Single-shot Path Integrated Panoptic Segmentation

Sukjun Hwang[1]   Seoung Wug Oh[2]   Seon Joo Kim[1]

[1]Yonsei University   [2]Adobe Research

Abstract

Panoptic segmentation, which is a novel task of unifying instance segmentation and semantic segmentation, has attracted a lot of attention lately. However, most of the previous methods are composed of multiple pathways with each pathway specialized to a designated segmentation task. In this paper, we propose to resolve panoptic segmentation in single-shot by integrating the execution flows. With the integrated pathway, a unified feature map called Panoptic-Feature is generated, which includes the information of both things and stuffs. Panoptic-Feature becomes more sophisticated by auxiliary problems that guide to cluster pixels that belong to the same instance and differentiate between objects of different classes. A collection of convolutional filters, where each filter represents either a thing or stuff, is applied to Panoptic-Feature at once, materializing the single-shot panoptic segmentation. Taking the advantages of both top-down and bottom-up approaches, our method, named SPINet, enjoys high efficiency and accuracy on major panoptic segmentation benchmarks: COCO and Cityscapes.

1. Introduction

Panoptic segmentation, first proposed by Kirillov *et al.* [15], is a task with the goal of annotating each pixel to the corresponding category. While semantic segmentation limits the term 'category' to class labels, panoptic segmentation extends the term to include the concept of instances. An early solution for the task was to use completely separate leading algorithms for instance and semantic segmentation [15]. However, separating models for each task not only conflicts with the motivation behind panoptic segmentation but also doubles the computation and the model size. As panoptic segmentation is based on the philosophy of unifying two different tasks, an ideal solution for the problem would be finding the best joint framework for both tasks.

To derive a joint solution for the new task, previous approaches started to adopt powerful instance or semantic segmentation models as baselines. Most of currently leading
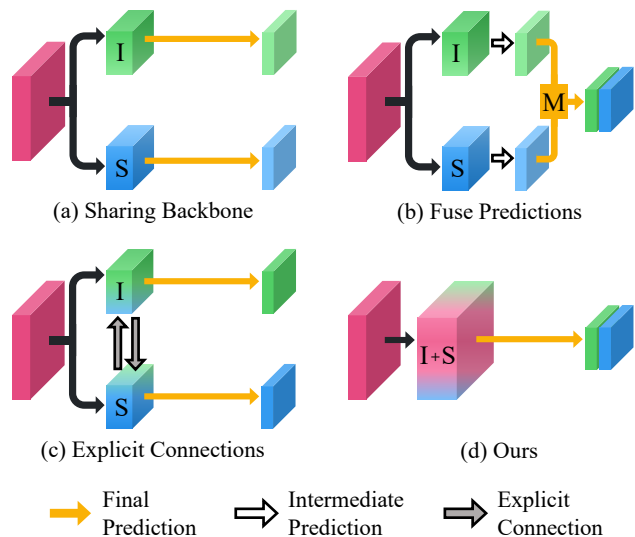

Figure 1. Panoptic segmentation with SPINet. (a) Most panoptic segmentation models tackle the task by separating the execution flow into two: instance branch I, and semantic branch S. (b) Intermediate predictions from I and S can generate better final prediction by using an additional module M. (c) Explicit connections between I and S alleviate the isolation of information. (d) Our model integrates I and S, performing panoptic segmentation within a single feature map. Through this integration, we can achieve improvement both in performance and efficiency.

panoptic segmentation models put their basis on either (top-down) Mask R-CNN [11] or (bottom-up) DeepLab [4]. As the baseline itself cannot fully perform panoptic segmentation, additional modules become necessary: a semantic branch for Mask R-CNN [14] and an instance branch for DeepLab [43].

Efficiency-wise, the strategy of supplying additional modules on top of the chosen basis was successful [7, 14, 26] (Fig. 1 (a)). Compared to executing two independent networks, the concept of sharing backbone layers for both instance and semantic branch led to a huge reduction of computations. Further improvement in performance was achieved by putting a connection between the two branches.

For instance, by including an extra module that takes intermediate predictions [17, 42] from the two branches resulted in performance improvement (Fig. 1 (b)). Moreover, supplying explicit connections between the branches [6, 18, 40] as shown in Fig. 1 (c), also enhanced the predictions as each branch complements from the information of their counterparts. However, these methods still have separate pathways for instance and semantic segmentation, leaving room for further structural unification.

In this paper, we propose **SPINet** with the goal of integrating the branches of instance and semantic segmentation (Fig. 1 (d)). By unifying two pathways, SPINet generates a single feature map called Panoptic-Feature, used for segmenting both thing and stuff. In contrast to previous structures with separate branches, Panoptic-Feature holds the information of both thing and stuff, thus it bridges the gap between the two without any explicit connections. Additionally, the computations necessary for capturing the context at high-level while retaining low-level fine details can be shared with the integrated pathway, making our model highly efficient.

As we integrate the execution flow of thing and stuff altogether, the panoptic segmentation task can be finalized in *single-shot* by applying a unified convolution to Panoptic-Feature. The weights of thing classes are generated dynamically by reading the context of the input scene similar to CondInst [31], and trainable parameters are used for the weights of stuff classes. By collecting all the weights and applying these to the Panoptic-Feature by the *single-shot* convolution, SPINet can generate masks for every things and stuffs in the scene at once.

The key to the success of the proposed method lies in the representation power of the Panoptic-Feature. To strengthen the Panoptic-Feature, we propose auxiliary tasks in order to learn a better latent space. The auxiliary tasks are designed to lead points in Panoptic-Feature to be clustered if they belong to the same class and moreover the same instance. Trained with additional guidance signals from the auxiliary tasks, we show that the performance of our panoptic segmentation gains further improvement.

SPINet achieves the state-of-the-art performance even without using heavy modules such as ASPP [4] or deformable convolution [9]. Moreover, by integrating the characteristics of both top-down and bottom-up, our model resolves the quality imbalance issue of previous approaches; top-down based models acquiring high PQ^{th} but comparatively low PQ^{st1}, and vice-versa. For the first time, SPINet achieves comparable results to two-stage methods on COCO [21] without the region proposal network. Moreover, our model delivers the state-of-the-art performance while yielding huge efficiency on Cityscapes [8]. With ResNet-50-FPN backbone, SPINet achieves 63.5% PQ on Cityscapes *val* set, and 43.0% PQ on COCO *val* set.

2. Related Works

2.1. Panoptic Segmentation Approaches

Mask R-CNN Based. Mask R-CNN [11] is a well-known instance segmentation model which detects instances first, then searches for finer details for segmentation. Panoptic-FPN [14] proposed using Mask R-CNN for the new panoptic segmentation task with an additional semantic segmentation branch. Numerous panoptic segmentation models [6, 16, 17, 18, 22, 40, 42] are built on top of Panoptic-FPN structure, increasing the performance with additional enhancement techniques. Since 'detect-then-segment' pipeline shows strength in capturing objects, panoptic segmentation methods based on Mask R-CNN show high PQ^{th}. However, as the models are not focused on preserving pixel-level details, PQ^{st} is relatively low.

DeepLab Based. DeepLab [4] is a strong semantic segmentation model, which focuses on fine-grained details by taking encoder-decoder like structure to recover spatial resolution. Panoptic segmentation models that adopt DeepLab [7, 43] not only generate pixel-level predictions for stuff but also for thing. With the fine-grained predictions, formation of instances is possible by a bottom-up methodology; aggregating the pixels that have similar aspects. As opposed to top-down models with Mask R-CNN, bottom-up panoptic segmentation models acquire high PQ^{st}, but low PQ^{th}.

DETR. Unifying thing and stuff classes altogether, DETR [3] predicts which categories exist in the input. With a number of attention heatmaps, generated per each predicted category by the transformer [33], mask logits corresponding to each category can be generated. The final prediction of panoptic segmentation can be done by a single pixel-wise `argmax` on the predicted mask logits. Though the methodology is noticeably simple, DETR requires an extremely large computing resource as it needs to decode the attention heatmaps for every instances separately in addition to the heavy base computation from the use of the transformer.

SPINet shows aspects of *bottom-up* as each point of Panoptic-Feature retains information about which class and instance they originate from. Meanwhile, the generation of dynamic filters catches the context of instances [31], hence our model can also be viewed as *top-down*. Similar to DETR, SPINet integrates the execution flow of thing and stuff by treating the both equally, but lighter as SPINet requires less steps to generate predictions. Taking the advantages of top-down, bottom-up, and unified execution flow, SPINet resolves the performance imbalance issue, showing competitive results on both COCO and Cityscapes with efficiency.

---

[1]PQ^{th} and PQ^{st} are *panoptic quality* (PQ) averaged over *thing* classes and *stuff* classes respectively. [15]
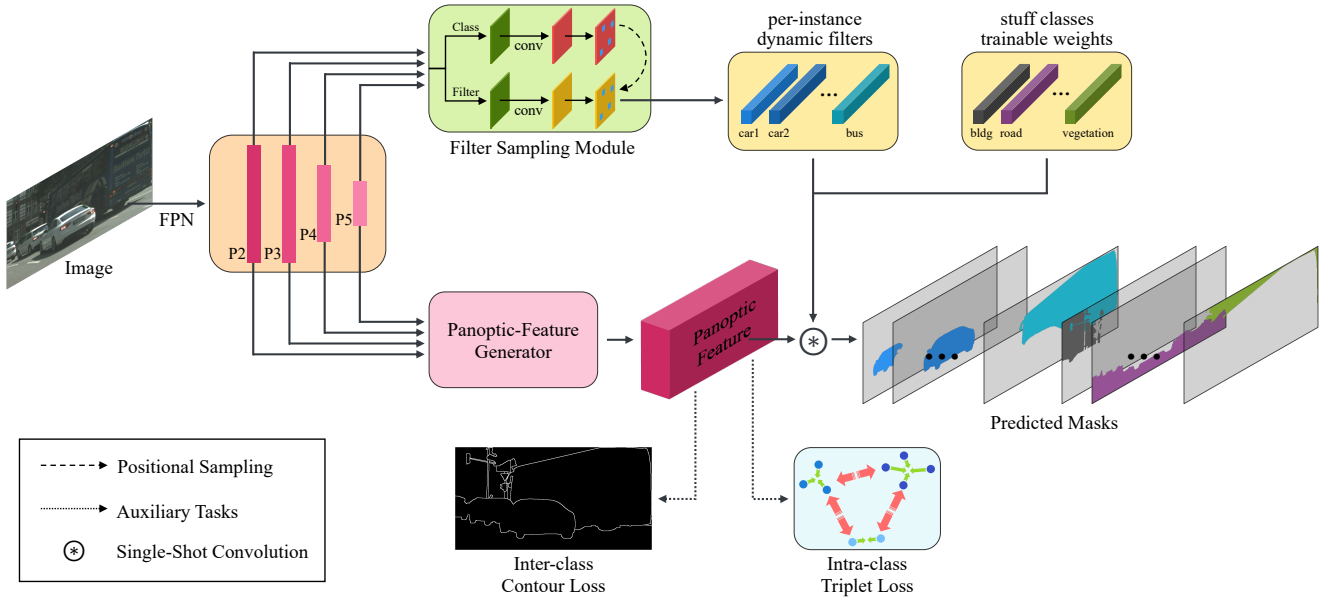
Figure 2. Overall execution flow of our proposed SPINet. The architecture is mainly composed of three modules: FPN, filter sampling module and Panoptic-Feature generator (Fig. 3). The three modules harmonize their use by the single-shot convolution, which generates masks for both instance and semantic segmentation at once.

2.2. One-stage Instance Segmentation

Latest object detection and instance segmentation models can be categorized by the existence of region proposal network (RPN) [28]: one-stage [20, 24, 27] and two-stage [2, 11, 28]. After the emergence of Mask-RCNN [11], the majority of instance segmentation models is built on top of two-stage detectors, hence improvement in object detection spontaneously led to performance gain in instance segmentation. RPN shows substantial performance, yet the extraction of regional features remains as a bottleneck. To overcome the issue, there has been a lot of improvement on one-stage detectors such as YOLO [27], and FCOS [32]. Many one-stage methods for instance segmentation [31, 35, 36] are now showing comparable performance to that of two-stage based methods.

Recently proposed CondInst [31] generates dynamic convolutional weights that correspond to an instance for each feature location. Our model adopts dynamic filters proposed by CondInst for segmenting instances. By equally treating the filters for thing and stuff, SPINet can finalize the panoptic segmentation with a single-shot convolution.

3. SPINet

As the primary goal of SPINet is to integrate the pathways of thing and stuff, our model has a simple execution flow as illustrated in Fig. 2. SPINet is composed of four steps: FPN backbone, filter sampling module, Panoptic-Feature generator, and single-shot convolution. First, the input image is encoded into multi-scale features through FPN backbone. Then, Panoptic-Feature generator takes the multi-scale features and constructs Panoptic-Feature. At the same time, the filter sampling module also operates on each stage of the multi-scale features to dynamically generate filters for segmenting instances in the scene. Filters for semantic segmentation are defined as trainable parameters and learned through back-propagation. Finally, single-shot convolution with dynamic filters for things and learned filters for stuffs are applied to Panoptic-Feature to finalize the overall prediction at once. Further improvement of performance is possible by adding auxiliary tasks that enhance the representation power of the Panoptic-Feature.

3.1. FPN

We utilize FPN [19] as multi-level features of our model. A considerable number of detection models adopting FPN use five levels of the feature maps $\{P_3, P_4, P_5, P_6, P_7\}$. However, using these levels results in a performance imbalance between instance and semantic segmentation. To deal with this issue, we make a slight adjustment, employing relatively low-level feature maps of $\{P_2, P_3, P_4, P_5\}$. Since a feature map of stride 4 has substantial spatial size, passing it to convolutional layers results in excessive computations. Therefore, we scale down C_2 in half, generating P_2 with a summation between halved C_2 and P_3. Note that for COCO, we include additional level P_6, and the generation of P_6 is the same as P_5, which is of stride 32.

3.2. Filter Sampling Module

We adopt dynamic convolutional filters, inspired by CondInst [31], to design our filter sampling module that is composed of the class head and the filter head. To deal with objects of different scales, the filter sampling module takes multi-level features P_l from FPN as inputs. The weights of the module are shared across all levels. With the class and the filter head, the filter sampling module predicts the class scores and generates dynamic convolutional filters, each corresponding to a designated instance. The class head consists of four sequential convolutions which outputs $C_l \in \mathbb{R}^{H_l \times W_l \times N_t}$, where H_l and W_l are the height and width of the feature P_l, and N_t is the number of thing classes (e.g. 8 for Cityscapes dataset). For the filter head, the structure is the same as the class head, but it additionally takes the absolute positional information encoded at each level [23]. The filter head produces dynamic filter $F_l \in \mathbb{R}^{H_l \times W_l \times D_f}$ as an output, where D_f is the number of output channels. Let k be the size of the kernel used for the single-shot convolution. A filter of $k^2 D_f$ dimensions can be obtained from the output of the filter head F_l; spatially pooled by the size of $k \times k$, putting the center to a given sampling location. During the training, the filters are uniformly sampled from the foreground locations. On the other hand, for the inference, the positions to be sampled are selected by the confidence score above a threshold, measured by C. The sampled filters can be finally used as convolutional weights after passing through a fully connected layer, each resulting in $k^2 D_\phi$ of the number of channels, where D_ϕ is the number of channels of the Panoptic-Feature. The separation between the convolution and the fully connected layer can save memory and computation by forwarding only the filters at interested locations.

3.3. Panoptic-Feature Generator

The structure of Panoptic-Feature generator is similar to the segmentation branch of Panoptic-FPN [14]. As the Panoptic-Feature is generated with the multi-level information from FPN, the features can simultaneously capture the context while fine-grained details are preserved. Note that per-instance relative coordinates can provide strong cues for segmenting instances as proposed in CondInst [31]. However, since all things and stuffs should be handled together in the Panoptic-Feature, per-instance relative coordinates cannot be used. Rather, as shown in Fig. 3, Panoptic-Feature generator utilizes the absolute positional encoding of CoordConv [23] to the merged output of the multi-level features, passing two additional convolutions followed by a deconvolutional layer. As a result, we can have a higher dimensional Panoptic-Feature with 256 channels compared to 8-channeled feature used in CondInst. From our experiments, it is found that adopting high-dimensional features is important to enable the integration of paths.
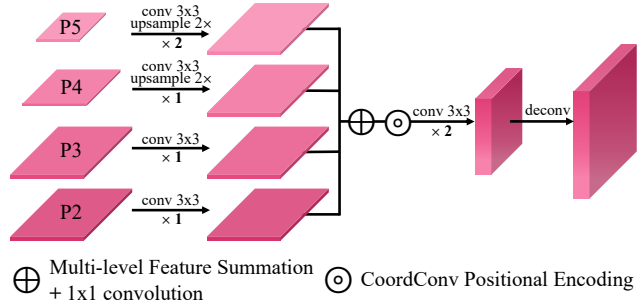

Figure 3. Panoptic-Feature generator. By merging multi-level features from P_2 to P_5, the feature map holds information from low-level to high-level altogether. Given the positional encoding, it passes through extra layers and become the size of stride 4.

To make the output features to have access to information of distant spatial locations, many previous works supply additional modules to their segmentation branch. The use of convolutions with deformation [9], dilation [5], or concatenating features from different levels have proven to be powerful [17, 26, 42]. Compared to general semantic segmentation branches that adapt each level to the size of stride 4, Panoptic-Feature generator internally maintains its spatial size to the stride of 8. As the size of the Panoptic-Feature is more compact, it can reach the information of distant locations by passing through convolutional layers. Therefore, SPINet can be accurate and efficient as it avoids using the aforementioned heavy computations. Moreover, by training with our proposed auxiliary tasks, the representation of the Panoptic-Feature can be further enriched.

3.4. Unified Single-shot Convolution

In our panoptic segmentation framework, the segmentation of both things and stuffs can be done by a single-shot convolution. The convolutional weights for things are sampled from the filter sampling module and the weights for stuffs are learned as trainable parameters. Applying these weights together on the Panoptic-Feature, the raw mask logits responsible for things $\hat{y}_t$ and stuffs $\hat{y}_s$ are generated.

We have separate losses for things and stuffs, denoted as $\mathcal{L}_t$ and $\mathcal{L}_s$. These losses are computed using the ground truth for segmentation (y_t and y_s). For the loss of things $\mathcal{L}_t$, we uniformly sample y_t and $\hat{y}_t$ from foreground locations, and scaled to the stride of 4. The loss is computed as follows:

$$\mathcal{L}_t(y_t, \hat{y}_t) = \text{DICE}(y_t, \sigma(\hat{y}_t)), \qquad (1)$$

where σ is the sigmoid function, and DICE is the dice loss as in [25]. $\mathcal{L}_t$ is normalized by the number of sampled sets.

For the loss of stuffs $\mathcal{L}_s$, we use a summation of two losses: bootstrapped cross entropy loss ($\mathcal{L}_{ce}$) [41] and the new multi-class dice loss ($\mathcal{L}_{md}$). We refer readers to [41] for the details of $\mathcal{L}_{ce}$. We make use of the dice loss again,
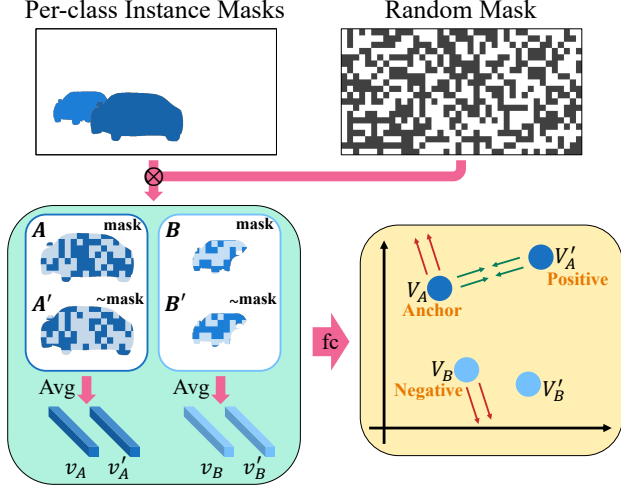
Per-class Instance Masks   **Random Mask**

Figure 4. Intra-Class Triplet loss. Multiple representations of instances can be generated from randomly generated masks; a pair per a random mask. Better arrangement of latent space assisted by the loss enhances the overall segmentation quality.

and $\mathcal{L}_{md}$ is computed as follows:

$$\mathcal{L}_{md}(y_s, \hat{y}_s) = \text{DICE}(y_s, \psi(\hat{y}_s)), \qquad (2)$$

where softmax – taken across stuff classes – is used for ψ, and $\mathcal{L}_{md}$ is normalized by the number of sampled sets, which in this case equals to the number of stuff classes.

The difference between Eq. (1) and Eq. (2) is the way it ranges the logits between 0 and 1 (e.g. sigmoid and softmax). Note that a finer semantic segmentation is shown possible with the simple use of Semantic Encoding Loss [44], which guides to better predict the presence of each class in the input. Similar to Semantic Encoding Loss, multi-class dice loss guides the mask of each class to be better fitted, enabling finer segmentation results.

4. Auxiliary Tasks for Clustering

One way to interpret the Panoptic-Feature is as follows: each point in Panoptic-Feature clusters together at latent space if they belong to a same class and moreover same instance. From this interpretation, Panoptic-Feature can be viewed as a huge composition of groups. In the latent space, there exists a large gap between the clusters of different classes, whereas the gap between instances of a same class is comparatively narrow.

Recently, many approaches improved performance by making use of auxiliary tasks that are designed properly with the consideration to the main task [1, 37]. We also offer new auxiliary tasks considering the interpretation of the Panoptic-Feature given above. The tasks are designed to give better guidance to the Panoptic-Feature. For example, the distance between points in the Panoptic-Feature increases if they are from different instances, and vice-versa.

Given the guidance, the latent space of the Panoptic-Feature arranges properly, allowing the single-shot convolution to make better predictions.

Intra-Class Triplet Loss. Let $\phi \in \mathbb{R}^{H \times W \times D_\phi}$ be the Panoptic-Feature with D_ϕ number of channels over its $H \times W$ spatial size. Let there exists an instance k with the ground-truth mask $m_k \in \{0,1\}^{H \times W}$. Given a random mask $m_r \in \{0,1\}^{H \times W}$, we can obtain a pair of partial masks M_k, M'_k, both corresponding to the instance k as follows:

$$M_k = m_k \odot m_r, \qquad M'_k = m_k \odot (\sim m_r), \qquad (3)$$

where $(\sim)$ operator inverses the value of each pixel, and $\odot$ is the element-wise multiplication.

With a partial mask M, we can generate a representation vector V as follows:

$$v^c = \frac{1}{N_M} \sum_{i,j} M_{i,j} * \phi^c_{i,j}, \quad v = \left[v^1, v^2, ..., v^{D_\phi}\right]$$
$$V = \text{FC}(v), \qquad (4)$$

where $M_{i,j}$ is the mask value of location (i,j), and N_M is the number of locations where $M_{i,j} = 1$. $\phi^c_{i,j}$ is the value of cth channel at location (i,j) of the Panoptic-Feature, and FC is a fully connected layer. By taking the above strategy, a representation pair V_k and V'_k can be obtained from the pair of partial masks M_k and M'_k.

A collection of representation pairs from instances can be gathered by applying the same to each instance. From the collection, triplets can be obtained by assigning representations from a same instance to pair up into $Anchor$ and $Positive$, while a representation from another instance become $Negative$. For tasks like re-identification and metric learning, off-the-shelf triplet margin loss [29] is a great choice to be considered. In order to better cluster points, we use Intra-Class Triplet Loss for narrowing the point features belonging to a same instance, and spreading the clusters of different instances. After some modifications from the triplet margin loss, our loss is defined as follows:

$$d_p = -\text{DIST}(V_a, V_p), \qquad d_n = -\text{DIST}(V_a, V_n),$$
$$\mathcal{L}_{intra}(V_a, V_p, V_n) = -\log\left(\frac{e^{d_p}}{e^{d_p} + e^{d_n}}\right), \qquad (5)$$

where $\mathcal{L}_{intra}$ is normalized by the number of triplet sets. DIST measures the L2 distance of an input pair, and V_a, V_p, V_n are representation vectors of $Anchor, Positive$, and $Negative$ respectively. An illustration of the triplet loss is given in Fig. 4.

Inter-Class Contour Loss. We assign an additional task to the Panoptic-Feature which is of predicting contours between different classes. A 1×1 convolutional layer is applied to the Panoptic-Feature, and the output channel of the

layer is 16. The spatial size of the output is the same as the Panoptic-Feature, which is of stride 4. Hence each location takes charge of 4×4 area with the pixel-shuffle [30] operation on the 16 channels. Inter-Class Contour Loss ($\mathcal{L}_{inter}$) is computed over every pixel as follows:

$$\mathcal{L}_{inter} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \text{FOCAL}(y_{i,j}, \hat{y}_{i,j}), \quad (6)$$

where FOCAL is the focal loss [20]. $y_{i,j}$ and $\hat{y}_{i,j}$ are the value of ground truth and predicted soft contour at location (i, j), respectively. The task of predicting contours from the Panoptic-Feature leads to enlargement of distances between class-wise clusters in latent space, and furthermore it eases the convolutional weights to distinguish classes.

Trained with the introduced Intra-Class Triplet Loss and Inter-Class Contour Loss, our model differentiates points from contrasting classes, and also clusters by instances. The auxiliary tasks stabilize our model's training, and SPINet improves performance while not compromising its efficiency as the tasks are used only for training.

5. Experiments

In this section, we compare our results with previous works on standard panoptic segmentation benchmarks. We analyze our model with the models that are on par with the state-of-the-art performance. We also provide ablation studies conducted on various settings. All experiments are conducted using ResNet-50 as the backbone.

5.1. Datasets

Our experiments are conducted on two popular benchmark datasets: COCO and Cityscapes. **COCO** [21] is a large dataset that has annotations of 118K, 5K, and 20K images for training, validation, and testing, respectively. The number of thing classes is 80, and stuff classes is 53. We use fine-grained annotations of **Cityscapes** [8] which is composed of 2975 images for training and 500 for validation. The dataset has 8 thing classes and 11 stuff classes, and is considered as a great benchmark for panoptic segmentation task due to its highly delicate annotations.

5.2. Implementation Details

Unless specified, our model is set without deformable convolutions, built on top of ImageNet pretrained ResNet-50 backbone, that replaces the first 7×7 convolution with three 3×3 convolutions. The replacement is proven effective by He *et al.* [12]. All convolutional layers of filter sampling module are followed by group normalization [38], and we freeze batch normalization of the layers of the backbone.

Training. The ground truth assignment for Filter Sampling Module is as allocating an instance to points that belong to

the instance mask which is interpolated to the feature size. If a point resides in an overlap of masks, an instance of smaller size is chosen.

With all the losses introduced, we can now finalize the total loss of SPINet as follows:

$$\mathcal{L} = \lambda_0 \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_s + \lambda_2 \mathcal{L}_t + \lambda_3 \mathcal{L}_{inter} + \lambda_4 \mathcal{L}_{intra}, \quad (7)$$

where $\mathcal{L}_{cls}$ uses focal loss [20]. For $\mathcal{L}_{ce}$ of $\mathcal{L}_s$, bootstrapped cross entropy loss of 0.2 top-k ratio is used for Cityscapes, while COCO uses general cross entropy. The coefficients $(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4)$ differs by the dataset. Cityscapes uses the coefficients of (1, 1, 5, 20, 1), while COCO adopts (1, 0.5, 3, 0, 1). Our model can benefit from training with $\mathcal{L}_{inter}$ on Cityscapes as the dataset is composed of fine-grained pixel-level annotations. However, the coarse annotations of COCO diminishes the enhancement.

Using 8 Tesla V100 GPUs, our models are trained for 270k iterations, allocating 2 images per GPU on COCO experiments. The learning rate starts from 0.01 and decreases by a scale factor of 0.1 at 180k and 240k. Allocating 4 images per GPU for Cityscapes experiments, the learning rate starts from 0.01 and decreases at 80k and 90k, terminating the training at 95k.

Inference. During inference, we preserve the full resolution of images from all datasets. We forward the input image through the network and give the confidence threshold of 0.45 for the sampling of dynamic filters for instances. Additional steps for the post-processing follows that of Panoptic-FPN [14], which marks down instances first, and adds up the predictions of stuff classes to the non-instance regions.

5.3. Comparison to the State of the Art

To highlight our model's performance, we compare our proposed SPINet to the state-of-the-art models for both COCO and Cityscapes dataset as shown in Table 1. For fair comparison, we listed performances of each models that are built on top of ResNet-50 and pretrained on ImageNet dataset.

COCO and Cityscapes have contrastive charateristic to each other. COCO is composed of numerous instances, thus the importance lies in the ability to capture instances. Meanwhile, to achieve high scores in Cityscapes, the ability to precisely predict the class of pixels is important. Hence, there has been a tendency of RPN based models being dominant on COCO, and non-RPN based models favoring Cityscapes until now. This tendency is clearly shown in Table 1, where previous models without RPN suffer from low performance on COCO dataset while being competitive on Cityscapes. However, with the strong ability of capturing instances, our model's performance on COCO is substantially higher than all non-RPN based model. Furthermore, the maintenance of pixel level details allows SPINet to sur-

	Method	*deform*	*atrous*	COCO *val* set					Cityscapes *val* set				
				PQ	PQ^{th}	PQ^{st}	AP	mIoU	PQ	PQ^{th}	PQ^{st}	AP	mIoU
w/ RPN	Pan-FPN [14]			39.0	45.9	28.7	33.3	41.0	57.7	51.6	62.2	32.0	75.0
	OCFusion [16]	✓		42.5	49.1	32.5	-	-	59.3	53.5	63.6	-	-
	UPSNet [42]	✓		42.5	48.6	33.4	34.3	<u>54.3</u>	59.3	54.6	62.7	33.3	75.2
	Seamless [26]		✓	-	-	-	-	-	60.3	56.1	63.3	33.6	77.5
	Li *et al.* [17]	✓		<u>43.4</u>	48.6	<u>35.5</u>	<u>36.4</u>	53.7	61.4	54.7	66.3	33.7	79.5
w/o RPN	DeeperLab† [43]		✓	33.8	-	-	-	-	56.6	-	-	-	-
	SSAP‡ [10]			36.5	-	-	-	-	58.4	50.6	-	34.4	-
	Hou *et al.* [13]			37.1	41.0	31.3	-	-	58.8	52.1	63.7	29.8	77.0
	Pan-DL [7]		✓	35.5	37.8	32.0	19.7	-	60.3	50.9	67.1	32.1	78.7
	Axial-DL⋆ [34]			**43.2**	48.1	**35.9**	-	-	63.1	-	-	35.6	80.3
	SPINet			42.2	49.3	31.4	33.2	43.2	63.0	57.0	67.3	35.3	80.0
	SPINet	✓		43.0	**<u>49.3</u>**	33.5	**34.0**	**43.9**	**<u>63.5</u>**	**<u>57.4</u>**	**<u>67.9</u>**	**36.3**	**<u>81.3</u>**

Table 1. We underline the highest numbers among all, and bold the highest numbers over the models without RPN. For fairness, the scores are measured without test time augmentation, and the models are trained on top of ResNet-50, using ImageNet pretrained weights. *deform* and *atrous* indicate the use of Deformable convolution and Atrous convolution, respectively. †: Xception-71 is used for backbone. ‡: ResNet-101 is used for backbone with horizontal flipping and mutliscale test. ⋆: Axial-DeepLab-M is used for backbone.

Method	PQ (%)	Params (M)	Inference Time (ms)		
			Total	Network	Post
UPSNet	59.3	44.5	501	191	310
Pan-DL	60.3	59.8	499	335	164
SPINet	63.0	42.2	201	171	30

Table 2. Inference time on Cityscapes *val* set. **Total:** Network + Post. **Network:** Time for a model to complete feed-forward execution. **Post:** Time for post-processing.

M.C.D	$\mathcal{L}_{inter}$	$\mathcal{L}_{intra}$	PQ	PQ^{th}	PQ^{st}	Params
			61.8	54.8	66.8	42.2
✓			62.3	56.0	66.8	42.2
✓	✓		62.7	56.2	67.4	42.3
	✓	✓	62.3	55.9	66.9	42.3
✓	✓	✓	**63.0**	**57.0**	**67.3**	42.3

Table 3. Use of different losses on Cityscapes *val* set. **M.C.D**: Multi-class dice loss. **Params** (M): Total number of parameters needed for training.

pass all previous models on Cityscapes, becoming the new state-of-the-art.

We also compare our model's inference speed with two well-known models: UPSNet [42] and Panoptic-DeepLab [7] where each represents a model with and without RPN. We measured time to output raw predictions, and time to post-process raw predictions. All times are measured on the same system environment, using TITAN XP. Note that Panoptic-DeepLab gains more accuracy than the proposed results in the paper by modifying the first convolutional layer of the backbone as described in Sec. 5.2 [39]. Our model enjoys the integrated structural pathway, where it omits both feature extraction at RPN and using heavy modules as ASPP. SPINet accelerates its prediction while performing the best score as shown in Table 2.

5.4. Ablation Study

We conduct ablation studies on COCO and Cityscapes validation sets using our model with ResNet-50. We show how our model can be improved by differentiating the model under various settings.

Use of Proposed Losses. As shown in Table 3, the performance of SPINet can be improved by making use of the pro-

posed losses. Note that the modules needed for the auxiliary tasks are not used during inference, thus the use of auxiliary tasks does not burden the model. Using multi-class dice loss and both auxiliary tasks enhances the segmentation quality, and also stabilizes the training. By taking a look at Fig. 5, we can understand how the auxiliary tasks affect the predictions. Compared to the model trained without auxiliary tasks, the model trained with auxiliary tasks tries to avoid the overflow of segmentation masks as Intra-Class Triplet loss restricts the intervention between instances as shown in Fig. 5 (b) and (d). Furthermore, SPINet trained with Inter-Class Contour loss detects the borders between semantic classes. It increases not only PQ^{st} but also PQ^{th} as it indirectly helps the model capturing small sized instances. Therefore, with the potential to estimate contours, our model can generate more accurate segmentation maps.

Different FPN Level. Since our model uses the single Panoptic-Feature, balancing the performance between thing and stuff becomes important. SPINet uses the multi-level features from FPN in two modules: Panoptic-Feature generator and filter sampling module. From experiments, we find that using the lower levels from FPN results in the best performance as shown in Table 4. Therefore, we lower the
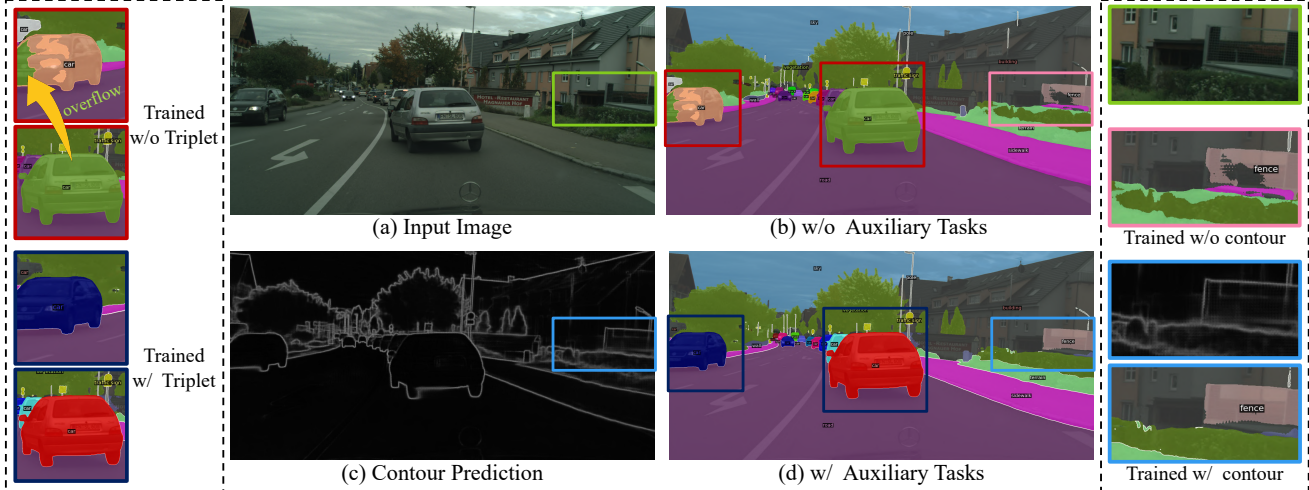
Figure 5. Visualized results of SPINet with ResNet-50 on Cityscapes *val* set. With the given image (a), we show two panoptic segmentation results, which are retrieved from the models trained without auxiliary tasks (b), and with auxiliary tasks (d). As the model trained with auxiliary tasks (d) uses two additional losses, Intra-Class Triplet loss and Inter-Class Contour loss, we visualize how the losses affect the prediction. We also visualized the contour map (c), predicted by (d), to ease the understanding of how the contours can increase the segmentation quality.

P.F.G	F.S.M	PQ (%)	PQth (%)	PQst (%)
P 3-7	P 3-7	40.5	48.3	28.6
P 2-6	P 3-7	40.5	48.6	28.3
P 2-6	P 2-6	**42.2**	**49.3**	**31.4**

Table 4. Various results measured on COCO validation set can be obtained by using features from different levels of FPN. **P.F.G**: Panoptic-Feature generator. **F.S.M**: Filter sampling module.

Integrated	PQ (%)	Speed (ms)	Params (M)	FLOPs (10^{12})
	62.3	202	46.6	0.96
✓	63.0	171	42.2	0.81

Table 5. The integration lightens both the speed and the computation, while also increasing the performance as implicit interchanges of information between thing and stuff become possible.

resolution of P2 to balance the performance and computation loads as described in Sec. 3.1, and use the downscaled feature for both Panoptic-Feature generator and filter sampling module.

Path Integration on CondInst. Though SPINet is inspired by CondInst thus share some similarities (e.g., both generate dynamic filters for each instance), it does not imply that CondInst is also suitable for path integration. The key difference is on the feature where instance-wise dynamic filters are applied to: F_{mask} in CondInst and Panoptic-Feature of SPINet. Specifically, CondInst customizes F_{mask} for each instance by using per-instance relative coordinates thus they need as many F_{mask}'s as the number of instances. As a result, they need to limit the channel size of F_{mask} to fit into GPU memory (8-channeled feature map is used as F_{mask} in CondInst). When we tried to directly integrate the instance and the semantic paths onto CondInst, the model could not be stably trained. Our conjecture is that the low dimensional feature map acts as an information bottleneck. We have tried gradually increasing the channel size as much as our GPU can hold, but the model still cannot be trained properly. On the contrary, our SPINet, by design, can have a larger 256-dimensional Panoptic-Feature and it allows us to integrate two paths stably.

Advantages of Path Integration. The integration of the separated branches makes the sharing of computations between things and stuffs become possible. Though it is clear that our model can benefit huge efficiency from the integration, it is uncertain to expect the gain in terms of performance as the number of parameters decreases. Therefore, we generated two feature maps from two individual Panoptic-Feature generators, each taking in charge of things and stuffs respectively. As shown in Table 5, the model with separate feature maps gets a lot heavier than the model with the integrated pathway. With fewer computations, the integrated model surprisingly overtakes the model with the separated branches by 0.7% increment of PQ.

6. Conclusion

We have introduced SPINet, a model with the novel integration of pathways between the instance and semantic segmentation, which makes efficient yet powerful panoptic segmentation become possible. With the simplification of the execution flow, SPINet can take the advantages of top-down and bottom-up models, improving even further with proposed auxiliary tasks. Finally, SPINet shows comparable results on COCO, and delivers the state-of-the-art performance on Cityscapes with a large margin.

References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019. 5

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2

[4] Liang-Cheih Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *IEEE TPAMI*, 2017. 1, 2

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 4

[6] Yifeng Chen, Guangchen Lin, Songyuan Li, Omar Bourahla, Yiming Wu, Fangfang Wang, Junyi Feng, Mingliang Xu, and Xi Li. Banet: Bidirectional aggregation network with occlusion handling for panoptic segmentation. In *CVPR*, 2020. 2

[7] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 1, 2, 7

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 2, 4

[10] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *ICCV*, 2019. 7

[11] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2, 3

[12] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019. 6

[13] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *CVPR*, 2020. 7

[14] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *CVPR*, 2019. 1, 2, 4, 6, 7

[15] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *CVPR*, 2019. 1, 2

[16] Justin Lazarow, Kwonjoon Lee, Kunyu Shi, and Zhuowen Tu. Learning instance occlusion for panoptic segmentation. In *CVPR*, 2020. 2, 7

[17] Qizhu Li, Xiaojuan Qi, and Philip H.S. Torr. Unifying training and inference for panoptic segmentation. In *CVPR*, 2020. 2, 4, 7

[18] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *CVPR*, 2019. 2

[19] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *ICCV*, 2017. 3, 6

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 6

[22] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *CVPR*, 2019. 2

[23] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NeurIPS*, 2018. 4

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 3

[25] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 4

[26] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *CVPR*, 2019. 1, 4, 7

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 3

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 3

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 5

[30] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 6

[31] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 2, 3, 4

[32] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 3

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[34] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. 7

[35] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *ECCV*, 2020. 3

[36] Yuqing Wang, Zhaoliang Xu, Hao Shen, Baoshan Cheng, and Lirong Yang. Centermask: Single shot instance segmentation with point representation. In *CVPR*, 2020. 3

[37] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. In *CVPR*, 2020. 5

[38] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 6

[39] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 7

[40] Yangxin Wu, Gengwei Zhang, Yiming Gao, Xiajun Deng, Ke Gong, Xiaodan Liang, and Liang Lin. Bidirectional graph reasoning network for panoptic segmentation. In *CVPR*, 2020. 2

[41] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016. 4

[42] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019. 2, 4, 7

[43] Tien-Ju Yang, Maxwell D. Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *CoRR*, abs/1902.05093, 2019. 1, 2, 7

[44] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018. 5