ColorRL: Reinforced Coloring for End-to-End Instance Segmentation

Tran Anh Tuan[1], Nguyen Tuan Khoa[1], Tran Minh Quan[2], Won-Ki Jeong[3*]
[1] Department of Computer Science and Engineering, UNIST, Ulsan, Korea
[2] Department of Applied Science, VinBrain and VinUniversity, Vietnam
[3] Department of Computer Science and Engineering, Korea University, Seoul, Korea
{anhtuanhsgs,ntkhoa}@unist.ac.kr, v.quantran@vinbrain.net, wkjeong@korea.ac.kr

Abstract

Instance segmentation, the task of identifying and separating each individual object of interest in the image, is one of the actively studied research topics in computer vision. Although many feed-forward networks produce high-quality binary segmentation on different types of images, their final result heavily relies on the post-processing step, which separates instances from the binary mask. In comparison, the existing iterative methods extract a single object at a time using discriminative knowledge-based properties (e.g., shapes, boundaries, etc.) without relying on post-processing. However, they do not scale well with a large number of objects. To exploit the advantages of conventional sequential segmentation methods without impairing the scalability, we propose a novel iterative deep reinforcement learning agent that learns how to differentiate multiple objects in parallel. By constructing a relational graph between pixels, we design a reward function that encourages separating pixels of different objects and grouping pixels that belong to the same instance. We demonstrate that the proposed method can efficiently perform instance segmentation of many objects without heavy post-processing.

1. Introduction

Instance segmentation is one of the challenging computer vision problems that assigns instance labels to pixels to separate objects, which is crucial for understanding a complex scene. Many existing methods are based on complex graphical models with deep neural networks (e.g., convolutional neural network [CNN] or recurrent neural network [RNN])) [20, 30]. However, most other methods are still trying to predict the intermediate representation [27] of the labeling map, which require extra post-processing steps. Some object proposal approaches use bounding boxes to capture the representation of instances [7,21]. Even though, they produce instance maps directly, the bounding boxes are often criticized for their coarse representation of the object's shape. A recent study by Araslanov *et al.* [2] aimed to address the issue by employing reinforcement learning for a sequential object detection and segmentation task. Although such sequential approaches have shown promising results when it comes to directly detecting and extracting individual instances in the image, they are only applicable to images with a small number of objects due to the sequential nature of the method.

The motivation behind the proposed work stems from our recent development of the cell image segmentation method. In this development, we observed that separating every individual cell in a large microscopy image is a slow and time-consuming task for conventional segmentation methods. To address this problem, we propose a novel end-to-end instance segmentation method using reinforcement learning, which separates multiple instances in parallel. Unlike in Araslanov *et al.* where a single agent handles segmentation sequentially, our method leverages multiple pixel-wise agents (similar to Furuta *et al.* [5]) working concurrently to differentiate multiple objects in an iterative, end-to-end fashion (Fig. 1). To make multiple instances concurrently labeled, we formulate the segmentation problem as an iterative binary graph coloring problem. To achieve this, we employ the asynchronous advantage actor-critic (A3C) algorithm to train the agents to choose the t-th bit value in a binary representation of the label (0 or 1) at step t of the coloring process. We demonstrate that the proposed method can efficiently handle images that have numerous objects of various shapes while maintaining a superior segmentation quality comparable to state-of-the-art methods. To the best of our knowledge, this is the first reinforcement learning-based end-to-end instance segmentation that runs in parallel.

2. Related Work

In this section, we briefly review the recent advances in image segmentation methods that are closely related to the

[*]Corresponding author

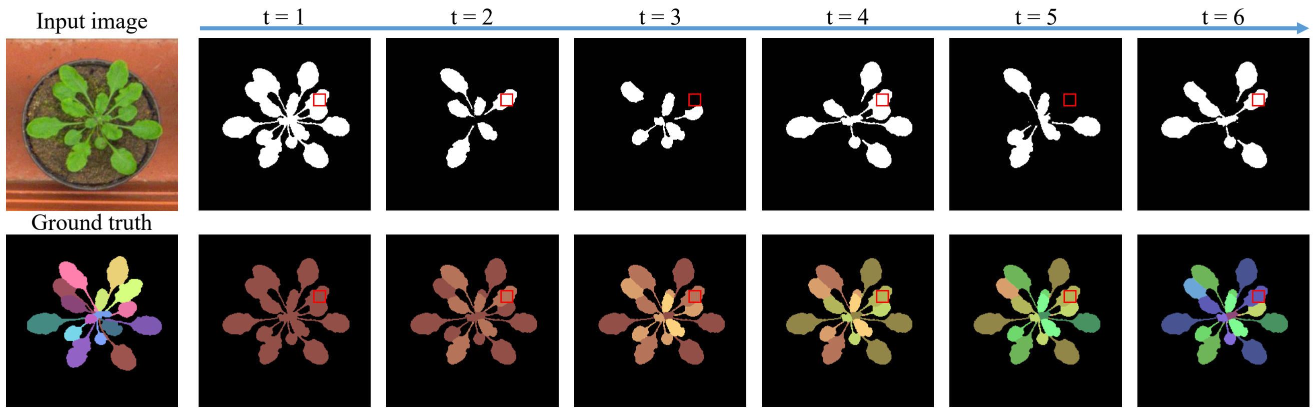


Figure 1. Illustration of our agent's coloring process through six coloring steps. At each step, multiple objects are segmented from the rest by the agent's action map (binary images, upper row) and have their color updated (colored images, bottom row). Also, the binary action map at each step is a binary digit map representing the segmentation label. For example, in the red box, the action values over the steps are $\{1,1,0,1,...\}$. Therefore, the binary label values are $\{1_2, 11_2, 011_2, 1011_2, ...\}$, which is equivalent to $\{1_{10}, 3_{10}, 3_{10}, 11_{10}\}$ in decimal form.

instance segmentation problem.

Object-proposal methods, such as Mask-RCNN [7] and PANet [29] , detect object bounding boxes and directly output the mask of each object. Nontheless, their use of bounding boxes can hurt the segmentation quality because of the coarse representation of the object's shape.

Another approach is using dense-prediction. For example, some methods predict a high dimensional encoding of the label map [4,11,17], which is then processed by clustering methods to extract individual instances. Specifically, in the electron microscope (EM) instance segmentation task, a commonly used approach to handle many densely-packed, irregularly-shaped objects is predicting binary object masks and then agglomerating information to produce the final results [13, 26]. 3C [14] predicts the mask of the target EM image in the base-k number system then agglomerate the over-segmentation labels across slides. These methods, however, relies on intermediate representations of the label and post-processing steps.

Iterative methods that predict and track the segmentation masks object-by-object do not need intermediate representation and bounding boxes. Araslanov *et al.* [2] used reinforcement learning (RL) agent to find a good order to predict each object at a time. Ren *et al.* [20] used a recurrent architecture to perform step-by-step attention and segmentation the mask of a single object. FFN [10] has been very successful in 3D EM segmentation by tracking a single object at a time. These methods, however, are constrained by their low scalability in processing many instances.

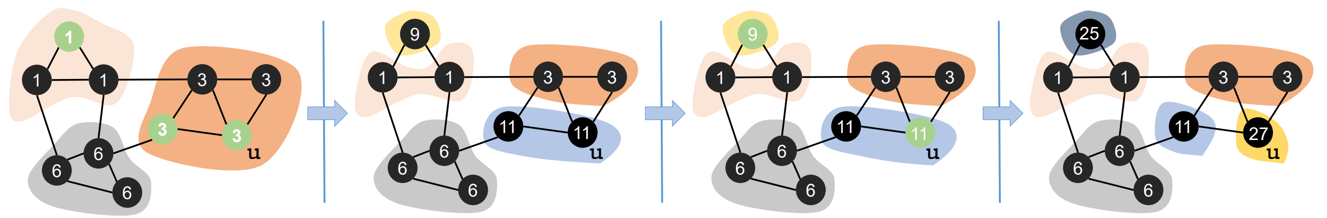
Since the seminal work by Mnih *et al.* [16], an increasing number of tasks requiring a complex sequence of decision-making processes have been solved by reinforcement learning [24, 25]. Inspired by this trend, reinforcement learning has been adopted to solve computer vision problems as well [2, 23, 25, 28]. For example, Furuta *et al.* presented an efficient way to train an asynchronous actor-critic agent (A3C), called PixelRL [5], that uses decision making per pixel for the denoising problem.

3. Method

3.1. Problem Formulation

In this work, similar to Gomez *et al.* [6], we formulate the instance segmentation problem into a multi-step graph coloring problem where the per-instance color (represented as a base-2 number) is determined by per-pixel RL agents via series of bit-value assignments. The feedback/rewards for each pixel's decision is determined by comparing the pixel's color with those of neighboring pixels, which are indicated by graph edges. More formal definitions and problem formulation are given below.

Let image *I* be the set of pixels $V = \{v_1, v_2, ..., v_N\}$, and a segmentation of *I* is a partition of V defined as $P = \{P_1, P_2, ..., P_M\}$, where each v_i belongs to exactly one subset of pixels P_j ($1 \leq i \leq N$ and $1 \leq j \leq M$). We denote the ground truth segmentation of *I* as the partition $\hat{P} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_K\}$. Then, the objective is to find P that is as close to $\hat{P}$ as possible. In our coloring approach, for each image *I*, we want to find a color mapping function that assigns a color to each pixel, $C : V \rightarrow \{c_1, c_2, ..., c_N\}, c_i \in \{0, 1, 2, ..., c-1\}$. Here, $C(V)$ denotes the color of all the pixels in V, and $C(V)[v]$ denotes the color of pixel v. The ideal condition for the pixels is that $C(V)[v] = C(V)[u]$ when $u, v \in \hat{P}_i$ for a subset $\hat{P}_i \in \hat{P}$ (i.e., pixels in the same object should have the same color). Also, $C(v)[v] \neq C(v)[u]$ when $u \in \hat{P}_i$ and $v \in \hat{P}_j$ for $i \neq j$ (i.e., pixels of different objects should have different colors). To relax the above conditions to be applied to only pixels of close proximity, we build a set of edges between pixels to indicate close proximity relationship. We



(a) Taking action at t = 3 (b) Partition of V at t = 4 (c) Taking action at t = 4 (d) Partition of V at t = 5

Figure 2. Illustration of the binary coloring process by the proposed agent. The number on each vertex represents its color label, and the green (or black) vertex during the action selection phase represents the action value of 1 (or 0). For example, at step $t = 4$, the vertex **u** with the color label of 11 ($C^{(4)}(V)[\mathbf{u}] = 11$, see (b)), it chooses the action value of 1 ($\mathcal{F}^{(4)}(V, C^{(4)}(V))[\mathbf{u}] = 1$ (green vertex), see (c)). Then, the color label of **u** at $t = 5$ becomes $11 + 2^4 \cdot 1 = 27$ (see (d))

define a graph G by considering V as a set of vertices (i.e., each pixel in I is a vertex in G) and can then construct the set of edges E that connect pixels. The graph coloring formulation for the instance segmentation problem is formally defined as follows: For a given graph $G = (V, E)$ and a ground truth partition $\hat{P} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_m\}$ of V, $C[V](v) = C[V](u)$ if there exists an edge $(u, v) \in E$ and $u, v \in \hat{P}_i$ for some $\hat{P}_i \in \hat{P}$. $C[V](v) \neq C[V](u)$ if there exists an edge $(u, v) \in E$ and $u \in \hat{P}_i, v \in \hat{P}_j$ for $\hat{P}_i, \hat{P}_j \in \hat{P}$ and $i \neq j$. The details for edge construction is stated in section 3.3.

Because directly finding a perfect color mapping C^* that satisfies the mentioned constraints is an intractable problem, we instead find an approximation of C^* via an iterative binary coloring process. We let $C^{(t)}(V)$ be the color mapping of V at time step t and define the binary coloring action $a^t = \mathcal{F}(V, C^{(t)}(V))$, in which $\mathcal{F}$ maps V to $\{0, 1\}^N$, and N is the size of V (i.e., each v in V is mapped to a value of either 0 or 1 through $\mathcal{F}$). For a pixel v, $\mathcal{F}^{(t)}(V, C^{(t)}(V))[v]$ denotes its binary mapped value. The color of v at time step $t + 1$ is computed as follows (illustrated in Fig. 2):

$$\begin{aligned} C^{(0)}(V)[v] &= 0 \\ C^{(t+1)}(V)[v] &= C^{(t)}(V)[v] + 2^t \mathcal{F}(V, C^{(t)}(V))[v] \end{aligned} \tag{1}$$

Here, $\mathcal{F}$ is a binary segmentation model that takes V and the color map $C^{(t)}(V)$ as its input. In this binary coloring scheme, at time step t, $a^{(t)}[v] = \mathcal{F}(V, C^{(t)}(V))[v]$, after being multiplied by 2^t, becomes the t-th least significant binary digit of the binary representation of the color of v. Through T coloring steps, we have the color mapping function $C^{(T)}$, which maps V to $\{c_1, c_2, ..., c_N\}, c_i \in \{0, 1, 2, ..., 2^T - 1\}$.

3.2. Coloring Agent

For the coloring problem, we can naturally think of a multi-agent system in which the agents decide the color $C^{(t+1)}$ of the pixels by taking actions $a^t = \mathcal{F}(V, C^{(t)})$. Our coloring agent consists of many pixel-level agents as in the PixelRL [5]. It processes the state $s^{(t)}$ at time step t to produce a binary map of N actions (one for each pixel), and each action decides the color of a single pixel. We formulate the Markov Decision Process for the instance segmentation problem as follows (Fig. 3 shows an overview of our agent architecture):

State: Like $\mathcal{F}^{(t)}$ in Eq. 1, our coloring agent takes the set of vertices V and the current color map $C^{(t)}(V)$ as the input. Thus, the state of the agent $s^{(t)}$ consists of V and $C^{(t)}(V)$. Given that image I has shape H×W×CH and there are T coloring steps, then the representation of V in the state is of shape H×W×CH. As for $C^{(t)}$, its state representation is the binary-valued array of shape H×W×T. The t-th binary map of shape H×W in the T channels is the binary map of the t-digit in the binary representation of the color map. Combining the two components, the state (input) of the agent is an array of shape H×W×(CH+T). Initially, at step $t = 0$, $C^{(t)}(V)$ is just a zero-valued array.

Actions: Action map a^t, which is a result of $\mathcal{F}(V, C^{(t)})[V]$, is a binary image of shape H×W. At each step t ($t > 0$), after the agent takes action, the color map $C^{(t)}$ of the agent's current state will have its t-th channel map replaced with a^t, resulting in $C^{(t+1)}$ for the next state $s^{(t+1)}$. Each action map a^t is a binary segmentation map of objects. By Eq. 1, the segmented objects by action a^t have their color updated and are separated from the unsegmented objects.

Rewards: Reward map $r^{(t)}$ is given after the agent takes an action a^t. At the pixel level, the per-pixel coloring agent that is in charge of pixel v takes an action $a^t[v]$ and receives a real-valued reward $r^t[v]$. $r^t[v]$ is the indicator that shows how good the action $a^t[v]$ is. The reward value is reduced when $a^t[v]$ generates merge (split) errors (i.e., pixels of different (same) objects have the same (different) colors). The reward value increases when the action rightly merges or splits pixels. To make the color comparison between pixels for the rewards, we construct a set of edges E between pixels from the ground truth partition (segmentation label) $\hat{P}$. Our reward has three components: one that encourages the splitting actions, another that encourages

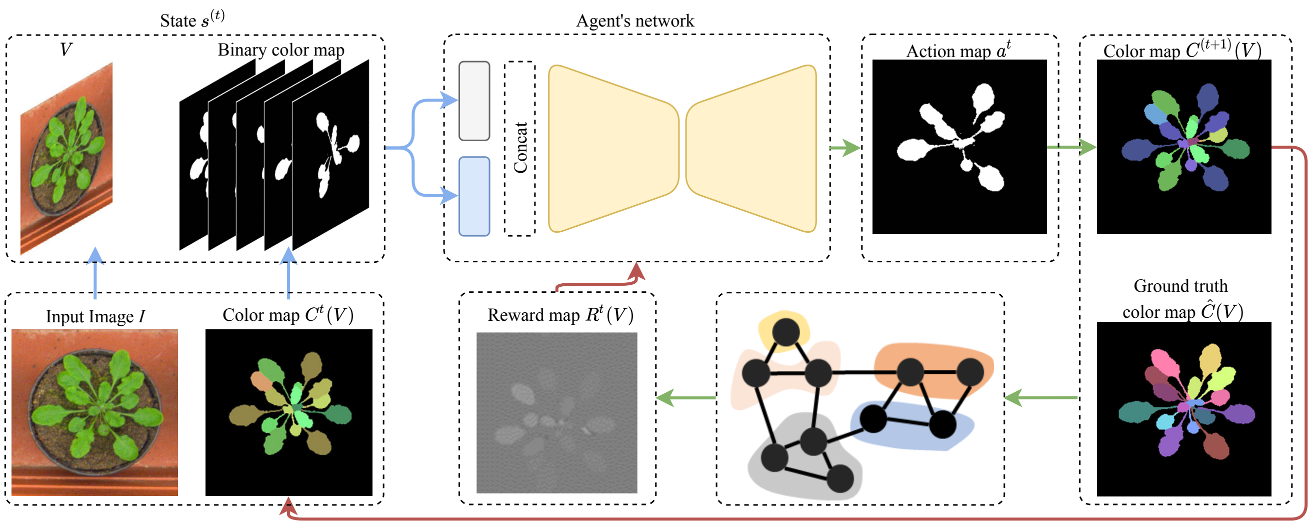
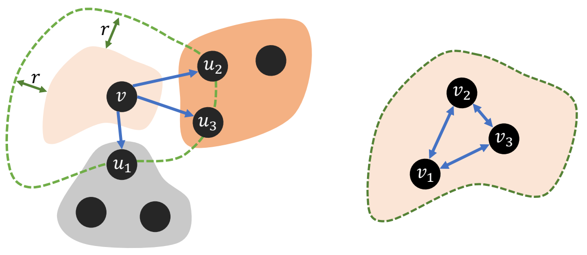


Figure 3. Overview of our coloring agent. The state of the agent comprises the sets of pixels V (input image I) and the binary color map $C^t(V)$. The input path (blue arrows) leads the current state to the agent network. The input image I and the binary color map are processed by two different modules. The outputs of the modules concatenate and then go through a CNN to produce the action map. The action-related path (green arrows) produces a new color map $C^{(t+1)}(V)$ from the action map a^t. Then, a graph algorithm will take the ground truth coloring $\hat{C}(V)$ and $C^{(t+1)}(V)$ to produce reward map $R^{(t)}(v)$. Red arrows indicate update-related paths where the network and state are updated using a new reward and a color map, respectively.



(a) Edges for splitting - $E^r(v)$ (b) Edges for merging - E_m

Figure 4. Edges for the splitting and merging rewards computation. (a): The splitting reward for v is computed on the edges connected to pixels of different objects within the distance r from the object containing v. (b): The merging reward is computed on the bi-directional edges within the same object.

merging actions, and a third that separates foreground and background objects.

3.3. Reward Function

For convenience, let $C(V)$ be the colormap for the set of pixels V while $C(v)$ is the mapping of a single-pixel v (the same meaning with $C(V)[v]$). We denote $\hat{C}$ as the perfect color mapping function (i.e., the ground truth coloring function) and let the background color be 0 ($\hat{C}(v) = 0$ if v is a background pixel). Also, $\hat{P}(v_i)$ is defined as the set of pixels of the ground truth object that contains v_i (i.e., $\hat{P}(v_i) = \{v_j \mid \hat{C}(v_i) = \hat{C}(v_j)\}$). As mentioned earlier in section 3.1, a relational graph and coloring constraints are constructed from the ground truth segmentation. During training, for each image, a graph is constructed with two types of edges (one for splitting and the other for merging) to provide feedback for the agent's coloring decisions.

Reward for predicting background-foreground: The background object has the most irregular shape and size, especially in the cell images. Thus, it is reasonable to first separate the background object from the other objects. In the first step, we let our agent separate foreground objects from the background; once the background has been separated, then its color is not changed in the later coloring steps (only the pixels belonging to foreground objects are changed). At time step t, the background-foreground reward R_{BF} of a pixel v is defined as follows:

$$R_{BF}^{(t)}(v) = \begin{cases} r_{bg} & \text{if } C^{(t+1)}(u) = 0, \hat{C}(v) = 0 \\ 0 & \text{if } \hat{C}^(v) \neq 0, t > 0 \\ -r_{bg} & \text{if } C^{(t+1)}(u) \neq 0, \hat{C}(v) = 0 \\ r_{fg} & \text{if } C^{(t+1)}(u) = 1, \hat{C}(v) \neq 0, t = 0 \\ -r_{fg} & \text{if } C^{(t+1)}(u) = 0, \hat{C}(v) \neq 0, t = 0 \end{cases}$$

where r_{bg} and r_{fg} are the proportions of pixels that are of foreground and of background respectively.

Reward for splitting actions: Pixel-level agents gain the spatial information of surrounding pixels through the relational graphs between pixels. We denote a directed edge from pixel u to pixel v as (u, v). Given a radius r, the set of directed edges between pixels constructed with r is denoted as $E^r$and $E^r(u) \subset E^r$ is the set of edges that originates from u. Let a pixel u belong to the foreground object $\hat{P}(u)$; if pixel v of another foreground object that is within the distance r from $\hat{P}(u)$, then (u, v) is in $E^r(u)$ (Fig. 4a). To state the definition of E^r formally, $(u, v) \in E^r(u)$ if $\hat{C}(v) \neq \hat{C}(u)$, $\hat{C}(v) \neq 0$, $\hat{C}(u) \neq 0$ and $\exists u' \in \hat{P}(u)$ such that $d(u', v) < r$, where $d(u', v)$ is the Manhattan distance between u' and v. We consider r as the radius of segments and call it the *splitting radius*. The sets of falsely merged pixels $FM_r^{(t)}(v)$ and truly split pixels $TS_r^{(t)}(v)$ for a pixel v at time step t are defined as follows:

$$TS_r^{(t)}(v) = \{u \mid (v, u) \in E^r, C^{(t)}(u) \neq C^{(t)}(v), \hat{C}^{(t)}(u) \neq \hat{C}^{(t)}(v)\}$$
$$FM_r^{(t)}(v) = \{u \mid (v, u) \in E^r, C^{(t)}(u) = C^{(t)}(v), \hat{C}^{(t)}(u) \neq \hat{C}^{(t)}(v)\}$$

With the set of edges E^r, at time step t $(1 \leq t)$, the splitting reward component $R_S^{(t)}(v \mid E^r)$ for the pixel-level agent that is in charge of pixel u is computed as follows:

$$R_{TS}^{(t)}(v \mid E^r) = \frac{|TS_r^{(t+1)}(v)| - |TS_r^{(t)}(v)|}{|E^r(v)|}$$
$$R_{FM}^{(t)}(v \mid E^r) = \frac{1}{T} \frac{|FM_r^{(t+1)}(v)|}{|E^r(v)|}$$

$R_{TS}^{(t)}(v \mid E^r)$ is the percentage of pixels that are truly (correctly) split from v by the action map $a^{(t)}$ to the total number of pixels that should be split from v. $R_{FM}^{(t)}(v \mid E^r)$ is the percentage of pixels that are still being falsely (wrongly) merged to the total number of pixels that should be split from v, which is normalized by the total number of coloring steps. By combining R_{TS} as the reward (positive component) and R_{FM} as the penalty (negative component) in Eq. 2, we have the splitting reward function R_S:

$$R_S^{(t)}(v \mid E^r) = R_{TS}^{(t)}(v \mid E^r) - R_{FM}^{(t)}(v \mid E^r)$$

We use multiple sets E^r with different r, which gives a higher priority for the splitting of closer pixel pairs.

Reward for merging actions: Similar to the splitting reward component, we also build edges between pixels that should have the same color. The set of directed edges E_m $E_m(u) \subset E_m$ are constructed as follows: if u and v are of foreground objects and $\hat{C}(u) = \hat{C}(v)$, then $(u, v) \in E_m(u)$ and $(v, u) \in E_m(v)$. An illustration of edge construction for merging reward is in Fig. 4b. The set of truly merged pixels $TM^{(t)}(v)$ and falsely splited pixels $FS^{(t)}(v)$ for a pixel of foreground object v at time step t is defined as follows:

$$FS^{(t)}(v) = \{u \mid (v, u) \in E_m, C^{(t)}(u) \neq C^{(t)}(v), \hat{C}^{(t)}(u) = \hat{C}^{(t)}(v)\}$$
$$TM^{(t)}(v) = \{u \mid (v, u) \in E_m, C^{(t)}(u) = C^{(t)}(v), \hat{C}^{(t)}(u) = \hat{C}^{(t)}(v)\}$$

With the set of edges E_m, at time step t, where $t \geq 1$, the merging reward component $R_M^{(t)}(v \mid E_m)$ for the pixel-level agent that is in charge of of pixel u is computed with the following:

$$R_{TM}^{(t)}(v \mid E_m) = \frac{1}{T} \frac{|TM_m^{(t+1)}(v)|}{|E_m[v]|}$$
$$R_{FS}^{(t)}(v \mid E_m) = \frac{|FS_m^{(t)}(v)| - |FS_m^{(t+1)}(v)|}{|E_m[v]|}$$

$R_{TM}^{(t)}(v \mid E_m)$ is the percentage of the number of pixels that is still truly (correctly) merged with v to the total number of pixels that should have the same color as v, which is normalized by the total number of step T. $R_{FS}^{(t)}(v \mid E_m)$ is the percentage of the pixels that are falsely (wrongly) split from v as a result of action map $a^{(t)}$. By using R_{TM} as the reward component and R_{FS} as the penalty component we form the merging reward function R_M:

$$R_M^{(t)}(v \mid E_m) = R_{TM}^{(t)}(v \mid E_m) - R_{FS}^{(t)}(v \mid E_m)$$

Reward for pixel v at time step t: The reward function $R^{(t)}(v)$ for the pixel-level agent at pixel v is:
When $1 \leq t < T$:

$$R^{(t)}(v) = R_{BF}^{(t)}(v) + w_m R_M^{(t)}(v \mid E_m) + w_s \sum_{E^r \in \mathcal{G}_s} R_S^{(t)}(v \mid E^r)$$

and when $t = 0$:

$$R^{(t)}(v) = R_{BF}^{(t)}(v)$$

Here, w_m and w_s are the weights for merging reward and splitting reward, respectively. $\mathcal{G}_s$ is the sets of E^r(s) for different values of r.

4. Experiments and Results

4.1. Experimental settings

For 2D image instance segmentation, we used a U-Net-based architecture [18] for the core network.For each dataset, we empirically chose the set of hyperparameters w_s, w_m, and r(s) for the best performance on the validation set and chose T so that 2^{T-1} (the maximum number of colors that our agent can assign) is greater than the maximum number of neighboring objects that an object can have in the training set. During the evaluation, because our agent allows far apart objects to have the same color, we perform connected-component labeling for the post-processing procedure. For more information related to the data preparation and implementational and experimental details, please see the appendix. Our code is published online [1].

4.2. Ablation Study

Splitting radius setting: With multiple radii r, the edges of a smaller radius are counted for a reward more times than the edges of a larger radius are. Thus, this gives a higher priority to closer pixel pairs and makes the coloring agent focus more on separating the objects that are close in proximity. Here, we analyze the behavior of our agent with two levels of splitting radii r_1 and r_2. We conduct this analysis in a simplified environment by letting the agent learn to color a single image with no augmentation with $w_s = w_m = 1$. We experimented with six pairs of radii on the CVPPP dataset with an input image size of

[1]https://github.com/anhtuanhsgs/ColorRL

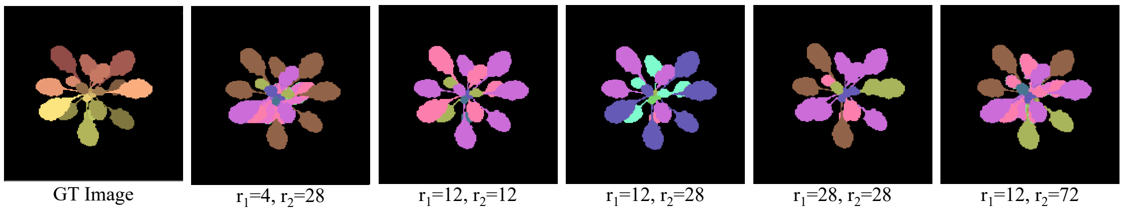


Figure 5. Coloring results from the agent after it learn to perform coloring on a single image with different settings for splitting radii. We experimented with two levels of radius r_1 and r_2, and let $r_1 \leq r_2$.

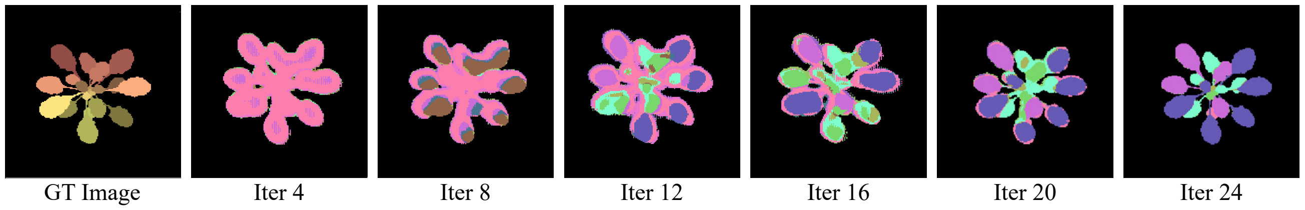


Figure 6. The ground truth segmentation and the intermediate results of the agent at different training iterations.

176×176. Fig. 5 shows the final results of the agent after six steps. We observed that $r_1 = 12$ and $r_2 = 28$ gave the best result among the trials. Small radii ($r = 4$) do not give the agents enough useful guidance to separate large and far apart objects. Large radii ($r = 72$), on the other hand, require each pixel-level agent to look further and process more information, thus making the task more complex.

Weights for splitting and merging rewards: To continue the single image coloring experiment, we analyze how splitting reward R_s and merging reward R_m influence the agent and change during the learning session. Influenced by the reward R_{BF} (separating only the foreground and background in the first step), the agent learns to segment between foreground and background first then gradually learns to segment the foreground leaves (Fig. 6).

Next, we analyze how the agent behaves with different pairs of w_s and w_m. We used 103 training images and 25 validation images of the CVPPP dataset in this experiment. Table 1 and Fig. 7 show the segmentation results. The agent tries to make more merging actions during early training iterations early on due to the influence of R_{BF} (Fig. 6). Thus, when $w_s < w_m$, it is much difficult for the agent to learn to separate the leaves since the reward is not high enough for the agent to explore for a better coloring solution and to go out of the local maxima. We conclude that during training, the agent should have $w_s \geq w_m$ so that there are incentives for it to explore more complex coloring decisions.

4.3. CVPPP Dataset

The Computer Vision Problems in Plants Phenotyping (CVPPP) dataset [15] is one of the popular datasets used for assessing the performance of instance segmentation algorithms. Our agent's results are shown in Fig. 8. The results in Table 2 show that our agent can generate results comparable to the state-of-the-art methods in terms of the object count and segmentation accuracy.

Table 1. Results of our reinforced graph coloring agent (GraphCL) on the CVPPP validation set with different weight settings

Model	w_s	w_m	r_1	r_2	SBD↑	|DiC|↓
ColorRL	0.00	2.00	12	28	21.2	15.6
	0.25	1.75			21.2	9.44
	0.50	1.50			70.7	2.72
	1.00	1.00			85.2	1.40
	1.50	0.50			87.3	1.34
	1.7	0.25			81.4	1.44
	2.00	0.00			5.60	92.8

Table 2. Segmentation quality of the CVPPP testset.

Model	SBD↑	|DiC|↓
MSU [22]	66.7	2.3
Nottingham [22]	68.3	3.8
IPK [19]	74.4	2.6
DLoss [4]	84.2	1.0
E2E [20]	84.9	0.8
AC-Dice [2]	79.1	1.12
Ours *(ColorRL)*	80.0	1.36

4.4. MoNuSeg Dataset

From the MoNuSeg 2018 nuclear segmentation challenge [12], we prepared two versions of the data that have the same scale but in different sizes: MNSeg-160 (image size of 160x160) and MNSeg-224 (image size of 224x224). The average (avg. cnt) and maximum (max cnt) number of objects are included for each dataset. The nuclei varies in sizes, positions, and density in each image. For this experiment, we used the Mask R-CNN (mrcnn) implementation

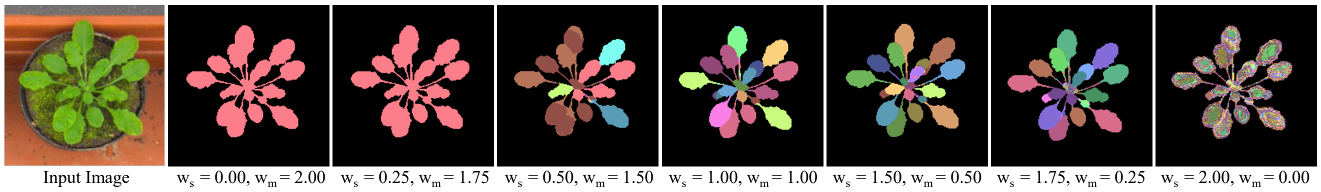


Figure 7. Coloring results of the agent trained with different pairs of splitting and merging weights.

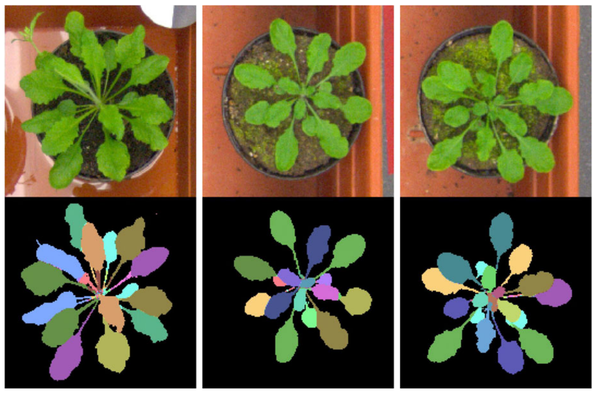


Figure 8. Coloring results of CVPPP test set before post-processing. Top row: input images, bottom row: coloring results.

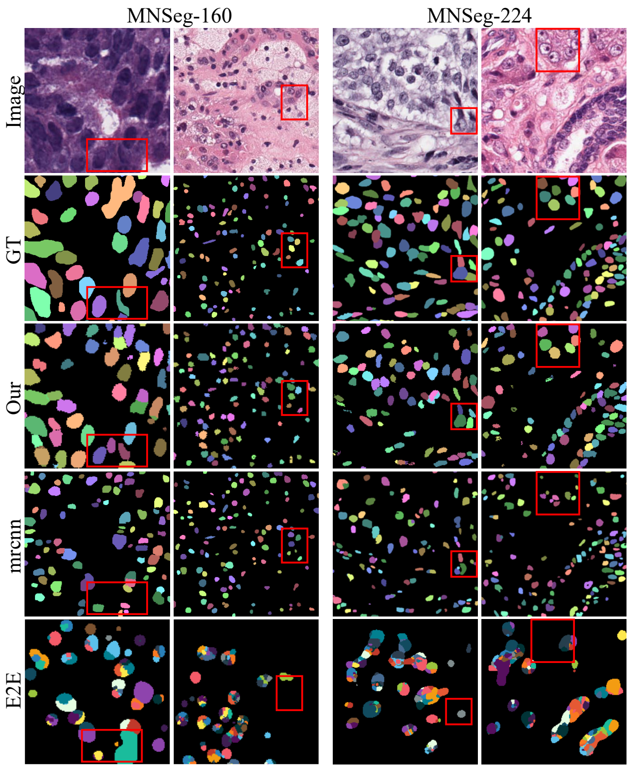


Figure 9. Experiment results of MoNuSeg-160 (two left columns) and MoNuSeg-224 (two right columns) test sets.

from [1] with the ResNet50 [8] backbone. The results are shown in Fig. 9 and Table 3.

Table 3. Segmentation quality of the MoNuSeg results. We evaluate the segmentation performance in terms of the aggregated Jaccard index (AJI) adapted rand index (ARand) and average inference time per image (avg. time)

Dataset	avg. cnt	max cnt	Model	avg. time (ms)	ARand↓	AJI↑
MNSeg-160	90	361	E2E	7860	0.779	0.110
			Mask RCNN	315.57	0.546	0.412
			ColorRL	**57.039**	**0.261**	**0.557**
MNSeg-224	182	603	E2E	9578	0.826	0.072
			Mask RCNN	642.93	0.6644	0.333
			ColorRL	**58.115**	**0.319**	**0.537**

Table 4. Segmentation quality of the CREMI testset. The metrics we used are VOI-split, VOI-merge, adapted rand index (ARand), and average inference time per image (avg. time). We omit the evaluation of ACIS for Cre-448 as it did not show any reasonable result.

Dataset	avg. cnt	max cnt	Model	avg. time (ms)	VOI-split↓	VOI-merge↓	ARand↓
Cre-160	16	25	E2E	223.72	0.463	0.394	0.129
			mrcnn	296.21	0.856	0.354	0.239
			ACIS	107.33	0.420	0.395	0.134
			Our	**81.13**	**0.248**	**0.135**	**0.034**
Cre-256	24	43	E2E	514.83	0.772	0.544	0.276
			mrcnn	359.71	0.660	0.493	0.206
			ACIS	118.50	0.657	1.316	0.365
			Our	**99.82**	**0.412**	**0.113**	**0.07**
Cre-448	65	80	E2E	910.46	1.178	3.082	0.660
			mrcnn	383.56	0.757	0.976	0.340
			Our	**98.33**	**0.379**	**0.230**	**0.095**

4.5. CREMI Dataset

From dataset A of CREMI [3], we prepared three different datasets: Cre-160, Cre-256, and Cre-448. The three datasets have the same image size of 224×224 but different scales and average number of objects per image. The cells in CREMI are densely packed and have very irregular shapes and sizes. We let our agent perform coloring through six steps. Table 4 shows our experiment results. The results show that conventional iterative methods, such as E2E and ACIS, do not handle many objects well. When the number of objects increases, the segmentation task becomes much

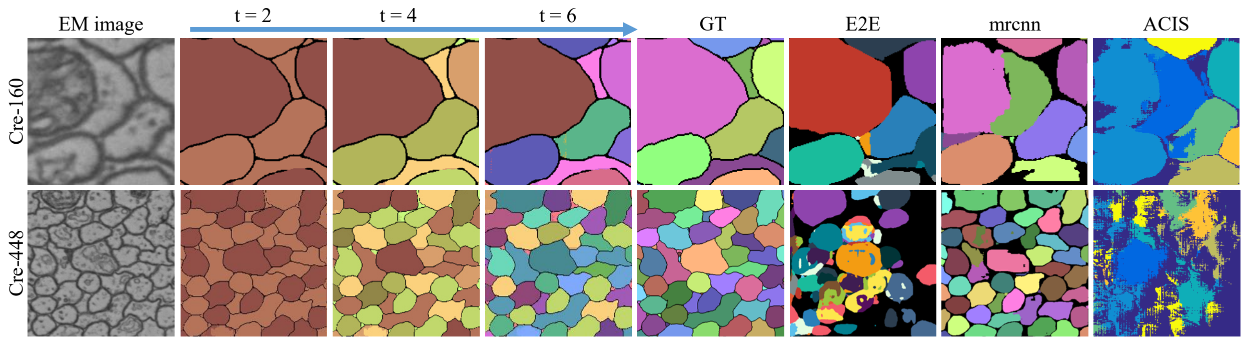


Figure 10. Experiment results of Cre-160 and Cre-448. We show our agent's coloring results from steps 2, 4, and 6.

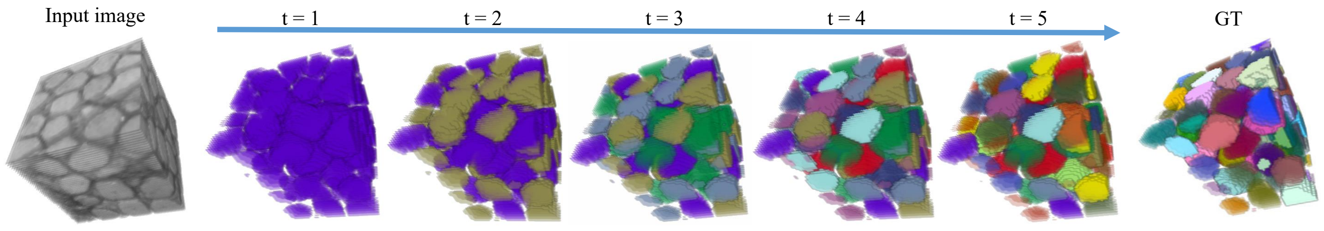


Figure 11. Step-by-step coloring result of the 3D electron microscopy volume of a Zebrafish through five steps.

Table 5. Segmentation quality of 3D larval zebrafish testset. We measured segmentation quality in terms of the symetric best dice score (SBD) and adapted RAND error (ARAND)

Method	SBD↑	ARand↓
U-Net3D + Connected Component	74.4	0.560
U-Net3D + Watershed	87.9	0.054
ColorRL	**90.0**	**0.038**

more challenging and takes longer. Mask R-CNN showed trouble when learning the irregular shape of the cells. Note also that our method scales well over a large number of objects (the running time does not increase as the number of objects increases). However, since our agent relies on the network for long range information, segmentation of extremely large or irregularly-shaped instances can be challenging (such as CREMI B and C datasets). We believe that designing a better core network with a more generalized reward formulation will solve this problem, which is left for future work.

4.6. Larval Zebrafish EM data

In this experiment, we tested ColorRL on a 3D stack of serial-section electron microscopy images of a of zebrafish brain [9]. The graph construction and coloring algorithm of ColorRL can be easily extended to a 3D setup without any change. We used a 3D U-Net as the core architecture for the coloring agent. We compared our method with a binary segmentation baseline from the same 3D U-Net model with a binary cross-entropy loss that was post-processed with a 3D watershed transform and connected component analysis. The results are shown in Fig. 11 and Table 5. It is shown that our agent takes advantage of the binary segmentation model's strength and performs segmentation in a more instance-oriented way to produce better results.

5. Conclusion

In this paper, we introduced a novel per-pixel label assignment method for end-to-end instance segmentation based on a graph coloring approach. Without the need of intermediate representations, our parallel per-pixel coloring agents directly label and separate many objects concurrently. We also demonstrated that our method can be easily extended to 3D volume instance segmentation problems. With this promising results, we plan to apply our method to large scale connectomics segmentation. We expect that designing a better core network with a more generalized reward formulation will handle EM volume of large and complex instances. Adopting constancy constraints between sections of EM images/volumes as in 3C is another interesting future direction to explore.

Acknowledgement

This work is supported by the Korea Health Industry Development Institute (HI18C0316), the National Research Foundation of Korea (NRF-2019M3E5D2A01063819), the Institute for Information & communications Technology Planning & Evaluation (IITP-2021-0-01819), and the Korea Institute of Science and Technology (KIST) Institutional Program, Republic of Korea (2E30970).

References

[1] Waleed Abdulla. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. `https://github.com/matterport/Mask_RCNN`, 2017.

[2] Nikita Araslanov, Constantin A Rothkopf, and Stefan Roth. Actor-Critic Instance Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8237–8246, 2019.

[3] CREMI. Miccai challenge on circuit reconstruction from electron microscopy images, 2017. `https://cremi.org/`.

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[5] Ryosuke Furuta, Naoto Inoue, and Toshihiko Yamasaki. Fully convolutional network with multi-step reinforcement learning for image processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3598–3605, 2019.

[6] D Gómez, J Montero, J Yáñez, and C Poidomani. A graph coloring approach for image segmentation. *Omega*, 35(2):173–183, 2007.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

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

[9] David Grant Colburn Hildebrand, Marcelo Cicconet, Russel Miguel Torres, Woohyuk Choi, Tran Minh Quan, Jungmin Moon, Arthur Willis Wetzel, Andrew Scott Champion, Brett Jesse Graham, Owen Randlett, et al. Whole-brain serial-section electron microscopy in larval zebrafish. *Nature*, 545(7654):345–349, 2017.

[10] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605–610, 2018.

[11] Shu Kong and Charless C Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 956–965, 2018.

[12] Neeraj Kumar, Verma, et al. A multi-organ nucleus segmentation challenge. *IEEE transactions on medical imaging*, 39(5):1380–1391, 2019.

[13] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the SNEMI3D connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[14] Yaron Meirovitch, Lu Mi, Hayk Saribekyan, Alexander Matveev, David Rolnick, and Nir Shavit. Cross-classification clustering: An efficient multi-object tracking technique for 3-D instance segmentation in connectomics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8425–8435, 2019.

[15] Massimo Minervini, Fischbach, et al. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern recognition letters*, 81:80–89, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[17] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[18] Ozan Oktay, Jo Schlemper, Folgoc, et al. Attention U-Net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[19] Jean-Michel Pape and Christian Klukas. 3-D histogram-based segmentation and leaf detection for rosette plants. In *European Conference on Computer Vision*, pages 61–74. Springer, 2014.

[20] Mengye Ren and Richard S Zemel. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6656–6664, 2017.

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

[22] Hanno Scharr, Massimo Minervini, et al. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications*, 27(4):585–606, 2016.

[23] Jianzhun Shao, Yuhang Jiang, Gu Wang, Zhigang Li, and Xiangyang Ji. PFRL: Pose-Free Reinforcement Learning for 6D Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11454–11463, 2020.

[24] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Sifre, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[25] Gwangmo Song, Heesoo Myeong, and Kyoung Mu Lee. Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1760–1768, 2018.

[26] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural computation*, 22(2):511–538, 2010.

[27] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.

[28] Burak Uzkent and Stefano Ermon. Learning when and where to zoom with deep reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12345–12354, 2020.

[29] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. PANet: Few-shot image semantic segmentation with prototype alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9197–9206, 2019.

[30] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected MRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 669–677, 2016.