YOLO-based Panoptic Segmentation Network

Manuel Diaz-Zapata, Özgür Erkent, Christian Laugier
Chroma Team, INRIA Grenoble Rhône-Alpes
Grenoble, France
Email: name.surname@inria.fr

***Abstract*—Autonomous vehicles need information about their surroundings to safely navigate them. For this, the task of Panoptic Segmentation is proposed as a method of fully parsing the scene by assigning each pixel a label and instance id. Given the constraints of autonomous driving, this process needs to be done in a fast manner. In this paper, we propose the first panoptic segmentation network based on the YOLOv3 real-time object detection network by adding a semantic and instance segmentation branches. YOLO-panoptic is able to do real-time inference and achieves a performance similar to the state of the art methods in some metrics.**

I. Introduction

Autonomous vehicles need to understand their surroundings in order to safely navigate from one point to another. For this, a perception pipeline that gives the vehicle enough information to work with is essential. Here, many of the tasks proposed in the Computer Vision field could be used to gather this information from one or many images taken from sensors located in the vehicle.

The drawback of some of these approaches is the limit to which they are able to parse the image. An object detection and classification algorithm would be able to describe how many foreground objects are around the vehicle in a fast manner, but is limited to a set of boxes describing the position and size of them as an output. Semantic segmentation on the other hand, aims to give each pixel in the image a class label taking into account both foreground and background classes, but no distinction is done between the pixels that belong to the same class and different objects. Having this problem in mind, instance segmentation aims to improve the description of the objects in the scene by predicting which pixels belong to which object, improving the bounding box description but not focusing on the background of the scene.

Panoptic Segmentation [1] is then proposed as a task that aims to perform parsing of the images at the pixel level by bringing together semantic and instance segmentation. In Panoptic segmentation, each pixel in an image must be given a class label and an id where the classes are divided in *things* and *stuff* subsets. Here, *things* represents the classes that usually have multiple instances in a scene, so an id is needed to differentiate pixels with same label that belong to different objects. And *stuff* classes are the classes that are usually in the background and pixels assigned with the same label don't have the need to be separated. For example, in the autonomous driving context classes such as pedestrian, car, bus or bicycle belong to the *things* subset and classes such as road, vegetation, sidewalk or building belong to the *stuff* subset.

Fig. 1. Output from YOLO-Panoptic on an image from the Cityscapes Dataset.

Knowing this, and the importance of low inference times for systems in the autonomous driving context, we propose as our contribution: the first Panoptic Segmentation network based on the You Only Look Once version 3 (YOLOv3) [2] real-time detector. As originally proposed, YOLOv3 is a single-shot fast object detection network that predicts bounding boxes using convolutional anchor boxes but is incapable of learning any type of mask representation. We augment a pretrained YOLOv3 network by adding two parallel heads to perform semantic and instance segmentation using features taken from the network's Darknet-53 backbone. This network is capable of performing inference between 6 and 17 fps, while achieving a performance close to some of the state of the art methods

II. Related Work

The two main differences between current Panoptic segmentation methods is how instance segmentation is done, and how the merging of semantic and instance segmentation masks is performed. For instance segmentation there are two main variants: detect-then-segment, also known as top-down; and grouping pixels to create the masks, also known as bottom-up. Out method follows the top-down strategy since we use the YOLOv3 [2] network as base. Related to the merging, the use of hand-crafted heuristics is the predominant approach but there are methods that resolve the overlap conflicts using learned parameters.

A. Top-down methods

This strategy usually builds on top of existing instance segmentation networks by adding a semantic segmentation

branch and resolving the overlaps between them to get the final prediction. Panoptic FPN [3] builds on top of Mask R-CNN[4] by adding a semantic segmentation head that predicts the *stuff* classes from the features given by the backbone. Instance and semantic segmentation masks are then merged by resolving overlaps using heuristics that prioritize masks classified as *things*. The attention-guided unified network (AUNet) [5] also builds on top of Mask R-CNN, but focuses on how the information known about the foreground objects can be used to create the background masks. For this, the proposal attention and mask attention modules are created. The role of both modules is to establish a relationship between foreground and background using the features from the Region Proposal Network[6] and mask segmentation branches of Mask R-CNN. In [7] as in Panoptic FPN, Mask R-CNN is used as the base network. A parallel branch for semantic segmentation is added and it uses features from different levels of the backbone. Once the stuff and instance masks are predicted, they are given to the Spatial Ranking Module that generates a ranking score map for merging by indicating which pixels belong to which instance or semantic class. The main drawback of these methods is their use of Mask R-CNN, which does not allow for high-speed inference.

B. Bottom-up methods

Bottom-up methods build the instances by clustering pixels together to then do classification of the different generated masks. Deeperlab [8] predicts agnostic instance masks based on different relationships between predicted keypoints, as well as masks for semantic segmentation. The Panoptic Segmentation map is created by assigning class labels and instance labels to each pixel based on majority voting, therefore there is no need to perform any other process for overlap resolution. Panoptic Deeplab[9] represents object instances by predicting the offset to the corresponding mass center for each foreground pixel. The instance id for a pixel classified as a *things* class in the semantic segmentation branch is assigned with respect to its closest predicted keypoint. The final panoptic prediction is created by adding the instance id information from the instance segmentation branch to the semantic segmentation prediction, so no overlap resolution is needed. AdaptIS [10] generates instance masks by finding a 'characteristic' vector that allows a set of AdaIN [11] modules to segment the instances based on point proposals. For panoptic segmentation, a parallel branch is added to perform semantic segmentation. Overlap between the masks is solved by a greedy algorithm that assigns points to the different predicted masks, where a mask is determined as segmented if it has an overlap lower than 0.5 with respect to other masks.

III. YOLO-based Panoptic Segmentation

We build our Panoptic segmentation network by adding two segmentation heads to the Darket-53 [2] backbone of the YOLOv3 network, one for semantic segmentation and another one for instance segmentation. The YOLOv3 network was chosen as due to its fast single-shot approach, where detections

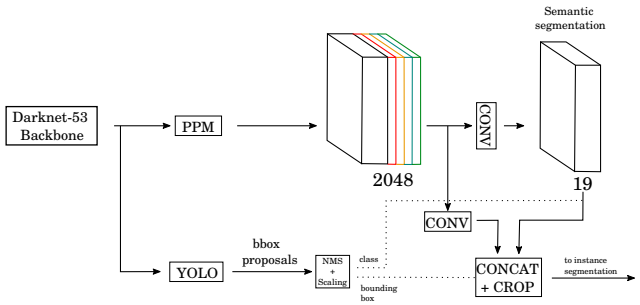

Fig. 2. Overview of the network architecture for YOLO-Panoptic

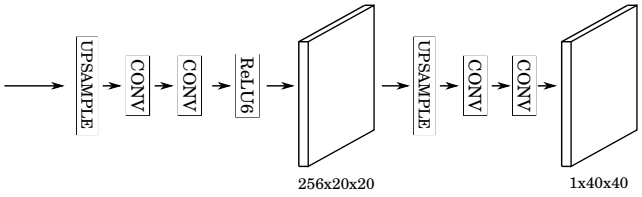

Fig. 3. Configuration of the instance segmentation head

do not go through a time-expensive refining process. Since we use the detections from the YOLOv3 network to find the instances, our network falls in the category of top-down methods. The predictions for each one of these are then merged using a set of rules in order to solve the overlaps between instances and semantic segmentation maps.

A. Input, output and size the network

The network takes as an input a square RGB image, which is an unsigned integer of 8 bits (uint8) tensor from the camera. This tensor is then converted to a 32-bit floating point (float32) tensor by dividing each value in the image by 255, which is the highest intensity value uint8 can represent. This normalization is done in order to assure the compatibility with the weights of the network, which are float32 tensors.

As outputs, the network gives a float32 tensor and an integer array. The float32 tensor is the set of output masks of equal spatial size than the input but with $C + n_I$ channels, where C are the number of semantic segmentation masks and n_I are the number of detected instances. An argmax operation is applied in the channel dimension of this tensor in order to assign a label to each pixel, changing the output mask tensor from float32 into an integer of 64 bits (int64) data type, which is the presented output here. For the case of the integer array, it describes the labels given to the instances since the amount of them can change from image to image and only the mask information can be inferred from the output, unlike the semantic segmentation masks.

The model has 76.6 million parameters, and occupies 1335 MB of VRAM when loaded to the GPU. When doing inference on both image sizes described in section IV, the model uses a maximum of 4191MB of VRAM.

B. Enriched features

To have enriched features to use for segmentation, we extract a set of features from three different parts of the backbone in order to use multi-scale information in the same fashion as [12]. We use the first three feature maps that are before the spatial reduction steps in the backbone with depths of 128, 256 and 512 channels respectively.

For the shallowest feature map, a 1×1 convolution is applied to increase its depth to 256 channels. The other two are bilinearly upsampled to match the spatial size of the first feature map. Then they are concatenated together to create a feature map of depth 1024.

This new feature map is passed through a Pyramid Pooling Module (PPM) [13] to extract information in different scales and varying among different subregions for our network to be able to perform segmentation in large, medium and small scales. The depth of the resulting features is decreased by four using a 1×1 convolution. The output of the PPM is upsampled and concatenated to its input, resulting in a set of enriched features that will be used for both semantic and instance segmentation. The network architecture can be seen in Figure 2.

C. Semantic Segmentation Branch

The semantic segmentation maps are obtained from the enriched features by applying first a 3×3 and and then a 1×1 convolution to predict a set of C feature maps, where C corresponds to the number of classes.

D. Instance Segmentation Branch

For the instance segmentation, the bounding boxes from the YOLO head are used to crop the enriched feature map. Non-maximum suppression is performed on the boxes from the detection head and a 1×1 convolution is applied to the enriched features to reduce the depth of the feature map.

The resulting boxes are used as attention masks on the enriched features for the foreground objects. The process is represented by the dotted line labeled 'bounding box' in Figure 2. Also, to add contextual information from the semantic segmentation branch the corresponding class feature map is added to the features before cropping them with the attention masks. This is the input dotted line named 'class' in Figure 2.

The cropped features are then fed through the instance segmentation branch. Figure 3 shows a diagram of the instance segmentation head, which is similar to the one proposed in [4]. Here, the input features are scaled to a fixed 20×20 spatial size, passed to two 3×3 convolutions and a ReLU6 [14] non-linearity, scaled to a size of 40×40 and finally passed to two more 3×3 convolutions to generate the final instance mask for each bounding box detection.

Once the masks are created, they are interpolated to their original size and location in the image using the bounding box coordinates.

E. Merging

To resolve overlaps, we use a set of rules similar to the ones proposed in [3]. First, we resolve overlaps between instances, where the pixels are assigned to the instance with the highest score. Second, overlaps between instances and semantic segmentation are resolved favoring the instance masks. Lastly, *stuff* regions under a certain area threshold are removed based on the assumption that these classes are usually not present in small areas.

IV. Experiments and Results

For training we use the original YOLOv3 architecture and its weights trained on the MS COCO Dataset [15], but keep the Darknet-53 and YOLO layers frozen. The parameters from the Pyramid Pooling Module, semantic segmentation branch and instance segmentation are the only ones being updated. Training is performed on the Cityscapes Dataset [16] using a Tesla V100 GPU capable of 14 TFLOPs on float32 data.

Training of both segmentation branch is performed simultaneously using the weighted loss function seen in Eq. 1, where $\mathcal{L}_{sem}$ is set as cross-entropy loss and $\mathcal{L}_{inst}$ is average of the binary cross-entropy loss between each instance and its ground truth. Here, the λ parameter is set to 0.1 and to used for two reasons. First, to force the network to learn first the semantic representations of the images; and second as a weight that allows to control the influence of misclassified small instances, which may not have as much importance in the scene, and whose loss affect the average. SGD is used as optimizer with a base learning rate of 0.1, momentum of 0.9 and polynomial learning rate scheduler with $\gamma = 0.9$.

$$\mathcal{L} = \mathcal{L}_{sem} + \lambda * \mathcal{L}_{inst} \tag{1}$$

We train the network for 100 epochs, with early stopping after 10 epochs of either the validation loss not decreasing, mean intersection over union of the semantic segmentation not increasing or mean intersection over union of the instance masks not increasing. For data augmentation we apply random scaling, random rotation, random horizontal flip, random Gaussian blur of 5×5 and random crop to the input images. We train the network on two image side sizes to validate speed tradeoffs: 640×640 and 1024×1024.

A comparison to current state of the art methods can be seen in tables I and II where we compare the panoptic quality metrics and inference time. Although the Panoptic Quality scores are not as high as other current methods, we do outperform [17] and achieve a low inference time of 161 ms on 1024×1024 images.

A. Image size and inference time

As stated on IV, we perform training on two different image sizes (640×640 and 1024×1024) to test if network performance stays the same by using a smaller image. And as can be seen in table III, using a smaller images increases inference speed by $2\times$ at the expense of 3 Panoptic Quality points. PQ, SQ and RQ correspond to the Panoptic Quality,

Architecture	PQ(%)	Time(ms)
JSISNet[17]	17.6	n/a
AUNet[5]	59.0	n/a
Panoptic FPN[3]	58.1	n/a
Single Network PS[18]	42.9	590
DeeperLab[8]	56.53	308
Panoptic Deeplab[9]	63.0	175
AdapIS[10]	62.0	n/a
FPSNet[19]	55.1	114
Real-Time PS[20]	58.8	99
YOLO-Panoptic	33.8	161

TABLE I
PANOPTIC QUALITY RESULTS ON THE CITYSCAPES VALIDATION SET.

Architecture	PQ^{Th}(%)	PQ^{St}(%)
JSISNet	10.0	23.5
AUNet	54.8	62.1
Panoptic FPN	52.0	62.5
Single Network PS	74.3	56.5
AdapIS	64.4	58.7
FPSNet	48.3	60.1
Real-Time PS	52.1	63.7
YOLO-Panoptic	29.7	45.4

TABLE II
PANOPTIC QUALITY OF THINGS (PQ^{Th}) AND STUFF (PQ^{St}) CLASSES RESULTS ON THE CITYSCAPES VALIDATION SET.

Segmentation Quality and Recognition Quality metrics as proposed in [1] respectively.

Image Size	640×640	1024×1024
PQ	0.3010	0.3385
SQ	0.5553	0.5426
RQ	0.5426	0.5913
Time (ms)	69	161

TABLE III
PERFORMANCE COMPARISON BETWEEN INFERENCE IN IMAGES OF SIZE 640 AND 1024 ON CITYSCAPES VALIDATION SET.

B. Qualitative analysis

Panoptic segmentation results on images from the Cityscapes Validation set can be seen in Figure 4. Here we can observe that by just updating the segmentation branches, the network is able to both predict the masks for the different instances, even small ones far away from the camera, and the masks for the background classes of the scene.

Also, we show how joint training improved the semantic segmentation which can be observed in Figure 5. We find that by sharing the same features before doing instance and semantic segmentation, information about the masks of the foreground objects in the semantic segmentation is improved. For example, on Figure 5, in the top row of images some pixels that were classified as car with semantic segmentation, are correctly classified as building; in middle row of images, we can see that the trash can is no longer classified as part of the car when adding instances to training; and in the bottom row, the space between the two cars is properly classified as part of the building class.

V. CONCLUSION AND FUTURE WORK

We present a YOLO-based panoptic segmentation network. This network is a modified version of the YOLOv3 detection model with parallel branches that predict semantic and instance masks that are merged to create a panoptic segmentation image. This network is capable of performing panoptic segmentation in a fast manner, which is a crucial requirement of autonomous vehicles.

As future work, we propose to fully train the network and evaluate if the segmentation can be used to generate a panoptic occupancy grid for autonomous systems.

ACKNOWLEDGMENTS

Many thanks go to Jilles Dibangoye and David Sierra-Gonzales for participating in discussions about some of the sections of this paper.

This work was supported by the European Union project Cyber-Physical Systems for the EU (CPS4EU).

Experiments presented in this document were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 9404–9413.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6399–6408.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[5] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, "Attention-guided unified network for panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7026–7035.

[6] 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*, 2015, pp. 91–99.

[7] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, "An end-to-end network for panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6172–6181.

[8] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen, "Deeperlab: Single-shot image parser," *arXiv preprint arXiv:1902.05093*, 2019.

[9] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 475–12 485.

[10] K. Sofiiuk, O. Barinova, and A. Konushin, "Adaptis: Adaptive instance selection network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7355–7363.

[11] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[12] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

Fig. 4. Output of the YOLO-Panoptic Network for different images taken from the Cityscapes Dataset, best viewed with digital zoom.

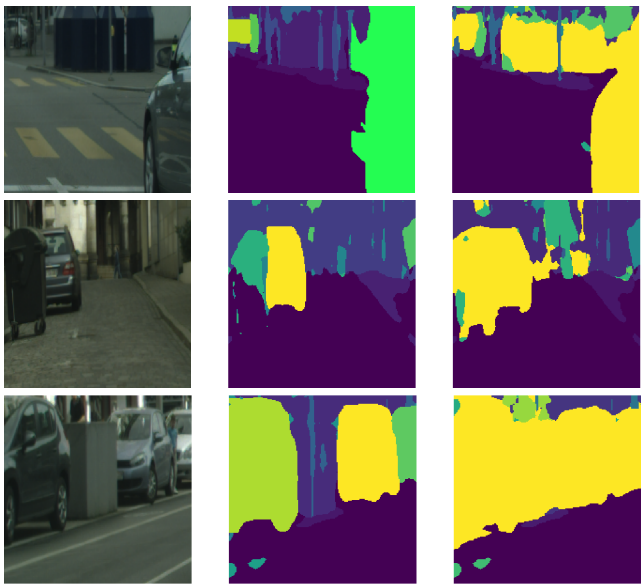

Fig. 5. Comparison of segmentation results between the panoptic (center) and semantic (right) segmentation results.

[13] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[14] T. Contributors, "Relu6," 2019. [Online]. Available: https://pytorch.org/docs/master/generated/torch.nn.ReLU6.html

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] D. de Geus, P. Meletis, and G. Dubbelman, "Panoptic segmentation with a joint semantic and instance segmentation network," *arXiv preprint arXiv:1809.02110*, 2018.

[18] D. de Geus, P. Meletis, and G. Dubbelman, "Single network panoptic segmentation for street scene understanding," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[19] D. de Geus, P. Meletis, and G. Dubbelman, "Fast panoptic segmentation network," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1742–1749, 2020.

[20] R. Hou, J. Li, A. Bhargava, A. Raventos, V. Guizilini, C. Fang, J. Lynch, and A. Gaidon, "Real-time panoptic segmentation from dense detections," *arXiv preprint arXiv:1912.01202*, 2019.