Panoptic segmentation in video

Project Report

Kristján Már Gunnarsson

Aalborg University
Electronics and IT

Title:
Panoptic segmentation in video

Theme:
Panoptic Segmentation

Project Period:
10^{th} semester, Spring 2023

Project Group:
1047

Participant(s):
Kristján Már Gunnarsson

Supervisor(s):
Andreas Møgelmose
Anders Skaarup Johansen

Page Numbers: 28

Date of Completion:
June 2, 2023

Abstract:

This project presents a study and application of video panoptic segmentation using Detectron2 as a 4^{th} semester project at VGIS. A challenging task that requires the coherent prediction of both object-level and pixel-level semantics in a video frame. Detectron2, with its flexibility and extensibility, has been employed as the foundation for the project, further augmented with ResNet101, a deep residual network, and Feature Pyramid Network (FPN), a structure that allows multi-scale object detection in an image. The project illustrates the robust capabilities of this combination for video panoptic segmentation, particularly in its ability to handle diverse scales and complex scenes in a video.

Contents

Preface **v**

1 Introduction **1**
1.1 Capture One . . . 1
1.2 Study Case . . . 2
1.3 Initial Problem Statement . . . 2

2 Problem Analysis **3**
2.1 Introduction . . . 3
2.2 Image Segmentation . . . 3
2.2.1 Capture One requests . . . 4
2.2.2 Other real world applications . . . 5
2.3 State-of-the-art methods for Panoptic Segmentation . . . 6
2.4 Datasets . . . 8
2.5 Final Problem Statement . . . 9

3 Project Scope **10**

4 Technical analysis **11**
4.1 Detectron2 . . . 11
4.1.1 Region Proposal Network . . . 12
4.1.2 ROI Head . . . 14
4.1.3 Semantic Segmentation Head . . . 15
4.1.4 Inference . . . 16
4.1.5 Dataset for training . . . 16

5 Tests & Results **17**
5.1 Detectron2 results . . . 17
5.1.1 Learning Rate and loss . . . 17
5.1.2 Segmentation Quality, Recognition Quality and Panoptic Quality . . . 18
5.1.3 Showcasing the results from the model . . . 19
5.2 Mask borders . . . 20

6 Discussion **24**
6.1 Evaluation of requirements 24
6.2 Future work 25

7 Conclusion **26**

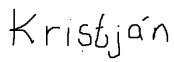
Bibliography **27**

Preface

Reading manual
This report is designed to be read digitally and is therefore prepared in a one-page format. Citations to the left of punctuation marks refer to the citation's particular sentence. When a citation appears to the right of a punctuation mark, it refers to the entire paragraph.

Documentation
https://drive.google.com/drive/folders/1cCP6Ua35$_Y$*wFj*4*kprTxw*6*PYSjo*9*NSePX*?*usp* = *sharing*

Aalborg University, June 2, 2023



Kristján Már Gunnarsson
<kgunna18student.aau.dk>

Chapter 1

Introduction

Within the field of computer vision, panoptic segmentation is still a relatively new concept. It is a unified framework that combines instance segmentation and semantic segmentation. The idea behind panoptic segmentation was introduced in 2018 along with the publication of a paper titled "Panoptic Segmentation" by Alexander Kirilov, et al [1]. Before this publication, instance segmentation and semantic segmentation were usually treated as separate tasks. The concept of panoptic segmentation can be seen as an extension to instance segmentation and semantic segmentation, creating a new task for computer vision.

1.1 Capture One

Capture One is a Danish software company that specializes in developing high-end image editing software for professional photographers and other creative professionals. The company was founded in 1992 as a division of Phase One, a leading manufacturer of digital medium format cameras and lenses. Today, Capture One is a standalone brand with a diverse product line that includes both desktop and mobile applications for editing and organizing images. Capture One's flagship product is Capture One Pro, a powerful image editing and management software that offers advanced features for tethered shooting, color grading, and image processing. The software is known for its exceptional image quality, robust workflow tools, and customization options that cater to the specific needs of individual users. In addition to Capture One Pro, the company also offers a range of other products including Capture One for Fujifilm, a version of the software optimized for Fujifilm camera users, and Capture One for Sony, a version optimized for Sony camera users. Capture One also offers mobile apps, including Capture One Express for mobile and Capture One Pro for iPad, which allow users to edit and organize images on-the-go. Capture One is committed to providing its users with exceptional support and resources, including online tutorials, webinars, and a community forum where users can connect and share their work. The company also offers personalized support through its Capture One Ambassador program, which provides expert advice and guidance to users. Capture One

has established itself as a leading provider of high-end image editing software, catering to the needs of professional photographers and other creative professionals. Its focus on exceptional image quality, robust workflow tools, and personalized support has made it a popular choice among users around the world. [2]

1.2 Study Case

The project is part of a research initiative called AI Color Fashion provided by Capture One. The primary goal of the project is to expand the capabilities of Capture One to include video. One of Capture One's specialties is their ability to produce accurate colors in their photo editing software. A feature of the software is color adjustment, which typically occurs on a per-object basis. For example, a person's skin, clothes, and background can be masked and then adjusted accordingly. However, this is a manual process for the editor, making it impractical for anything other than still images. This is where panoptic segmentation comes in, as it can automate the process of creating masks in the image frames of a video.

1.3 Initial Problem Statement

To initialize the project an initial problem statement has been made, which will form the basis for the upcoming problem analysis:

How can the process of masking different objects in a scene be automated frame by frame?

Chapter 2

Problem Analysis

2.1 Introduction

The upcoming section will cover Image segmentation and its subtasks, semantic segmentation, instance segmentation, and panoptic segmentation. Furthermore, the section will explore state-of-the-art methods for panoptic segmentation, as well as their Panoptic Quality scores, for which the measurement will be explained. After that, leading datasets that are available for the task at hand will be looked at. Finally, the section will conclude with a project scope that discusses the requirements and delimitations for the project.

2.2 Image Segmentation

In computer vision, image segmentation is the process of dividing an image into multiple segments, or regions, each of which corresponds to a different object or part of the image. There are two main types of image segmentation: semantic segmentation and instance segmentation.

Semantic segmentation is a technique that involves assigning a label to each pixel in an image based on the category of object it belongs to. In other words, each pixel is labeled with a class label that corresponds to the type of object or "stuff" it represents. "Stuff" refers to regions in an image that do not have a well-defined boundary or shape. Semantic segmentation is typically used for identifying "Stuff" in an image because this type of object does not have a well-defined boundary. Instead, they are typically defined by their texture or material properties. For example, in a street scene, semantic segmentation could be used to identify areas of the image that correspond to road, sidewalk, sky, buildings. [1] [3]

On the other hand, instance segmentation is a technique that involves identifying and delineating every individual object in an image. This technique combines object detection with semantic segmentation, allowing for the identification and classification of each

object in an image, while also delineating its boundaries with a segmentation mask or bounding box. Therefore, instance segmentation is typically used for "things" which have clear boundaries and can be easily distinguished from one another. For example, in a street scene, instance segmentation could be used to identify and segment individual cars, pedestrians, and bicycles. [1] [3]

The main difference between semantic and instance segmentation is in how they treat objects of the same class. In semantic segmentation, all objects of the same class are treated as a single entity and given the same label. In instance segmentation, each individual object of the same class is separately identified and segmented. This allows for more detailed information to be extracted about each object within an image. This can be seen in Figure 2.1.

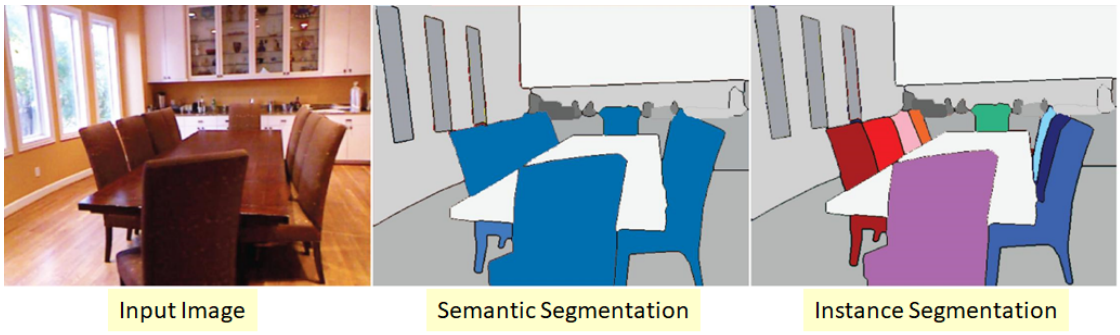


Figure 2.1: The difference output results between semantic segmentation and instance segmentation.[3]

Panoptic segmentation is a more recent technique that aims to unify the strengths of both semantic and instance segmentation. It involves labeling each pixel in an image with a class label (for "stuff" regions) or instance ID (for individual objects), with the additional feature of distinguishing between "stuff" and "things". "Things" refers to objects that have a well-defined boundary or shape, while "stuff" refers to regions that do not have a well-defined boundary or shape. This technique is particularly useful in complex scenes where there are multiple objects and "stuff" regions that need to be identified and delineated. [1]

2.2.1 Capture One requests

Outside of being able to do what the current state-of-the-arts methods can do with panoptic segmentation, the following is useful for Capture One's implementation. In addition to being able to mask objects in frames using panoptic segmentation, Capture One would like the solution to be able to operate on images and videos in 8k resolution. This makes panoptic segmentation a bit more challenging mainly because of two reasons. The first being that with higher resolution the number of pixels that need to be processed by the panoptic segmentation model also increases. The results of this is increased computational complexity, which will make processing in real-time challenging. The second challenge

with increased resolution is decreased context information. The objects in a scene can become smaller and densely packed. This can lead to difficulties differentiating between the objects and stuff. Additionally, the context information that helps identify the objects in a scene becomes more scattered and less informative, which may lead to inaccurate boundaries. [4]

As it is a software used primarily by photographers, they will in many cases want to be able to have an influence on what will be produced by the model. One way of having a person in the loop of what will be produced by the model could be through Interactive segmentation. This means that the user could guide the model, by allowing the user to select regions of interest, provide more context or have the ability to refine the segmentation results produced by the model at the end.

2.2.2 Other real world applications

Panoptic segmentation is useful in robotics applications for a wide range of tasks such as object detection, localization, and manipulation. In robotics, robots need to be able to perceive and interpret their environment accurately to perform tasks autonomously. Panoptic segmentation can provide a detailed understanding of the environment by detecting and segmenting all objects, including both things and stuff, in a scene. This helps robots to navigate in complex environments and perform tasks efficiently. For example, a robot equipped with a panoptic segmentation model can detect and segment different objects in a cluttered workspace and plan its path to avoid collisions with obstacles. [1]

Panoptic segmentation is a useful tool for autonomous driving as it can help self-driving cars to detect and recognize different objects in the environment, including vehicles, pedestrians, and road markings. By providing pixel-level segmentation of all objects, including both things and stuff, in a scene, panoptic segmentation can enable self-driving cars to make more accurate and safe decisions. For example, a self-driving car can use panoptic segmentation to identify pedestrians and other vehicles in its surroundings and adjust its speed and trajectory accordingly to avoid collisions. [1]

Panoptic segmentation can be used in augmented reality applications to provide a more realistic and immersive experience. In augmented reality, virtual objects need to be accurately overlaid on real-world objects to create a seamless experience. Panoptic segmentation can help achieve this by providing accurate and detailed segmentation of real-world objects. For example, a user can point their smartphone camera at a real-world object, and a panoptic segmentation model can accurately detect and segment the object. This allows virtual objects to be overlaid on the real-world object in a way that is realistic and visually appealing. [1]

Modern image segmentation techniques are based on deep neural networks where the following section will briefly showcase handpicked state-of-the-art methods.

2.3 State-of-the-art methods for Panoptic Segmentation

In this section a brief overview of a handful state-of-the-art methods will be described. The following state-of-the-art methods have been chosen as they have good Panoptic Quality (PQ) scores, which is a widely used standard for evaluating the performance of panoptic segmentation models, which will be described as well..

Panoptic Quality
To be able to determine the performance of a model, a measurement has been made for panoptic segmentation. The measurement combines both instance and semantic segmentation tasks in a unified way to evaluate panoptic segmentation. It measures both the recognition and localization of each individual object in an image or video, as well as the correct labeling of all pixels. The PQ score is needed as panoptic segmentation combines instance and semantic segmentation together, so for semantic predictions there are no clear cut confidence scores.

$$PQ = \frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP|} \cdot \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}[5] \tag{2.1}$$

In equation 2.1 TP refers to True Positives, which are the correctly predicted segments. FP refers to False Positives, which are the predicted segments that do not correspond to a ground truth segment. FN refers to False Negatives, which are the ground truth segments that were not detected. The function IoU(p,g) is the Intersection over Union between the predicted segment p and the ground truth segment g. The left side of the equation is the same as segmentation quality and the right side is the same as recognition quality. The segmentation quality is evaluating how closely matched the segments are with their ground truth. When this value comes close to 1 it indicates that the predicted segments are closely matched with their ground truths. It however does not take into account any of the bad predictions. This however, the recognition quality takes into account. The combination of precision and recall attempts to identify how effective the model is at getting a prediction right. [1] [5]

Unified Panoptic Segmentation Network (UPSNet)
UPSNet is a neural network architecture that unifies instance segmentation and semantic segmentation tasks into a single framework. It achieves this by using a shared feature encoder to extract features from an input image and then applying two parallel prediction branches. One branch predicts the class label for each pixel in the image (semantic segmentation), while the other predicts a binary mask for each object instance in the image (instance segmentation). UPSNet is designed to handle both things at once, and can be

trained end-to-end. [6]

Video Panoptic Segmentation Network (VPSNet)
VPSNet is based on the same unified panoptic segmentation framework as UPSNet, but it includes several modifications to handle video inputs. Specifically, VPSNet adds a temporal consistency module that leverages information from neighboring frames to improve the accuracy of the segmentation predictions. Additionally, VPSNet includes a tracking module that helps to ensure that objects are consistently labeled across multiple frames. This is achieved by using a convolutional neural network to learn a mapping between the current and its neighboring frames, which maximizes the similarity between the segmentation predictions. Overall, VPSNet represents a significant advance in the field of video segmentation, as it is able to segment all objects in every frame of a video while maintaining temporal consistency and object tracking. [7]

Panoptic-Deeplab
One of the main strengths of Panoptic-Deeplab is its ability to perform panoptic segmentation in a fully convolutional manner. This means that the model can process input images of arbitrary sizes and produce segmentation maps of the same size without requiring any post-processing steps. This is in contrast to previous state-of-the-art methods that relied on cropping or resizing the input images to fixed sizes. Panoptic-Deeplab achieves this fully convolutional operation by using dilated convolutions and a feature pyramid network (FPN) to capture multi-scale contextual information. The model also incorporates a spatial attention mechanism that enables it to selectively attend to informative regions of the input image. The Panoptic-Deeplab model is currently one of the leading models for panoptic segmentation, achieving state-of-the-art results on several benchmark datasets.[8]

EfficientPS
EfficientPS is notable for its high accuracy and efficiency, achieved through several innovations in its architecture, including the use of a feature fusion module, a lightweight attention mechanism, and a new type of feature pyramid network. These innovations allow EfficientPS to achieve state-of-the-art results on several benchmarks while being faster and more memory-efficient than previous models. However, training EfficientPS requires significant computational resources, and the model may struggle with rare or unusual objects that are not well-represented in the training data. [9]

Detectron2
Detectron2 Is a PyTorch based modular object detection library. Detectron2 is flexible and extensible and can provide fast training. The library includes high-quality implementations of state-of-the-art object detection algorithm. The Detectron2 architecture consists of a backbone network for feature extraction, a feature pyramid network for multi-scale feature representation, and task-specific heads for different computer vision tasks. The

open source library has been used to achieve state-of-the-art performance on a wide range of benchmark datasets within panoptic segmentation. [10]

Panoptic FCN
This solution is based on Detectron2. The framework of Panoptic FCN (Pano-FCN) is mainly consisting of three components. A kernel generator, kernel fusion and a feature encoder. The task for the kernel generator is designed to locate and classify thing centers along with stuff regions. The kernel head in each stage generates kernel weights for both things and stuff. Then the kernel fusion merges kernel weights with the same identity from the different stages. The feature encoder encodes the high-resolution features with details. Additionally, the backbone for the model consists of a FPN. [11]

Detectron2 will be looked further into in this project. As was mentioned a bit the Detectron2 is a modular extensible design, meaning that a user can plug custom module implementation into most object detection systems, thus the specific modules used to create the architecture which will complete the panoptic segmentation will be discussed in the technical analysis.

2.4 Datasets

The dataset which is chosen plays a crucial role in the development and evaluation for panoptic segmentation solutions. As is the case with most neural networks, the network is trained using datasets which provide input/output examples on specific classes. Therefore it is important to know what the classes within the datasets are to be certain of their relevance for the specific use case of the method. For a dataset to be viable for doing panoptic segmentation it needs pixel-level annotations. This means that for each image or video that it has, the dataset should provide annotations that indicate which parts of the image correspond to objects and which parts correspond to stuff on the pixel-level. Furthermore, the dataset should define a set of object and stuff categories. These need to be well defined as to be able to assign each pixel to a single category. The dataset should also include instance-level annotation which can identify individual instances of each object category.. Commonly used datasets for the state of the art methods are as follows: Common Objects in Context (COCO), Cityscapes, ADE20K and Mapillary Vistas.

COCO is one of the largest and most diverse datasets for panoptic segmentation. It contains over 328k images of complex scenes with 80 object categories and 53 stuff categories. COCO is commonly used as a benchmark dataset for object detection, instance segmentation, and panoptic segmentation. [12]

Cityscapes is a dataset of urban scenes, designed for autonomous driving applications. It contains over 5k high-resolution images of street scenes with 8 stuff categories (such as

road, sidewalk, building) and 19 object categories (such as car, person, bicycle). [13]

ADE20K is a large-scale dataset for scene understanding, including semantic and instance segmentation. It contains over 20k images with 150 stuff categories and 2,000 object categories. The images represent diverse indoor and outdoor scenes, including natural and man-made environments. [14]

Video Panoptic Segmentation in the Wild (VIPSeg) is a dataset consisting of 3536 videos and 84750 frames with pixel-level panoptic annotations, with a wide range of real-world scenarios and categories. The length of the videos vary from 3 to 10 seconds. In total there are 124 categories, where the split is 58 things and 66 stuff classes. The dataset also consists of diverse scenes covering also indoor scenes, so it is not only focused on street view scenes. [15]

The dataset which will be used further on will be the COCO dataset. COCO is one of the most commonly used dataset for video panoptic segmentation and is widely used as a benchmark for models. This is also done since a dataset including just people and the features of people such as hair, as well as clothes has not been found, which would have been the most relevant for Capture One's use case. However, as it is machine learning that is being used, then it should possible to use the knowledge gained from using COCO and apply that to other datasets.

2.5 Final Problem Statement

From what was learned throughout the problem analysis the problem statement can be narrowed to the following:

How can Detectron2 be used to create video panoptic segmentation which takes into account the necessities desired by Capture One?

Chapter 3

Project Scope

Within the project scope a set of requirements can be set for expectations from the solution. These are both from the study case given and by looking at it from the problem analysis. The requirements set are as follows.

Requirements

- 1. Working Panoptic segmentation for video
- 2. A panoptic quality score of 50 >=
- 3. Different type of borders for the mask, depending on class.
- 4. Multiple masks for a single pixel

Delimitations
There is a delimitation with the hardware used for the project. As the personal computer used does not contain an NVIDIA graphic card. This means that doing machine learning locally will not allow for the usage of CUDA, which allows for GPU processing, thus speeding the process of machine learning significantly by having more cores available for computations. The workaround for this was to use Google Colab Notebooks. Through this, it is possible to use a remote GPU that has access to CUDA.

Chapter 4

Technical analysis

From the problem analysis several state-of-the-art methods were looked at as well as their performances along with various datasets. By using these factors as well as looking at other implementations, it was decided to proceed with Detectron2 as it both has okay performances and is implemented eusing PyTorch, which is familiar. Furthermore, Detectron2 probably has the most documentation out of the ones that were looked into, which is helpful.

4.1 Detectron2

The model used for Detectron2 to be able to perform panoptic segmentation is based on Base (faster) R-CNN-FPN. The backbone of the model consists of a ResNet and a FPN. In the used Detectron2 model ResNet-101 is utilised, which is pre-trained on ImageNet and serves to extract a hierarchy of features. ResNet models are favored for their ability to handle a wide range of visual complexities with the help of residual blocks and skip connections that allow for training of deep neural networks. The number behind the ResNet indicates the depth of the model. In this case the depth is 101. For ResNet models the depth is the number of layers in the model, both convolutional layers and fully-connected (fc) layers. The ResNet is among one of the reasons as to why the Mask-RCNN from Detectron2 is efficient and its throughput time is higher for image per second compared to other models [16]. The Mask-RCNN is the one most responsible for the instance segmentation. The output from the ResNet-101 network is fed into the FPN. Here it constructs a multi-scale, pyramidal hierarchy of features. The idea behind FPN is to exploit the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with high-level semantic feature maps. By combining low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections, FPN is able to create feature maps that are rich in semantic information at all scales. The downscaling is done by using several different strides. In this specific model, the FPN will provide five feature maps, where the height and width of

these feature maps will be 4, 8, 16, 32 and 64 times smaller than the original frame. [4] [17] [18]

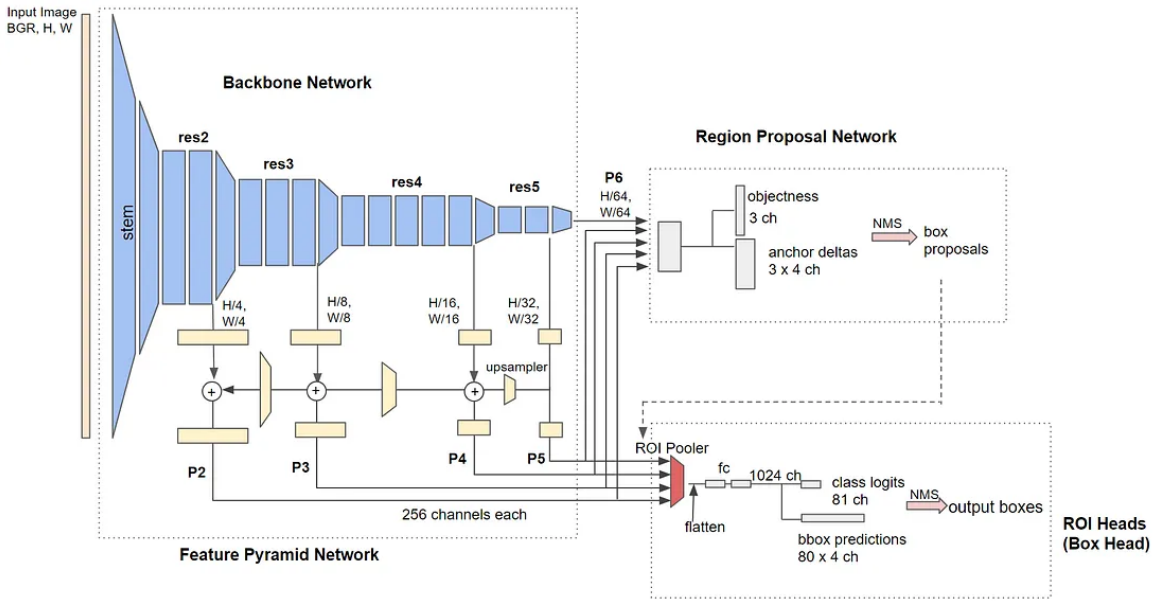


Figure 4.1: The architecture of the Detectron2 implementation. Showing the three main building blocks used to achieve its results. It is however missing the semantic segmentation head, needed to get semantic labels. [18]

4.1.1 Region Proposal Network

These feature maps are fed into the Region Proposal Network (RPN). The main objective of the RPN is to identify regions within the image that are likely to contain an object. This is important as this can reduce the number of regions that need to be examined significantly, making the model more efficient in terms of speed. Inside of the RPN component there is a neural network and non-neural network functionalities. The RPN head, which can be seen on Figure 4.2 consists of three convolution layers. The first step is to lay out a series of anchor boxes. These are predefined bounding boxes of various scales and aspect ratios that cover the entire image. The goal of the anchor boxes is to provide a set of reference points that the RPN can use to identify potential objects of different shapes and sizes. The RPN applies a small neural network to this feature map through a sliding window. This small network is slid across the feature map, and at each spatial position, it is connected to all possible anchor boxes. For each anchor box, the RPN computes an objectness score and a bounding box regression. The objectness score is used to measure the probability that the anchor box contains an object of any kind. Its sole job is to determine if the box has an object or is part of the background. There is also a bound box regression, which

provides four values. These values are used to adjust the coordinates of the anchor box in order to make sure that if there is an object in the box, then the box will be adjusted to fit the object more accurately. [19]

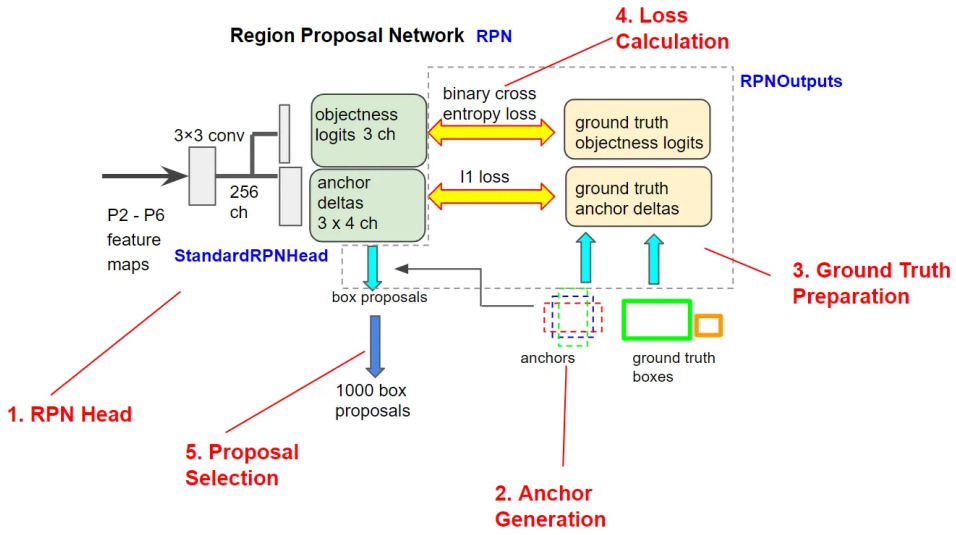


Figure 4.2: Closer look at the RPN, which is a core component of the Base R-CNN-FPN model. [19]

From calculating all of the objectness scores an objectness map is created. By visualising the objectness map it can be seen from the previous output from the FPN that the smaller objects can be found from the P2 and P3 outputs and the larger objects from P4 to P6. This can be seen in the following example on Figure 4.3
This showcases the strength of using a multi-scale network which can detect various sizes that single-scale networks might miss.

When the objectness scores and bounding box regressions have been calculated for all anchor boxes, the model goes on to the next step which is to apply Non-Maximum Suppression (NMS). The goal of using NMS is to reduce the amount of proposals for objects. The NMS technique does so by removing anchors that have significant overlap and only keeping bounding boxes that have the highest objectness scores. This makes sure that the model does not have multiple detections of the same object. The final step for the RPN is to go through the Region Proposal. Each region proposal is a candidate bounding box that may contain an object. These region proposals, refined by the bounding box regressions and filtered by non-maximum suppression, are passed along to the next stage of the model for further processing [19]

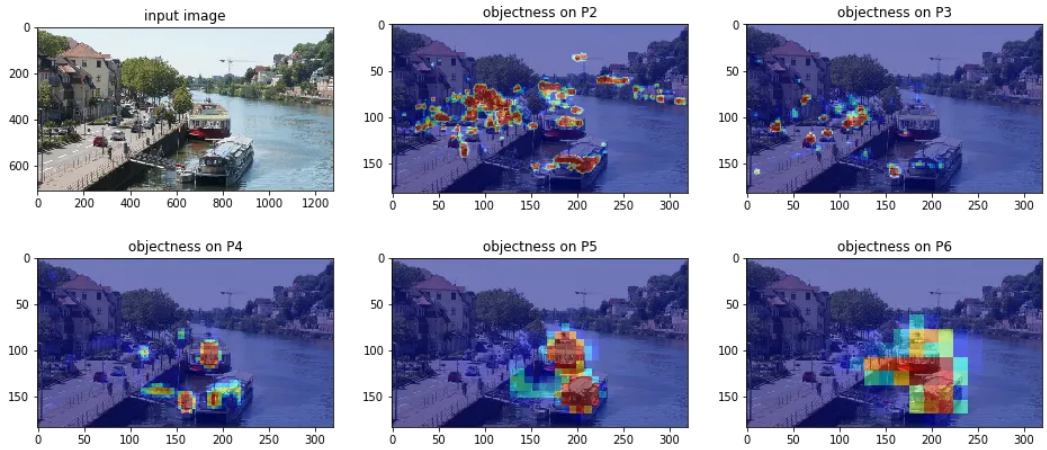


Figure 4.3: It can be seen for P2 and P3 that smaller objects are being detected for the objectness score and for P4 to P6 only the larger objects are being detected. [19]

4.1.2 ROI Head

In Detectron2, the ROI Head is responsible for predicting class labels, bounding box coordinates, and in the case of Mask R-CNN, object instance masks.

ROI Pooling
First, an ROI is defined in the input feature map. The ROI is defined by the four corner points that establish the bounding box. This bounding box is known from the previous RPN output. The ROI is then divided into a grid of smaller sections, typically of size HxW (where H and W are predefined, for Detectron2 this is 7x7). For each of these grid cells, max pooling is performed. This means that the maximum value within each section is taken as the representative of that section. Max pooling helps to reduce the dimensionality and to mitigate the effects of small translations in the image. After max pooling is performed on each section, the output is a fixed-size HxW feature map. This fixed-size output is what enables the subsequent fully connected layers in the network to process the ROIs, regardless of their original sizes. While ROI Pooling is effective, it can cause some spatial misalignment due to the quantization (rounding down) operation. ROI Align fixes this by using bilinear interpolation to accurately map the extracted features back to the original image, preserving exact spatial locations. This precision is particularly beneficial for tasks such as instance segmentation, where the exact contours of an object need to be determined. [20]
Box Head
After the region proposals have been normalized to a fixed size by ROI Pooling or ROI Align, they are forwarded to the ROI Head. The ROI Head consists of several fully connected layers that carry out further processing on the proposals. Before going into the FC layer the input is flattened into 12544 channels and then fed into the FC layers. Next is the

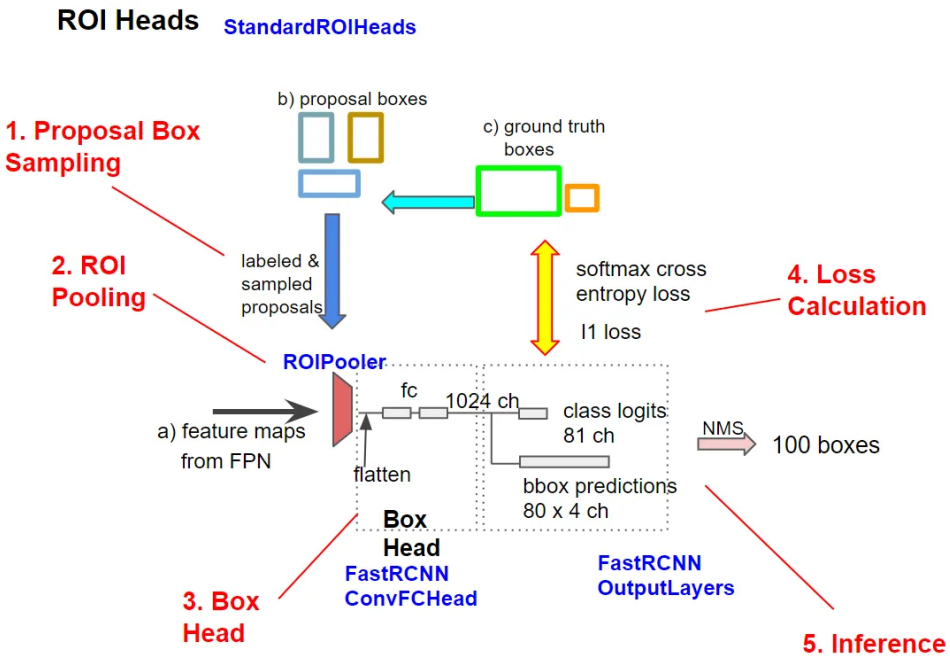


Figure 4.4: The structure of the ROI Head used in the Detectron2 model. [20]

process of calculating the loss for the outputs during training. [20]

Loss Calculation
For the Detectron2 model here, there are two types of loss that need to be calculated. The first one is for the classification. This is typically a softmax cross-entropy loss that measures the error between the predicted class probabilities and the true class labels. It encourages the model to assign a high probability to the correct class for each object. The other second one is for the bounding box regression loss. This loss measures the error between the predicted bounding box coordinates and the ground-truth bounding box coordinates. It encourages the model to predict bounding boxes that closely match the actual locations and sizes of the objects. In Detectron2, this is typically a Smooth L1 loss, which is less sensitive to outliers than the mean squared error loss. [4]

4.1.3 Semantic Segmentation Head

This is a fully convolutional network head that outputs a semantic label for every pixel in the image. It's used for "stuff" classes. In the final panoptic segmentation output, each pixel in the image is assigned a semantic label (from the semantic segmentation head) and an instance ID (from the instance segmentation head). The instance ID is used to distinguish between different object instances of the same class. [4]

4.1.4 Inference

For panoptic inference, the panoptic output format requires each output pixel to be assigned a single class label or empty and instance id, which for stuff classes is ignored. As the instance and semantic segmentation outputs from the model may overlap, the NMS can again be utilized to resolve overlaps. Additionally, the post-processing will take favour to instance segmentation outputs over semantic segmentation. At last it will remove any stuff regions labeled "other" or under a given threshold for the confidence score. [4]

4.1.5 Dataset for training

The dataset which will be used for training in this project is the COCO dataset. More specifically it is the COCO dataset from 2017. Looking at an analysis of 5000 randomly chosen images from the COCO dataset made by Roboflow, shows that the distribution is quite uneven, where the five most represented classes are "person", "car", "chair", "book" and "bottle" [21]. However, as it is from randomly chosen images, it could either be showing a class to be too well represented or low according to what it would be in the overall view. Furthermore, not all classes are represented from the random images. However, it does quickly show that "person" is significantly more represented than the rest. A possibility that could be used to counteract the uneven distribution if the full one was known, would be to set weights when training on it. However, as a full distribution of the dataset has not been found, this will not be applied.

Chapter 5

Tests & Results

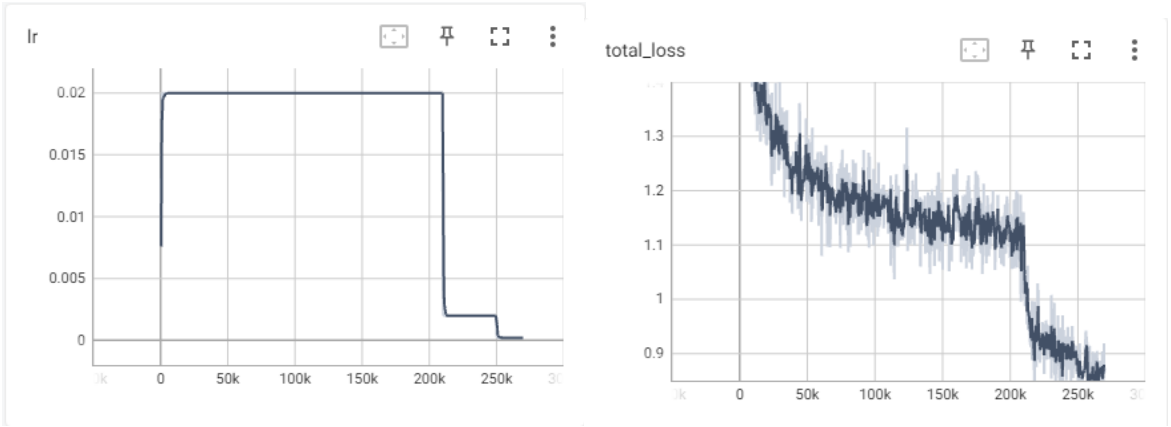
This chapter will be describing the results from the Detectron2 training, as well as showcase results showing panoptic segmentation. In the report it will only showcase images,

5.1 Detectron2 results

In this section, the results from the Detectron2 training will be shown as well as certain metrics during training. These will be the learning rate throughout training, the total loss for the model, the segmentation quality (SQ), the recognition quality (RQ) and the Panoptic Quality. In general for the training, parameters such as the batchsize especially are set to be very low. The reason for this is due to the GPU memory being quite low for such a large dataset and complex model. For the evaluation and gathering the metrics, all of it was done on a validation set.

5.1.1 Learning Rate and loss

In the configuration for the ResNet101-FPN, the learning rate scheduler is a step based one. The first step is set to take place 210000 iterations in and the second time at 250000 iterations. In total there are 270000 iterations to go through. The learning rate starts at 0.02, and with each step it gets divided by ten. So at 210000 iterations the learning rate becomes 0.002 and at 250000 iterations it becomes 0.0002. Comparing the figure showing the learning rate with the total loss of the model. It can be seen at around 200000 iterations the loss starts to flatten out. Applying a lower learning rate helps the model get its total loss down, getting closer to 0, with its lowest loss reaching 0.8. All of this can be seen on Figure 5.1.1
The total loss is a combination of the losses mentioned in the architecture in Section 4.1. The loss has been calculated after each epoch during the training process. The losses that are included in the total loss are classification loss, box regression loss and mask loss. Some of these losses occur multiple times during the architecture of the model, for example the



(a) The learning rate for the model during training.

(b) The total loss for the model during training.

Figure 5.1

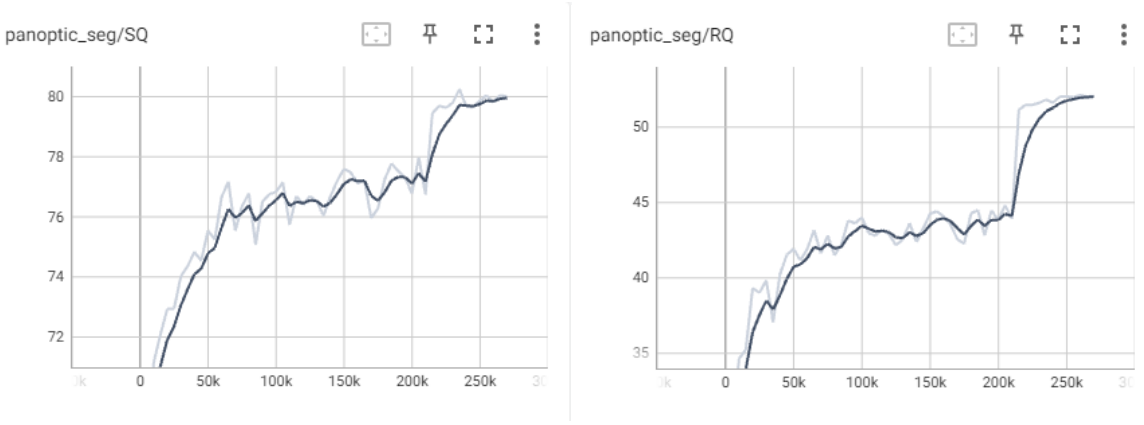
classification and bounding box losses happen throughout the FPN and also in the ROI Head.

5.1.2 Segmentation Quality, Recognition Quality and Panoptic Quality

As was shown in an earlier chapter in Equation 2.1, the equation for the panoptic quality was shown. In this equation it was shown in order to calculate the PQ, the SQ and RQ could be used to do so. Therefore, it is relevant to show. The Figure 5.1.2 shows that these values are upwards around 80 for the SQ and 52 for the RQ. These values need to be in the range 0-1. This means that when using the equation for PQ, it would be

$$PQ = SQ \cdot RQ \longrightarrow 0.80 \cdot 0.52 = \sim 42 \tag{5.1}$$

This can also be seen to be the case when looking at the PQ scores during training.
When looking at it, the lower RQ score might indicate that the model is struggling with false positives or false negatives. This however, may be a result of various factors and does indicate room for improvement for the panoptic segmentation. With a model ready it is possible to try to input videos into the model and see the results.



(a) Segmentation Quality of the model.

(b) Recognition Quality of the model.

Figure 5.2

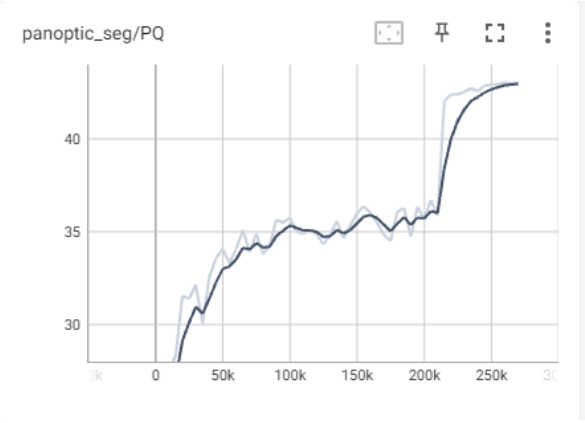


Figure 5.3: Panoptic Quality score for the model

5.1.3 Showcasing the results from the model

The two examples which will be used are videos from Aalborg and New York City (NYC) [22] [23]. The Aalborg video is a walk through a portion of the town and showcases a bit more of what classes the COCO dataset has to offer. Whereas, the NYC video is quite short but shows how it performs at a busy intersection. Figures 5.4 and 5.6 show at a quick glance a still image which can be used as a comparison before any panoptic segmentation is applied.
After having tested the model on a couple of video, the possibility to look into the other wanted requests from Capture One can start. The next focus will be to try and look into handling the borders between masks.

Figure 5.4: Still image showing approximately the same frame before the panoptic segmentation to the Aalborg video. [22]

Figure 5.5: Dynamic camera showcasing the video panoptic segmentation walking through Aalborg. [24]

5.2 Mask borders

As was mentioned in the problem analysis and in the study case. Capture One are interested in seeing how borders between masks can be handled. This could mean that for certain classes the mask should be softer, probably fitting best for thing classes. There are multiple ways to try and tackle this issue. In this case it is attempted through using the output from the combined semantic and instance labels. This function from Detectron2 returns two parameters. The one of importance is the one called segments_info. This list includes dictionaries, where each dictionary represents information about a single instance

Figure 5.6: Still image showing approximately the same frame before the panoptic segmentation to the NYC video. [23]

Figure 5.7: Static video showcasing the video panoptic segmentation for an active street. [25]

in the panoptic segmentation result. The dictionaries used to try and alter the borders are the "id" and "isthing". The latter is of more importance and is used to split the information in the segments_info into things and stuff. This was to try and put softer borders around the thing classes first. The id was used to specify what color should be used for the specific instances. Only two colors were chosen to be used. These are red and green, more specifically, stuff is outlined with a green line and things are outlines with a red line. This can be seen in Figure 5.8. [26]

Figure 5.8: Outlines from the segmentations found in the image. Green outlines are of stuff in the image and red outlines the things. [12]

With the possibility of outlining the masks in the image based off the panoptic segmentation information. It should be possible to alter the border of the masks given by what category it belongs to. With this knowledge it should be feasible to move on to the next step, which is applying the panoptic segmentation inference first, and then try to use what was learned from before to get the outlines of the classes. This can be seen in Figure 5.9 and 5.10.
The idea for the algorithm was to apply Gaussian blur to 'thing' segments before applying a Canny edge detector. This should result in smoother, thicker outlines for 'thing' segments as the blurring process causes some of the mask's 'True' values to 'blend' into neighboring 'False' pixels. When the edge detector is applied, it ought to produce thicker, smoother edges that include this 'blend'. However, this did not seem to be the case as Figures 5.9 and 5.10 seem to be identical. The reason for this is probably due to adjustments happening in the wrong place.

Figure 5.9: Figure showing the regular inference from the Detectron2 model [12]

Figure 5.10: After applying the outlines for the boundaries of the segments. Furthermore, adding blur to the masks of the "thing" classes. [12]

Chapter 6

Discussion

This chapter will discuss the results and flaws of this project. Additionally, a future work section mentioning potential improvements that can be made to the project.

6.1 Evaluation of requirements

In chapter 3 certain requirements were set out for the project based on the project proposal given from Capture One. These requirements will be reviewed in this section to be able to determine if they were successful or not. The first requirement set was:

- Working Panoptic Segmentation for video

This particular requirement was fulfilled using Detectron2 using the COCO dataset from 2017 and was the main task of this project.

The second requirement set was:

- A panoptic quality score of 50 >=

The specific number requirement for the PQ was set by looking at other more current solutions, where many exceed the 50 mark in PQ [27]. This requirement was not fulfilled as the PQ only went up to about 42, but still showed promise from the videos.

The third requirement set was:

- Different type of borders for the mask, depending on class.

The idea behind this requirement was to be able customize the borders, so that certain classes could be different in terms of how the mask blends with other masks. For example certain thing classes do not need to be delineated. The best use case example from Capture One's perspective could be a person's hair, as this would be close to impossible to mask

perfectly without getting anything else into the mask. Therefore, putting a soft border here could be beneficial. In this project there was an attempt to smoothen the border between thing and stuff classes as a starting point, since if this would work then using the segments_info parameters, desired classes could be tackled instead.

The fourth and last requirement which was set was the following:

- Multiple masks for a single pixel

Due to time limitation, this requirement was not looked into. But it is believed that this could be achieved while still using Detectron2, but most likely the internals of the model would need to be looked at as to get multiple confidence scores. If there were multiple high scoring confidence scores, then this could potentially be used to have multiple masks for a single pixel.

6.2 Future work

In this project the model used was Detectron2, where in hindsight it might have been better to use other models that have shown to be able to get a higher PQ scores. However, there are pros and cons to most implementations, and due to Detectron2 being somewhat well documented it served as a fine introduction to the subject.

For the next implementation it would definitely be beneficial to try out a new dataset as well as model to broaden the knowledge around the subject and as to see what works best for the task at hand. The new dataset should also preferably be a video dataset, as this can give unique temporal information, that images would not be able to provide, thus potentially leading to an increase in the quality of the output from the model. A direct comparison between using a video dataset and image dataset would be interesting.

In general it would also be preferable to have a computer with at least one or more GPUs that are compatible with CUDA to be able to set up training locally on the computer [28]. Setting up environments and working with the files locally is usually more manageable than setting up a project remotely. This can especially become troublesome in google colab with the limited time allocated per user to a GPU, and the limited GPU memory available.

Chapter 7

Conclusion

In section 2.5 a final problem statement was given:

How can Detectron2 be used to create video panoptic segmentation which takes into account the necessities desired by Capture One?

In addition to the final problem statement, four requirements were set for the project. Only one of four requirements were fulfilled by the solution. The one fulfilled was to get a model capable of doing VPS.

However, the solution was not capable to perform at the same level as other implementations out there, peaking at around a PQ score of 42.

The third requirement was to be able to handle the borders between the masks. At current time this was not fulfilled, but is believed with more time this could perhaps have been solved.

The fourth requirement was to be able to have multiple masks for the same pixel. This was prioritised the least, therefore this was not really tested or attempted in any way.

In regards to the final problem statement, it is possible to use Detectron2 to create VPS, however, the supplementary requests from Capture One were not taken into account in the solution.

Bibliography

[1] Alexander Kirillov et al. "Panoptic Segmentation". In: *https://arxiv.org/abs/1801.00868* (2018).

[2] *About Capture One*. URL: https://www.captureone.com/en/about-capture-one.

[3] *Semantic Segmentation vs. Instance Segmentation: Explained*. URL: https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/ (visited on 04/03/2023).

[4] Alexander Kirillov et al. "PPanoptic Feature Pyramid Networks". In: *https://arxiv.org/abs/1901.02446* (2019).

[5] *Panoptic Segmentation - The Panoptic Quality Metric*. URL: https://medium.com/@danielmechea/panoptic-segmentation-the-panoptic-quality-metric-d69a6c3ace30 (visited on 04/04/2023).

[6] *Review — UPSNet: A Unified Panoptic Segmentation Network*. URL: https://sh-tsang.medium.com/review-upsnet-a-unified-panoptic-segmentation-network-3754561fe497.

[7] *VPSNet for Video Panoptic Segmentation*. URL: https://github.com/mcahny/vps.

[8] *Panoptic-DeepLab (CVPR 2020)*. URL: https://github.com/bowenc0221/panoptic-deeplab.

[9] *EfficientPS: Efficient Panoptic Segmentation*. URL: http://panoptic.cs.uni-freiburg.de/.

[10] *Detectron2: A PyTorch-based modular object detection library*. URL: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/.

[11] *Fully Convolutional Networks for Panoptic Segmentation*. URL: https://github.com/dvlab-research/PanopticFCN.

[12] *Common Objects in Context*. URL: https://cocodataset.org/#home.

[13] *The Cityscapes Dataset*. URL: https://www.cityscapes-dataset.com/.

[14] *ADE20K*. URL: https://paperswithcode.com/dataset/ade20k.

[15] *Large-scale Video Panoptic Segmentation in the Wild: A Benchmark*. URL: https://github.com/VIPSeg-Dataset/VIPSeg-Dataset.

[16] *Benchmarks*. URL: https://detectron2.readthedocs.io/en/latest/notes/benchmarks.html.

[17] Xiangyu Zhang et al. "Deep Residual Learning for Image Recognition". In: *https://arxiv.org/pdf/1512.03385.pdf* (2015).

[18] *Digging into Detectron 2 — part 1*. URL: https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd (visited on 01/05/2023).

[19] *Digging into Detectron 2 — part 4*. URL: https://medium.com/@hirotoschwert/digging-into-detectron-2-part-4-3d1436f91266 (visited on 01/05/2023).

[20] *Digging into Detectron 2 — part 5*. URL: https://medium.com/@hirotoschwert/digging-into-detectron-2-part-5-6e220d762f9 (visited on 01/05/2023).

[21] *Dataset Health Check*. URL: https://universe.roboflow.com/jacob-solawetz/microsoft-coco/health (visited on 05/05/2023).

[22] *Walking street, Aalborg 4K*. URL: https://youtu.be/4RS_-XbajlM.

[23] *Driving Downtown - New York City 4K - USA*. URL: https://youtu.be/7HaJArMDKgI.

[24] *Aalborg VPS*. URL: https://youtu.be/3R7jKiI7n9g (visited on 01/06/2023).

[25] *VPS - NYC*. URL: https://youtu.be/0qzaw4zq5hA (visited on 01/06/2023).

[26] *PanopticFPN*. URL: https://github.com/facebookresearch/detectron2/blob/main/detectron2/modeling/meta_arch/panoptic_fpn.py.

[27] *Panoptic Segmentation on COCO test-dev*. URL: https://paperswithcode.com/sota/panoptic-segmentation-on-coco-test-dev.

[28] Dahun Kim et al. "Video Panoptic Segmentation". In: *https://arxiv.org/pdf/2006.11339v1.pdf* (2020).