HAL
open science

YOLO-Based Panoptic Segmentation

Manuel Alejandro Diaz-Zapata

To cite this version:

Manuel Alejandro Diaz-Zapata. YOLO-Based Panoptic Segmentation. Computer Vision and Pattern Recognition [cs.CV]. 2020. hal-02884735v1

HAL Id: hal-02884735

https://inria.hal.science/hal-02884735v1

Submitted on 30 Jun 2020 (v1), last revised 9 Feb 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Master of Science in Informatics at Grenoble
Master Informatique
Specialization Graphics, Vision and Robotics

YOLO-Based Panoptic Segmentation

Manuel Alejandro Diaz Zapata

June, 2020

Research project performed at CHROMA

Under the supervision of:
Ozgur Erkent, PhD
Christian Laguier, HDR

Defended before a jury composed of:
Dominique Vaufreydaz, HDR
Massih-Rezza Amini, HDR
Victor Romero-Cano, PhD

June 2020

Abstract

Given the recent challenge of Panoptic Segmentation, where every pixel in an image must be given a label, as in semantic segmentation, and an instance id, a new YOLO-based architecture is proposed here for this computer vision task. This network uses the YOLOv3 architecture, plus parallel semantic and instance segmentation heads to perform full scene parsing. A set of solutions for each of these two segmentation tasks are proposed and evaluated, where a Pyramid Pooling Module is found to be the best semantic feature extractor given a set of feature maps from the Darknet-53 backbone network. The network gives good segmentation results for both stuff and thing classes by training with a frozen backbone, where boundaries between background classes are consistent with the ground truth and the instance masks match closely the true shapes of the objects present in a scene.

Acknowledgement

I would like to express my sincere gratitude to Dr. Erkent for all of his help and guidance during this project, and to Dr. Laugier for the opportunity to be part of the CHROMA's research team. Lastly, to my family, for all of their support during these times abroad.

The work presented in this report was partially supported with funds from Toyota Motor Europe. 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).

Résumé

Compte tenu du défi récent de la segmentation panoptique, où chaque pixel d'une image doit recevoir une étiquette, comme dans la segmentation sémantique, et un identifiant d'instance, une nouvelle architecture basée sur YOLO est proposée ici pour cette tâche de vision par ordinateur. Ce réseau utilise l'architecture YOLOv3, ainsi que des têtes de segmentation sémantique et d'instance parallèles pour effectuer une analyse complète de la scène. Un ensemble de solutions pour chacune de ces deux tâches de segmentation est proposé et évalué, où un Pyramid Pooling Module se révèle être le meilleur extracteur de caractéristiques sémantiques compte tenu d'un ensemble de caractéristiques du réseau de base Darknet-53. Le réseau donne de bons résultats de segmentation pour les classes de choses et d'objets en s'entraînant avec une backbone figée, où les frontières entre les classes d'arrière-plan sont cohérentes avec la ground-truth et les masques d'instance correspondent étroitement aux vraies formes des objets présents dans une scène.

Contents

Abstract **i**

Acknowledgement **i**

Résumé **i**

1 Introduction **1**

2 State-of-the-Art **3**
2.1 Panoptic Segmentation 3
2.2 Attention-guided Unified Network (AUNet) [23] 4
2.2.1 Backbone 4
2.2.2 Proposal Attention Module 5
2.2.3 Mask Attention Module 5
2.3 Panoptic Feature Pyramid Network [21] 5
2.4 Unified Panoptic Segmentation Network [42] 6
2.4.1 Semantic Segmentation Head 6
2.4.2 Panoptic Segmentation Head 6
2.5 Single Network Panoptic Segmentation for Street Understanding [10] 7
2.5.1 Inter-branch information exchange 7
2.5.2 Advanced Merging Heuristics 7
2.6 End-to-End Network for Panoptic Segmentation [26] 8
2.6.1 Spatial Ranking Module 8
2.7 Deeperlab: Single-shot image parser [43] 9
2.7.1 Encoder 9
2.7.2 Decoder 9
2.7.3 Semantic Segmentation Head 9
2.7.4 Instance segmentation heads 10
2.7.5 Prediction Fusion 10
2.7.6 Parsing Covering metric 11
2.8 Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation [3] 11
2.8.1 Backbone 11
2.8.2 Semantic Segmentation head 11

2.8.3 Instance segmentation head . . . 12
2.8.4 Panoptic Segmentation head . . . 12
Instance Segmentation as output . . . 12
2.9 AdaptIS: Adaptive Instance Selection [40] . . . 12
2.9.1 Backbone . . . 12
2.9.2 Instance Selection Network . . . 12
2.9.3 Semantic Segmentation and Point Proposal branches . . . 13
2.9.4 Merging . . . 14
2.10 Fast Panoptic Segmentation Network [11] . . . 14
2.10.1 Backbone . . . 14
2.10.2 Attention Mask . . . 14
2.10.3 Panoptic Head . . . 15
2.11 Real-Time Panoptic Segmentation from Dense Detections [16] . . . 15
2.11.1 Backbone . . . 15
2.11.2 Parameter-Free Mask Construction . . . 15
2.11.3 Panoptic Segmentation . . . 16
Target assignment . . . 16
Unified Panoptic Head . . . 16

3 Panoptic Segmentation using Darknet-53 and YOLOv3 17
3.1 You Only Look Once Architecture . . . 17
3.1.1 You Only Look Once (YOLO) . . . 17
3.1.2 YOLO9000 . . . 18
3.1.3 YOLOv3 . . . 18
3.2 Semantic Segmentation Network . . . 18
3.2.1 Experimentation with Single Feature Maps . . . 18
3.2.2 Experimentation with Multiple Feature Maps . . . 21
3.3 Instance Segmentation Network . . . 24
3.4 Panoptic Segmentation . . . 26
3.4.1 Merging semantic and instance segmentation maps . . . 26
3.4.2 Refining of Segmentation using Instances . . . 28
3.4.3 Comparison with state-of-the-art results . . . 28

4 Conclusion and Future Work 31

A Semantic Segmentation results per class 33

B Panoptic Quality results on the Cityscapes dataset 39

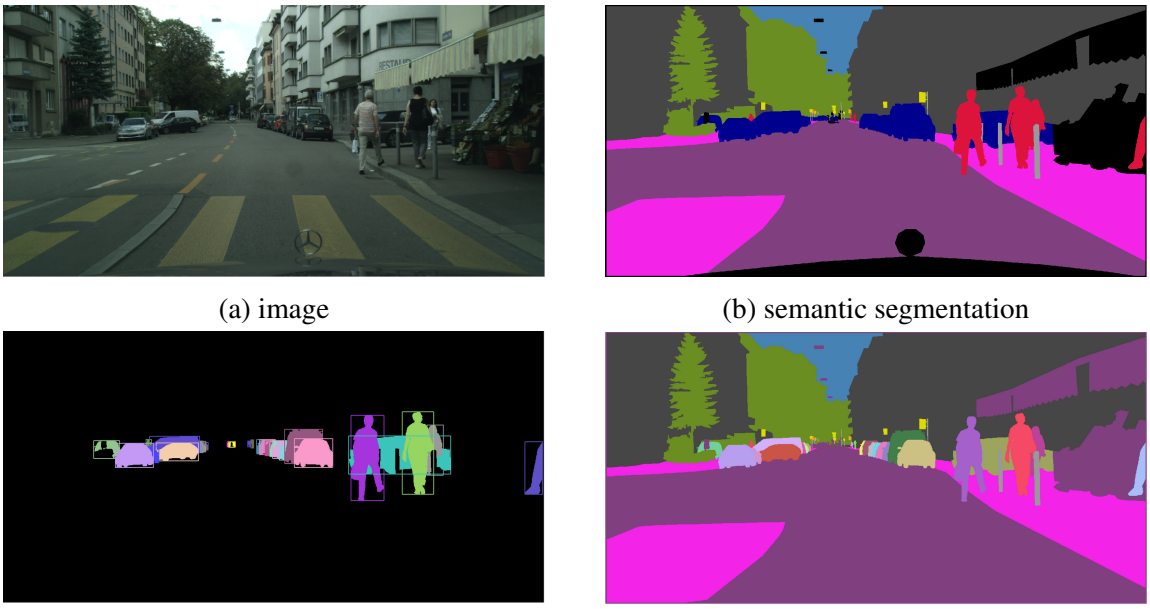
C Panoptic Segmentation Results 41
C.1 Results on images from the Cityscapes Dataset . . . 41
C.2 Results on images from Grenoble . . . 43

Bibliography 45

— 1 —
Introduction

Due to the recent advancements in Artificial Intelligence, and more specifically Deep Learning, the field of Computer Vision has experienced increased attention for the improvements in performance of artificial neural networks over systems based in hand-crafted features. Here, scene parsing has become an important task that can be divided into three broad categories that are being used in different fields such as robotics and autonomous driving: image classification, object detection and semantic segmentation.

Given an image, a neural network can be trained to output a label corresponding to the class of the main object in the image, this task is called image classification. A more challenging goal than image classification is to say where in an image an object can be found, where the bounding boxes that enclose the objects on an image are predicted, this is object detection. Most of the time image classification is integrated to object detection in order to describe each found object in an image. But although these two categories can help with image parsing, their results can only give information about the general position and class of the objects in a scene



(a) image

(b) semantic segmentation

(c) instance segmentation

(d) panoptic segmentation

Figure 1.1 – Semantic, instance and panoptic segmentation in the Cityscapes Dataset [7]

since their descriptions lack information such as the true shape of the objects in the image or even description of the background.

With semantic segmentation, the goal is to assign a class to each pixel of an image to get a dense description of the scene. By assigning labels at the pixel level, amorphous regions of similar texture or material that belong to the same class can be found in an image. Although this type of dense scene description can give as a result the true shape of some regions or objects in the image, the main downside of this method is the loss of separation between countable objects because all pixels of the same class are classified together. A clear example of this can be seen for the cars on figures 1.1a and 1.1b.

In order to bridge the gap between semantic segmentation and object detection and classification, instance segmentation was proposed as a combination of both. With instance segmentation, detection and pixel-wise description of each object in the scene is done in order to achieve a better parsing quality for the objects in the image. Still, important information about the scene such as background description is not given, as shown on figure 1.1c, due to the focus on foreground objects for object detection.

In 2018, Kirillov et al. [22] noted that there has been a gap on the methods used to detect *stuff* or uncountable objects (e.g. sidewalk, road, building), and *things* or countable objects (e.g. car, person, bicycle, truck). Where semantic segmentation has been mainly focused towards stuff and instance segmentation towards things. Having this in mind, the task of Panoptic Segmentation is proposed, where the information from semantic and instance segmentation can be merged in order to achieve full scene parsing as can be seen on figure 1.1d.

As a solution to the Panoptic segmentation task, a panoptic segmentation network based on the YOLOv3 detector and Darknet-53 feature extraction neural networks is created. First, the capability of semantic segmentation are be tested by using features from the Darknet-53 network with multiple candidate configurations. Then, these semantic features are used with the detections from the YOLOv3 layers in order to perform instance segmentation. A panoptic image is then created by merging the information from each of these networks using a set of rules. Finally, the resulting architecture is compared to some of the current solutions using the Panoptic Quality score [22].

— 2 —
State-of-the-Art

2.1 Panoptic Segmentation

Proposed by Kirillov et al. [22], given a predetermined set of L semantic classes encoded by $\mathcal{L} := \{0, \ldots, L-1\}$, a panoptic segmentation algorithm is required to map each pixel i of an image to a pair $(l_i, z_i) \in \mathcal{L} \times \mathbb{N}$, where l_i represents the semantic class of pixel i and z_i represents its instance id. Here, z_i helps to differentiate pixels with the same label into distinct segments. There is also the possibility for pixels to have a void label or no label at all, relieving the constraint for each pixel to have a semantic label.

The semantic class set $\mathcal{L}$ is composed by the subsets of stuff $(\mathcal{L}^{\mathrm{St}})$ and things $(\mathcal{L}^{\mathrm{Th}})$ classes, such that $\mathcal{L} = \mathcal{L}^{\mathrm{St}} \cup \mathcal{L}^{\mathrm{Th}}$ and $\mathcal{L}^{\mathrm{St}} \cap \mathcal{L}^{\mathrm{Th}} = \emptyset$. If a pixel is labeled with $l_i \in \mathcal{L}^{\mathrm{St}}$, its corresponding id z_i will be irrelevant. Otherwise, all pixels with the same (l_i, z_i) assignment where $l_i \in \mathcal{L}^{\mathrm{Th}}$, belong to the same instance.

The panoptic segmentation task format serves as a strict generalization of the semantic segmentation task format. Where if the input image does not have any instances or all of its classes are stuff, both tasks would be the same. Regarding instance segmentation, since the goal is to give each pixel a semantic label and an instance id, the panoptic segmentation task does not admit instance overlapping as instance segmentation does. In order to address segment matching, this task format also specifies that a predicted segment and a ground truth mask are a match if their intersection over union (IoU) metric, also known as the Jaccard Index [19], is strictly greater than 0.5. By joining the two last requirements: non-overlapping instances and segment matching, a unique matching is given where there can only be at most one predicted segment with each ground truth segment.

Since Panoptic segmentation combines segmentation and recognition tasks, a new metric called Panoptic Quality (PQ) is needed to measure the performance of new algorithms. Panoptic Quality measures the quality of a predicted panoptic segmentation relative to the ground truth and can be explained by two terms, one is the Segmentation Quality (SQ) and the other one is the Recognition Quality (RQ). The definition for PQ, SQ and RQ can be seen on eq. 2.1, 2.3 and 2.4 respectively.

$$PQ = \frac{\sum_{(p,g) \in TP} \mathrm{IoU}(p,g)}{|TP| + 0.5|FP| + 0.5|FN|} \tag{2.1}$$

$$PQ = SQ * RQ \tag{2.2}$$

$$SQ = \frac{\sum_{(p,g)\in TP} \text{IoU}(p,g)}{|TP|} \tag{2.3}$$

$$RQ = \frac{|TP|}{|TP| + 0.5|FP| + 0.5|FN|} \tag{2.4}$$

Where p is the mask prediction given by either the semantic or instance segmentation and g is the ground truth. IoU is the intersection over union, also known as the Jaccard Index, or the overlap between both masks. And TP, FP and FN are respectively the amount of True Positives, False Positives and False Negatives.

It is worth noting that PQ treats both types of classes (stuff and things) uniformly. The division of PQ into SQ and RQ is only done in order to facilitate interpretation of results. Panoptic Quality is not a combination of semantic and instance segmentation metrics, it provides a unified evaluation over all classes. It is also common the separate use of the PQ metric for the two main class groups, known as PQ^{St} for stuff classes and PQ^{Th} for things classes. PQ_{St} can be seen as the mean intersection over union for semantic segmentation in most cases, where as PQ^{Th} can be influenced by the detection aspect of the metric if objects are not detected in the scene, even if the resulting masks describe very closely the instances' shapes.

2.2 Attention-guided Unified Network (AUNet) [23]

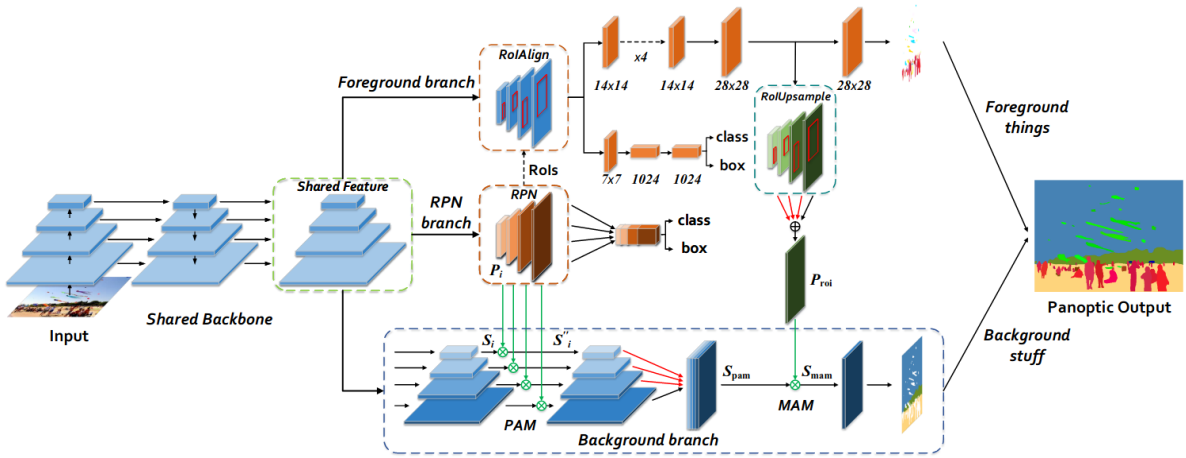


Figure 2.1 – AUNet's architecture, image taken from [23]

2.2.1 Backbone

On AUNet, a semantic segmentation model and an instance segmentation model are fused by sharing their Feature Pyramid Network (FPN) [24] backbone in order speed up computation and extract features from different scales. These features are then divided into three branches: foreground, background and region proposal network. This model uses the same process as Mask R-CNN [14] for instance segmentation and uses these feature maps to improve the semantic segmentation. This architecture can be seen on fig 2.1.

2.2.2 Proposal Attention Module

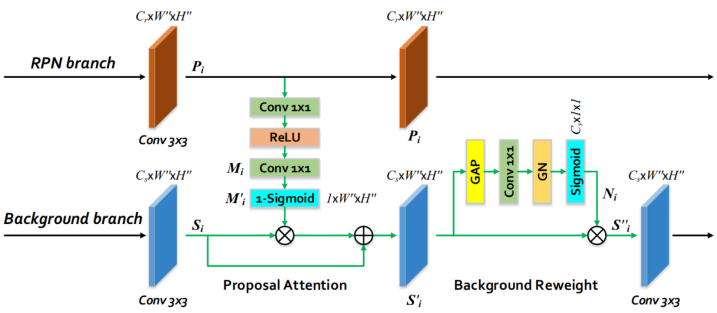


Figure 2.2 – Proposal Attention Module, image taken from [23]

Since Region Proposal Networks (RPN) [36] are used in two-stage detectors to do binary class separation (foreground and background), their features contain information about the background related to the objects in the foreground. Knowing this, a complementary relationship between foreground (FG) and background (BG) classes is established by a Proposal Attention Module (PAM) that can be seen on fig 2.2.

Here, the attention map S_i is generated by passing the ith RPN pyramid level through a series of 1x1 convolutions and activation functions, to then be multiplied and summed element-wise with the ith pyramid level resulting from the backbone. S_i goes then through a background reweight function in order to ignore useless background information.

2.2.3 Mask Attention Module

The Mask Attention Module (MAM) creates a feature map that will be added to the BG branch in order to further increase the relationship between FG and BG feature maps. Since the instance segmentation on the FG branch is done like in Mask R-CNN, and it implements the RoI-Align operation to get $m \times m$ masks, the RoI-Upsample module is proposed as an inverse operation for RoI-Align to get an $1 \times W' \times H'$ activated feature map. This activated map is then passed through a similar attention operation as in PAM and then passed through a background reweight function to get the final feature maps for the stuff classes.

2.3 Panoptic Feature Pyramid Network [21]

This network uses a FPN [24] as backbone and Mask R-CNN [14] as its instance segmentation branch. Here, Kirillov et al. propose a modification to the Mask R-CNN pipeline in order to add semantic segmentation.

The feature maps for semantic segmentation are generated by a series of convolutions and $2\times$ upsampling of the levels from the FPN in order to reach feature maps of 128 channels and a quarter the size of the input image. These upsampled feature maps are then summed together, upsampled by 4 and passed through a convolution in order to get C feature maps of the same size of the input image, where C is the number of stuff classes. These together with the instance masks are then fused to obtain the panoptic image, after resolving the different overlaps between the instance and semantic features.

2.4 Unified Panoptic Segmentation Network [42]

UPSNet uses FPN as its backbone for feature extraction, these features are then divided into two branches: one for instance segmentation by following the Mask R-CNN pipeline, and another for semantic segmentation. The output from these two branches are fused by a Panoptic Segmentation head to get the final panoptic segmentation image.

2.4.1 Semantic Segmentation Head

For semantic segmentation, levels 2 through 5 of the FPN are used. Each of these features are passed through a deformable convolution [8] network to then be upsampled to 1/4 scale. All of these feature maps are concatenated together and a 1×1 convolution is aplied with softmax to predict the semantic logits.

2.4.2 Panoptic Segmentation Head

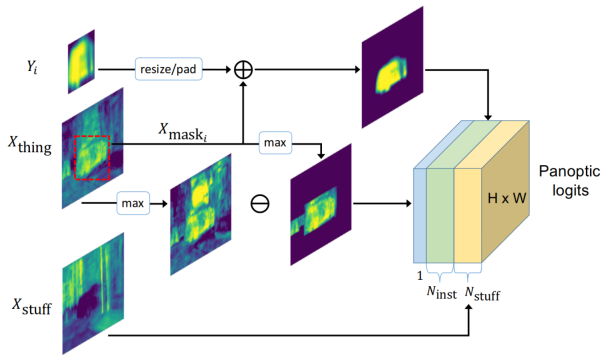


Figure 2.3 – Panoptic segmentation head of UPSNet, image taken from [42]

Once all the semantic and instance segmentation logits are processed, they are concatenated together to form a feature map of size $[(N_{stuff} + N_{inst}) \times H \times W]$ that will go through the Panoptic Segmentation head in order to uniquely determine both the class and instance ID for each pixel. This module can be seen on fig 2.3.

Since a bounding box (B_i) and class (C_i) are available for each instance, a second instance representation X_{mask_i} is extracted. This is done by taking the values in the space B_i of layer C_i from the semantic segmentation and setting the values outside of B_i in X_{mask_i} to zero. The instance mask is resized and padded in order to be of size $H \times W$ and summed with X_{mask_i} to be the final representation of the instance.

At inference time, if the mask is consistent with the segmentation maps, an instance ID is given. If this is not the case and the frequency of the mode of the segmentation maps is not higher than 0.5 and the class is one of the stuff classes, an instance ID is given, the ID is assigned to the segmentation maps otherwise. When facing inconsistency, the majority decision made by the segmentation maps is trusted more if only it prefers a stuff class.

Unknown pixel classification is added as a way to evade doing wrong predictions. Here, the unknown logits are defined as $Z_{unknown} = max(X_{thing}) - max(X_{mask})$, where X_{mask} is a concatenation of all instance masks along channel dimension and X_{thing} are the logits for thing classes.

This means that for any pixel if the maximum of X_{thing} is larger than the maximum of X_{mask}, an instance might be missing.

2.5 Single Network Panoptic Segmentation for Street Understanding [10]

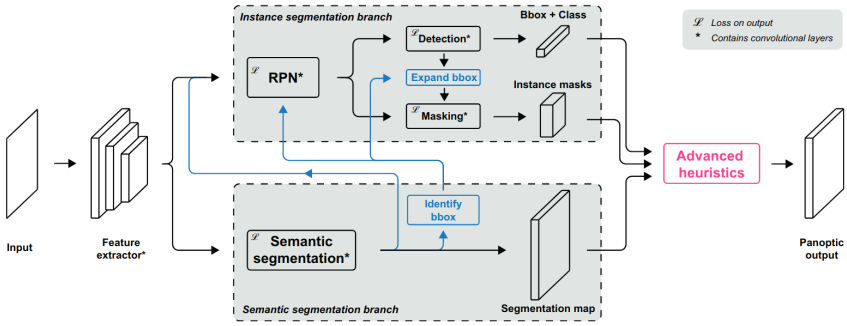


Figure 2.4 – Architecture of the network, image taken from [10]

For this network, a ResNet backbone was used, where the original output stride of 32 is replaced by an output stride of 8 in order to allow denser semantic segmentation predictions. This backbone is shared between the instance and semantic segmentation branches, in order to do the panoptic prediction in one pass. For instance segmentation, Mask R-CNN is used.

Semantic segmentation is done following the implementation done on Fully Convolutional Networks for Semantic Segmentation [28], where the predictions are made directly after the feature extractor. In order to increase performance in the segmentation branch, a Pyramid Pooling Module (PPM) [44] is added and a learnable deconvolution [30] plus upsampling is applied to resize the predictions to the original image size and generate the final output features.

2.5.1 Inter-branch information exchange

As a method of adding box proposals to the RPN module in the detection branch, clusters from the semantic segmentation output are used to generate extra bounding boxes. The predicted bounding boxes are matched with the segmentation output and expanded if the segment extends beyond the boundary of the box.

The network also implicitly uses semantic segmentation information to improve the segmentation of instances. This is done by taking the semantic features pre-softmax, normalizing and concatenating them to the normalized features from the backbone. A 3×3 convolution is then used and the resulting features are used as input for the instance segmentation branch.

2.5.2 Advanced Merging Heuristics

As a form to resolve overlaps on the instance segmentation branch, each pixel is assigned to the instance with the highest confidence score at that specific pixel. Then, when merging things and stuff, all the pixels from the instance masks are replaced as *void* label in the semantic segmentation output if their confidence value is higher than an α threshold.

Finally, any predicted stuff class that has a pixel count below a certain threshold is considered as an unlikely prediction and then replaced by either stuff classes above this threshold or a *void* label. This process is done since it is very unlikely that a stuff class consists of a small number of pixels.

2.6 End-to-End Network for Panoptic Segmentation [26]

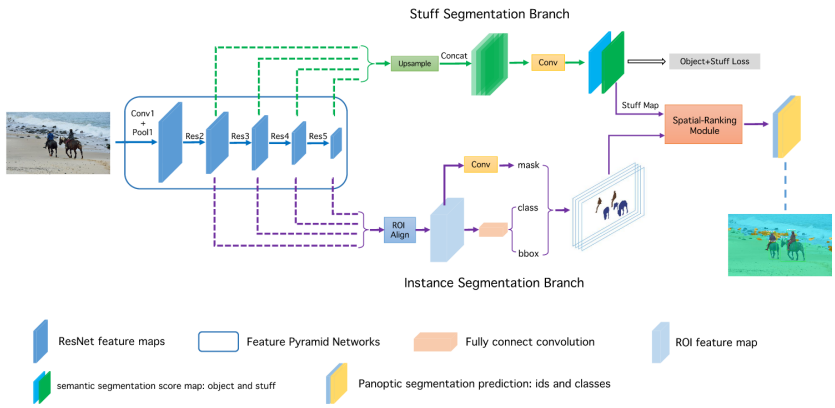


Figure 2.5 – Architecture of the network, image taken from [26]

The network uses a FPN as its backbone, and Mask R-CNN as its instance segmentation architecture. The feature maps are generated by applying a top-down pathway and lateral connections. A 3×3 convolution is applied to these maps to get the RPN feature maps. Instance masks are generated by following Mask R-CNN's method. For stuff segmentation, the RPN feature maps are passed through two 3×3 convolutions, concatenated and followed by two other 3×3 and 1×1 convolution layers.

2.6.1 Spatial Ranking Module

The Spatial Ranking Module is proposed as an alternative to heuristic-based overlap resolution of instances. First, the instance segmentation results are mapped to a single tensor for each class, resulting in a channel dimension equal to the amount of thing classes. Then a large kernel convolution [32] is applied to the feature map in order to generate a ranking score map. With this feature map, a ranking score of each instance is calculated using eq. 2.5. The final ranking score of the whole instance is computed by the average of pixel ranking scores in a mask.

$$P_{objs} = \frac{\Sigma_{(i,j)\in objs} S_{i,j,cls} \cdot m_{i,j}}{\Sigma_{(i,j)\in objs} m_{i,j}} \tag{2.5}$$

$$m_{i,j} = \begin{cases} 0 & \text{(i,j)} \in \text{instance} \\ 1 & \text{(i,j)} \notin \text{instance} \end{cases} \tag{2.6}$$

2.7 Deeperlab: Single-shot image parser [43]

2.7.1 Encoder

Yang et al. propose two encoders on [43], one focused on high quality predictions and another one focused on the task of real-time inference. The first is based on the standard Xception-71 [4]; and the second one is a modification of MobileNetV2 [38], referred as *Wider MobileNetV2*, where all the (3×3) convolutions are replaced by (5×5) convolutions in order to increase the receptive field from (491×491) to (981×981) while maintaining the same amount of memory footprint.

Both of these encoders are followed by an Atrous Spatial Pyramid Pooling [2] (ASPP) module, which helps to further increase the receptive field by applying several parallel atrous convolutions.

2.7.2 Decoder

The authors combine the activations at the output of the encoder (with stride 16) and low-level feature maps (with stride 4) in order to recover object boundaries. These ASPP feature maps are first reduced by a (1×1) convolution to then be concatenated together. The low level features are fed through a space-to-depth (S2D) [39] operation which keeps the memory usage of the feature maps the same.

All of these features are then concatenated and two (7×7) kernel operations are applied in order to further increase the receptive field. These resulting features are then reduced by applying a depth-to-space (see fig. 2.6) operation in order to have a feature map of 256 channels and stride 4. This feature map will be the input for both the semantic and instance segmentation heads.

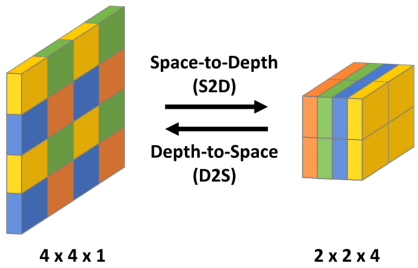


Figure 2.6 – An example of the space-to-depth (S2D) and depth-to-space (D2S) operations. The S2D operation moves activations from the spatial dimension to the channel dimension and the D2S operation is the inverse. Image and caption taken from [43].

2.7.3 Semantic Segmentation Head

Here the features from the decoder pass through two convolution layers, (7×7) with 256 channels and (1×1), respectively. For semantic segmentation the bootstrapped cross-entropy loss [41] is minimized, where the pixels are sorted based on the cross-entropy loss and only the top-K positions are propagated. Here, $K = 0.15 \cdot N$, where N is the total number of pixels in the image. Also, the pixel loss is weighted with respect to the instance sizes, following this bootstrapped cross-entropy loss:

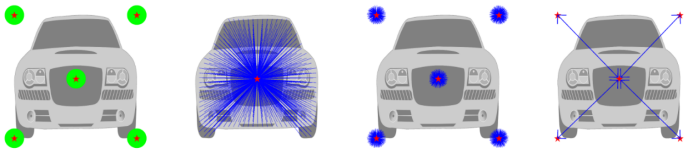
$$l = -\frac{1}{K}\sum_{i=1}^{N} w_i \cdot \mathbb{1}[p_{i,y_i} < t_K] \cdot \log p_{i,y_i} \tag{2.7}$$

where y_i is the target class label for pixel i, $p_{i,j}$ is the predicted posterior probability for pixel i and class j, and $\mathbb{1}[x] = 1$ if x is true and 0 otherwise. t_K is the threshold such that only the pixels with top-K highest losses are selected. The weight w_i is set to 3 for pixels that belong to instances with an area lower than 64x64 and 1 otherwise, this is done so that the network focuses on both hard pixels and small instances.

2.7.4 Instance segmentation heads

For instance segmentation, this model uses a keypoint-based representation and it contains four heads: keypoint prediction heatmap, long-range offset map, middle-range offset map and short-range offset map, these prediction maps can be seen on fig. 2.7. Where each one of them receive the features from the decoder and fed them through two convolutions, a (7×7) convolution with 64 channels and a (1×1) convolution. Here the four bounding box corners plus the center of mass of the object are considered, giving a total of $P = 5$ object keypoints.

The keypoint prediction heatmap predicts whether or not a pixel is within a disk of radius R pixels centered in the corresponding keypoint. The long-range offset map predicts the position offset from a pixel to all the corresponding keypoints, encoding the long-range information for each pixel. The short-range offset map is similar to the long-range offset map except that it only focuses on within the disk of radius R. Finally, the middle-range offset map predicts the offset among keypoint pairs, defined in a *directed keypoint relation graph (DKRG)*, which is used to group keypoints related to the same instance.



(a) (b) (c) (d)

Figure 2.7 – Four prediction maps generated by our instance-related heads: (a) keypoint heatmap, (b) long-range offset,(c) short-range offset, and (d) middle-range offset. The red stars denote the keypoints, the green disk denotes the target for keypoint prediction, and the blue lines/arrows denote the offsets from the current pixel to the target keypoint. Image and caption taken from [43].

2.7.5 Prediction Fusion

Keypoint refinement is done as in Personlab [31]. Keypoints are found by a mixture of Hough-voting on the short and long-range maps, which are merged into one. Local maxima is used to localize the keypoints, which are then scored using the *Expected-OKS* score [31].

To detect the instance, the keypoints are pushed into a queue and popped one a the time. If a popped keypoint is in the proximity of the corresponding keypoint of a detected instance, it is rejected and another one is popped. If it is not, the middle-range map is used to find the

positions of the remaining four keypoints. The confidence score of the instance is defined as the average of its keypoint scores. Overlapping instances are removed using non-maximum suppression after all instances are detected.

Instance labels are assigned to pixels using the predicted long-range offset map. Here, they assign each pixel to the detected instance whose keypoints have the smallest L_2-distance to the pixel's predicted keypoints.

Finally, the semantic label of each pixel is determined by considering 'stuff' and 'thing' classes differently. Pixels whose predicted label is in the 'stuff' category are given a unique instance id. For the other pixels, each id is determined by the instance segmentation result and their label is determined by a majority vote of the semantic labels.

2.7.6 Parsing Covering metric

The Parsing Covering (PC) metric is proposed as an added metric to evaluate the quality of image results. This metric accounts for instance sizes, which are not a factor in the PQ metric.

$$Cov_i = \frac{1}{N_i} \sum_{R \in S_i} |R| \cdot \max_{R' \in S'_i} IoU(R, R'), \tag{2.8}$$

$$N_i = \sum_{R \in S_i} |R|, \tag{2.9}$$

$$PC = \frac{1}{C} \sum_{i=1}^{C} Cov_i, \tag{2.10}$$

Where R and R' are groundtruth regions and predicted regions respectively, S_i and S'_i are the groundtruth segmentation and predicted segmentation for the i-th semantic class respectively, and N_i is the total number of pixels of groundtruth regions from S_i. A notable difference between PQ and PC is that PC does not require matching.

2.8 Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation [3]

2.8.1 Backbone

Panoptic-Deeplab uses an encoder (ResNet50, Xception-71 or MobileNetV3) which is shared for both semantic and instance segmentation branches. Then, two separate ASPP and decoder modules are used for each branch. The decoder follows DeepLabV3+ [2] with two modifications: (1) an additional low-level feature is introduced to the decoder with output stride 8, and (2) in each upsampling a (5×5) depth-wise separable convolution is applied [17].

2.8.2 Semantic Segmentation head

The bootstrapped cross entropy loss (eq. 2.7) introduced in [43] is used for semantic segmentation to predict both 'stuff' and 'thing' classes.

2.8.3 Instance segmentation head

Object instances are represented by their center of mass, where for every foreground pixel the offset to the corresponding mass center is predicted. During training these centers are represented as a 2D Gaussian with a standard deviation of 8 pixels, and during inference predicted foreground pixels are grouped to their closest predicted mass center.

2.8.4 Panoptic Segmentation head

Since the objects are represented by their center of mass, max-pooling is performed on the instance center heatmap prediction as a form of keypoint-based NMS. Then, a threshold is applied to filter out predictions with low confidence, where only locations with top-k highest confidence scores are kept.

Instance center regression is used to obtain the instance id for each pixel, where an offset vector $O(i, j)$ to its center is predicted. The instance id for the pixel is thus the index of the closest instance center after moving the pixel location (i, j) by the offset $O(i, j)$. That is:

$$\hat{k}_{i,j} = \underset{k}{\operatorname{argmin}} ||C_k - ((i, j) + O(i, j))||^2 \tag{2.11}$$

where $\hat{k}_{i,j}$ is the predicted instance id for pixel at (i,j) and C_k is the center of mass for the k-th object. Pixels with 'stuff' classes are set to have 0 as their instance id.

For the merging, "majority vote" is used. Where the semantic label of a predicted instance mask is inferred by the majority vote of the corresponding predicted semantic labels.

Instance Segmentation as output

This model can also generate instance segmentation models as a by-product, where the confidence score for the mask is given by:

$$Score(Objectness) \times Score(Class) \tag{2.12}$$

Where $Score(Objectness)$ is the unnormalized objectness score obtained from the class-agnostic center point heatmap, and $Score(Class)$ is obtained from the average of semantic segmentation predictions within the predicted mask region.

2.9 AdaptIS: Adaptive Instance Selection [40]

2.9.1 Backbone

AdaptIS uses a standard backbone such as ResNet-50, ResNet-101 or ResNeXt-101 pretrained on the ImageNet dataset[37].

2.9.2 Instance Selection Network

Using Adaptive Instance Normalization (AdaIN) [18], the authors propose the generation of object masks based on a 'characteristic' vector. Where the network can be parameterized to vary the output by providing different parameters to AdaIN, resulting in an instance selection

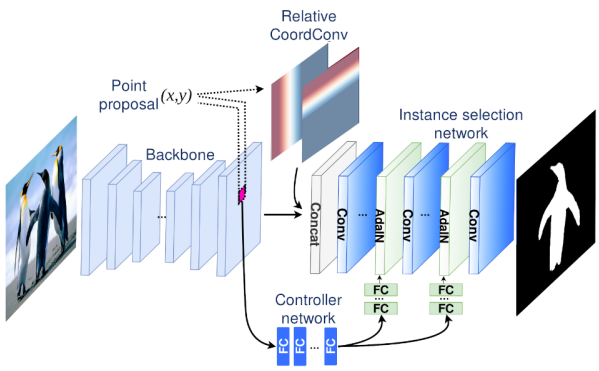


Figure 2.8 – AdaptIS architecture for class agnostic instance segmentation, image taken from [40]

network that generates an object mask using features from the backbone and a "characteristic" vector.

To generate said "characteristic" vector for a point (x,y), a corresponding embedding $Q(x,y)$ is found by bilinearly interpolating the output of the backbone. $Q(x,y)$ is then processed in a *controller network*, which is composed by fully connected layers, to get the 'characteristic' vector. By using this mechanism, the instance selection network is able to adapt to the selected object and the backbone is forced to produce rich features due to the feedback loop provided by the controller network.

To aid in the disambiguation of different objects, a *Relative CoordConv* block based on the CoordConv [27] is proposed. The main purpose of this block is to produce two maps for x and y coordinates, these are predicted based on the point proposals and then concatenated with the features produced by the backbone.

Overlaps between instances are resolved by the use of a greedy algorithm. It begins by initializing a map S of segmented objects where all elements are initialized as 'unknown'. At each iteration, a point (x,y) is sampled to obtain an AdaptIS output C_i, where the mask M_i is obtained from by applying a threshold T as $M_i = C_i > T$. The resulted mask is added to the map as 'segmented' if it has an overlap lower than 50% with other masks in S, it is ignored otherwise. This process is finished once all the pixels in S are marked as 'segmented' or when no more point proposals are available. The pixels are assigned the instance id that has the highest confidence among all segmented objects.

2.9.3 Semantic Segmentation and Point Proposal branches

For the semantic segmentation task, a standard pipeline is put in parallel with AdaptIS on a shared backbone.

For the task of Panoptic Segmentation, a point proposal branch that predicts the priority of sampling different point proposals is trained. This network predicts whether a point (x,y) would make a good or a bad proposal and follows the same structure as the semantic segmentation branch.

2.9.4 Merging

To obtain the final panoptic image, all the "stuff" labels are found in one pass of the semantic segmentation branch. Then the instance segmentation is done with the semantic pixels added to the map S as 'segmented' instead of using an empty map.

In order to reduce the number of point proposals for evaluation, local maximums are found in the result of point proposal branch by using a breadth-first algorithm.

2.10 Fast Panoptic Segmentation Network [11]

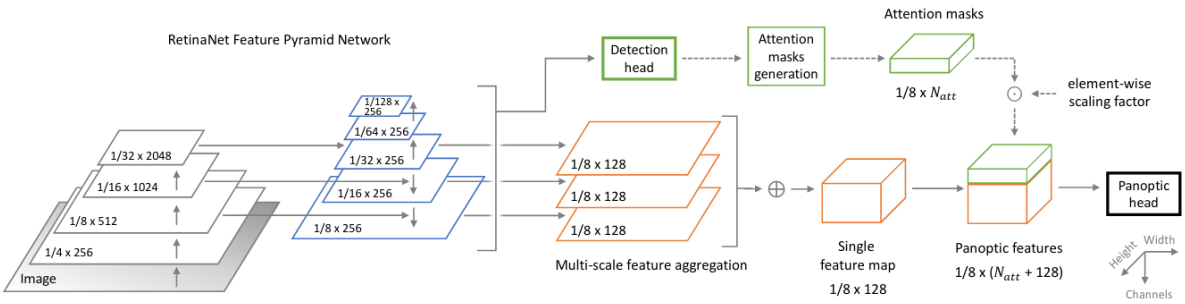


Figure 2.9 – FPSNet architecture for panoptic segmentation, image taken from [11]

2.10.1 Backbone

For FPSNet, a ResNet-50-based Feature Pyramid Network (FPN) is used as a backbone. But since their implementation is based on a single feature map, the authors take feature maps P3, P4 and P5 with stride of 8. Where two upsampling steps are applied to P5 and one to P4 in order to create S5 and S4 respectively. These upsampling steps are defined as a (3×3) convolutional layer with ReLU, followed by $2\times$ bilinear upsampling. To create S3, a (3×3) convolutional layer with ReLU is applied to P3. Finally, the single feature map S is created as $S = S3 + S4 + S5$.

2.10.2 Attention Mask

This is part of what the authors call Panoptic Module. It is assumed that bounding box predictions are given by a detector. It works by projecting the bounding boxes on a tensor with the dimensions of the feature map and then filling the bounding box with values of a Gaussian distribution with mean $\mu = (x_c, y_c)$ and covariance $C = Diag(0.25 \cdot w_b, 0.25 \cdot h_b)$, where (x_c, y_c), w_b, h_b are the center coordinates, width and height of the bounding box, respectively. Outside the bounding boxes the values are set to zero.

A hyperparameter N_{att} is used to define the number of attention masks to be used. Where if there are more than N_{att} bounding boxes given by the detector, only the N_{att} with highest scores are chosen. If there are less than N_{att}, zero-filled masks are used as the remaining masks. These masks are shuffled in order to divide the instances among the channels of the convolutional layer as equally possible during training.

The attention masks are then stacked in the outer (or channel) dimension of a tensor so it has shape [N_b,H,W,N_{att}], where N_b is the batch size, H and W are the height and width of the image, respectively. The masks are then scaled in the range [0,1] and multiplied by a hyperparameter C_{att} to facilitate training by ensuring that the attention masks are in the same order of magnitude as the features.

This feature map is then merged to the features of the backbone by concatenating them to the attention masks and then applying a (3×3) convolution, ReLU activation and batch normalization.

2.10.3 Panoptic Head

This head consists of four more sets of (3×3) convolution, ReLU activation and batch normalization. To get the final panoptic map, a (1×1) convolution is applied to predict N_{out} outputs for each pixel, where $N_{out} = N_{att} + N_{stuff} + 2$. Here the pixels for the nth output belong to the nth attention mask, and the instance or class is defined for each pixel by picking the layer with the highest score (argmax) after bilinearly upsampling the logits to the original dimensions of the image.

2.11 Real-Time Panoptic Segmentation from Dense Detections [16]

2.11.1 Backbone

For this network, the authors use a ResNet-50 together with a Feature Pyramid Network (FPN). Where the FPN module has 5 levels corresponding to strides 128, 64, 32, 16 and 8 with respect to the input image.

2.11.2 Parameter-Free Mask Construction

Dense bounding box detection is performed where at least one bounding box is predicted at each location in the input image:

$$\begin{gathered} \mathcal{B}(x,y) = \mathbf{B}, \quad \mathbf{B} = (\mathbf{b}, c) \\ \mathbf{b} = (x_1, x_2, y_1, y_2) \in \mathbb{R}^4, \quad c \in \{1, ..., N_{things}\} \end{gathered} \tag{2.13}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the left-top and bottom-right corners for bounding box $\mathbf{B}$ that pixel (x,y) belongs to; c is the predicted class ID for the corresponding bounding box. Given all detected boxes, a reduced set (B_{query}) is obtained through NMS:

$$\mathcal{B}_{query}(x,y) = \mathbf{B}_j, \quad \mathbf{B}_j = (\mathbf{b}_j, c_j) \tag{2.14}$$

This reduced set will be used to search for instance masks, where for each box $\mathbf{B}_j$ a global mask probability is created as:

$$\mathcal{M}(x,y,j) = \hat{P}_{loc}(x,y,j) \cdot \hat{P}_{sem}(x,y,c_j), \quad \mathbf{M}_j(x,y) = \mathcal{M}(x,y,j) > \sigma, \tag{2.15}$$

where $\hat{P}_{loc}(x,y,j)$ is an estimated probability that pixel (x,y) shares the same bounding box as object j, and $\hat{P}_{sem}(x,y,c_j)$ is the probability that pixel (x,y) shares the same semantic class

as object j, which is the value from the semantic map for (x,y) in c_j. The final set of instance masks $\mathbf{M}_j$ is constructed by applying a simple threshold σ to the global mask probability map.

2.11.3 Panoptic Segmentation

In this paper a single-stage panoptic segmentation network is proposed in order to achieve real-time performance.

Target assignment

Let $\mathbf{F}_i \in \mathbb{R}^{h_i \times w_i \times C}$ be the feature map at layer i of the FPN, with stride z. Since the pixel (x,y) on the feature map is the center of a receptive field, its original pixel location (x_0, y_0) in the input image can be found. If this original pixel location falls within one of the ground truth masks $\mathcal{M}_{GT} : \{\mathbf{M}_{GT_j}\}$, then it is considered to be in the foreground, and will be associated to the ground truth mask $\mathbf{M}_{GT_j}$ and its corresponding bounding box $\mathbf{B}_{GT}$.

If location (x_0, y_0) is associated with $\mathbf{B}_{GT}$, the following regression offsets $\mathbf{t}_{xy} = (l,t,r,b)$ are assigned to it:

$$\begin{aligned} l &= x_0 - x_1, \quad t = y_0 - y_1, \\ r &= x_2 - x_0, \quad b = y_2 - y_0, \end{aligned} \tag{2.16}$$

where $\mathbf{b}_{\mathbf{GT_j}} = (x_1, y_1, x_2, y_2)$ are the bounding box coordinates as defined in Eq 2.13. And since it is possible that locations on multiple FPN levels have the same (x_0, y_0), disambiguation is done by removing offset $\mathbf{t}_{xy}$ from the assigned set of regression targets $\mathcal{T}_i$ for feature map F_i if it does not satisfy:

$$\mathbf{t}_{xy} \in \mathcal{T}_i, \text{iff} \quad m_{i-1} \leq \max(l,t,r,b) \leq m_i, \tag{2.17}$$

where m_i is the maximum regression size for $\mathbf{F}_i$.

A centerness o_{xy} value is also predicted to down-weight predicted bounding boxes near boundaries through NMS. Resulting on a final 6-dimensional label, $(\mathbf{t}_{xy}, c_{xy}, o_{xy})$ at each foreground location.

$$o_{xy} = \sqrt{\frac{\min(l,r)}{\max(l,r)} \times \frac{\min(t,b)}{\max(t,b)}} \tag{2.18}$$

Unified Panoptic Head

Since the main focus of the paper is to eliminate redundant operations between the instance and semantic segmentation, the authors propose a head whose task is to do localization and semantics in parallel from the features found at the selected levels of the FPN.

For this, a Localization tower and a semantics tower are proposed, where each tower contains 4 sequential convolutional blocks (Conv + GroupNorm + ReLU). In addition to the per level predictions, these processed features are also used to globally predict:

1. Levelness $\mathcal{I}$: The FPN level that the bounding box at each (x,y) belongs to, where 0 is given to background pixels.

2. Semantic segmentation $\mathcal{S}$: the semantic class probability distribution over N classes.

— 3 —

Panoptic Segmentation using Darknet-53 and YOLOv3

Most of the state of the art implementations for panoptic segmentation use Residual Networks [15] and multi-level feature map aggregation strategies like Feature Pyramid Networks (FPN) [24]. By using feature maps from different scales, the network would be able to extract information that is semantically strong and could be used for different tasks such as detection or segmentation. Having this in mind, together with the consideration that a shared feature extractor backbone would eliminate the necessity of using parallel networks and shorten inference time for the tasks associated to panoptic segmentation, we decided to test how could You Only Look Once version 3 (YOLOv3) [35], and its backbone Darknet-53, be used for real-time Panoptic Segmentation. For this, the goal is to create a segmentation head, which would be added in parallel to the YOLOv3 architecture, in order to do panoptic segmentation using feature maps from Darknet-53 and the object proposals from the YOLO layers.

This chapter addreses first the different network architectures considered for the task of semantic segmentation and how they compare to each other in terms of speed and segmentation accuracy. Then, after the final semantic segmentation architecture is selected, instance segmentation architectures that rely on the predictions from the semantic segmentation network and the YOLO layers will be evaluated and one will be selected. Finally, the merging strategy for all of the final predictions will be explained and results of the final panoptic segmentation network will be shown.

3.1 You Only Look Once Architecture

It is important to do a small introduction to the architecture which the panoptic segmentation network will be based on. YOLO has been a competitive single-shot object detection network for real time inference. Up to date it has four different versions, here we will do a brief explanation of the first three.

3.1.1 You Only Look Once (YOLO)

The original YOLO architecture [33] divides the given image into an $S \times S$ grid, where each cell predicts only one object and a number of bounding boxes are predicted for the object. Each boundary box contains 5 elements: (x, y, w, h, C), where (x, y) are the coordinates of the box

center relative to the bounds of the grid cell, w and h are the box width and height with respect to the image, and C are the conditional class probabilities for the cell.

Since YOLO frames detection as a regression problem it does not need a complex pipeline. This allows for very short inference times, reaching performance levels of 45 FPS or even 150 FPS if a smaller network version is used.

Between the limitations of YOLO are: spatial constraints on bounding box predictions (two boxes and one class per grid), problem to generalize to objects in new or unusual aspect ratios and a loss function that treats errors of small and large bounding boxes equally.

3.1.2 YOLO9000

Given the shortcomings of YOLO, some changes were implemented in order to improve its performance in YOLO9000 [34]. First, Batch Normalization was added after each convolutional layer, which resulted in a 2% increase in mAP. Second, the classification network was fine tuned on the full 448×448 images from ImageNet, resulting on an almost 4% mAP increase. Third, convolutional anchor boxes are predicted and the fully-connected layers are removed. Fourth, the network is shrunk to operate on 416×416 input images in order to get a feature map with a center box after convolutions.

Top-K bounding boxes were identified for the Pascal VOC2007 and COCO datasets and the priors that gave the best tradeoff between model complexity and recall were chosen ($k = 5$). The anchor prediction is constrained by using the sigmoid function in order to predict location coordinates relative to the cell location.

Darknet-19 is used as a new backbone. This allows faster inference while keeping 72.9% top-1 and 91.2% top-5 accuracy on ImageNet.

3.1.3 YOLOv3

For YOLOv3 [35], multi-label classification is considered. This is done by replacing the softmax function with independent logistic classifiers to calculate the likeliness of the input belonging to a specific label.

A newer version of the backbone is presented as Darknet-53, which achieves the same classification accuracy than ResNet at half the speed. This network uses successive 1×1 and 3×3 convolutions together with shortcut connections and its architecture can be seen on figure 3.1.

3.2 Semantic Segmentation Network

3.2.1 Experimentation with Single Feature Maps

In order to create the segmentation head, a decoder architecture that could take feature maps from the backbone was considered. Following the fully convolutional decoder architecture of the Segnet network [1], and by taking feature maps from the middle layers of the backbone a good segmentation result was expected to be achieved. The architecture for the decoder head is composed by a series of 3×3 convolutions, batch normalization layers and $2\times$ upsampling, it can be seen on fig. 3.2.

At first, the convolutional features right before the YOLO prediction heads were considered. But they were discarded as potential inputs for the segmentation decoder due to their small

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.1 – Darknet-53 architecture, taken from [35]

spatial size and possible specialization towards detection and not segmentation. Instead, it was decided to take the features from middle layers in the backbone, since their spatial resolution is considered to be big enough to carry relevant information for the semantic segmentation task.

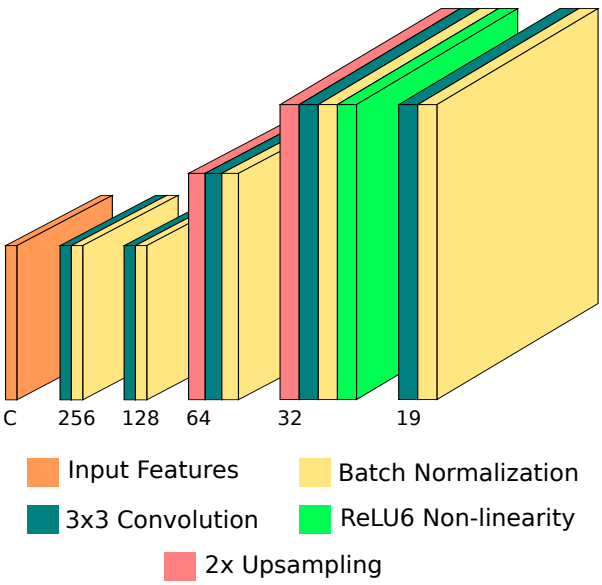


Figure 3.2 – Decoder Architecture proposed for semantic segmentation using single feature maps from Darknet-53

The chosen feature maps from the backbone to be used in the decoder network were the outputs from the 12th, 37th and 62nd netork layers. These layers were chosen since they are

followed by a convolutional layer with stride 2, making the feature maps the last features before a spatial scale reduction. The feature sizes can be observed in table 3.1.

The backbone network was frozen and only the decoder was trained. Training was performed for each chosen feature map on the Cityscapes Dataset [7] for 50 epochs, using Cross entropy Loss as the loss function, the Adam optimizer [20] and the Cosine Annealing [29] learning rate scheduler. Early stopping was set at the condition of the validation loss not decreasing after 5 epochs and the decoder networks were initialized following the Golorot initialization method [13]. The following transformations were applied in order to perform data augmentation to the dataset and maximize the network's performance:

- Random scaling between $[0.25, 4]$ times the spatial size of the image.
- Random rotations between $[-10, 10]$ degrees.
- Random probability of Gaussian blur with a 5×5 kernel.
- Random horizontal flip.
- Random crop of size 640×640.

In order to standardize the input to the decoder network, all features are upsampled to $1/16$ times the size of the original image if needed. On tables 3.1 and 3.2, can be seen the results for each of the tested configurations on the train and validaton sets. Here, the mean Intersection over Union (mIoU) is used as the metric to compare how much do the predictions match the ground truth. For the mIoU metric, two sizes of images were tested in order to evaluate how does each candidate perform on its original training image size and an upscaled one. Per class scores can be found on Appendix A.

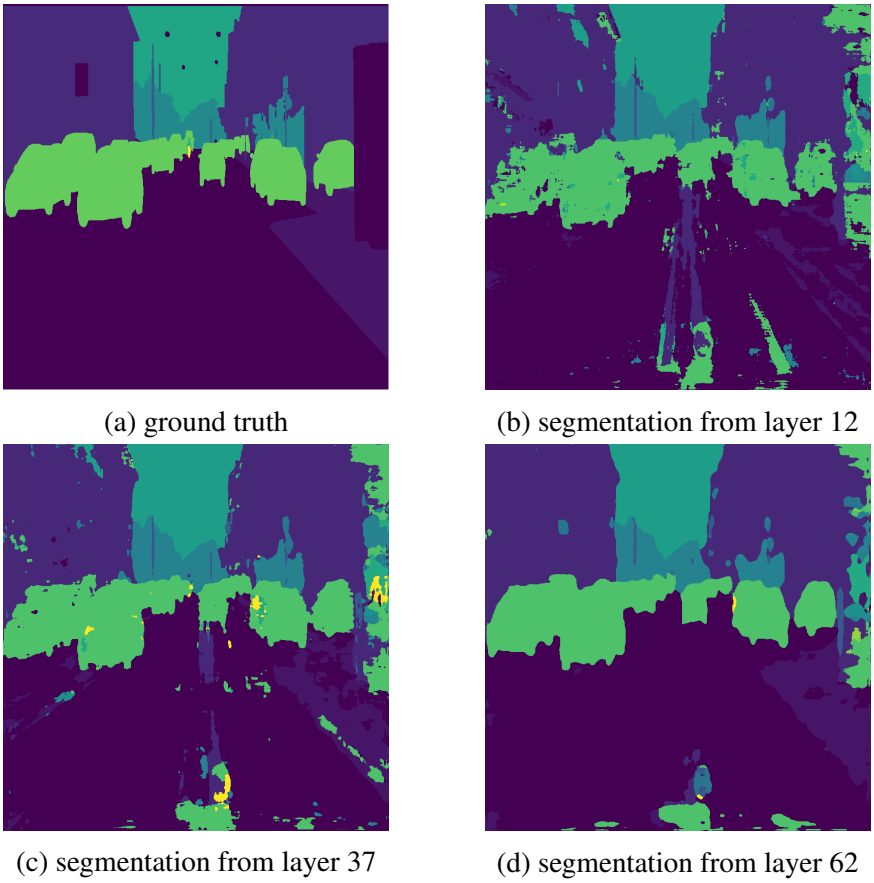
Layer taken from	**Shape**	**mIoU** (640×640)	**mIoU** (1024×1024)
12	$128 \times H/4 \times W/4$	0.2454	0.2429
37	$256 \times H/8 \times W/8$	0.3027	0.3045
62	$512 \times H/16 \times W/16$	**0.3533**	**0.3683**

Table 3.1 – Mean Intersection over union for each tested group of features on the training set of the Cityscapes Dataset. H and W are the original image's height and width respectively.

Layer taken from	**mIoU** (640×640)	**mIoU** (1024×1024)
12	0.2529	0.2340
37	0.3081	0.2959
62	**0.3560**	**0.3566**

Table 3.2 – Mean Intersection over union for each tested group of features on the validation set of the Cityscapes Dataset.

Here it can be seen that the features from the deepest proposed layer give the highest performance result despite their reduced spatial size. The segmentation results from each of the proposed configurations can be seen on fig. 3.3.



(a) ground truth

(b) segmentation from layer 12

(c) segmentation from layer 37

(d) segmentation from layer 62

Figure 3.3 – Segmentation ground truth and results for the three tested configurations.

Even though the results show that the features from the Darknet-53 backbone can be used for semantic segmentation, they could be improved by leveraging multi-scale information. In order to do this, all three feature maps will be used together in different architectures to find one with the best performance.

3.2.2 Experimentation with Multiple Feature Maps

In order to use all three feature maps to recover information from multiple scales, two strategies were used to merge the information between the layers. The first one follows a very simple approach, by upsampling and concatenating all three, the features could be input together into the decoder architecture. The second tested strategy is the one used with Feature Pyramid Networks (FPN), where by upsampling and adding features together in a per-level process, multi-scale information could be used to improve semantic segmentation performance. Three network architectures are proposed to find the one that gives the best segmentation performance.

For the first architecture, denominated as *Segnet multi-feature*, the features from the 37th and 62nd layers first get upsampled to match the spatial size of the features from the 12th layer. Then, they are concatenated together and processed through a 1×1 convolution in order to reduce channel size and be passed through the previous convolutional decoder architecture (fig. 3.2). This architecture can be seen on fig. 3.4.

The second architecture processes the features as an *FPN* in order to add the feature maps

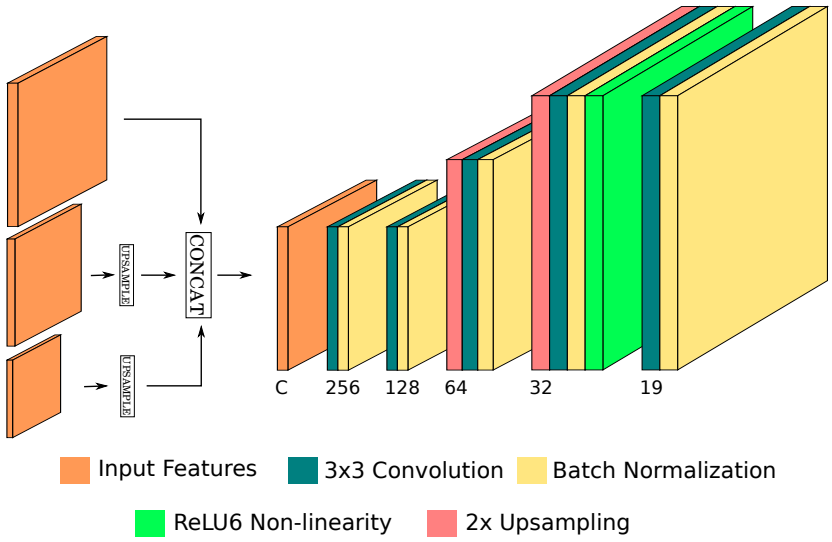


Figure 3.4 – Segnet multi-feature architecture for segmentation, modified from [44].

together. This is done by taking one set of feature maps from the backbone, using a 1x1 convolution plus upsampling to match the shape of the next feature maps, and then performing addition between both feature maps. Once the last feature map is added, a $2\times$ upsampling is performed and two sets of 3×3 convolutions are used to get the final segmentation output. Here, the feature maps from the 5th, with size $(64 \times H/2 \times W/2)$ convolution are added in order to have more contextual information in the pyramid. On figure 3.5, the convolutional blocks represent a convolution plus batch normalization.

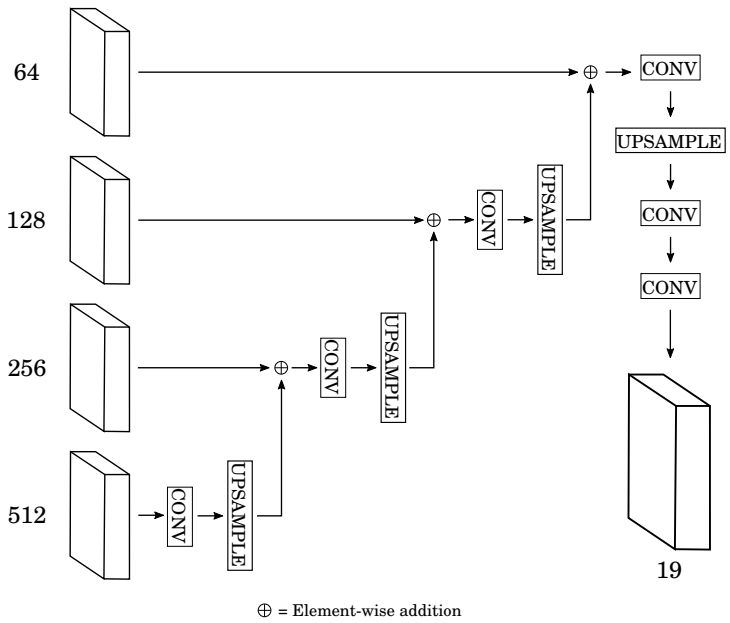


Figure 3.5 – FPN architecture for segmentation.

For the last architecture, the *Pyramid Pooling Module* (PPM), from the Pyramid Scene Parsing Network [44], is used. Here, the features are preprocessed and concatenated to create

a feature map with depth of 1024 channels and $1/16$ the size of the input image. For the preprocessing step a 1×1 convolution is used to increase the channel depth of the 12th layer feature map from 128 to 256, also the 37th and 62nd features are upsampled by 4 and 16 times respectively. Once all the features are concatenated, the PPM does a series of pooling operations "*to extract information with different scales and varying among different subregions from different levels*" [44], by doing global average pooling with bins of size 1, 2, 3 and 6. After each pooling operation, a 1×1 convolution is applied to reduce its channel depth by four, followed then by batch normalization and a ReLU non-linearity. The resulting feature maps are upsampled to the input feature size and concatenated with it. Finally, the semantic segmentation is obtained after a 3×3 and a 1×1 convolution. In this process, the auxiliary loss specified in [44] was not used since all training was done with a frozen backbone.

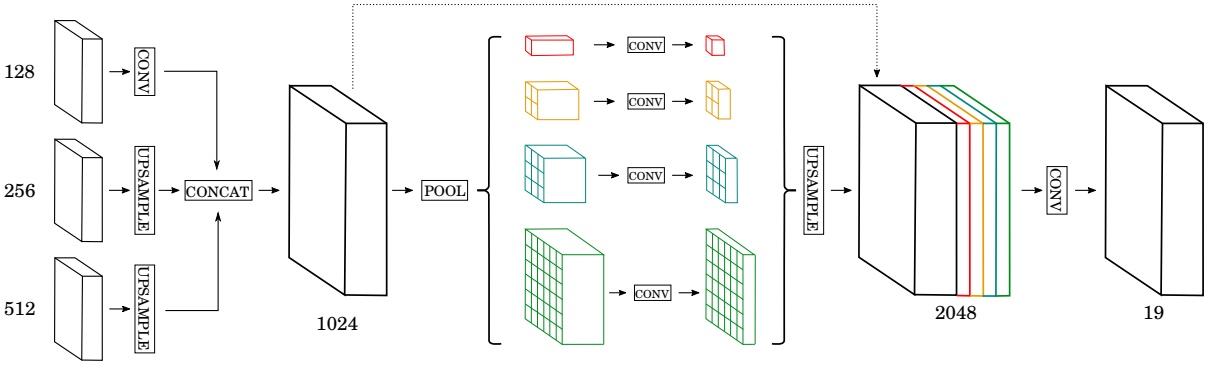


Figure 3.6 – Pyramid Pooling module-based architecture for segmentation, modified from [44].

For these three candidate architectures the same training procedure and data augmentation techniques from the single feature experimentation were used. This time the training was extended to 70 epochs and early stopping to 7 epochs after the validation loss not decreasing. On tables 3.3 and 3.4, the performance of each of these networks is displayed together with their inference speeds on the test and validation sets of the Cityscapes dataset running on a Nvidia Tesla V100 GPU.

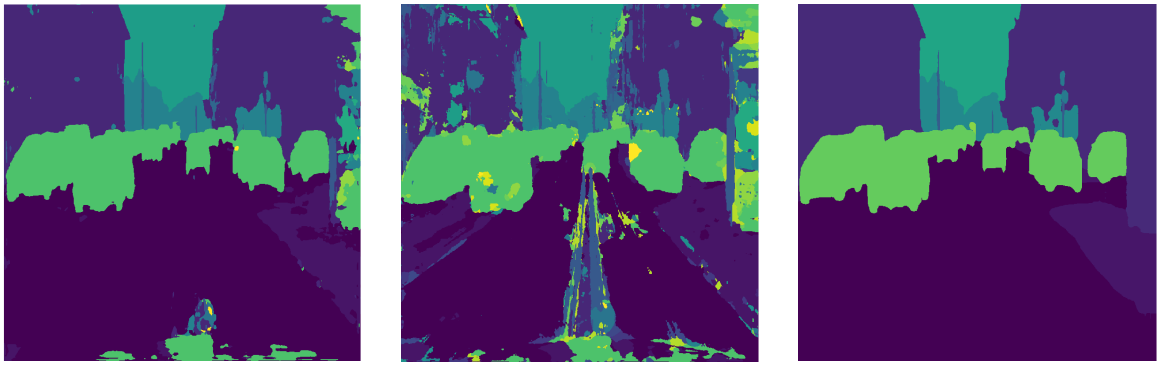
Architecture	**Train mIoU**	**Validation mIoU**	**time** (ms)
Segnet multi-feature	0.3878	0.3702	20.0
FPN	0.3409	0.3217	20.2
PPM	**0.5848**	**0.5879**	**19.1**

Table 3.3 – Mean Intersection over union for the proposed multi-feature architectures on images of size (640×640).

From these results can be seen that, in general, adding features from multiple scales improves the segmentation quality of the network. On fig. 3.7, a qualitative comparison can be done for each of the proposed networks. The FPN network gives an image with some inconsistencies, where patches of some class labels appear on top of others when they do not appear in the ground truth (fig. 3.3a). The Segnet multi-feature and PPM architectures show consistent results towards the center of the image, but the former fails to properly classify the pixels near the bottom and right part of the image.

Architecture	Train mIoU	Validation mIoU	**time** (ms)
Segnet multi-feature	0.3961	0.3824	**34.6**
FPN	0.3403	0.3242	34.7
PPM	**0.5864**	**0.5684**	36.0

Table 3.4 – Mean Intersection over union for the proposed multi-feature architectures on images of size (1024×1024).



(a) Segnet multi-feature (b) FPN (c) PPM

Figure 3.7 – Segmentation results for the three architectures using multiple feature maps.

All networks still perform at real-time speeds (~30 frames per second) when doing both detection and segmentation in high resolution images (1024×1024). From the presented options, the architecture based on the Pyramid Pooling Module shows the best performance at both low and high resolution images, outperforming the other methods by a difference of more than 15% and giving cleaner segmentation images. According to these results, the PPM architecture is the semantic feature extraction head that the panoptic segmentation network will use to perform instance segmentation.

3.3 Instance Segmentation Network

With the detection task done by the YOLO layers, and the semantic segmentation task done by the Pyramid Pooling Module, the next task is to merge the information from both in an instance segmentation architecture in order to find the masks for the instances in the images. For this, an architecture similar to the one used for the mask branch in [14] is used to do foreground segmentation.

First, non-maximum suppression is performed on the bounding box predictions from the YOLO layers in order to find the best detections. Then, a (1×1) convolution is applied to the features from the PPM in order to reduce the depth dimension, plus an upsampling operation to increase the spatial dimensions. The boxes are then resized to match the scale of the output features, from the Pyramid Pooling Module preprocessed features, in order to be used as attention masks for the foreground segmentation process. Also, in order to add contextual information from the semantic segmentation prediction, the feature plane from the corresponding class given by the detection is concatenated to the preprocessed features, resulting in the final instance input features. This process is illustrated in fig. 3.8.

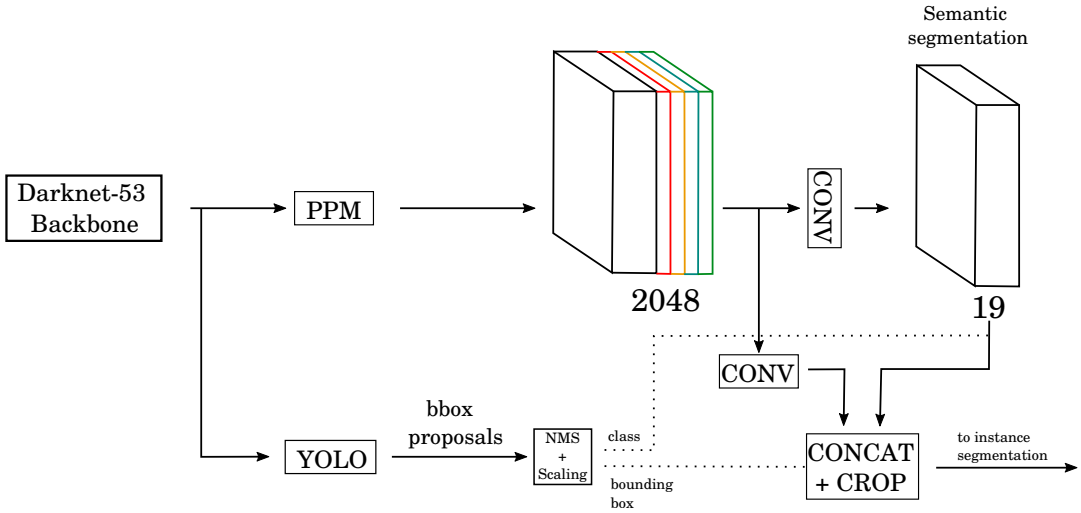
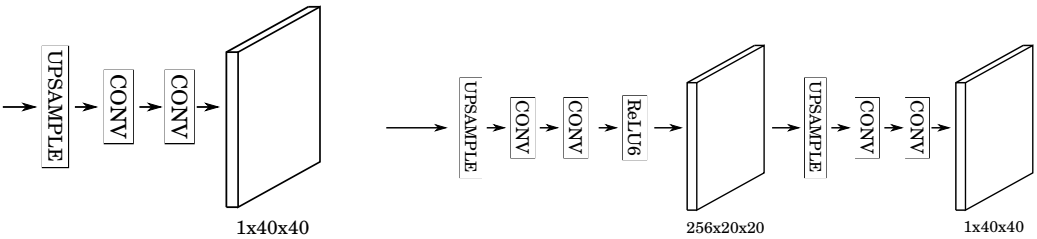


Figure 3.8 – Preprocessing done to the features from the PPM.



(a) Shallow instance segmentation network

(b) Deep instance segmentation network

Figure 3.9 – Proposed instance segmentation networks

For the foreground segmentation task two architectures are proposed, these will be referred as the *shallow* and *deep* architectures. In both cases the instance input features are cropped using the detections from the YOLO head. For the *shallow* architecture, this crop is resized to a (40×40) shape and then processed by a (3×3) convolution followed by a (1×1) convolution. In the *deep* architecture, the crop is resized to a (20×20) shape and passed to two (3×3) convolutions, a ReLU6 [6] non-linearity to then be processed by the same operations of the shallow architecture. Each instance class is determined in both cases from the class indicated in by the object detection branch. The two networks proposed can be seen on fig. 3.9.

The goal of having these two separate architectures is to compare if the same mask quality can be achieved by using less convolutional layers than the proposed on [14]. Also, to test another set of features from the backbone, new feature maps were taken from the 6th, 13th and 38th layers. These features instead of being the ones before a spatial size reduction in the backbone, they are the ones after.

In the interest of training the Pyramid Pooling Module for the instance segmentation task, joint training of the semantic segmentation and instance segmentation networks was done. In the weighted loss function shown in Eq 3.1, $\mathcal{L}_{sem}$ is set as cross-entropy loss and $\mathcal{L}_{inst}$ is binary cross-entropy loss between each instance and its ground truth.

$$\mathcal{L} = \mathcal{L}_{sem} + 0.1 * \mathcal{L}_{inst} \tag{3.1}$$

Training on the Cityscapes Dataset was set at 100 epochs, with early stopping after 10 epochs of either the validation loss not decreasing, mean IoU of the semantic segmentation not increasing or mean IoU of the instances not increasing. Here the data augmentation strategies described in subsection 3.2.1 are used again. Stochastic Gradient Descent is chosen as the optimizer with a base learning rate of 0.1, momentum of 0.9 and the polynomial learning rate scheduler shown in Eq 3.2.

$$\mathrm{lr} = \mathrm{lr}_{base} * \left(\frac{1 - \text{current iter}}{\text{max iter}} \right)^{0.9} \tag{3.2}$$

On tables 3.5 and 3.6 the performance of each combination of the proposed networks and set of features is presented together with their respective inference times on the test and validation sets of the Cityscapes dataset running on a Nvidia Tesla V100 GPU. Here the numbers between brackets specify which features were used in each case.

Architecture	**Train mIoU**	**Validation mIoU**	**time** (ms)
Deep [12,37,62]	**0.7190**	**0.7134**	66.98
Shallow [12,37,62]	0.7044	0.6983	66.64
Deep [6,13,38]	0.7072	0.7010	66.98
Shallow [6,13,38]	0.6822	0.6794	**65.06**

Table 3.5 – Mean Intersection over union for the instance segmentation architectures on images of size (640×640).

Architecture	**Train mIoU**	**Validation mIoU**	**time** (ms)
Deep [12,37,62]	**0.7301**	**0.7184**	153.73
Shallow [12,37,62]	0.7101	0.7012	151.85
Deep [6,13,38]	0.7146	0.7069	155.86
Shallow [6,13,38]	0.6857	0.6814	**151.33**

Table 3.6 – Mean Intersection over union for the instance segmentation architectures on images of size (1024×1024).

According to the results in the comparison tables all alternatives perform similarly with a mean Intersection over Union of 0.7 on average. But as can be seen on fig. 3.10, when comparing qualitatively the results, the combination of the deep architecture and the group of features from the 12th, 37th and 62nd layers yield the highest quality results. It is worth noting that as can be seen on fig. 3.10, these networks fail to detect and segment small instances that are far away.

3.4 Panoptic Segmentation

3.4.1 Merging semantic and instance segmentation maps

According to the definition for the panoptic segmentation task, each pixel an image must be given one label and an instance id, so all overlaps must be resolved between instances and

(a) ground truth (b) Deep [12,37,62]

(c) Shallow [12,37,62] (d) Deep [6,13,38] (e) Shallow [6,13,38]

Figure 3.10 – Instance segmentation ground truth and results for the four tested configurations.

semantic segmentation. To do this, a set of heuristics similar to the rules used in [21] are applied.

First, overlaps between instances are resolved based on their objectness score from the detection branch, where the pixels are assigned to the instance with the highest score. Second, overlaps between instances and semantic segmentation maps are resolved in favor of the instances. Lastly, for stuff classes, regions under a certain area threshold are removed based on the assumption that these classes do not usually have small areas.

On tables 3.7 and 3.8 the scores for panoptic quality (PQ), segmentation quality (SQ) and recognition quality (RQ) can be seen for each of the features-architecture pairs from section 3.3. Each one of these was evaluated on images with sizes (640×640) and (1024×1024) from the Cityscapes Dataset training (T) and validation (V) sets. More extensive evaluation results on the panoptic quality, segmentation quality and recognition quality for the stuff and things classes can be found on appendix B.

Architecture	**PQ** (T)	**SQ** (T)	**RQ** (T)	**PQ** (V)	**SQ** (V)	**RQ** (V)
Deep [12,37,62]	**0.3361**	**0.5640**	**0.5944**	**0.3010**	**0.5553**	**0.5426**
Shallow [12,37,62]	0.3217	0.5491	0.5849	0.2863	0.5373	0.5349
Deep [6,13,38]	0.3070	0.5200	0.5913	0.2759	0.5115	0.5424
Shallow [6,13,38]	0.2891	0.5009	0.5771	0.2556	0.4902	0.5230

Table 3.7 – Panoptic segmentation evaluation on images of size (640×640)

Architecture	**PQ** (T)	**SQ** (T)	**RQ** (T)	**PQ** (V)	**SQ** (V)	**RQ** (V)
Deep [12,37,62]	**0.3692**	**0.5789**	**0.6372**	**0.3385**	**0.5738**	**0.5913**
Shallow [12,37,62]	0.3512	0.5635	0.6232	0.3198	0.5551	0.5796
Deep [6,13,38]	0.3351	0.5311	0.6328	0.3058	0.5256	0.5852
Shallow [6,13,38]	0.3103	0.5071	0.6129	0.2808	0.4996	0.5656

Table 3.8 – Panoptic segmentation evaluation on images of size (1024×1024)

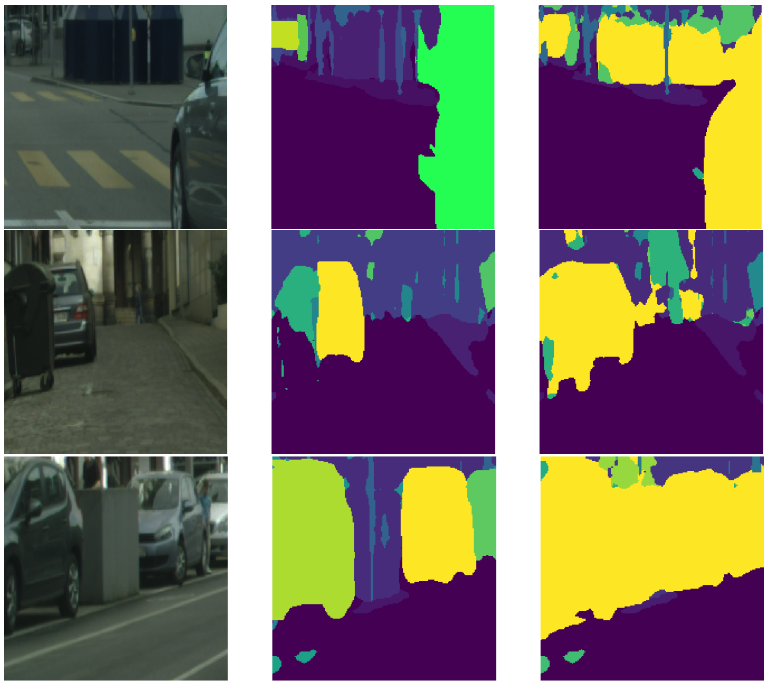


Figure 3.11 – Comparison of segmentation results between the panoptic (center) and semantic (right) segmentation networks.

3.4.2 Refining of Segmentation using Instances

On fig. 3.11, three cropped zones of different images are shown, followed by the their corresponding panoptic and semantic segmentation results. On each one of these images it can be seen that by combining information from the semantic segmentation branch and the instance segmentation head, better segmentation results were be achieved than by just using the semantic segmentation maps. This can be explained by the focus of the instance segmentation network on finding the pixels that belong to the foreground instead of zones of similar textures, which can result on inaccurate pixel predictions like in the segmentation head.

3.4.3 Comparison with state-of-the-art results

With the final YOLO-based panoptic segmentation network defined, a comparison to some of the state of the art methods mentioned on chapter 2 is done in order to compare its performance on the Cityscapes Validation set. PQ^{St} and PQ^{Th} are the separated Panoptic Quality scores for things and stuff classes respectively as explained in section 2.1.

Although the results show that the final network, denoted as YOLO-Panoptic, only outper-

forms JSISNet [9] from the architectures presented in tables 3.9 and 3.10, it is worth noting again that due to time constraints all presented experiments were trained with the Darknet-53 and YOLO layers frozen. It is expected for the panoptic quality scores to increase by performing training for the full network. Results of the final panoptic segmentation network on different images from the Cityscapes dataset and images taken in the city of Grenoble can be seen on appendix C.

Architecture	**Backbone**	**OD / IS method**	**Merging**
JSISNet [9]	ResNet-50	Mask R-CNN	Heuristics
AUNet [23]	ResNet-101-FPN	Mask Attention Module	Attention Operation
Panoptic FPN [21]	FPN	Mask R-CNN	Heuristics
Single Network Panoptic Segmentation [10]	ResNet-50	Mask R-CNN	Advanced Heuristics
DeeperLab [43]	Xception-71 [5]	Keypoint-based method	Majority Vote
Panoptic Deeplab [3]	Xception-71	Class Agnostic Instance Segmentation	Majority Vote
AdapIS [40]	ResNeXT-101	AdaptIS	Heuristics
FPSNet [11]	Resnet-50-FPN	RetinaNet[25]	Panoptic Head
Real-Time Panoptic Segmentation [16]	ResNet-FPN	Parameter-free mask reconstruction	Heuristics
YOLO-Panoptic	Darknet-53	YOLOv3	Heuristics

Table 3.9 – Comparison of between the backbones, object detection (OD) or instance segmentation (IS) methods, merging and pretraining for some of the state-of-the-art networks.

Architecture	**PQ**$(\%)$	$\mathbf{PQ^{Th}}(\%)$	$\mathbf{PQ^{St}}(\%)$	**Time**(ms)
JSISNet	17.6	10.0	23.5	-
AUNet	59.0	54.8	62.1	-
Panoptic FPN	58.1	52.0	62.5	-
Single Network Panoptic Segmentation	42.9	74.3	56.5	590
DeeperLab	56.53	-	-	308
Panoptic Deeplab	63.0	-	-	175
AdapIS	62.0	64.4	58.7	-
FPSNet	55.1	48.3	60.1	114
Real-Time Panoptic Segmentation	58.8	52.1	63.7	99
YOLO-Panoptic	33.8	29.7	45.4	161

Table 3.10 – Comparison of the YOLO-Panoptic Architecture to other Panoptic segmentation networks.

— 4 —
Conclusion and Future Work

A YOLO-based Panoptic Segmentation network was introduced. This network leverages multi-scale features together with a Pyramid Pooling Module to extract semantic features that are used to do semantic and instance segmentation. These instance and semantic segmentation results are then fused following a set of overlap removals in order to comply with the requirements for panoptic segmentation.

For the semantic feature extraction, the use of multiple feature scales was found to give better semantic segmentation results than the use of single feature maps. These features were then tested with three semantic segmentation architectures that leveraged multiscale information differently, here the Pyramid Pooling Module architecture had the highest performance between them on semantic segmentation.

Using the Pyramid Pooling Module and the object predictions from the YOLOv3 detector, an instance segmentation network was proposed for creating masks that separate individual objects. A panoptic segmentation image was created using the masks together with the semantic segmentation features through a process of overlap removal.

In order for the task to be done in real-time, the neural network must minimize the amount of processing done. This can be achieved by using a common feature extraction backbone for multiple tasks such as segmentation and detection.

Since the results presented were done while keeping the backbone and detection module frozen, full training or just fine-tuning of these two parts of the network is recommended as future work. By doing this using semantic and instance segmentation tasks as part of the loss, better performance from the network could be achieved.

To continue with previous work done at the Chroma Team, an extension of the Semantic Occupancy Grid [12] could be done using Panoptic images to give a more precise description of the occupancy grid.

— A —

Semantic Segmentation results per class

Class	**Layer 12** (mIoU)	**Layer 37** (mIoU)	**Layer 62** (mIoU)
Road	0.7311	0.7871	0.7853
Sidewalk	0.2794	0.3992	0.4672
Building	0.5937	0.6452	0.6847
Wall	0.0033	0.0424	0.1309
Fence	0.0559	0.1044	0.1939
Pole	0.1492	0.1689	0.1052
Traffic Light	0.0384	0.1663	0.1728
Traffic Sign	0.2095	0.2828	0.2780
Vegetation	0.6999	0.7388	0.7357
Terrain	0.1971	0.2811	0.2364
Sky	0.7476	0.7576	0.7388
Person	0.1843	0.3420	0.3905
Car	0.4604	0.5540	0.6559
Truck	0.0000	0.0010	0.0352
Bus	0.0020	0.0021	0.2071
Train	0.0015	0.0049	0.1215
Motorcycle	0.0000	0.0020	0.1268
Bicycle	0.0635	0.1695	0.2927

Table A.1 – Mean Intersection over union for each tested group of single features on the training set of the Cityscapes Dataset with size (640×640)

Class	**Layer 12** (mIoU)	**Layer 37** (mIoU)	**Layer 62** (mIoU)
Road	0.7426	0.7914	0.7878
Sidewalk	0.3239	0.4114	0.4618
Building	0.5952	0.6435	0.6800
Wall	0.0042	0.0487	0.1289
Fence	0.0683	0.1191	0.2141
Pole	0.1597	0.1807	0.1109
Traffic Light	0.0418	0.1856	0.1878
Traffic Sign	0.1919	0.2636	0.2706
Vegetation	0.7194	0.7533	0.7454
Terrain	0.2406	0.3117	0.2478
Sky	0.7383	0.7497	0.7312
Person	0.1332	0.2790	0.3190
Car	0.5241	0.6182	0.7180
Truck	0.0000	0.0000	0.0228
Bus	0.0007	0.0017	0.1803
Train	0.0015	0.0053	0.1534
Motorcycle	0.0000	0.0038	0.1535
Bicycle	0.0668	0.1791	0.2953

Table A.2 – Mean Intersection over union for each tested group of single features on the validation set of the Cityscapes Dataset with size (640×640).

Class	**Layer 12** (mIoU)	**Layer 37** (mIoU)	**Layer 62** (mIoU)
Road	0.7103	0.7729	0.7814
Sidewalk	0.2530	0.3702	0.4544
Building	0.5680	0.6260	0.6746
Wall	0.0020	0.0465	0.1446
Fence	0.0468	0.0906	0.1909
Pole	0.1580	0.2018	0.1561
Traffic Light	0.0324	0.1785	0.2432
Traffic Sign	0.2278	0.3165	0.3195
Vegetation	0.7109	0.7431	0.7626
Terrain	0.2141	0.2957	0.2711
Sky	0.7549	0.7634	0.7666
Person	0.2013	0.3559	0.4617
Car	0.4201	0.5237	0.6618
Truck	0.0001	0.0008	0.0246
Bus	0.0022	0.0021	0.1704
Train	0.0014	0.0030	0.0786
Motorcycle	0.0000	0.0028	0.1426
Bicycle	0.0685	0.1877	0.3247

Table A.3 – Mean Intersection over union for each tested group of single features on the training set of the Cityscapes Dataset with size (1024×1024).

Class	**Layer 12** (mIoU)	**Layer 37** (mIoU)	**Layer 62** (mIoU)
Road	0.7276	0.7896	0.7961
Sidewalk	0.2407	0.3858	0.4638
Building	0.5333	0.5936	0.6438
Wall	0.0011	0.0342	0.1277
Fence	0.0396	0.0697	0.1565
Pole	0.1732	0.2184	0.1640
Traffic Light	0.0274	0.1609	0.2387
Traffic Sign	0.2161	0.3160	0.3312
Vegetation	0.7346	0.7623	0.7755
Terrain	0.1516	0.2328	0.2111
Sky	0.7068	0.7109	0.6966
Person	0.2022	0.3403	0.4489
Car	0.3812	0.4831	0.6153
Truck	0.0000	0.0000	0.0321
Bus	0.0026	0.0019	0.1578
Train	0.0008	0.0024	0.0935
Motorcycle	0.0000	0.0002	0.1152
Bicycle	0.0724	0.2235	0.3515

Table A.4 – Mean Intersection over union for each tested group of single features on the validation set of the Cityscapes Dataset with size (1024×1024).

Class	**Segnet multi-feature** (mIoU)	**FPN** (mIoU)	**PPM** (mIoU)
Road	0.8080	0.7787	0.8782
Sidewalk	0.4723	0.3677	0.6281
Building	0.6950	0.5083	0.7621
Wall	0.1377	0.1684	0.3022
Fence	0.2004	0.1843	0.3309
Pole	0.2316	0.1569	0.1550
Traffic Light	0.2578	0.1293	0.2069
Traffic Sign	0.3325	0.1564	0.3143
Vegetation	0.7764	0.7157	0.7687
Terrain	0.2699	0.2865	0.3662
Sky	0.8013	0.7494	0.7593
Person	0.4561	0.3391	0.3299
Car	0.6115	0.5642	0.7428
Truck	0.0646	0.1444	0.3473
Bus	0.2118	0.2436	0.3911
Train	0.1341	0.2249	0.4348
Motorcycle	0.1960	0.2017	0.1398
Bicycle	0.3232	0.2173	0.2985

Table A.5 – Mean Intersection over union for each architecture on the Training set of the Cityscapes Dataset with (640×640) images.

Class	**Segnet multi-feature** (mIoU)	**FPN** (mIoU)	**PPM** (mIoU)
Road	0.8201	0.7885	0.7922
Sidewalk	0.4618	0.3710	0.6120
Building	0.6646	0.4766	0.7355
Wall	0.1006	0.1422	0.2066
Fence	0.1566	0.1379	0.2441
Pole	0.2497	0.1725	0.1608
Traffic Light	0.2370	0.1158	0.1956
Traffic Sign	0.3530	0.1649	0.3308
Vegetation	0.7888	0.7238	0.7706
Terrain	0.2036	0.2261	0.2916
Sky	0.7486	0.6905	0.6844
Person	0.4320	0.3125	0.2962
Car	0.5562	0.5231	0.7226
Truck	0.0730	0.1442	0.3283
Bus	0.2104	0.2643	0.4220
Train	0.1180	0.1358	0.3318
Motorcycle	0.1383	0.1619	0.0682
Bicycle	0.3517	0.2397	0.3195

Table A.6 – Mean Intersection over union for each architecture on the validation set of the Cityscapes Dataset with (640×640) images

Class	**Segnet multi-feature** (mIoU)	**FPN** (mIoU)	**PPM** (mIoU)
Road	0.7971	0.7683	0.8802
Sidewalk	0.4551	0.3480	0.6682
Building	0.6897	0.4903	0.7827
Wall	0.1485	0.1697	0.3699
Fence	0.2014	0.1834	0.4035
Pole	0.2745	0.1709	0.2351
Traffic Light	0.3185	0.1536	0.3001
Traffic Sign	0.3427	0.1734	0.4240
Vegetation	0.7900	0.7350	0.8081
Terrain	0.2904	0.2843	0.4273
Sky	0.8128	0.7390	0.7995
Person	0.5174	0.3631	0.4202
Car	0.6287	0.5603	0.7966
Truck	0.0547	0.1230	0.4382
Bus	0.1625	0.2353	0.4951
Train	0.0996	0.1971	0.5170
Motorcycle	0.2067	0.2133	0.1932
Bicycle	0.3391	0.2166	0.3944

Table A.7 – Mean Intersection over union for each architecture on the Training set of the Cityscapes Dataset with (1024×1024) images.

Class	**Segnet multi-feature** (mIoU)	**FPN** (mIoU)	**PPM** (mIoU)
Road	0.8136	0.7848	0.8041
Sidewalk	0.4683	0.3597	0.6488
Building	0.6581	0.4587	0.7429
Wall	0.1147	0.1439	0.2137
Fence	0.1622	0.1397	0.2926
Pole	0.2989	0.1911	0.2448
Traffic Light	0.3127	0.1416	0.2847
Traffic Sign	0.3544	0.1815	0.4365
Vegetation	0.8037	0.7422	0.8055
Terrain	0.2122	0.2240	0.3240
Sky	0.7654	0.6801	0.7279
Person	0.5015	0.3456	0.3739
Car	0.5745	0.5172	0.7686
Truck	0.0555	0.1144	0.3628
Bus	0.1506	0.2377	0.4918
Train	0.0966	0.1536	0.3632
Motorcycle	0.1695	0.1683	0.1147
Bicycle	0.3702	0.2516	0.4021

Table A.8 – Mean Intersection over union for each architecture on the validation set of the Cityscapes Dataset with (1024×1024) images

— B —

Panoptic Quality results on the Cityscapes dataset

Architecture	$\mathbf{PQ^{St}}$	$\mathbf{SQ^{St}}$	$\mathbf{RQ^{St}}$	$\mathbf{PQ^{Th}}$	$\mathbf{SQ^{Th}}$	$\mathbf{RQ^{Th}}$
Deep [12,37,62]	0.4602	0.4754	0.9687	0.2806	0.7190	0.3860
Shallow [12,37,62]	0.4623	0.4698	0.9843	0.2561	0.7044	0.3596
Deep [6,13,38]	0.4086	0.4185	0.9766	0.2674	0.7072	0.3745
Shallow [6,13,38]	0.4087	0.4133	0.9895	0.2366	0.6822	0.3431

Table B.1 – Panoptic segmentation evaluation for proposed architectures on images of size (640×640) in the train set of the Cityscapes dataset.

Architecture	$\mathbf{PQ^{St}}$	$\mathbf{SQ^{St}}$	$\mathbf{RQ^{St}}$	$\mathbf{PQ^{Th}}$	$\mathbf{SQ^{Th}}$	$\mathbf{RQ^{Th}}$
Deep [12,37,62]	0.4490	0.4666	0.9626	0.2406	0.7134	0.3327
Shallow [12,37,62]	0.4482	0.4571	0.9817	0.2182	0.6983	0.3080
Deep [6,13,38]	0.3996	0.4098	0.9755	0.2297	0.7010	0.3233
Shallow [6,13,38]	0.3968	0.4028	0.9858	0.1995	0.6794	0.2888

Table B.2 – Panoptic segmentation evaluation for proposed architectures on images of size (640×640) in the validation set of the Cityscapes dataset.

Architecture	$\mathbf{PQ^{St}}$	$\mathbf{SQ^{St}}$	$\mathbf{RQ^{St}}$	$\mathbf{PQ^{Th}}$	$\mathbf{SQ^{Th}}$	$\mathbf{RQ^{Th}}$
Deep [12,37,62]	0.4612	0.4756	0.9701	0.3345	0.7301	0.4540
Shallow [12,37,62]	0.4724	0.4763	0.9918	0.2985	0.7101	0.4162
Deep [6,13,38]	0.4063	0.4137	0.9825	0.3146	0.7146	0.4366
Shallow [6,13,38]	0.4031	0.4053	0.9951	0.2752	0.6857	0.3974

Table B.3 – Panoptic segmentation evaluation for proposed architectures on images of size (1024×1024) in the train set of the Cityscapes dataset.

Architecture	$\mathbf{PQ^{St}}$	$\mathbf{SQ^{St}}$	$\mathbf{RQ^{St}}$	$\mathbf{PQ^{Th}}$	$\mathbf{SQ^{Th}}$	$\mathbf{RQ^{Th}}$
Deep [12,37,62]	0.4542	0.4726	0.9618	0.2969	0.7184	0.4091
Shallow [12,37,62]	0.4630	0.4672	0.9914	0.2646	0.7012	0.3736
Deep [6,13,38]	0.4006	0.4092	0.9791	0.2779	0.7069	0.3893
Shallow [6,13,38]	0.3938	0.3962	0.9943	0.2421	0.6814	0.3512

Table B.4 – Panoptic segmentation evaluation for proposed architectures on images of size (1024×1024) in the validation set of the Cityscapes dataset.

— C —

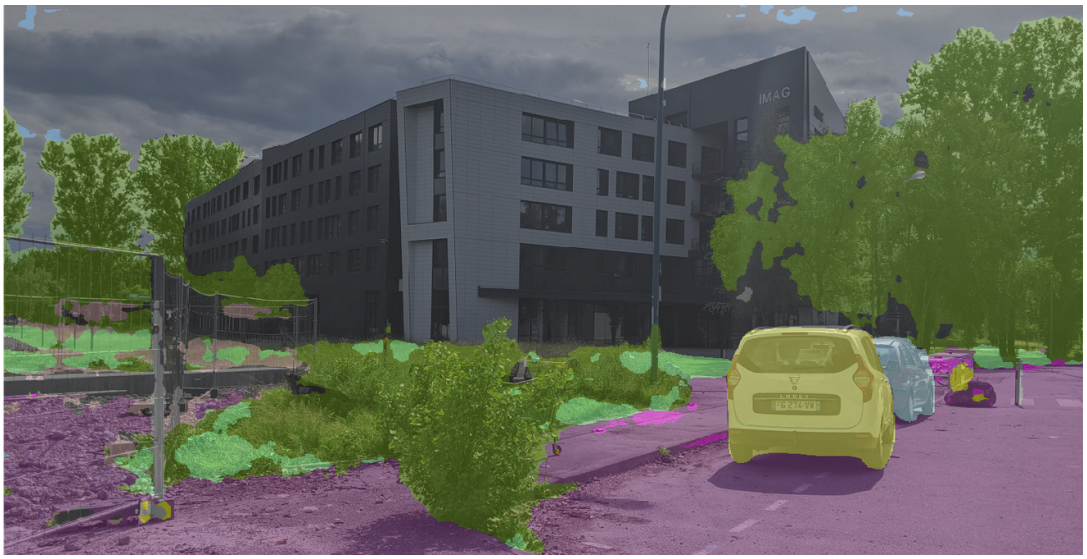
Panoptic Segmentation Results

C.1 Results on images from the Cityscapes Dataset

These images are best viewed with digital zoom

SPIEL
PARADIES

C.2 Results on images from Grenoble



Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[3] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[6] Torch Contributors. Relu6, 2019.

[7] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[9] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Panoptic segmentation with a joint semantic and instance segmentation network. *arXiv preprint arXiv:1809.02110*, 2018.

[10] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Single network panoptic segmentation for street scene understanding. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[11] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Fast panoptic segmentation network. *IEEE Robotics and Automation Letters*, 5(2):1742–1749, 2020.

[12] Özgür Erkent, Christian Wolf, and Christian Laugier. Semantic grid estimation with occupancy grids and semantic segmentation networks. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1051–1056. IEEE, 2018.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[14] 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.

[15] 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.

[16] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. *arXiv preprint arXiv:1912.01202*, 2019.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[19] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[22] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9404–9413, 2019.

[23] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7026–7035, 2019.

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[26] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6181, 2019.

[27] 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 *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018.

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[30] Panagiotis Meletis and Gijs Dubbelman. Training of convolutional networks on multiple heterogeneous datasets for street scene semantic segmentation. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1045–1050. IEEE, 2018.

[31] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[32] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017.

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[34] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[35] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[36] 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.

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[39] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[40] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019.

[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.

[42] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.

[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. *arXiv preprint arXiv:1902.05093*, 2019.

[44] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.