Prediction Rating and Performance Improvement for Segmentation Networks by Time-Dynamic Uncertainty Estimates

Dissertation

Bergische Universität Wuppertal
Fakultät für Mathematik und Naturwissenschaften

eingereicht von
Kira Maag, M. Sc.
zur Erlangung des Grades eines Doktors der Naturwissenschaften

Betreut durch Prof. Dr. Hanno Gottschalk und Dr. Matthias Rottmann

Wuppertal, 03.08.2021

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-urn:nbn:de:hbz:468-20211007-111502-8
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20211007-111502-8

DOI: 10.25926/jtyb-cd34
[https://doi.org/10.25926/jtyb-cd34]

Acknowledgments

First of all, I wish to thank Hanno Gottschalk and Matthias Rottmann for raising my interest in computer vision, for giving me the opportunity to work under their supervision and for their support while doing so. I have benefited greatly from their knowledge of basics for neural networks, deep learning to the latest ai-technologies and enjoyed the very pleasant working atmosphere during our several meetings. I am grateful to Andreas Frommer for also being a part of his research group.

In addition, I would like to thank Peter Schlicht and Fabian Hüger for motivating the scope of automated driving, for great workshops including fruitful discussions and for also making this thesis possible. My work was founded in part by Volkswagen AG.

I also wish to thank all my colleagues of the *AI* Group, i.e., Applied Computer Science and Stochastic Group, for an awesome time. In particular, I thank my seat neighbors, Robin Chan and Pascal Colling, for proofreading this thesis.

Last but not least, I thank my family and friends for mental and continuous support.

Foreword

Parts of this thesis have already been published in the following works:

- K. Maag, M. Rottmann, and H. Gottschalk, *Time-dynamic estimates of the reliability of deep semantic segmentation networks*, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), (2020), pp. 502–509. © 2020 IEEE

- K. Maag, M. Rottmann, S. Varghese, F. Hüger, P. Schlicht, and H. Gottschalk, *Improving video instance segmentation by light-weight temporal uncertainty estimates*, IEEE International Joint Conference on Neural Networks (IJCNN), (2021). © 2021 IEEE

  Parts of these publications are incorporated in Chapters 2, 3 and 4.

- K. Maag, *False negative reduction in video instance segmentation using uncertainty estimates*, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), (2021). © 2021 IEEE

  Parts of this publication are incorporated in Chapter 5.

Contents

Chapter 1

Introduction

In recent years, neural networks have demonstrated outstanding performance in computer vision tasks like image classification [154, 156], object detection [129, 130], semantic segmentation [25, 141], instance segmentation [13, 59], face recognition [26, 27] and depth estimation [142, 165]. In 2012, the approach of Alex Krizhevsky et al. [83] achieved such impressive results in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC-2012) for image classification that consequently research on neural networks was further intensified. Their deep convolutional neural network won the competition with a test error rate of 15.3% compared to 26.2% obtained by the second-best method. The application areas of computer vision tasks are diverse, such as in sports performance analysis [114], agriculture [153] as well as in safety critical ones like medical diagnosis [77] and automated driving [24]. Over the last years, more and more technical assistance systems have been developed which customer can choose to install in their vehicles. These systems include for example traffic sign recognition, adaptive cruise control and lane keeping assistance. In automated driving, the human driver becomes less necessary and the vehicle is guided by the system. To this end, information of the environment is extracted from different sensors like camera, radar or LiDAR. Computer vision tasks like object detection, semantic segmentation and instance segmentation are usually applied to monocular images, but several cameras are attached to the vehicle to increase the field of vision. These tasks serve as important tools for scene understanding. State-of-the-art approaches are mostly based on convolutional neural networks. However, neural networks as statistical models produce probabilistic predictions, which are therefore prone to error with a certain probability. A well-known failure in the perception of an automated vehicle is the Uber accident from 2018 [115]. A 49-year-old pedestrian crossing the street was not detected by the system and thus, struck by the ego car. The different sensors identified this pedestrian a few seconds before the accident as car,

bicycle or other, i.e., something that has been detected but does not belong to any predefined class. As a result of these class changes, the path of the person could not be clearly determined and the speed was incorrectly estimated as the person was predicted to be a vehicle. The accident occurred due to the failed perception. This example demonstrates the importance of understanding and minimizing the errors of neural networks. For this reason, the reliability of neural networks in terms of prediction quality estimation [33, 136] and uncertainty quantification [44] is of highest interest, in particular in safety critical applications like medical diagnosis [120, 169] and automated driving [70, 87]. Furthermore, two different errors arise when using object detection methods, namely false positive and false negative object predictions, respectively. False positives are objects predicted by the neural network which do not appear in the image, and false negatives are objects which are overlooked by the network and thus, not detected.

In this thesis, we propose methods for prediction quality rating and performance improvement in terms of accuracy by using time-dynamic uncertainty estimates. In particular, we present a false positive detection method and a reduction approach minimizing non-detected objects of a neural network. We consider the problems of semantic segmentation, i.e., the pixel-wise classification of image content, and instance segmentation, which consists of categorizing as well as localizing objects by predicting each pixel that corresponds to a given instance. Our methods serve as post-processing steps, applicable to any semantic or instance segmentation network, which we test on different datasets and networks. The main focus of our methods lies on the application of automated driving. Using the availability of image sequences, we obtain further information of predictions without much effort for enhancing our approaches.

For the first time, the tasks of temporal meta classification and meta regression for semantic and instance segmentation are presented, which were previously introduced only for single frames and semantic segmentation in [136]. Meta classification refers to the task of predicting whether a predicted segment/instance intersects with the ground truth or not. As performance measure for prediction quality, the commonly used intersection over union (IoU) is considered which quantifies the degree of overlap of prediction and ground truth [72]. In semantic segmentation, an object is called false positive if the IoU is equal to zero and in instance segmentation, if the IoU is less than 0.5. Hence, the meta classification task corresponds to (meta) classifying between $IoU = 0$ and $IoU > 0$ for every predicted segment and to classifying between $IoU < 0.5$ and $IoU \geq 0.5$ for every predicted instance, respectively. Meta regression is the task of predicting the IoU for each predicted segment/instance directly. This prediction of the IoU serves as a performance estimate. Therefore, both meta classification and regression are able to reliably evaluate the quality of a segmentation obtained from a neural network. For learning both tasks, segment/instance-wise metrics are constructed

characterizing uncertainty and geometry of a given object. By tracking objects over time, time series of single-frame metrics are generated. To this end, we introduce a light-weight tracking approach for predicted segments/instances. Our tracking algorithm matches objects based on their overlap in consecutive frames by shifting objects according to their expected location in the subsequent frame predicted via linear regression. For instance segmentation, we extend this set of single-frame metrics by novel and truly time-dynamic ones. These metrics are based on survival time analysis, on changes in the shape, on expected position of instances in an image sequence and on depth estimation. From the respective set of metrics, we estimate the prediction quality on segment/instance-level by means of temporal uncertainties.

Moreover, we employ meta classification to improve the network performance in terms of accuracy for instance segmentation networks. These networks provide for each instance a confidence value, also called score, which is not well calibrated [53]. These scores can have low values for correctly predicted instances and high ones for false predictions. During inference of instance segmentation networks, all instances with score values below a threshold are removed. This is done to balance the number of false positive and false negative instances. Nevertheless, it can happen that correctly predicted instances disappear, while many false positives remain. For this reason, meta classification is used as advanced score value improving the networks' prediction accuracy by replacing the score threshold by the estimated probability of correct classification during inference.

Furthermore, we focus on the reduction of non-detected objects by an instance segmentation network. In applications like automated driving, the detection of road users, i.e., the reduction of false negatives, is of highest interest. In other words, it is preferable to predict road users incorrectly to missing them. We use a relatively small score threshold value during inference and apply our light-weight false negative detection algorithm to these remaining instances to find possible overlooked ones. This approach is based on inconsistencies in the time series of tracked instances such as a gap in the time series or a sudden end and uses this information of previous frames to detect these cases. Subsequently, new instances are constructed that the neural network may have missed. Since the sensitivity toward predicting instances of road user can be greatly increased by our algorithm to reduce false negatives, we use meta classification as false positive detector. As a result, our fused approach reduces the number of false negative instances and improve the networks' performance.

This thesis is structured as follows. In Chapter 2, we present the basics on neural networks containing semantic segmentation, instance segmentation, depth estimation, the motivation as well as related work on uncertainty quantification. In addition, the object tracking task and related work are introduced. The basics also include different methods for meta classification and regression. Following

in Chapter 3, we describe our prediction rating method for semantic segmentation. In more detail, our method consists of the tracking algorithm for segments, the construction of segment-wise metrics using uncertainty and geometric information as well as temporal meta classification and regression. We conclude this chapter with numerical results and a brief discussion. We proceed analogously in Chapter 4 for instance segmentation and explain the differences to segments, the time-dynamic metrics as well as how we use meta classification to improve the networks' performance. Subsequently in Chapter 5, we propose and evaluate our temporal false negative detection approach consisting of a detection step fused with uncertainty based meta classification. Finally, in Chapter 6, we discuss the results of this thesis and provide an outlook on interesting extensions.

Chapter 2

Review of Basic Material and Related Work

This chapter gives an overview of basic material that is employed in the remainder of this thesis. Firstly, in Sec. 2.1, we introduce the basics of neural networks including feed forward and convolutional neural networks, respectively, which are mainly based on the textbook [51]. In addition, the network architectures for semantic segmentation, instance segmentation and depth estimation are described as well as associated performance metrics. Afterwards, it is explained why the uncertainty evaluation of neural networks is of high interest and the related work on this topic is presented in Sec. 2.2. Secondly, we turn to object tracking, which is an essential task in video applications and the basis for the methods developed in this thesis. Related work and evaluation tools for object tracking are introduced in Sec. 2.3. Next, in Sec. 2.4, we discuss classification and regression methods which we use for our meta classification and regression tasks. Finally, the details of datasets used in this thesis are given in Sec. 2.5.

2.1 Neural Networks

Deep learning is a subset of machine learning in artificial intelligence, which is motivated by the functionality of the human brain. The function of the brain can be imitated by creating similar patterns to make decisions on given data. Neural networks form a particular type of deep learning models and have applications in computer vision, for example for automated driving and medical diagnosis.

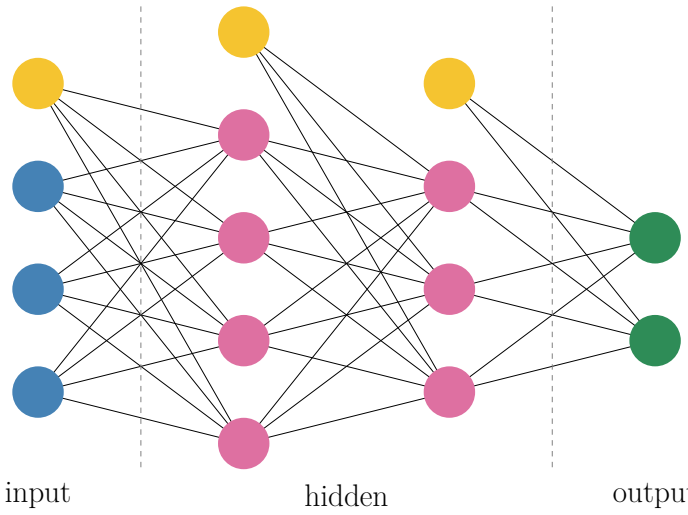


Figure 2.1: A feed forward neural network with input neurons (blue), neurons in the hidden layers (pink), output neurons (green) and bias units (yellow) is shown.

2.1.1 Feed Forward Neural Networks

The origins of artificial neurons date back to 1943 [106]. At this period, simplified models of a (McCulloch–Pitts) neural network were used for the computation of logical and arithmetic functions. The feed forward neural networks (FFNs) or multilayer perceptrons are originated by this McCulloch–Pitts neural network model and are the basis for deep learning models. An exemplary network structure of a FFN is shown in Fig. 2.1. A neural network consists of an input and output layer as well as hidden layers, whose number may vary. In addition, there are bias units that do not receive any input themselves. If there is a connection between all neurons of consecutive layers, the network is called *fully-connected*, as depicted in Fig. 2.1. The power of the connection between two neurons is expressed by *weights*. Neural networks learn through weight adjustments. If the network is only passed through from input to output, it is called a feed forward network (as the name says). The output of the network can be determined by a chain of functions based on the order of the layers

$$f(x, w) = (f^{(l')} \circ f^{(l'-1)} \circ \ldots \circ f^{(1)})(x, w) \tag{2.1}$$

with a number of layers l' as well as given input data x and weights w. So each layer l represents a function of the previous layer $l-1$

$$h^{(l)} = f^{(l)}(h^{(l-1)}) = \phi^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)}) = \phi^{(l)}(a^{(l)}) \; \forall \; l = 1, \ldots, l' \tag{2.2}$$

where $h^{(l)} \in \mathbb{R}^{N^{(l)}}$ is the output using $f^{(l)} : \mathbb{R}^{N^{(l-1)}} \to \mathbb{R}^{N^{(l)}}$, $\phi^{(l)} : \mathbb{R}^{N^{(l)}} \to \mathbb{R}^{N^{(l)}}$ the activation function applied on activation $a^{(l)} \in \mathbb{R}^{N^{(l)}}$ with $N^{(l)}$, the number

of neurons in layer l. The weight vector of the bias unit of layer l is denoted by $b^{(l)} \in \mathbb{R}^{N^{(l)}}$ and the weight matrix between neurons of layer $l-1$ and l by $W^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l-1)}}$.

In addition to feed forward neural networks, other well-known architectures are considered, such as recurrent neural networks (RNNs) [104]. These networks are designed to handle a series of inputs with no predefined size limit (in comparison to FFNs). During training, RNNs use additional information from prior inputs influencing the production of the current output. The networks' output is affected by weights applied on inputs and additionally by a hidden state vector providing the context based on prior input. In [163], the time-delay neural networks (TDNNs) are introduced which are able to recognize relationships between time-delayed events. To obtain time-invariants, the inputs of multiple time steps are applied simultaneously by the TDNN. A typical application for these networks is speech recognition. Furthermore, spiking neural networks (SNNs) have also become widely known [102]. These networks are more similar to biological neural networks than FFNs. Using bio-inspired neuron and synapse models, SNNs mimic the functionality of the human brain. The main difference between both networks is that the neurons of the SNN (in comparison to those of a FFN) do not transfer information during every propagation cycle but only when a membrane potential attains a certain threshold. Reaching this threshold, the neuron gives a signal to the other neurons which as a result decrease or increase their potentials.

Universal Approximation Theorem The question in deep learning is how accurate the function $f(x, w)$ obtained by a neural network approximates the function $f^* : \mathbb{R}^n \to \mathbb{R}$. Given the input data x and labels y, we assume that there is a *correct* labeling function, i.e., $f^*(x) = y$. The universal approximation theorem states that feed forward neural networks are able to approximate ($f(x, w) \approx y$) any continuous function to any degree of accuracy on a closed and bounded subset of $\mathbb{R}^n$ (see [67]). To fulfill the property, the FFN requires at least one hidden layer and a non-linear activation function. A proof for the sigmoid activation function is introduced in [31]. In a more recent work [151], the universal approximation property for the ReLU function is also shown.

Activation Function The activation functions are applied to the hidden layers and the output layer. The choice of activation functions in neural networks depends on the type of function that is approximated. If a nonlinear function is to be approximated, a nonlinear activation function is required in the hidden layers. In Fig. 2.2 three different activation functions are shown, which are explained in the following. The *sigmoid* function and its derivative are defined by

$$s(a) = \frac{1}{1+e^{-a}} \quad \text{and} \quad s'(a) = \frac{e^{-a}}{(1+e^{-a})^2}\,. \tag{2.3}$$

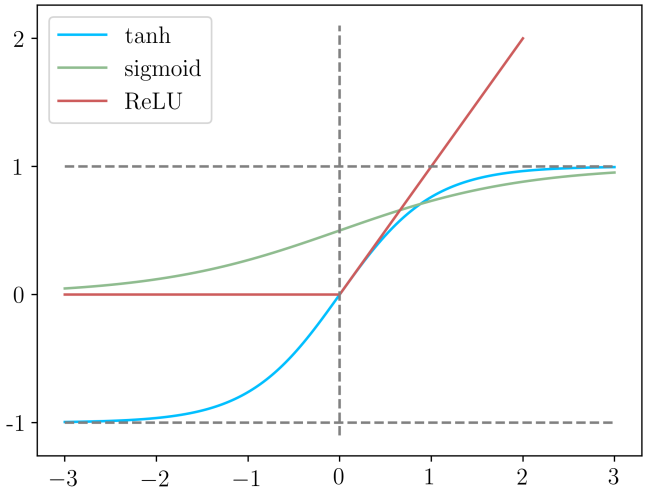


Figure 2.2: Activation functions.

This function is differentiable which is relevant for the training of a neural network. The output of a network is compared with the true labeled data (*ground truth*) and the resulting difference should be minimized. The rapidly flattening ends of the sigmoid function lead to the *vanishing gradient problem* [65], which means that the gradients of the weights in early layers decrease towards zero, causing numerical problems. An improvement is the *hyperbolic tangent* which is described by a scaled sigmoid function

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad \text{with} \quad \tanh'(a) = \frac{4 \cdot e^{-2a}}{(1 + e^{-2a})^2} \,. \tag{2.4}$$

At the origin the derivative has higher values resulting in faster convergence than using the sigmoid function, however, at the edges the same vanishing gradient problem occurs [7]. A solution is to use the *rectified linear unit* (ReLU) defined by the function

$$g(a) = \max\{0, a\} \,, \tag{2.5}$$

which outperforms sigmoid and tanh in practice [101]. The ReLU activation function is inspired by the human brain where the percentage of active neurons at the same time is rather low, i.e., for all negative inputs ReLU returns zero [50]. Meanwhile and due to the piece-wise linear parts of ReLU, the computation of its derivative is very efficient and can prevent the vanishing gradient problem. Note, the non-differentiability in zero is not a problem in practice.

The presented activation functions can be used in the hidden layers as well as in the output layer. If a binary classification problem is considered, the sigmoid or tanh functions are suitable choices as activation functions for the output layer. Both functions return a value in $[0, 1]$ which can be interpreted as probability for a

binary variable having the value 1. However, for example in image segmentation, more classes are needed and thus, we want to obtain a probability distribution over c different classes. A possible activation function is the *softmax* function [19] which is used in the last layer to provide a probability distribution over predefined classes. The softmax function for activation $a_i \in \mathbb{R}$ of neuron i (see (2.2)) is given by

$$\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{c} e^{a_j}} \in (0,1)\,, \tag{2.6}$$

that indicates the probability of predicting the input of the neural network belonging to class i. We obtain $\sum_{i=1}^{c} \text{softmax}(a_i) = 1$.

Loss Function First, we introduce the theoretical basics of learning models in general [146]. We define the input set $\mathcal{X}$, the label set $\mathcal{Y}$ and the training data $\mathcal{S} = ((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}))$ with n examples. The training data is generated by a probability distribution $\mathcal{D}$ over $\mathcal{X}$ representing the environment. Moreover, we assume a correct labeling function $f^* : \mathcal{X} \to \mathcal{Y}$ with $y^{(i)} = f^*(x^{(i)})\ \forall\ i$. The learner shall construct a prediction rule $r_\mathcal{S} : \mathcal{X} \to \mathcal{Y}$ to classify the examples. By $\mathcal{A}(\mathcal{S})$, we denote the rule returned by a learning algorithm $\mathcal{A}$ after receiving the training data $\mathcal{S}$. The error of $r_\mathcal{S}$ corresponds to the probability that $r_\mathcal{S}(x)$ is not equal to $f^*(x)$ on a random example x generated by the distribution $\mathcal{D}$. More formally, the error of the prediction rule is defined by

$$L_{\mathcal{D},f^*}(r_\mathcal{S}) = \mathcal{D}(\{x : r_\mathcal{S}(x) \neq f^*(x)\}) \tag{2.7}$$

where $\mathcal{D}(\mathcal{A})$ describes how likely an observation of example $x \in \mathcal{A}$ is with $\mathcal{A} = \{x \in \mathcal{X} : \pi(x) = 1\}$, $\pi : \mathcal{X} \to \{0,1\}$. The goal of the learning algorithm is to find a $r_\mathcal{S}(x)$ minimizing the error. In general the correct labeling function f^* and the probability distribution $\mathcal{D}$ are unknown, information can only be extracted from the training data. For this reason, we consider the empirical error (also called risk) over the training examples defined by

$$L_\mathcal{S}(r_\mathcal{S}) = \frac{|\{i \in \{1,\ldots,n\} : r_\mathcal{S}(x^{(i)}) \neq y^{(i)}\}|}{n}\,. \tag{2.8}$$

As the training data is available and depicts a small representation of the environment, it is useful to employ them for the training error. This paradigm is called *Empirical Risk Minimization* (ERM) [161].

Minimizing the empirical risk may lead to an *overfitting* problem of the ERM learning rule [75], i.e., the performance on the training set is very high, while the performance on the true environment is poor. A solution would be to restrict the set of classifiers (hypothesis class) before actually presenting the training data

to the learner. This restriction is called inductive bias. Consequently, the errors made during learning can be divided into the estimation and the approximation error [118]. The latter depends on the inductive bias, so this error is determined by the chosen hypothesis class. The estimation error is caused due to the fact that the empirical risk is only an estimate of the true risk. As the total error shall be reduced, the bias-complexity trade-off [47] occurs. If the size of the hypothesis class is increased, the approximation error decreases on the one hand and the estimation error increases on the other hand.

Maximum likelihood estimation [3] is the task of estimating distribution parameters w based on the training set $\mathcal{S}$ for modeling the underlying distribution $\mathcal{P}_w$. The maximum likelihood estimator is an empirical risk minimizer for the log loss function

$$-\log(\mathcal{P}_w(x)) \tag{2.9}$$

given parameters w and input x. This formula is also called negative log-likelihood of input x under the condition that the data is independently and identically distributed according to $\mathcal{P}_w$. Minimizing the empirical risk

$$\arg\min_w \sum_{i=1}^{n}(-\log(\mathcal{P}_w(x^{(i)}))) = \arg\max_w \sum_{i=1}^{n}\log(\mathcal{P}_w(x^{(i)})) \tag{2.10}$$

is equivalent to the maximum likelihood principle. Then, the true risk of parameter w is given by

$$\mathbb{E}_x[-\log(\mathcal{P}_w(x))] = -\sum_x \mathcal{P}(x)\log(\mathcal{P}_w(x)) \tag{2.11}$$

$$= \sum_x \mathcal{P}(x)\ \log\left(\frac{\mathcal{P}(x)}{\mathcal{P}_w(x)}\right) + \sum_x \mathcal{P}(x)\ \log\left(\frac{1}{\mathcal{P}(x)}\right) \tag{2.12}$$

$$= D_{\mathrm{KL}}(\mathcal{P}(x)||\mathcal{P}_w(x)) + H(\mathcal{P}(x)) \tag{2.13}$$

where we assume that the data is distributed as $\mathcal{P}(x)$, not necessarily $\mathcal{P}_w(x)$. The true risk corresponds to the combination of the (Shannon) entropy $H(\mathcal{P}(x))$ [147] and the Kullback–Leibler divergence $D_{\mathrm{KL}}(\mathcal{P}(x)||\mathcal{P}_w(x))$ [84]. The latter measures the divergence between two probabilities.

In practice, after calculating the output of the neural network, this value is compared with the ground truth value. This is done by means of the loss function. The error should be as small as possible, so the loss function has to be minimized. We consider the multi-class classification problem with c classes. For this, we define the one-hot-encoded label vector $y(x) \in \{0,1\}^c$ consisting of zeros, only at the position i it has the value 1, if i corresponds to the true class of the input x. Let $\hat{y}_i := \mathrm{softmax}(a_i)$ be the output of the softmax function in the last layer. Thus, we define the *cross-entropy* (see (2.11)), a commonly used loss function, by

$$L(\hat{y}, y) = -\sum_{i=1}^{c} y_i \log(\hat{y}_i)\,. \tag{2.14}$$

In multi-class classification problems, the predicted probabilities $\hat{y}$ shall converge to the original probability distribution y, and this is accomplished by minimizing the cross-entropy loss, which measures the difference between these two probability distributions. Furthermore, minimizing the cross-entropy achieves a faster convergence and more robustness during training with noisy examples (generalization) compared to other loss functions, see [73].

Stochastic Gradient Descent Algorithm The loss function is minimized using the stochastic gradient descent (SGD) algorithm [14]. We start with initial weights, descend by a learning rate ϵ in the direction of the minimum and adjust the weights accordingly. This process is repeated until a stopping condition is met, e.g. a local minimum is reached or a predefined number of iterations is exceeded. Descending by the learning rate ϵ, i.e., the step in the direction of the steepest descent, is described by

$$w := w - \epsilon \cdot \nabla_w \bar{\mathcal{L}} \tag{2.15}$$

where $\bar{\mathcal{L}}$ denotes the average over the loss function $L(\hat{y}, y)$ applied to the n training examples

$$\bar{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}^{(i)}, y^{(i)}) \,. \tag{2.16}$$

In practice, we do not use all n examples from the training dataset, instead so-called *minibatches* are considered using only $m \ll n$ training examples. Note, this can reduce the computing effort. We define analogously the gradient of the minibatch by

$$\nabla_w \mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \nabla_w L(\hat{y}^{(i)}, y^{(i)}) \,. \tag{2.17}$$

In each weight update, different randomly sampled minibatches are created. For convex problems, the gradient of the minibatch $\nabla_w \mathcal{L}$ is an unbiased estimator for gradient $\nabla_w \bar{\mathcal{L}}$:

$$\mathbb{E}[\nabla_w \mathcal{L}] = \nabla_w \bar{\mathcal{L}} \,. \tag{2.18}$$

By the law of large numbers [148], we obtain

$$\lim_{m \to \infty} \nabla_w \mathcal{L} = \lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} \nabla_w L(\hat{y}^{(i)}, y^{(i)}) \tag{2.19}$$

$$= \mathbb{E}[\nabla_w L(\hat{y}^{(i)}, y^{(i)})] \tag{2.20}$$

$$= \nabla_w \bar{\mathcal{L}} \,. \tag{2.21}$$

To guarantee the convergence of the SGD, the sufficient conditions

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty \tag{2.22}$$

shall be satisfied [133]. In applications, it is common that the learning rate is reduced over the number of iterations. To study the corresponding convergence rate, the error between the loss function after k iterations $L^k(\hat{y}, y)$ and the minimal possible loss $L^*(\hat{y}, y)$ is considered. If we apply the SGD to a convex problem, this error is $\mathcal{O}(\frac{1}{\sqrt{k}})$ and for a strongly convex problem $\mathcal{O}(\frac{1}{k})$ [15].

Learning (descending) can be comparatively slow in the SGD, for this reason, various methods have been developed to speed up learning, such as the method of *momentum* [126]. This method collects an exponentially decreasing average of past gradients ν (called velocity) and moves in this direction, i.e., the former gradients gain less importance. The update of the weights is modified to

$$\nu := \alpha\nu - \epsilon \cdot \nabla_w \mathcal{L} \tag{2.23}$$

$$w := w + \nu \tag{2.24}$$

where $\alpha \in [0, 1)$ denotes a hyperparameter determining how fast the contribution of the former gradient decays exponentially. The step size increases if many consecutive gradients have the same direction. The SGD with momentum is often used in segmentation networks, e.g. instance segmentation [13, 59].

A modified version of the momentum method is introduced in [155], called *Nesterov momentum*, which is also considered in segmentation tasks like semantic segmentation [24]. The weight update remains the same and only the position where the gradient is evaluated varies. The gradient is evaluated at new position

$$w_{intermediate} = w + \alpha\nu \tag{2.25}$$

after the current velocity is deployed. This modification works like a correction step. When the momentum leads into the wrong direction, the gradient can correct this during the same update iteration.

Another gradient descent method is the *adaptive moment estimation* (Adam) [80]. The learning rates are adjusted depending on the estimation of the first and second moments of the gradient. This optimizer is also well-known in the field of deep learning and is for example commonly-used in depth estimation [90]. In summary, all optimization algorithms presented are widely used, and there is no consensus on which optimizer to choose. We refer to [143] for a comparison of different optimization methods for different learning tasks.

Backpropagation The gradients used by the stochastic gradient descent algorithm (or other gradient based optimization methods) are determined by backpropagation [139]. The architecture of neural networks allows for efficient computation of gradients by making use of already-computed derivatives of weights. The backpropagation procedure can be summarized in the following three steps:

1. Forward propagation:
   - Initialization of weights w
   - Input x is propagated forward through the network
   - Calculation of $\hat{y}$
2. Error calculation:
   - Error is determined using the loss function $L(\hat{y}, y)$
3. Backward propagation:
   - Error is propagated back through the network (from output to input)
   - Weights are changed depending on the effect on the error
   - Computation of the gradients

For a deeper understanding, we consider an example of applied backpropagation. We denote by $w_{i,j}^{(l)}$ the weight of neuron i in layer $l-1$ to neuron j in layer l, by $a_j^{(l)}$ the activation and by $h_j^{(l)}$ the output of neuron j in layer l using the activation function $\phi : \mathbb{R} \to \mathbb{R}$ (the same in all hidden layers). We compute the gradient of the loss function $L = L(\hat{y}, y)$ for an example (x, y):

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial w_{i,j}^{(l)}} . \tag{2.26}$$

In the following, we replace $\frac{\partial L}{\partial a_j^{(l)}}$ by $\delta_j^{(l)}$ which is referred to as the error of neuron j in layer l. Next, we reformulate the latter fraction into

$$\frac{\partial a_j^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial}{\partial w_{i,j}^{(l)}} (b_j^{(l)} + \sum_{k=1}^{N^{(l-1)}} w_{k,j}^{(l)} \cdot h_k^{(l-1)}) = h_i^{(l-1)} \tag{2.27}$$

where $N^{(l-1)}$ is the number of neurons in layer $l-1$. Using this equation and the definition of the neuron error, we obtain

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} \cdot h_i^{(l-1)} \; \forall \; i = 1, \ldots, N^{(l-1)} \; \forall \; j = 1, \ldots, N^{(l)} \tag{2.28}$$

as formulation of the gradient. For the hidden layers the error $\delta_j^{(l)}$ can be rewritten as

$$\delta_j^{(l)} = \sum_{k=1}^{N^{(l+1)}} \frac{\partial L}{\partial a_k^{(l+1)}} \cdot \frac{\partial a_k^{(l+1)}}{\partial a_j^{(l)}} \tag{2.29}$$

$$= \sum_{k=1}^{N^{(l+1)}} \delta_k^{(l+1)} \cdot \frac{\partial a_k^{(l+1)}}{\partial a_j^{(l)}} \tag{2.30}$$

$$= \phi'(a_j^{(l)}) \sum_{k=1}^{N^{(l+1)}} w_{j,k}^{(l+1)} \cdot \delta_k^{(l+1)} \tag{2.31}$$

resulting in the following representation of the gradient of the loss function

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \phi'(a_j^{(l)}) \cdot h_i^{(l-1)} \sum_{k=1}^{N^{(l+1)}} w_{j,k}^{(l+1)} \cdot \delta_k^{(l+1)} \;\forall\; i = 1, \ldots, N^{(l-1)} \;\forall\; j = 1 \ldots, N^{(l)} \,. \tag{2.32}$$

Regularization Previously, we have focused on the training of neural networks, although the algorithms have to achieve good results even with new input. The error caused by new input is called *generalization error*. Overfitting describes the gap between this error and the training error. Regularization refers to any change made to the learning algorithm to prevent overfitting and thus, reduce this generalization error.

One possibility to regularize is to add a *penalty parameter* $\Omega(w)$ with hyperparameter $\lambda \in [0, \infty)$ to the loss function

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda \cdot \Omega(w) \,. \tag{2.33}$$

Typically the weights of the bias unit are not penalized, here w describes all weights except the ones from the bias unit. A commonly used regularization is the application of a ℓ_2-penalization, also known as *weight decay* [54],

$$\Omega(w) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2} w^\top w \,. \tag{2.34}$$

The loss function and its derivative are modified to

$$\tilde{\mathcal{L}} = \mathcal{L} + \frac{\lambda}{2} w^\top w \tag{2.35}$$

$$\nabla_w \tilde{\mathcal{L}} = \nabla_w \mathcal{L} + \lambda w \tag{2.36}$$

and the weight update changes to

$$w := w - \epsilon(\nabla_w \mathcal{L} + \lambda w) \tag{2.37}$$

$$= (1 - \epsilon\lambda) w - \epsilon \nabla_w \mathcal{L} \tag{2.38}$$

(here, for simplification the SGD update rule). Using this penalty term, the weight vector shrinks by a constant factor in each iteration multiplicative by the learning rate ϵ. As a result, the model capacity is reduced which can help to prevent overfitting. However, we obtain non-sparsity as the weights are non-zero (are only decreased and maybe close to zero), and this regularization is not robust to outliers due to the squared term.

Similar to weight decay, the ℓ_1-penalization [157] can be considered for regularization

$$\Omega(w) = \|w\|_1 = |w| \,. \tag{2.39}$$

We rewrite again the loss function and its derivative

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda|w| \tag{2.40}$$

$$\nabla_w \tilde{\mathcal{L}} = \nabla_w \mathcal{L} + \lambda \cdot \operatorname{sign}(w) \,. \tag{2.41}$$

By ℓ_1-regularization, a sparse solution is obtained as weights are forced to be zero during the training process. The associated features are no longer considered which can be interpreted as feature selection. One drawback is the piece-wise non-differentiable solution resulting in a more expensive computation since it is not solvable in terms of matrix measurement [134].

Another method preventing overfitting is *early stopping* [11]. The intuition is that training too much can also lead to overfitting. To this end, the training set is split into two disjoint sets, the training and the validation set. The latter has the task of estimating the generalization error. The training error is expected to decrease steadily over time, while the validation error begins to rise again after a certain time. For this reason, we want to stop training when the validation error is small and has not measurably improved for a number of epochs. The problem of which criterion is used for stopping is addressed in [127]. Using this early stopping during training, the generalization error shall be reduced. This method is widely employed due to its effectiveness, ease of use and the reduced computational effort as the number of training epochs is decreased. There is just one problem if the amount of training data is very limited since not all available training data can be used by employing early stopping.

Dropout is another regularization method where neurons drop out randomly [64]. For each training iteration, independent mask vectors are formed deciding whether a neuron (of the input or a hidden layer) is switched off or not. We denote by $N = \sum_{l=0}^{l'-1} N^{(l)}$ the number of neurons of the input and all hidden layers where $N^{(l)}$ is the corresponding number of neurons in layer l and l' the number of layers. By $\mu \in \{0,1\}^N$, we define the mask vector. The probability for a neuron dropping out is a hyperparameter which is chosen before training. Typically, the probability for an input neuron to stay on is 80% and for neurons in the hidden layers 50%. Dropout training consists of neurons dropping out, forward

and backward propagation performed by the reduced network, a weight update, followed by the application of dropout again (the previously dropped out neurons turn on with their weights from the previous iteration). The goal of this training is to minimize $\mathbb{E}_\mu[\mathcal{L}]$ receiving the mask parameter μ. Although, this expectation includes an exponential number of terms (2^N), we get an unbiased estimate of the gradient by sampling values of μ. In practice, 10 to 20 mask are sufficient. Each reduced model defined by μ outputs a probability distribution $p(y|x, \mu)$. Computing the arithmetic mean over all masks

$$\sum_{\mu} p(\mu) \cdot p(y|x, \mu) \tag{2.42}$$

with $p(\mu)$ denoting the probability distribution used for sampling μ, we obtain the output of the neural network including dropout. In summary, dropout is an effective method for reducing overfitting and yields large improvements over other regularization approaches in various computer vision tasks [152]. There is just one drawback, dropout increases the training time in comparison to a standard neural network of the same architecture.

2.1.2 Convolutional Neural Networks

In fully-connected feed forward networks, the number of parameters and thus, the computing effort increases rapidly with the complexity of the model. The convolutional neural networks (CNNs) [88] offer a solution by replacing the matrix multiplication by convolution operations in at least one layer. Convolutions reduce the connections between neurons and decrease the number of parameters. A convolutional layer consists of a convolution, a nonlinear activation function (like ReLU) and in general a pooling step.

In Sec. 2.1.1, we have introduced the universal approximation theorem for feed forward neural networks. Analogously, in [180, 189] a universal approximation theorem for CNNs is formulated. Any continuous function can be approximated (up to an arbitrary accuracy) by a CNN if the depth of the neural network is large enough.

Convolution The convolution operation [18] for two functions $o_1 : \mathbb{Z} \to \mathbb{R}$ and $o_2 : \mathbb{Z} \to \mathbb{R}$ in the discrete case is defined by

$$(o_1 * o_2)(p) = \sum_{\eta} o_1(\eta) \cdot o_2(p - \eta) \quad \forall\ p \in \mathbb{Z} \tag{2.43}$$

where $*$ denotes the convolution. The function o_1 is referred to as input, the function o_2 as *kernel* and the output as *feature map*. If the input is a gray scaled

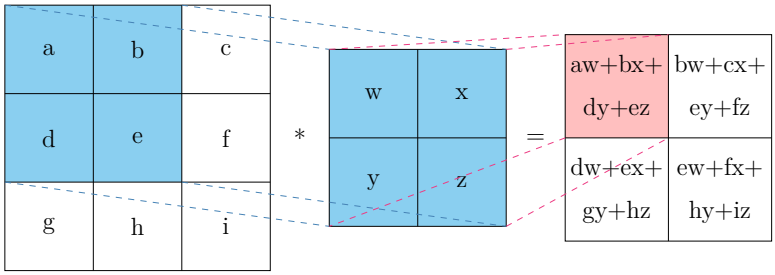


Figure 2.3: An example of convolution with a kernel size of 2×2 and stride of 1.

image, the kernel is also two-dimensional and the convolution is changed to

$$(o_1 * o_2)(p, q) = \sum_{\eta_1} \sum_{\eta_2} o_1(\eta_1, \eta_2) \cdot o_2(p - \eta_1, q - \eta_2) \quad \forall\ (p, q) \in \mathbb{Z}^2\,. \tag{2.44}$$

We can equivalently rewrite this equation to

$$(o_1 * o_2)(p, q) = \sum_{\eta_1} \sum_{\eta_2} o_1(p - \eta_1, q - \eta_2) \cdot o_2(\eta_1, \eta_2) \quad \forall\ (p, q) \in \mathbb{Z}^2 \tag{2.45}$$

as the convolution is a commutative operation. In case of an RGB image a third dimension is added

$$(o_1 * o_2)(p, q) = \sum_{\eta_1} \sum_{\eta_2} \sum_{d} o_1(p - \eta_1, q - \eta_2, d) \cdot o_2(\eta_1, \eta_2, d) \quad \forall\ (p, q) \in \mathbb{Z}^2\,. \tag{2.46}$$

The commutative property of convolution (see (2.44) and (2.45)) is obtained by kernel flipping, i.e., the kernel is flipped relative to the input. Kernel flipping is only of interest from a theoretical perspective and thus, in the application of neural networks cross-correlation is used

$$(o_1 * o_2)(p, q) = \sum_{\eta_1} \sum_{\eta_2} o_1(p + \eta_1, q + \eta_2) \cdot o_2(\eta_1, \eta_2) \tag{2.47}$$

which is the same as a convolution, without flipping the kernel. An example is shown in Fig. 2.3. The number of pixels by which the kernel is moved on the image is called *stride*. If the stride is smaller than the dimension of the kernel, many parameters of the previous equation are jointly used, which is also visible in Fig. 2.3. Convolutions reduce the dimension of the image and support the network to become invariant to small translations of the input.

Pooling Pooling is the last step in the convolutional layer. Pooling summarizes different regions of the input image (or feature maps) and replaces the output with a summarized statistic of adjacent outputs. This operation reduces the number of parameters and improves statistical efficiency. There are different

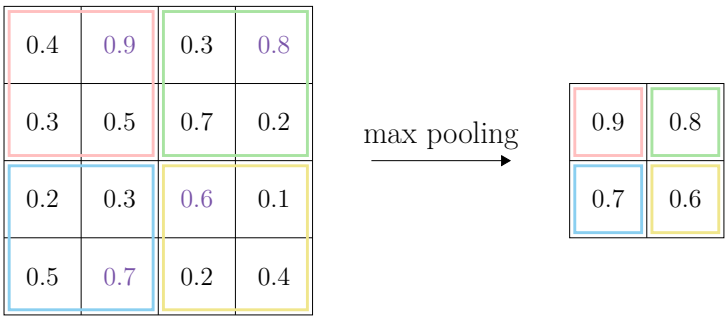


Figure 2.4: Example of max pooling with a kernel size of 2×2 and stride of 2.

pooling methods, such as max pooling [190], which is the most commonly used one. Max pooling chooses the maximum of neighboring values, see Fig. 2.4. Similar to the use of convolutions, a reduction of parameters and dimensions is caused by the application of pooling. Other well-known pooling functions are averaging, applying the ℓ_2-norm and the weighted average based on the distance from the center pixel [16]. The idea of pooling is that the exact position of a feature is not so important, rather the position in relation to other features.

Zero Padding The convolution (Fig. 2.3) and pooling (Fig. 2.4) examples are called *valid*, i.e., the kernel size and the stride are adjusted such that the kernel does not cross the boundary. Zero padding adds zeros to the input image (or feature maps) to ensure validity and to reduce dimensional shrinkage due to convolutions and pooling. We consider a gray scaled image as input with size of $g_x \times g_y$, a kernel with size $k_x \times k_y$ and stride $\tilde{s}$. The resulting output after passing the image through a convolution has a size of

$$\left(\frac{g_x - k_x}{\tilde{s}} + 1\right) \times \left(\frac{g_y - k_y}{\tilde{s}} + 1\right) \tag{2.48}$$

(see [36]). For validity, it is required that both fractions result in integer values. Zero padding describes the approach to add a boundary of new entries with a value of zero to the input image. An example is given in Fig. 2.5 (left). The output size is modified to

$$\left(\frac{g_x - k_x + 2 \cdot p}{\tilde{s}} + 1\right) \times \left(\frac{g_y - k_y + 2 \cdot p}{\tilde{s}} + 1\right) \tag{2.49}$$

using zero padding where p defines the number of boundaries with zero entries by which the input image is extended. As before, the fractions have to be integers. By using zero padding, valid input images with arbitrary kernel sizes and strides can be obtained. A problem, which can be solved by zero padding, is the reduction of the dimensionality. This reduction decreases the computational effort and the

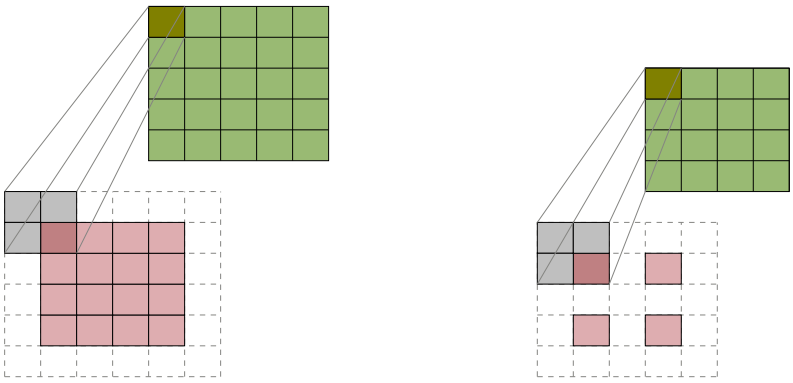


Figure 2.5: *Left*: An example of deconvolution with an input size of 4×4 (pink), a kernel size of 2×2, stride of 1 and zero padding of 1. The operation results in an output of size 5×5 (green). *Right*: Another example with an input size of 2×2 (pink), a kernel size of 2×2, stride of 1, zero padding of 1 and one row with zeros. The operation results in an output of size 4×4 (green).

number of parameters, however, it can happen that the output is reduced down to a size of 1×1. Reducing the size of the filters also prevents this problem, although small filters provide less information. Therefore, zero padding counteracts the reduction and allows the application of multiple stacked convolutional layers, in turn allowing the construction of ever deeper neural networks.

Deconvolution The deconvolution [131] undoes a convolution. While the convolution operation combines multiple feature inputs to a single output, the deconvolution operation associates a single input to multiple outputs. The input and output dimensions are switched. Thus, the deconvolution operation provides a method for increasing the output dimension. We need this dimensionality increase for networks in which the input and output dimension shall be equal. One possibility of deconvolution is zero padding. Using zero padding, the input dimension is kept or increased with several zero padding levels. In Fig. 2.5 (left) an example is shown in which the dimension is increased. Another variant of deconvolution is to add zeros between the single entries of the input image. This method is additionally combined with zero padding in Fig. 2.5 (right). Note, these operations are also called transposed [36] or backward convolutions, since they can be reformulated as convolutions with padding.

Unpooling Unpooling is another method of increasing output dimensions [184]. Instead of summarizing information from areas as in pooling, unpooling uses information to extend a value to an area. As an example, the max unpooling (complement to max pooling) is given in Fig. 2.6. In a network, max pooling is

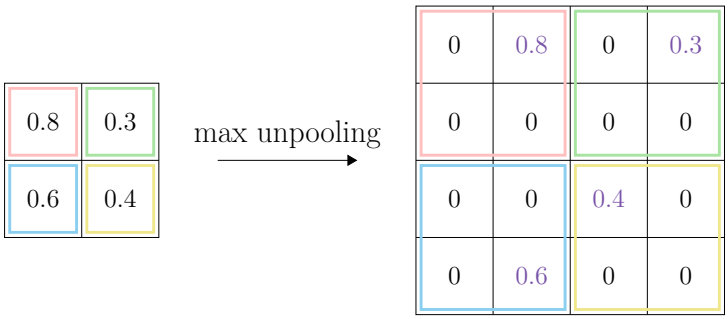


Figure 2.6: Example of max unpooling with stored pooling indices from Fig. 2.4.

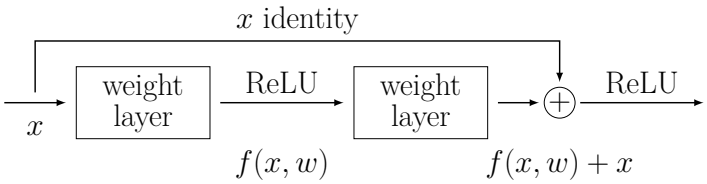


Figure 2.7: Architecture of a residual block. Input x is feet into a weight layer, followed by ReLU activation and another weight layer. The resulting output of these stacked layers is added to identity x (using the shortcut connection).

applied first and the pooling indices of the maximum values per area are stored. After further different layers, the max unpooling is used and the current values are placed at the previously stored indices, while the empty values are filled with zeros (or densified by convolutions). The dimension of the input in max pooling has to be equal to the dimension of the output in max unpooling and vice versa. There are further unpooling methods, e.g. nearest neighbor, where the areas are not filled up with zeros, instead the current values are copied.

Residual Block In many works on visual recognition tasks very deep models, i.e., a high number of used layers, show great performance improvements compared to the the use of more shallow neural networks [48, 49, 96]. However, as depth increases, accuracy becomes saturated and then quickly degrades [60]. This degradation problem is tackled in [61] by introducing deep residual learning. While in common neural networks each layer feeds its output into the next layer, in networks using residual blocks each layer additionally is merged with the output of stacked layers, see Fig. 2.7. These shortcut connections do not add additional parameters or increase computational complexity, they just perform

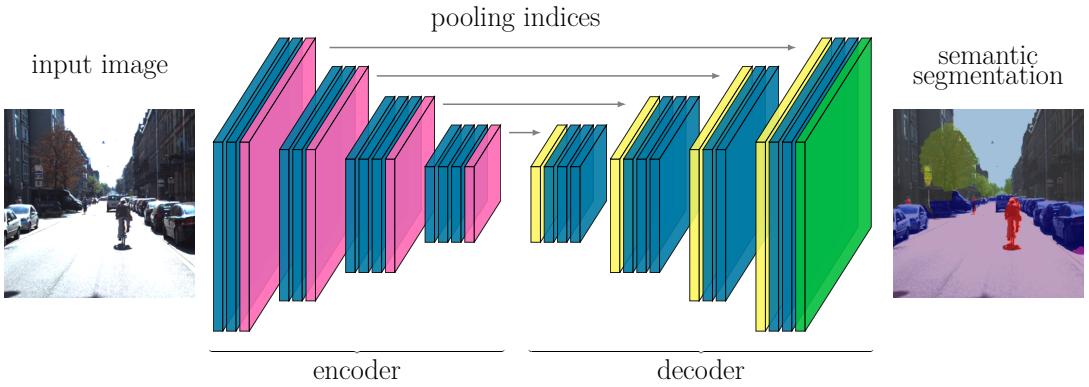


Figure 2.8: Architecture of a semantic segmentation network with an RGB image as input. The convolutional layers (blue) are followed by a pooling step (pink) during downsampling, while the pooling indices are passed to the unpooling step (yellow) during upsampling. Last, the semantic segmentation is obtained using the softmax function (green).

identity mapping. We can perform an identity mapping directly by relying only on skip connections. The stacked layers do not directly fit a desired underlying mapping $\mathcal{R}(x)$, instead these layers explicitly fit a residual mapping of

$$f(x, w) = \mathcal{R}(x) - x \tag{2.50}$$

where the original mapping is modified to $f(x, w) + x$. It is easier to optimize the residual mapping than the original one. As the outputs from previous layers are added to the outputs of stacked layers using skip connections, the training of very deep networks becomes possible. Note, the network architectures presented in [61] using these shortcut connections are called ResNets.

2.1.3 Semantic Segmentation

A fully convolutional network (FCN) is a neural network in which all fully-connected layers are replaced by convolutional layers [96]. The benefits of the FCN compared to feed forward neural networks are the reduced number of weights and the arbitrary dimension of the input. FCNs first perform downsampling (convolution, pooling) for dimension reduction, followed by upsampling (deconvolution, unpooling) to restore the input dimension. Semantic image segmentation is the pixel-wise classification of image content within a pre-defined label space. The objects contained in the image are of interest as well as their position. Fig. 2.8 shows an exemplary network architecture for a semantic segmentation. The pooling indices are stored during downsampling and passed to corresponding unpooling layer in the mirrored upsampling step. The pixel-wise classification as output is

obtained by applying the argmax function pixel-wise to the softmax layer. During training, the cross-entropy loss is applied pixel-wise to prediction (after the softmax layer) and ground truth, and afterwards averaged. This pixel-wise approach corresponds to the assumption that the presence of a class in a pixel is independent of the other pixels.

Since information is lost during downsampling, a possible solution is the use of skip layers. Skip layers combine upsampling with the output of previous layers to retain information of shallow layers for the upsampling step, for example concerning the position of objects. Another development, which is used in many efficient neural network architectures [25, 141, 186], are the so-called *depthwise separable convolutions*. This convolution replaces a full convolution operation by splitting the convolution into two separate layers. The first layer is given by a depthwise convolution where a single convolution kernel is used per channel. The second layer consists of an 1×1 convolution (pointwise convolution). The depth of this kernel corresponds to the number of channels of the input. The pointwise convolution creates linear combinations of the input channels as new features. Contrary, in a full convolution operation the kernel consists of as many channels as the number of dimensions. Due to this split into two layers, the computing effort can be reduced and the networks can be made more efficient.

The MobilenetV2 [141] and the Xception65 network, a modified version of Xception network [25], use this depthwise separable convolutions and are used as backbone networks for the DeepLabv3+ [24], which is a semantic image segmentation network. The architecture of the DeepLabv3+ network is shown in Fig. 2.9. *Atrous convolution* (also known as dilated convolution [183]) generalizes the standard convolution operation and is used to extract features obtained by the backbone network, a deep CNN, with arbitrary resolution by adding space between the values in a kernel. It is defined for each location q by

$$(o_1 *_{\tilde{r}} o_2)(q) = \sum_k o_1(q + \tilde{r}k) \cdot o_2(k) \tag{2.51}$$

where o_1 denotes input, o_2 the kernel and $\tilde{r}$ the atrous rate which corresponds to the stride using to sample the input. For the standard convolution $\tilde{r} = 1$ is valid. The principle of depthwise convolution can also be applied to atrous convolutions. Consequently, the computing effort can be significantly reduced, and the MobilenetV2 as well as the Xception65 network can be used as a backbone. The DeepLabv3+ backbone is followed by the application of the Atrous Spatial Pyramid Pooling module consisting of an 1×1 convolution, three atrous convolutions with different rates and image pooling. Combining with a pointwise convolution, an atrous separable convolution is applied. The decoder module concatenates the upsampled encoder features with the low-level features from the backbone network. Due to the large number of channels in the low-level features, an 1×1 convolution is applied before concatenation to reduce the number of channels. To

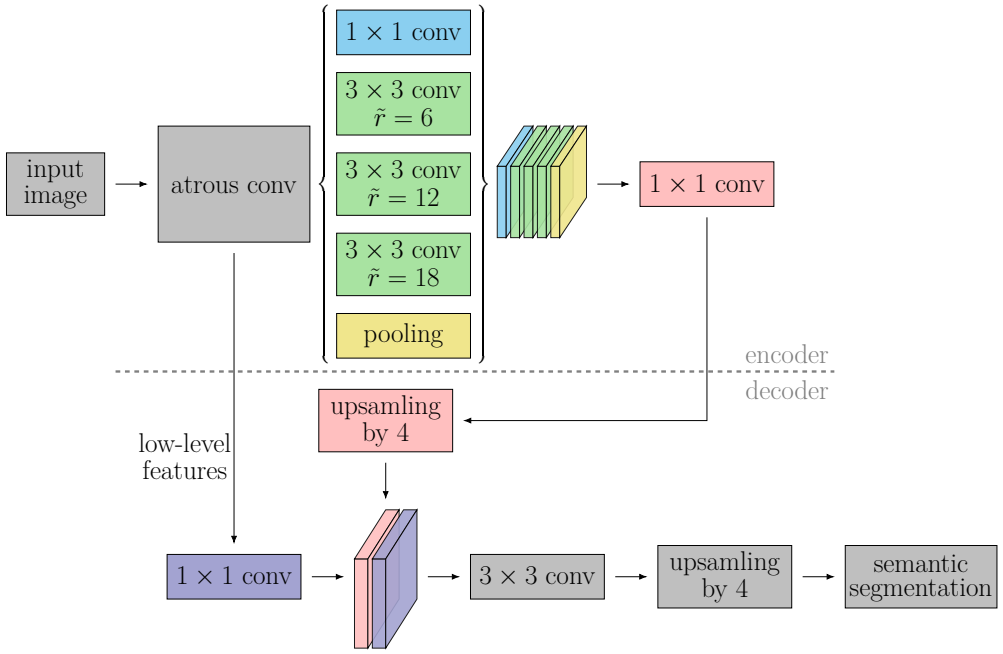


Figure 2.9: Architecture of the DeepLabv3+ network. The input image is feed into a deep CNN (backbone network) including atrous convolutions followed by the Atrous Spatial Pyramid Pooling module. In the decoder, the encoder features are upsampled and concatenated with the low-level features extracted by the backbone network. To obtain the semantic segmentation, a convolution and an upsampling are applied before.

refine the features, 3×3 convolutions are used followed by upsampling to obtain the semantic segmentation.

2.1.4 Instance Segmentation

Object detection describes the task of identifying and localizing objects of a set of given classes. With respect to image data, state-of-the-art approaches are mostly based on convolutional neural networks. On the one hand, localization can be performed by predicting bounding boxes. This corresponds to ordinary object detection [48, 129, 130]. On the other hand, localization can be done by predicting each pixel that corresponds to a given instance. This is also known as instance segmentation, which is an extension of object detection. For this reason, in some instance segmentation networks an already existing object detection model is extended by adding a mask branch [13, 59]. We distinguish between one-stage and two-stage instance segmentation networks. An example of a two-stage network, the Mask-R CNN [59], architecture is given in Fig. 2.10. This network extends Faster R-CNN [130], an object detection network, by adding a branch for the mask prediction to the branch for recognition using bounding boxes. An input

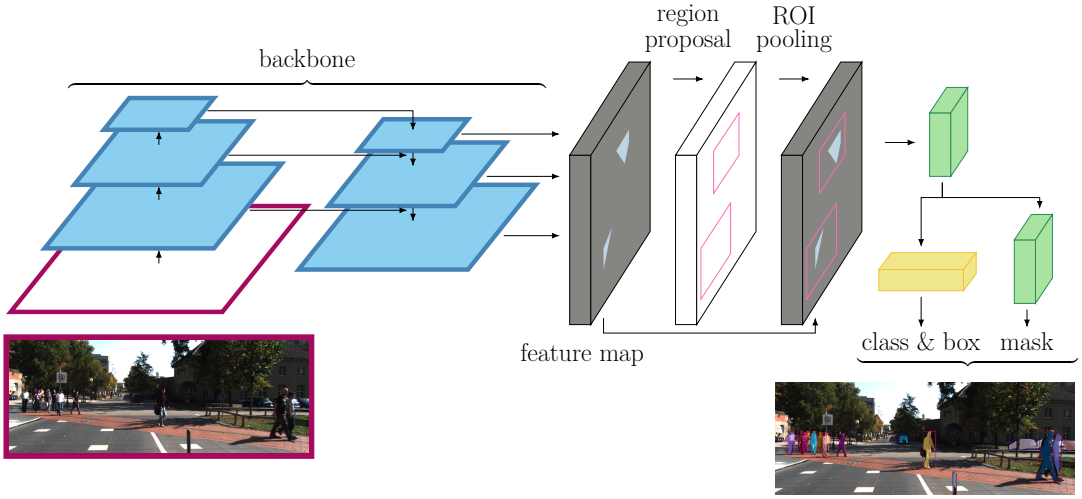


Figure 2.10: Architecture of a two-stage instance segmentation network. The image serves as input for the backbone network (including a FPN) and results in a feature map. Using the feature map, the RPN and ROI pooling, ROIs are obtained. For each ROI there is a branch for the class and box prediction as well as a branch for the mask prediction.

image is fed into the backbone consisting of a standard feature extraction pyramid (for example ResNet50 or ResNet101 [61]) and a feature pyramid network (FPN). The FPN is an improvement over the standard pyramid, which takes the high level features from the first pyramid and passes them to the lower layers. Thus, the features on each level have access to the lower and higher level features. The resulting feature map is scanned by the region proposal network (RPN) that locates areas containing objects, also called *regions of interest* (ROIs). ROI pooling describes the step of cropping a part of the feature map and resizing it to a predefined size. Then, for each ROI, in one branch the class and bounding box coordinates and in another branch the mask representation is predicted by a CNN and merged into an instance segmentation. On the resulting instances the *non-maximum suppression* (NMS) is applied to reduce the predicted instances of one instance to the one with highest confidence. Then a confidence threshold is chosen to discard instances with low confidence values. The detection confidence, also called *score* value, corresponds to the estimated class probability given an instance. During the training, a multi-task loss function is defined for each ROI

$$L = L_{cls} + L_{box} + L_{mask}\,. \tag{2.52}$$

The class loss L_{cls} corresponds to the log loss for the true class. For the bounding box loss L_{box} a robust ℓ_1-loss is applied to the difference of the bounding box coordinates (top left corner, height and width) between prediction and ground

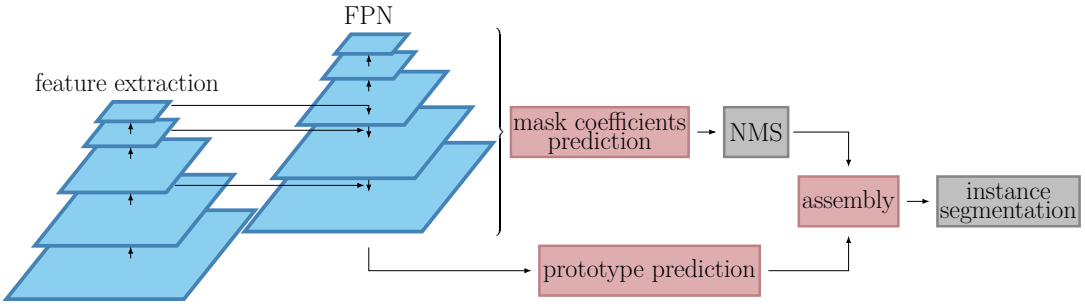


Figure 2.11: Architecture of an one-stage instance segmentation network. An image serves as input for the backbone network consisting of a feature extraction pyramid and a FPN. One branch predicts the prototypes and the other one predicts the mask coefficients and applies a NMS. Both outcomes are merged together and results in the instance segmentation.

truth [48]. The robust ℓ_1-loss is less sensitive to outliers and given by

$$smooth_{\ell_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise}\,. \end{cases} \tag{2.53}$$

The mask loss L_{mask} is defined by the pixel-wise binary cross-entropy between prediction and ground truth.

The presented network structure is a two-stage instance segmentation method. In the first step, ROIs are generated and in the second step, these ROIs are classified and segmented. To obtain real-time speed, an one-stage instance segmentation approach is introduced in [13], the YOLACT network, where the localization step is omitted. While the YOLO network [129] is well-known for its real-time speed in object detection, the YOLACT network fulfills this property analogously for instance segmentation. The architecture of this one-stage network is given in Fig. 2.11. The instance segmentation is divided into two simpler and parallel executable tasks after the backbone. The first branch uses a fully convolutional network to produce a set of prototype masks (similar semantic segmentation masks) of the same size as the input image. Visually and spatially similar instances are depicted in the prototypes which number is independent of the number of classes. In contrast to semantic segmentation, this model has no loss on the prototypes, instead the loss function L (2.52) is applied to the final mask after assembly. The second branch adds a prediction of a mask coefficients vector for each prototype to an object detection (bounding box and class prediction). Afterwards, a NMS is applied. The instance masks are formed by linear combinations of prototypes and coefficients. The resulting instance segmentation is obtained by using a confidence threshold. This confidence (score) value reflects how confident the model is about its instance prediction with corresponding class probability and how well the predicted instance fits the ground truth instance.

2.1.5 Depth Estimation

Depth estimation is another computer vision task where the goal is to obtain a representation of the spatial structure of a scene, reconstructing the appearance and the three-dimensional shape of objects in the image. Applications of depth estimation are for instance in automated driving [56], scene understanding and robotics [34]. For measuring the distance relative to a camera, monocular or stereo images are often considered using supervised, semi-supervised or self-supervised methods. We focus on supervised depth estimation from monocular images. The seminal work for this task is [142]. The depth is estimated from monocular cues, like size and texture, in images by supervised learning minimizing a regression loss.

Using neural networks for depth estimation is gaining popularity. The depth is predicted by learning depth cues through gradient-based methods. An example of a deep CNN for supervised depth estimation from monocular images is presented in [90]. This network utilizes Local Planar Guidance (LPG) layers at multiple stages of the decoding phase for an effective guidance of densely encoded features. The network architecture is shown in Fig. 2.12. The backbone network is used as a dense feature extractor (DFE) and followed by a denser version [177] of the Atrous Spatial Pyramid Pooling module (see Sec. 2.1.3) with dilation rates $\tilde{r} \in \{3, 6, 12, 18, 24\}$. In general, the encoding-decoding scheme reduces the feature map resolution to $h/8$ and then returns to the original resolution h for dense prediction. During the decoding phase, LPG layers are employed to locate geometric guidance to the desired depth estimation at each stage. For the finest estimation, an 1×1 reduction layer is used. The outputs of the different layers are concatenated and fed to a convolutional layer to produce the final depth estimation.

The main idea for the Local Planar Guidance layer is to define direct and explicit relations between the final output and the internal features in an effective way. This layer is not intended to directly estimate the depth values for the particular scale, instead in combination with the other LPG layers and the reduction, each output contributes to the final depth estimate through the last convolutional layer. A $k \times k$ LGP layer starts with 1×1 convolutions to obtain a feature map of size $h/k \times h/k \times 3$. To create a unit normal vector (n_1, n_2, n_3), the first two channels of this feature map are considered as the polar θ and azimuthal ϕ angles and converted by

$$n_1 = \sin(\theta)\cos(\phi)\,, \quad n_2 = \cos(\phi)\,, \quad n_3 = \sin(\theta)\sin(\phi)\,. \tag{2.54}$$

The last channel of the feature map defines the perpendicular distance n_4 between plane and origin using the sigmoid function. To guide features, these local plane

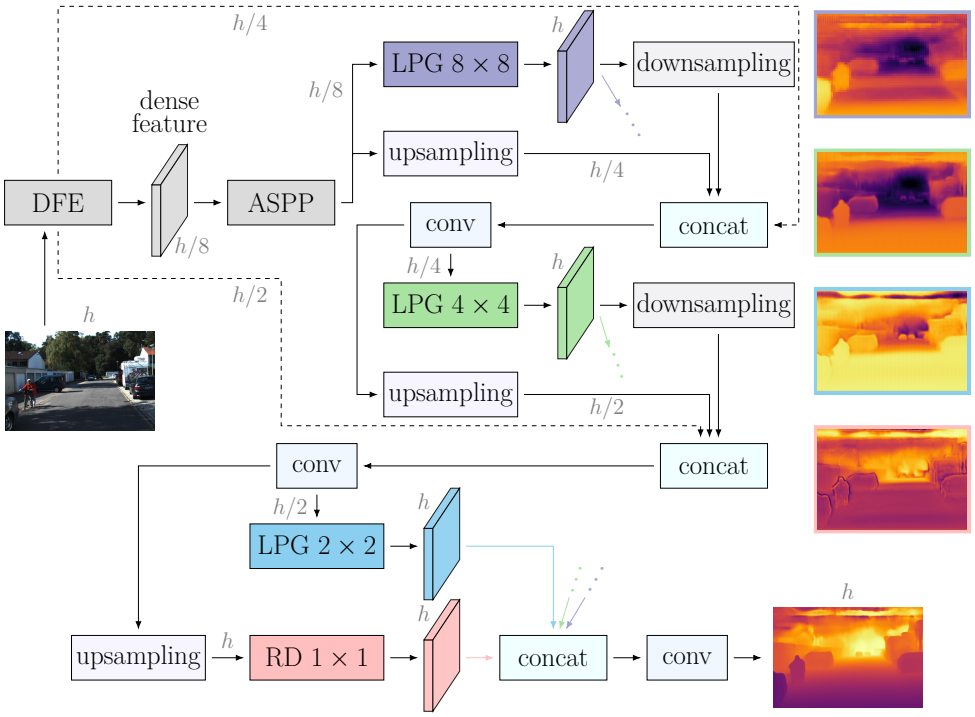


Figure 2.12: Architecture of a depth estimation network. The input image is fed into the feature extractor which is followed by a Atrous Spatial Pyramid Pooling (ASPP) module for contextual information extraction. In the decoding phase, the LPG layers output estimations with full spatial resolution h which are concatenated for final depth estimation. In addition, skip layers (dashed lines) link the base network and internal outputs with corresponding spatial resolutions.

coefficient estimations $n_i, i = 1, \dots 4$, are adopted to $k \times k$ local depth cues using the ray-plane intersection

$$\frac{n_4}{n_1 u_z + n_2 v_z + n_3} \tag{2.55}$$

where (u_z, v_z) are $k \times k$ normalized and patch-wise coordinates of pixel z. This results in locally-defined relative depth estimations by using only four parameters. As features at the same spatial position in different stages are used to predict the final depth, the global shapes can be learned at coarser scales and local details at finer scales to obtain an efficient representation.

The loss function used for training is based on the *scale invariant logarithmic error* (silog) [38]

$$L_s = \frac{1}{Z} \sum_z (\log \hat{y}_z - \log y_z)^2 - \frac{\lambda}{Z^2} (\sum_z \log \hat{y}_z - \log y_z)^2 \tag{2.56}$$

where Z denotes the number of pixels of an image having valid ground truth values, y_z the depth ground truth and $\hat{y}_z$ the depth prediction per pixel z. This

definition is specific for a single image. For an overall silog metric, the mean over all training images is considered. Higher values of λ focus the network on minimizing variance of the error (here $\lambda = 0.85$ is used) since the loss can be reformulated to the sum of the variance and a weighted squared mean of the error in log space. The training loss used in [90] is defined as

$$L = 10\sqrt{L_s} \tag{2.57}$$

since scaling of the loss function range improves convergence and the final training result. Note, the silog also serves as performance measure for the quality of depth estimation networks. The definition of this measure corresponds to the loss function (2.56) without the parameter λ.

2.1.6 Evaluation Metrics

There are different evaluation metrics for neural networks. In semantic segmentation, the mean intersection over union and in object detection as well as instance segmentation, the mean average precision are often used as performance measures. These metrics are possibilities to assess the prediction quality of neural networks.

Intersection over Union The intersection over union (IoU, also known as Jaccard index [72]), is a commonly used performance measure for semantic segmentation. The $IoU \in [0, 1]$ quantifies the degree of overlap of prediction and ground truth, it is equal to zero if and only if any predicted pixel does not intersect with the ground truth. In Fig. 2.13 the object-level definition of the IoU is given, as well as a few examples. To evaluate the prediction performance of a neural network in semantic segmentation, the mean IoU is used. To this end, the IoU is calculated for each class on pixel-level (not, as shown in the examples, on object-level) and then averaged over classes.

Mean Average Precision The calculation of the average precision [40] is illustrated by the following example. In Fig. 2.14, different ground truth and predicted objects are given in two images. In object detection, each predicted object receives a score value. This value describes the confidence of the network prediction for an object. The score values associated with the predicted objects are given in Table 2.1. The predicted objects are sorted by the score value and matched in this order with ground truth objects using the IoU on object-level. The prediction p_4 has an IoU value equal to or greater than 0.5 with ground truth g_3 and thus, p_4 is considered as a true positive with matched ground truth g_3. The next prediction p_3 also has a large overlap with ground truth g_3, however, once

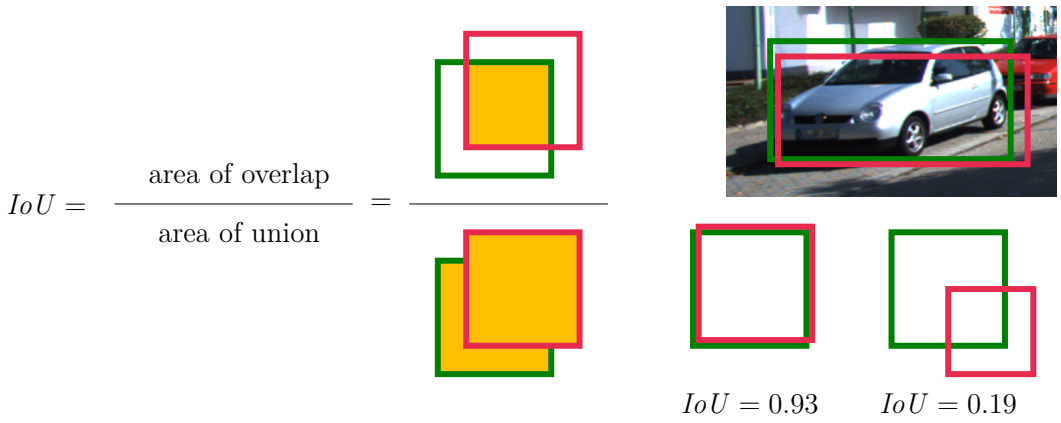


Figure 2.13: Intersection over union on object-level. Green boxes correspond to ground truth (provided by humans) and red boxes to predictions obtained by neural networks.

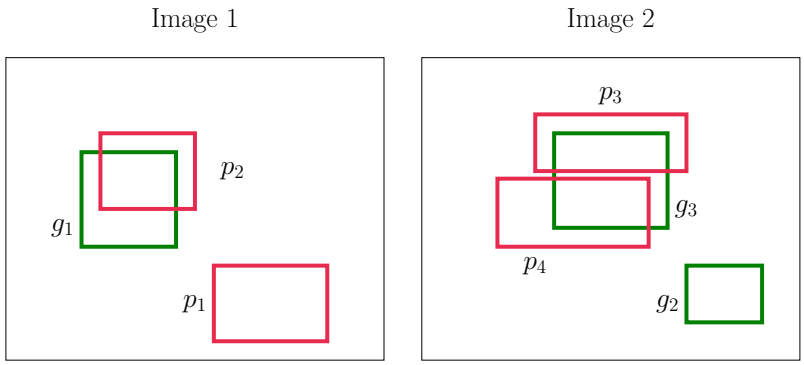


Figure 2.14: Two input images for average precision calculation. Green boxes correspond to ground truth objects g_i, $i = 1, 2, 3$, and red boxes to predicted objects p_i, $i = 1, \ldots, 4$.

the ground truth object is matched, it cannot be matched with other predictions. Therefore, p_3 is treated as false positive. To match a prediction with a ground truth object, the IoU has to be greater than or equal to a threshold, here 0.5. To calculate the precision and recall in the next step, the true and false positives are accumulated. The precision is the percentage of correct positive predictions to all predicted objects and is given by

$$\text{precision} = \frac{\text{true positives}}{\text{all predicted objects}}\,. \tag{2.58}$$

The recall is the percentage of true positive objects detected to all ground truth objects

$$\text{recall} = \frac{\text{true positives}}{\text{all ground truth objects}}\,. \tag{2.59}$$

Plotting the precision and recall values for varying thresholds, we are able to measure the degree of detection ability, and the resulting *precision-recall curve*

Table 2.1: From left to right the following data is given: image number, predicted object, associated score value, matched ground truth object, IoU, true positive or false positive, accumulated true positives, accumulated false positives, precision and recall.

image	prediction	score	gt	IoU	tp/fp	acc tp	acc fp	precision	recall
2	p_4	0.97	g_3	≥ 0.5	tp	1	0	1	0.33
2	p_3	0.86	g_3	≥ 0.5	fp	1	1	0.5	0.33
1	p_2	0.73	g_1	≥ 0.5	tp	2	1	0.67	0.67
1	p_1	0.57	–	< 0.5	fp	2	2	0.5	0.67

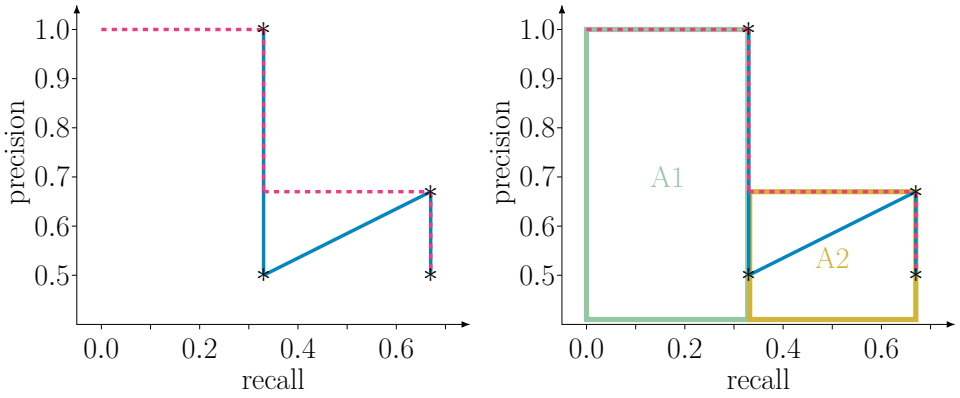


Figure 2.15: *Left*: The precision and recall values are given by asterisks and the corresponding precision-recall curve corresponds to the blue line. The pink line represents the interpolation of this curve. *Right*: The area under the curve is divided into two areas, A1 and A2.

is shown in Fig. 2.15 (left). The precision-recall curve evaluates the performance of an object detection network. A strong performance is achieved when both measures are high, and to provide an overall measure of different networks, the *area under precision-recall curve* ($AUPRC$) is calculated. The average precision can be interpreted as an approximated $AUPRC$ by interpolating the given points. The intention is to reduce the effects of wobbling in the curve. The n points of the curve are interpolated as follows

$$\sum_{n}(r_{n+1} - r_n) \max_{\hat{r} \geq r_{n+1}} p(\hat{r}) \tag{2.60}$$

where $p(\hat{r})$ is the corresponding precision value to recall $\hat{r}$. By applying this equation, we obtain the two areas that are given in Fig. 2.15 (right) as estimated $AUPRC$. By calculating the total area, we achieve an average precision value of $AP = \text{A1} + \text{A2} = 0.56$. The mean average precision (mAP) results of the average over all classes.

Figure 2.16: *Top*: Pixel-wise entropy heatmap obtained by the softmax output of a semantic segmentation network. *Bottom*: Two predictions of an instance segmentation network.

2.2 Uncertainty Quantification

Semantic segmentation, i.e., the pixel-wise classification of image content, and instance segmentation, i.e., the identification and localization of objects of a set of given classes represented by pixel-wise masks, are important tools for scene understanding. With respect to image data, state-of-the-art approaches are mostly based on convolutional neural networks. In recent years, CNNs have demonstrated outstanding performance for these two tasks. There is the drawback that neural networks as statistical models produce probabilistic predictions prone to error, and it is necessary to understand these errors. Prediction quality estimates [33, 136] as well as uncertainty quantification [44] of neural networks are of highest interest in safety critical applications like medical diagnosis [120, 169] and automated driving [70, 87]. However, semantic segmentation networks do not provide uncertainty values by default. In Fig. 2.16 (top), the pixel-wise entropy applied to the softmax output of a semantic segmentation network is shown providing pixel-wise uncertainties. We observe segment filling dispersion caused by a misspecified loss function as neighboring pixels are assumed to be independent. This results in a non-convergence of the uncertainty. In instance segmentation networks such as YOLACT [13] as well as Mask R-CNN [59] uncertainty estimations are given, however, these are not well adjusted. In [53], it is observed that an increasing model capacity and a lack of regularization are correlated with the miscalibration of the model. The instance segmentation networks provide a confidence value (the opposite of uncertainty), the score value, for each instance which can have

high values for false predictions and low ones for correct predictions. In Fig. 2.16 (bottom left), an uncertain prediction of a rider is shown where many instances are predicted, but only with low confidences. In contrast, in Fig. 2.16 (bottom right), an example of the overconfidence of a neural network is given. The stand with bags is predicted to be a person with a score value of 95%, while the real person in the background has a confidence of only 30%. During inference of instance segmentation networks, all instances with score values below a threshold are removed. It can happen that correctly predicted instances disappear, while many false positives remain. For these reasons, different methods are developed which provide uncertainty estimates for segmentation neural networks.

We distinguish between two types of uncertainty, *epistemic* uncertainty caused by a lack of knowledge and *aleatoric* uncertainty due to inherent randomness [71]. Aleatoric (also called statistical) uncertainty refers to the variability in the outcome of an experiment that is due to inherent random effects. A typical example of aleatoric uncertainty is coin flipping. Even the best fitting model cannot provide a definite answer, head or tail, only the probabilities of both outcomes can be determined. As a consequence, this type of uncertainty cannot be reduced by any additional source of information due to the stochastic component. On the other hand, epistemic (also called systematic) uncertainty is caused by a lack of information, e.g. about the best fitting model. For instance, when asked about the meaning of the italian word "testa", the possible answers can be head or tail, same as for the coin flipping. In contrast to uncertainty caused by randomness like coin flipping, in this example the uncertainty can be reduced by using additional information. Consequently, epistemic uncertainty refers to the reducible part of uncertainty and is considered throughout this work.

Related Work A very important type of uncertainty is the model uncertainty resulting from the fact that the ideal model parameters are unknown and have to be estimated from data. Bayesian models are one possibility to consider these uncertainties [103]. Therefore, different frameworks based on variational approximations for Bayesian inference exist [4, 37]. Recently, Monte-Carlo (MC) Dropout [44] as approximation to Bayesian inference has aroused a lot of interest. In classification tasks, the uncertainty score can be directly determined on the network's output [44]. Threshold values for the highest softmax probability or threshold values for the entropy of the classification distributions (softmax output) are common approaches for the detection of false predictions (false positives) of neural networks, see e.g. [63, 93]. Uncertainty metrics like classification entropy or the highest softmax probability are usually combined with model uncertainty (MC Dropout inference) or input uncertainty, see [44] and [93], respectively. Alternatively, gradient-based uncertainty metrics are proposed in [119], and an alternative to Bayesian neural networks is introduced in [86] where the idea of ensemble

learning is used to determine uncertainties. Some of these uncertainty measures have also been transferred to semantic segmentation tasks, such as MC Dropout [89], which also achieves performance improvements in terms of segmentation accuracy [78]. The works presented in [76] and [169] also make use of MC Dropout to model uncertainty and filter out predictions with low reliability. This line of research is further developed in [70] to detect spatial and temporal uncertainty in the semantic segmentation of videos. Based on MC Dropout, structure-wise metrics are presented in [138] as well as voxel-wise uncertainty metrics based on maximum softmax probability in [66]. In semantic segmentation tasks, the concepts of *meta classification* and *meta regression* are introduced in [136]. Meta classification refers to the task of predicting whether a predicted segment intersects with the ground truth or not. To this end, the object-wise intersection over union, as performance measure for semantic segmentation, is considered. The meta classification task corresponds to (meta) classifying between $IoU = 0$ and $IoU > 0$ for every predicted segment. Meta regression is the task of predicting the IoU for each predicted segment directly. The main aim of both tasks is to have a model that is able to reliably assess the prediction quality of a semantic segmentation obtained from a neural network. As input both methods use segment-wise metrics extracted from the segmentation network's softmax output. This idea is extended in [137] by adding resolution dependent uncertainty, also applied to semantic segmentation, and in [144] to object detection. Similar works on a single object per image basis are introduced in [33] and [68], instead of hand crafted metrics they utilize additional CNNs. In [39], performance measures for the segmentation of videos are introduced, these measures are also based on image statistics and can be calculated without ground truth. Some methods for uncertainty evaluation are transferred to object detection, such as the MC Dropout [81]. False positive detection is presented based on MC Dropout and an ensemble approach in [120]. In [87], false positive objects are assigned high uncertainties using two methods, loss attenuation and redundancy with multi-box detection. Based on a dropout sampling approach for object detection [110], the work of [112] investigates the semantic and spatial uncertainty in instance segmentation.

In object detection and instance segmentation, the problem of miscalibrated score values (gap between prediction confidence and network accuracy) is addressed by confidence calibration [53]. For single object localization, a Platt scaling [125] inspired method is introduced in [124] where the predictions are recalibrated in a post-processing step. A modification of the standard softmax layer varying the probabilistic confidence score is presented in [117] for the object detection task. In [85], additional information of the regression output is employed to estimate calibrated confidences with respect to image location and bounding box size. An uncertainty-aware neural network is proposed in [185] using smoothed labels with respect to pixel-wise uncertainty and a relaxed sigmoid function to produce calibrated output. For long-tailed (many categories with few training

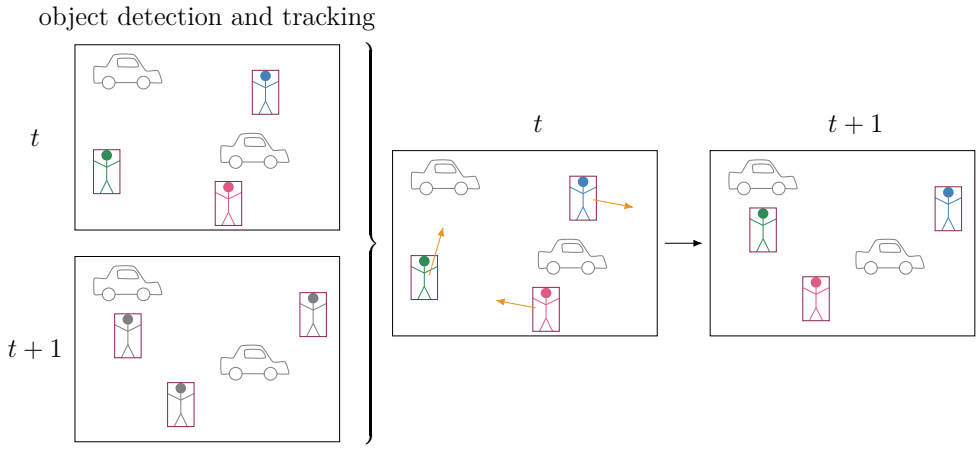


Figure 2.17: Multiple object tracking. Object detection is performed in frame t as well as in $t+1$ and a tracking algorithm is applied to the objects found (here pedestrians). The objects in frame t assign random IDs (given by different colors). In the right panel the results of the tracked objects are shown.

samples) object detection and instance segmentation, a post-processing method reweighs the predicted scores of each class by its training sample size to obtain calibrated confidence scores [121]. The task of medical image segmentation is tackled in [107] which is based on model ensembling for confidence calibration of fully convolutional networks using batch normalization.

2.3 Object Tracking

While most works, like those presented before, focus on uncertainty quantification for single frames, there is often video data available. To use the benefit of videos over single frames, object tracking methods can be applied. Object tracking is an essential task in video applications, such as automated driving, robot navigation and many others [109]. The task of object tracking consists of detecting objects and then propagating the location of objects to subsequent frames, eventually studying their behavior [182]. There are two levels of object tracking, *single* object tracking (SOT) and *multiple* object tracking (MOT). In SOT, the appearance of the object is known and the task is to track this object using the information from previous frames, while MOT requires the detection of multiple objects in the frames, also objects which leave or enter frames. An illustration for two frames is given in Fig. 2.17. The main difficulty in tracking multiple objects simultaneously is caused by occlusions and interactions between objects. An example is shown in Fig. 2.18. The objects are represented by bounding boxes and pixel-wise masks. The individual IDs of the objects are given by the different colors. For single as

Figure 2.18: An example of multiple object tracking, where pedestrians are tracked.

well as multiple object tracking different methods are proposed.

2.3.1 Related Work

In this section, we discuss different approaches for single and multiple object tracking.

Single Object Tracking In most works, the target object is represented as an axis-aligned [170] or rotated [82] bounding box such as in the following approaches. A popular strategy for single object tracking is the tracking-by-detection approach [5]. A discriminative classifier is trained online while performing the tracking to separate the object from the background only by means of the information where the object is located in the first frame. Another approach for tracking-by-detection uses adaptive correlation filters that model the targets appearance, the tracking is then performed via convolution [12]. In [32] and [160], the trackers based on correlation filters are improved with spatial constraints and deep features, respectively. Another object tracking algorithm [113] combines Kalman filters and adaptive least squares to predict occluded objects where the detector shows deficits. In contrast to online learning, there are also tracking algorithms that learn the tracking task offline and perform tracking as inference, only. The idea behind these approaches [10, 62] is to train a similarity function on pairs of video frames offline instead of training a discriminative classifier online. In [10], a fully-convolutional siamese network is introduced. This approach is improved by making use of region proposals [91], angle estimation and spatial masking [58] as well as memory networks [178]. Another approach for single object tracking with bounding boxes is presented in [179] where semantic information is used for tracking. Most algorithms and also the ones described here use bounding boxes, mostly for initializing and predicting the position of an object in the subsequent frames. In contrast, [28] uses coarse binary masks of target objects instead of rectangles. There are other procedures that initialize and/or track an object without

bounding boxes, since a rectangular box does not necessarily capture the shape of every object well. In [74], a temporal quad-tree algorithm is applied, where the objects are divided into squares getting smaller and smaller. Other approaches use semantic image segmentation such as [55], where the initialization includes a segmentation for predicting object boundaries. A segmentation-based tracking algorithm is presented in [35] based on an adaptive model. The approaches presented in [6] and [150] are also based on segmentation and use particle filters for the tracking process. There is also a superpixel-based approach [181] that creates binary masks and starts from a bounding box initialization.

Multiple Object Tracking The tracking task (association problem) in [191], [149] and [187] is solved by dual matching attention networks, a CNN using quadruplet losses and a CNN based on correlation filters, respectively. Another approach for object association involves finding a network flow via backpropagation [145]. The previously described algorithms use one model for the object detection and another for the association problem. The work of [168] presents a shared model for both tasks. A focus on the improvement of long-term appearance models is demonstrated in [79] based on a recurrent network. Most works, including those described here, work with bounding boxes, while there are other methods that use a binary segmentation mask representation of the object [166]. Starting from a bounding box initialization, binary masks are created based on a fully-convolutional siamese approach [166]. In [2] and [123], segmentation and tracking are jointly solved using a pixel-level probability model and a recurrent fully convolutional network, respectively. To perform the detection, segmentation and tracking tasks simultaneously, the Mask R-CNN network is extended by a tracking branch [176], by 3D convolutions to incorporate temporal information [162] and by a mask propagation branch [9]. In contrast, the sub-problems classification, detection, segmentation and tracking are treated independently in [97]. Another work for multiple object tracking is based on the optical flow and the Hungarian algorithm [20].

2.3.2 Evaluation Methods

There exists different metrics for the evaluation of object tracking methods, some of them are described in the following. These metrics are designed to evaluate not only the tracking process, but also the prediction of the objects by the neural network. To this end, we define for each frame t of an image sequence (with a length of T) the ground truth objects by $\{g_1, \ldots, g_n\}$ and the predicted objects by $\{p_1, \ldots, p_m\}$. In Fig. 2.19 some cases for matches and non-matches between ground truth and predicted objects are shown. Mismatches are incremented as the associated predicted object changes. The same applies to switches between

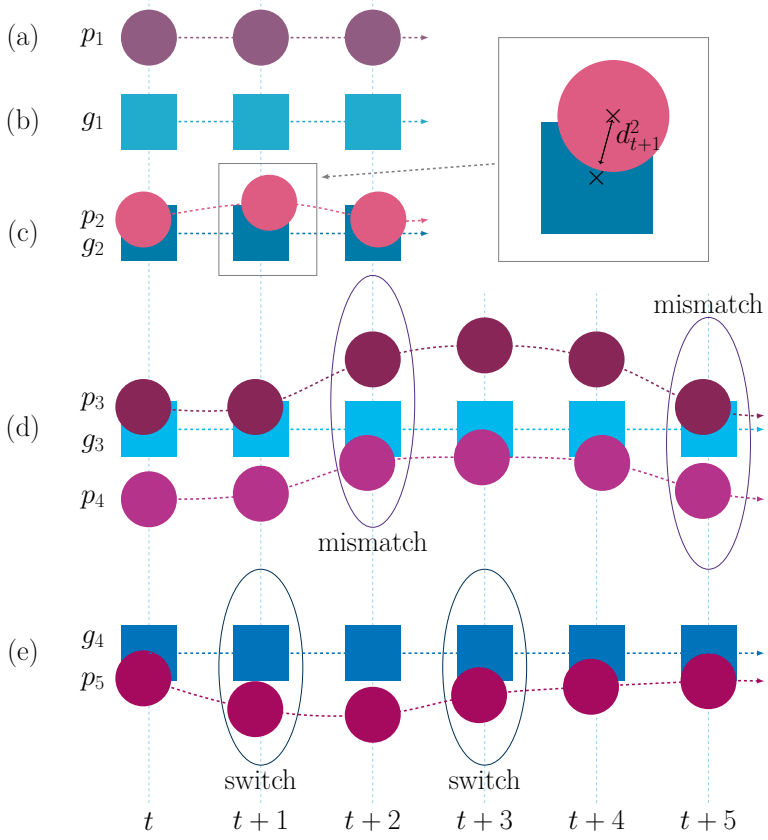


Figure 2.19: Optimal matches and error measures between ground truth and predicted objects. (a): False positives. (b): False negatives. (c): Matched ground truth g_2 and predicted object p_2 (true positives). The zoom shows the distance between the geometric centers of the two objects. (d): Mismatch error. First, g_3 and p_3 are matched. After two frames, there is another match with p_4 and after another three frames, it switches back to p_3. We count two mismatch errors in this case. (e): First, g_4 and p_5 are matched for one frame. In the next two frames, the objects are not close enough to be a match before they are matched ones again. There are two cases of a switch between matched and non-matched for the ground truth object g_4.

matched and non-matched, it will count up after the first match as soon as the status changes.

For the tracking evaluation metrics, we define some terms in frame t:

- gt_t: number of ground truth objects
- fn_t: number of false negatives
- fp_t: number of false positives
- tp_t: number of true positives
- mme_t: number of mismatch errors
- smn_t: number of switches between matched and non-matched
- d_t^i: distance between geometric centers of the ground truth object g_i and its associated predicted object

Two well-known object tracking metrics are *MOTP* and *MOTA* [8]. The multiple object tracking precision

$$MOTP = \frac{\sum_{t=1}^{T} \sum_i d_t^i}{\sum_{t=1}^{T} tp_t} \tag{2.61}$$

is the distance between geometric centers of matched pairs of ground truth and predicted objects, averaged by the total number of matches. The multiple object tracking accuracy

$$MOTA = 1 - \frac{\sum_{t=1}^{T} (fn_t + fp_t + mme_t)}{\sum_{t=1}^{T} gt_t} \tag{2.62}$$

is based on three error ratios: the ratio of false negatives

$$\overline{fn} = \frac{\sum_{t=1}^{T} fn_t}{\sum_{t=1}^{T} gt_t}\,, \tag{2.63}$$

the ratio of false positives

$$\overline{fp} = \frac{\sum_{t=1}^{T} fp_t}{\sum_{t=1}^{T} gt_t} \tag{2.64}$$

and the ratio of mismatches over the total numbers of ground truth objects in all frames

$$\overline{mme} = \frac{\sum_{t=1}^{T} mme_t}{\sum_{t=1}^{T} gt_t}\,. \tag{2.65}$$

The following metrics are focused on the evaluation of object detection. The

$$\text{precision} = \frac{\sum_{t=1}^{T} tp_t}{\sum_{t=1}^{T} (tp_t + fp_t)} \tag{2.66}$$

relates the true positives to all predicted objects and the

$$\text{recall} = \frac{\sum_{t=1}^{T} tp_t}{\sum_{t=1}^{T}(tp_t + fn_t)} \tag{2.67}$$

relates the true positives to all ground truth objects [182]. The F-measure [22] is a combination of recall and precision, defined by

$$\text{F-measure} = \frac{\sum_{t=1}^{T} 2 \cdot tp_t}{\sum_{t=1}^{T}(2 \cdot tp_t + fn_t + fp_t)} \,. \tag{2.68}$$

The *FAR* metric [109] is a different representation of the sum of false positives and is given by

$$FAR = \frac{\sum_{t=1}^{T} fp_t}{T} \cdot 100 \,. \tag{2.69}$$

These described measures are summed up over all frames of an image sequence. The metrics described in the following do not describe the ground truth objects frame-wise, instead a ground truth object can occur in several frames and is assigned a common ID. We define by GT all ground truth objects of an image sequence which are identified by different IDs. We divide these ground truth objects into three cases: mostly tracked MT, partially tracked PT and mostly lost ML (see [109]). An object is mostly tracked if it is tracked for at least 80% of frames, out of the total number of frames in which it appears. It is independent whether the ground truth object is matched with one same predicted object or different predicted objects. An object is mostly lost if it is only tracked for less than 20% and the other cases are partially tracked. We focus on the object tracking metrics described here as these are well-known.

2.4 Classification and Regression Methods for the Prediction of the IoU

For the quality evaluation of neural networks, which we discuss in detail in Chapter 3 and Chapter 4, we use the object-wise intersection over union as a performance measure. In Chapter 3, we consider the task of (meta) classifying between $IoU = 0$ and $IoU > 0$ for every predicted object obtained by a semantic segmentation to identify false positive segments. In Chapter 4, we consider the problem of instance segmentation and thus, we classify between $IoU < 0.5$ and $IoU \geq 0.5$ for every predicted instance. Moreover, we predict the IoU for each predicted object directly (meta regression) to evaluate the quality of a segmentation obtained from a neural network. In the following, we consider different classification and regression methods, such as linear models, gradient boosting as well as shallow neural networks with corresponding evaluation metrics [57].

2.4.1 Linear Regression

A general linear regression model is given by

$$y = x\beta + \epsilon \tag{2.70}$$

where y denotes the true IoU values, x the input variables, β the coefficients and ϵ the error term. The task of fitting a linear model is to minimize the residual sum of squares and thus, to estimate the coefficients

$$\min_{\beta}(\|y - x\beta\|_2^2 + \lambda \cdot \Omega(\beta))\,. \tag{2.71}$$

To prevent overfitting a regularization term $\Omega(\beta)$ with parameter λ is added (see Sec. 2.1.1, paragraph Regularization). Using ℓ_1-penalization, the coefficient vector is thinned out, i.e., coefficients are set to zero, while using the ℓ_2-penalization, large coefficients are reduced.

A linear model to binary classification is the logistic regression. The logistic regression is fitted by

$$\min_{\beta}(\sum_{i} -y_i \log(\vartheta(\beta^\top x_i)) - (1 - y_i)\log(1 - \vartheta(\beta^\top x_i))) \tag{2.72}$$

where

$$y_i = \begin{cases} 0, & \text{if } IoU = 0 \\ 1, & \text{if } IoU > 0\,, \end{cases} \tag{2.73}$$

x_i denotes the i^{th} input variable and $\vartheta(\cdot)$ the logistic function (equal to the sigmoid function, see (2.3)). The *least absolute shrinkage and selection operator* (LASSO [157]) method makes use of ℓ_1-penalization and is given by

$$\min_{\beta}(\sum_{i} -y_i \log(\vartheta(\beta^\top x_i)) - (1 - y_i)\log(1 - \vartheta(\beta^\top x_i)) + \lambda\|\beta\|_1) \tag{2.74}$$

where λ denotes the regularization parameter. The LASSO method investigates the predictive power of different combinations of input variables and serves as feature selector [128, 158, 167]. Besides the standard LASSO method, there exists also modifications such as using Bayesian models [122] or support vector machines [135].

2.4.2 Gradient Boosting

Gradient boosting is a learning technique for classification and regression tasks that generates a prediction model in the form of an ensemble of weak prediction models [42, 43, 105]. Typically, binary decision trees are used as models. Decision

Algorithm 2.1: Gradient boosting

initialize model $f_0(x) = \arg\min_{\beta} \sum_{i=1}^{N} L(y_i, \beta)$

for $m = 1, \ldots, M$ **do**

  for $i = 1, \ldots, N$ **do**

    $r_{im} = -\dfrac{\nabla L(y_i, f_{m-1}(x_i))}{\nabla f_{m-1}(x_i)}$

  end

  fit a regression tree to the pseudo-residuals r_{im} giving regions $R_{jm}, j = 1, \ldots, J_m$

  for $j = 1, \ldots, J_m$ **do**

    $\beta_{jm} = \arg\min_{\beta} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \beta)$

  end

  update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \beta_{jm} \cdot I(x \in R_{jm})$ where $I(x \in R_{jm}) = \{1 \text{ if } x \in R_{jm}, 0 \text{ else } \}$

end

output $\hat{f}(x) = f_M(x)$

trees form a hierarchical structure from a set of training data with as few decision paths as possible. This is done in a recursive way, starting at the root of the tree, until the best and distinct class assignment is found based on the features. Gradient boosting creates a strong learner by means of additive base learners like decision trees. The approach is described in Algorithm 2.1. At each iteration m, $m = 1, \ldots, M$, a subsample $\{(x_i, y_i)\}_{i=1,\ldots,N}$ is randomly generated of the training data. The gradients of the loss function L to be minimized are called pseudo-residuals. Then the regression tree is fitted to these pseudo-residuals. The decision tree is partitions into J_m disjoint regions $\{R_{jm}\}_{j=1,\ldots,J_m}$ where the number J_m depends on each iteration m. By minimizing the loss function, the coefficients β for the individual regions are obtained. Subsequently, the model can be updated using these coefficients β. In summary, gradient boosting constructs an additive regression model by fitting base learners in each iteration. For classification and regression tasks, different loss functions are applied. For example, logistic loss can be used for classification and squared loss for regression tasks. The possibility to optimize different loss functions and the various hyperparameters available allow gradient boosting to be quite flexible. Furthermore, this method can work with different data like categorical or numerical values and also handle missing data.

2.4.3 Shallow Neural Networks

In Sec. 2.1, neural networks are introduced. Neural networks can be applied to different tasks, especially shallow neural networks can be used for classification and regression. In the following, we focus on the architectures considered for meta classification and regression. For classification, the network contains only one hidden layer with a small number of neurons (50 ones). As activation function the sigmoid function is used and as regularization a ℓ_2-penalization. The shallow neural network considered for regression also contains only one hidden layer (with 50 neurons). A ReLU activation function is applied as well as ℓ_1- and ℓ_2-penalization, respectively. Classification and regression are simple problems compared to, for example, object detection and semantic image segmentation. For this reason and to prevent overfitting, only shallow neural networks are sufficient for these meta classification and regression tasks.

2.4.4 Performance Measures

For the evaluation of the prediction quality of classification and regression models, we introduce two metrics for each case.

Classification Measures To evaluate the prediction of a binary classification model, the *accuracy* is often considered. The accuracy is the number of correctly predicted observations divided by the total number of observations.

Another commonly used quality measure for the prediction of classification models is the *area under receiver operating characteristics curve* ($AUROC$) [41]. To this end, we apply the true categorized variable $y_k = \{0 \text{ if } IoU = 0, 1 \text{ if } IoU > 0\}$ and the predicted variable

$$\hat{y}_k = \begin{cases} 0, & \text{if } \hat{p}_k > \tilde{s} \\ 1, & \text{else} \end{cases} \tag{2.75}$$

where $\tilde{s} \in [0, 1]$ is a threshold and $\hat{p}_k \in [0, 1]$ denotes the predicted probability for an object k to be a false positive. The $AUROC$ is based on the

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false negatives}} \tag{2.76}$$

(also called true positive rate) and on the

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{false positives}} \tag{2.77}$$

(also called true negative rate). The ROC is obtained by varying the decision threshold $\tilde{s}$ in a binary classification problem, here for the decision between $IoU =$

0 and > 0 (or $IoU < 0.5$ and $IoU \geq 0.5$). The sensitivity and specificity are compared, so that in the best case (everything is predicted correctly) the $AUROC$ reaches a value of 1.

Regression Measures To evaluate the regression model, we use the *coefficient of determination* R^2 [116]. Let y_i be the IoU value of an object i and $\hat{y}_i$ the IoU predicted by the model, with corresponding mean values $\bar{y}$ and $\bar{\hat{y}}$ over all observations, respectively. The coefficient of determination is given by

$$R^2 = \frac{\sum_i (\hat{y}_i - \bar{\hat{y}})^2}{\sum_i (y_i - \bar{y})^2} \,. \tag{2.78}$$

The higher the value of $0 \leq R^2 \leq 1$ is, the more accurate is the fitting of the regression model. $R^2 = 0$ corresponds to the baseline model (each value is predicted as the mean) and $R^2 = 1$ to a perfect prediction.

Furthermore, for regression models the *standard errors* σ can be computed as the square root of the mean squared error.

2.5 Datasets

In this section, we give an overview of the datasets which we consider in our experiments. The KITTI dataset [45, 46] consists of real street scene images from Karlsruhe in Germany, and the MOT dataset [109] provides scenes from pedestrian areas and shopping malls. The synthetic VIsual PERception (VIPER) dataset [132] is obtained from the computer game GTA V and consists of more than 250K images acquired at different times of the day and under various weather conditions. We focus on the images from the day category, i.e., bright images and no rain, for further processing. Examples of raw images of these datasets are shown in Fig. 2.20. We use these datasets for different tasks like semantic segmentation, instance segmentation and depth estimation. The detailed information about how many images and for which tasks we use the datasets is given in Table 2.2. We provide the resolution of the images and the number of labeled images for semantic segmentation, instance segmentation as well as depth estimation. Furthermore, we give the splittings of the images into video sequences. The KITTI dataset and the VIPER dataset have a framerate of 10 frames per second (fps) and the MOT dataset of 30 fps. For semantic and instance segmentation, the number of classes for the specific task are specified. The classes in semantic segmentation cover the categories flat (e.g. road and sidewalk), construction (e.g. building and wall), nature (vegetation and terrain), vehicle (e.g. car, bus

Figure 2.20: *Top*: Two raw images of the KITTI dataset. *Center*: VIPER dataset. *Bottom*: MOT dataset.

Table 2.2: Overview of the datasets including the number of labeled images for the three different tasks, i.e., semantic segmentation, instance segmentation and depth estimation.

	KITTI	VIPER	MOT
resolution	1,242 × 375	1,920 × 1,080	1,920 × 1,080 640 × 480
semantic segmentation	142 (29 videos)	8,740 (33 videos)	–
number of classes	19	23	–
instance segmentation	8,008 (21 videos)	–	2,862 (4 videos)
number of classes	2	–	1
tracking IDs	✓	–	✓
depth estimation	24,268 (61 videos)	–	–

and train), sky, object (e.g. pole and traffic sign) and human (person and rider). In the VIPER dataset, classes like chair and trash are added. In instance segmentation, the classes correspond to the most important ones, persons and cars. Since the MOT dataset is dealing with pedestrian areas and shopping malls, only pedestrians are labeled. For both datasets, multi-object tracking IDs are available for the instances [162].

Chapter 3

Time-Dynamic Estimates of the Reliability of Deep Semantic Segmentation Networks

Semantic segmentation, i.e., the pixel-wise classification of image content, is an important tool for scene understanding (see Sec. 2.1.3). In recent years, neural networks have demonstrated outstanding performance for this task. In safety relevant applications like automated driving [70] and medical diagnosis [169], the reliability of predictions and thus, uncertainty quantification is of highest interest. While most works focus on uncertainty quantification for single frames, video data is often available as well. In this thesis, we study uncertainties by taking temporal information into account. To this end, we construct metrics that express uncertainties in single frames. By tracking predicted segments over time, we obtain time series of metrics that quantify the dynamics of predicted objects (the prediction of one object instance corresponds to a segment). From this information, we assess the prediction quality on segment-level.

In this chapter, we elaborate on the meta classification and regression approach from [136] which provides a framework for post-processing a semantic segmentation. Our method aggregates pixel-wise uncertainty heatmaps obtained by the softmax output of a semantic segmentation network on segments, such as pixel-wise entropy, probability margin or variation ratio. An example of pixel-wise variation ratio is given in Fig. 3.1 (right). In addition to these metrics, further quantities derived from the predicted segments are used, for instance various measures corresponding to the geometry of segments. We extend these segment-wise and single-frame metrics by taking time-dynamics into account. To this end, we present a light-weight approach for tracking segments over time, matching them according to their overlap in consecutive frames. This is accomplished by shifting

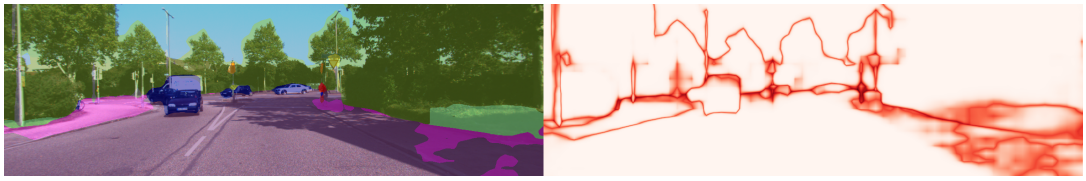


Figure 3.1: *Left*: Semantic segmentation predicted by a neural network. *Right*: Pixel-wise variation ratio.

segments according to their expected location in the subsequent frame. The time series of metrics, yielding a structured dataset (metrics in columns and predicted segments in lines), is presented to meta classifier/regressor to either classify between $IoU = 0$ and $IoU > 0$ or predict the IoU directly. For both prediction tasks, we study different types of models and their dependence on the length of the time series. Given the fact that for automated driving image sequences are available, the proposed light-weight tracking algorithm uses information from consecutive frames to improve meta classification and regression performance with only little additional effort.

A core assumption for our method is that a semantic segmentation network and a video stream of input data are available. In our tests, we employ the publicly available DeepLabv3+ network with two different network backbones (Sec. 2.1.3) and apply them to the VIPER dataset and to the KITTI dataset (see Sec. 2.5). For the VIPER dataset, there are labeled ground truth images for each frame, while for the KITTI dataset, for only a few (142) frames ground truth is given. For this reason, we use alternative sources of information to address the lack of ground truth by incorporating *pseudo* ground truth provided by a stronger network. The source code of our method is publicly available at `http://github.com/kmaag/Time-Dynamic-Prediction-Reliability`. Our contributions are summarized as follows:

- We propose our light-weight tracking approach for segments and the construction of segment-wise metrics using uncertainty and geometry information. In addition, we introduce the meta regression and classification methods including the construction of their inputs consisting of time series of metrics.
- For the KITTI dataset, we present alternative sources of information to address the lack of ground truth annotation in video data.
- We perform meta classification and regression to assess the prediction quality of neural networks on segment-level. Moreover, we study the influence of time-dynamics on meta classification and regression as well as the incorporation of various classification and regression methods. For meta regression, we achieve R^2 values of up to 87.51% as well as for the meta classification,

AUROC values of up to 88.68%. Furthermore, we compare our approach with different baselines.

The remainder of this chapter is organized as follows. In Sec. 3.1, we demonstrate the differences to related work. Our method is described in Sec. 3.2. In more detail, in Sec. 3.2.1, we introduce our tracking algorithm for single-frame based semantic segmentation networks applied to video data. This is followed by the construction of segment-wise metrics using uncertainty and geometry information in Sec. 3.2.2. In Sec. 3.2.3, we describe the meta regression and classification methods including the construction of their inputs consisting of time series of metrics. The numerical results are presented in Sec. 3.3. We study the impact of time-dynamics on meta classification and regression as well as the incorporation of various classification and regression methods. Finally, we discuss our results and next steps to improve our methods in Sec. 3.4.

3.1 Related Work

In Sec. 2.3.1, the related work on multiple object tracking is presented. Most works [5, 10, 12, 32, 58, 62, 82, 91, 113, 160, 170, 173, 178, 179] in the field of object tracking make use of bounding boxes, while our approach is based on semantic segmentation. There are some approaches using segmentation masks. However, only a coarse binary mask is used in [28] and in [55], the segmentation is only used for initialization. In [2, 35], segmentation and tracking are executed jointly. In our procedure, a segmentation is inferred first, tracking is performed afterwards making our method independent of the neural network. In addition to the different forms of object representations, there are various algorithms for object tracking. In the tracking-by-detection methods, a classifier for the difference between object and background is trained and therefore, only information about the location of the object in the first frame is given [5, 12, 32, 160]. We do not train classifiers as this information is contained in the inferred segmentations. Another approach is to learn a similarity function offline [10, 58, 62, 91, 178]. The works of [2, 6, 35, 150, 166] are based on segmentation and they use different tracking methods, like probability models, particle filters and fully-convolutional siamese networks, respectively. Our tracking method is solely based on the degree of overlap of predicted segments in consecutive frames.

The related work on uncertainty quantification is introduced in Sec. 2.2. With respect to uncertainty quantification, MC dropout is widely used, see [44, 76, 78, 169]. Whenever dropout is used in a segmentation network (we do not use dropout), the resulting heatmap can be included into our framework. There are alternative measures of uncertainty like gradient based ones [119] or measures based on spatial and temporal differences between the colors and movements of

the objects [39]. We construct metrics based on aggregated dispersion measures from the softmax output of a neural network on segment level. The works [33, 68] closest to ours are constructed to work with one object per image, instead of hand crafted metrics they are based on (post-processing) CNNs. We extend the work of [136], where meta classification and regression for single frames are introduced, by a temporal component. The authors post-processed semantic segmentation predictions in order to estimate the quality of each predicted segment. We construct time series from the segment-wise uncertainty metrics and further, study additional methods for the meta classification and regression, e.g. gradient boosting and neural networks.

3.2 Method

In this section, we present our light-weight tracking method and the construction of segment-wise metrics based on object's geometric and dispersion measures. From this information, we assess the prediction quality on segment-level by means of meta classification as well as regression. Furthermore, we predict false positive segments by meta classification.

3.2.1 Tracking Segments over Time

We introduce a light-weight tracking algorithm for the case where semantic segmentation is available for each frame of an image sequence. Semantic image segmentation aims to segment objects in an image. It can be viewed as a pixel-wise classification of image content (e.g. Fig. 3.1 (left)). To obtain a semantic segmentation, the goal is to assign to each image pixel z of an input image x a label y within a prescribed label space $\mathcal{C} = \{y_1, \ldots, y_c\}$ with c different class labels. Here, this task is performed by a neural network that provides for each pixel z a probability distribution $f_z(y|x, w)$ over the class labels $y \in \mathcal{C}$, given learned weights w and an input image x. The predicted class for each pixel z is obtained by

$$\hat{y}_z(x, w) = \arg\max_{y \in \mathcal{C}} f_z(y|x, w)\,. \tag{3.1}$$

Let $\hat{\mathcal{S}}_x = \{\hat{y}_z(x, w)|z \in x\}$ denote the predicted segmentation and $\hat{\mathcal{K}}_x$ the set of predicted segments. A segment is defined as a connected component of which all pixels belong to the same class (set of pixel locations). The idea of the proposed tracking method is to match segments of the same class according to their overlap in consecutive frames. We denote by $\{x_1, \ldots, x_T\}$ the image sequence with a

length of T and x_t corresponds to the t^{th} image. Furthermore, we formulate the overlap of a segment k with a segment j through

$$O_{j,k} = \frac{|j \cap k|}{|j|}. \tag{3.2}$$

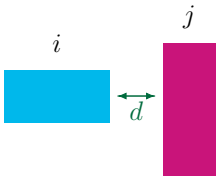
Considering the movement of objects, we also compute geometric centers of predicted segments. The geometric center of a segment $k \in \hat{\mathcal{K}}_{x_t}$ in frame t is defined as

$$\bar{k}_t = (\bar{k}_v, \bar{k}_h)_t = \frac{1}{|k|} \sum_{z \in k} z \tag{3.3}$$

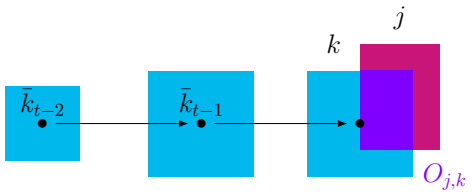
where $z = (z_v, z_h)$ is given by its vertical and horizontal coordinates of pixel z.

Our tracking algorithm is applied sequentially to each frame t, $t = 1, \ldots, T$, and we aim at tracking all segments present in at least one of the frames. To give the segments different priorities for matching, the segments of each frame are sorted by size and treated in descending order. When a segment in frame t is matched with a segment from a previous frame, it is ignored in further steps, and matched segments are assigned an unique ID. Within the description of the matching procedure, we introduce parameters c_{near}, c_{over}, c_{dist} and c_{lin}, the respective numerical choices are given in Sec. 3.3. More formally, our algorithm consists of the following five steps to match segments $k \in \hat{\mathcal{K}}_{x_{t-1}}$ with segments in frame t:

Step 1 (aggregation of segments). The minimum distance between segment $i \in \hat{\mathcal{K}}_{x_t}$ and all $j \in \hat{\mathcal{K}}_{x_t} \setminus \{i\}$ of the same class is calculated. If the distance is less than a constant c_{near}, the segments are so close to each other, they are considered as one segment and receive a common ID.



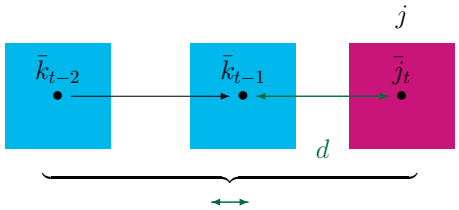
Step 2 (shift). If the algorithm was applied to at least two previous frames, the geometric centers $(\bar{k}_{t-2})$ and $(\bar{k}_{t-1})$ of segment $k \in \hat{\mathcal{K}}_{x_{t-1}}$ are computed. The segment from frame $t-1$ is shifted by the vector $(\bar{k}_{t-1} - \bar{k}_{t-2})$ and the overlap $O_{j,k}$ with each segment $j \in \hat{\mathcal{K}}_{x_t}$ from frame t is determined. If $O_{j,k} \geq c_{over}$ or $j = \arg\max_{i \in \hat{\mathcal{K}}_{x_t}} O_{i,k}$, the segments k and j are matched and receive the same ID.



If there is no match found for segment k during this procedure, the quantity

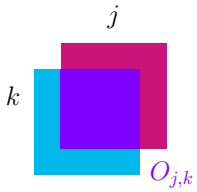
$$d = \min_{j \in \hat{\mathcal{K}}_{x_t}} \|\bar{j}_t - \bar{k}_{t-1}\|_2 + \|(\bar{k}_{t-1} - \bar{k}_{t-2}) - (\bar{j}_t - \bar{k}_{t-1})\|_2 \tag{3.4}$$

is calculated for each available j and both segments are matched if $d \leq c_{dist}$. This allows for matching segments that are closer to $\bar{k}_{t-1}$ than expected. The first norm measures the distance of geometric centers while the second norm ensures that the direction and shift of segment k from frame $t-2$ to $t-1$ is similar to those of segment k from frame $t-1$ to segment j in frame t.



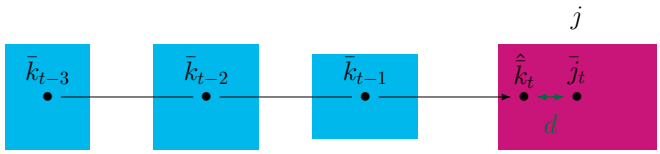
If segment k exists in frame $t-1$ and not in $t-2$, then step 2 is simplified: only the distance between the geometric center of $k \in \hat{\mathcal{K}}_{x_{t-1}}$ and $j \in \hat{\mathcal{K}}_{x_t}$ is computed and the segments are matched if the distance is smaller than c_{dist}.

Step 3 (overlap). If $t \geq 2$, the overlap $O_{j,k}$ of the segments $k \in \hat{\mathcal{K}}_{x_{t-1}}$ and $j \in \hat{\mathcal{K}}_{x_t}$ in two consecutive frames is calculated. If $O_{j,k} \geq c_{over}$ or $j = \arg\max_{i \in \hat{\mathcal{K}}_{x_t}} O_{i,k}$, the segments k and j are matched.

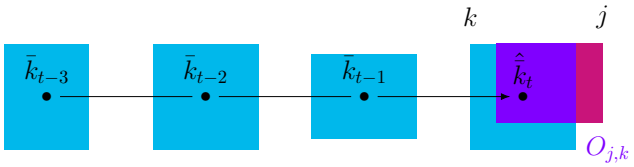


Step 4 (regression). In order to account for flashing predicted segments, either due to false prediction or occlusion, we implement a linear regression and match segments that are more than one, but at most $n_{lr} - 2$, frames apart in temporal direction. If the ID of segment $k \in \hat{\mathcal{K}}_{x_*}$, $* \in \{t - n_{lr}, \ldots, t-1\}$, has not yet been

assigned in frame t and $t \geq 4$, i.e., three frames have already been processed, then the geometric centers of segment k are computed in frames $t - n_{lr}$ to $t - 1$ (in case k exists in all these frames). If at least two geometric centers are available, a linear regression is performed to predict the geometric center ($\hat{\bar{k}}_t$). If the distance between the predicted geometric center and the calculated geometric center of the segment $j \in \hat{\mathcal{K}}_{x_t}$ is less than a constant value c_{lin}, k and j are matched.



If no match was found for segment k, segment $k \in \hat{\mathcal{K}}_{x_{t_{max}}}$ is shifted by the vector $(\hat{\bar{k}}_t - \bar{k}_{t_{max}})$, where $t_{max} \in \{t - n_{lr}, \ldots, t - 1\}$ denotes the frame where k contains the maximum number of pixels. If $O_{j,k} \geq c_{over}$ or $j = \arg\max_{i \in \hat{\mathcal{K}}_{x_t}} O_{i,k}$ applied to the resulting overlap, k and j are matched.



Step 5 (new IDs). All segments $j \in \hat{\mathcal{K}}_{x_t}$ that have not yet received an ID are assigned with a new one.

In Fig. 3.2 our tracking algorithm is applied to a short image sequence of predicted segments obtained by a semantic segmentation network. Note, our approach allows multiple segments in one frame to have the same ID.

3.2.2 Segment-wise Metrics and Time Series

In the previous section, we presented the semantic segmentation and the resulting probability distribution $f_z(y|x, w)$ for pixel z of an image x and weights w. The degree of randomness in $f_z(y|x, w)$ is quantified by (pixel-wise) dispersion measures. We consider the normalized entropy

$$E_z(x, w) = -\frac{1}{\log(c)} \sum_{y \in \mathcal{C}} f_z(y|x, w) \log f_z(y|x, w)\,, \tag{3.5}$$

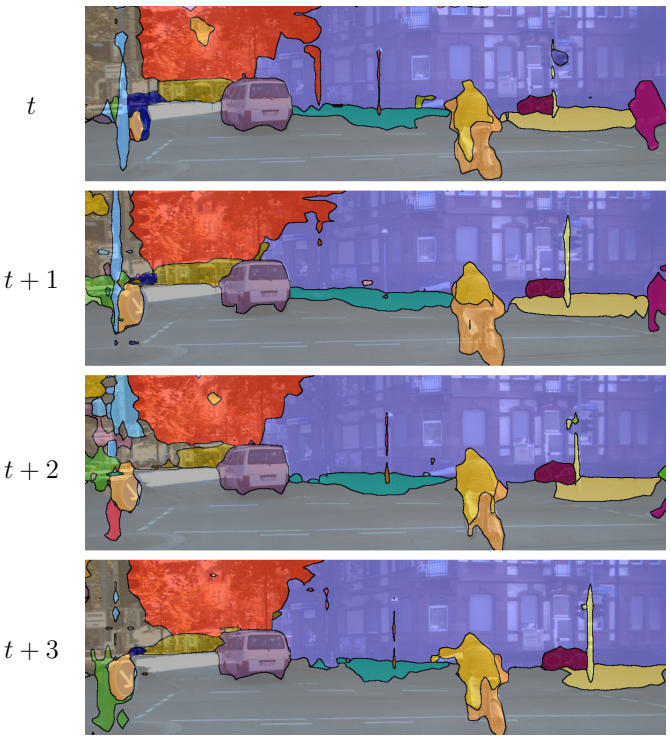


Figure 3.2: Our tracking method is applied to predicted segments obtained by a semantic segmentation network in four consecutive frames. Each color corresponds to a different ID.

the variation ratio

$$V_z(x, w) = 1 - f_z(\hat{y}_z(x, w)|x, w) \tag{3.6}$$

and the probability margin

$$M_z(x, w) = 1 - f_z(\hat{y}_z(x, w)|x, w) + \max_{y \in \mathcal{C} \setminus \{\hat{y}_z(x,w)\}} f_z(y|x, w)\,. \tag{3.7}$$

A visualization of pixel-wise variation ratio is shown in Fig. 3.1 (right). Note, that also other heatmaps (like MC Dropout variance) can be processed. To obtain metrics per segment from these pixel-wise dispersion measures for each segment $k \in \hat{\mathcal{K}}_x$, we define *mean dispersions* $\bar{D}$ as

$$\bar{D} = \frac{1}{S} \sum_{z \in k} D_z(x) \tag{3.8}$$

where $D_z \in \{E_z, V_z, M_z\}$ and $S = |k|$ denotes the *segment size*. We further distinguish between the inner $k_{in} \subset k$, where a pixel $z \in k_{in}$ if all eight neighboring pixels of z are elements of k, and the boundary $k_{bd} = k \setminus k_{in}$. At segment level, the segment size and the mean disperions are provided as metrics, divided into inner and boundary. From these metrics, additional metrics are derived such as

the *relative segment sizes* $\tilde{S} = S/S_{bd}$ and $\tilde{S}_{in} = S_{in}/S_{bd}$ as well as the *relative mean dispersions* $\tilde{\bar{D}} = \bar{D}\tilde{S}$ and $\tilde{\bar{D}}_{in} = \bar{D}_{in}\tilde{S}_{in}$ where $D \in \{E, V, M\}$. Our set of metrics per segment is constructed by these measures and the *geometric center* $\bar{k}$ (defined in (3.3)) as well as the *mean class probabilities* for each class $y \in \mathcal{C}$

$$P(y|k) = \frac{1}{S} \sum_{z \in k} f_z(y|x, w)\,. \tag{3.9}$$

In summary, we use the following set of metrics:

$$\begin{aligned} U^k = &\{\bar{D}, \bar{D}_{in}, \bar{D}_{bd}, \tilde{\bar{D}}, \tilde{\bar{D}}_{in} \,:\, D \in \{E, V, M\}\} \\ &\cup \{S, S_{in}, S_{bd}, \tilde{S}, \tilde{S}_{in}\} \cup \{\bar{k}\} \cup \{P(y|k) \,:\, y = 1, \ldots, c\}\,. \end{aligned} \tag{3.10}$$

The separate treatment of inner and boundary in all dispersion measures is motivated by typically large values of D_z for $z \in k_{bd}$ (see Fig. 3.1 (right)). In addition, we find that poor or false predictions are often accompanied by fractal segment shapes (which have a relatively large amount of boundary pixels, measurable by $\tilde{S}$ and $\tilde{S}_{in}$) and/or high dispersions $\bar{D}_{in}$ on the segment's inner. An example of such a fractal segment is the sidewalk in the front right area of Fig. 3.1 (left). The presented metrics are single-frame based and the proposed light-weight tracking method provides the identification of predicted segments in consecutive frames. Hence, we obtain time series for each of the defined metrics, that are subject to further analysis.

3.2.3 Prediction of the IoU from Time Series

A measure to determine the prediction accuracy of the segmentation network with respect to the ground truth is the IoU (see Sec. 2.1.6). To this end, we define by $\mathcal{K}_x$ the set of connected components in the ground truth $\mathcal{S}_x$, analogously to $\hat{\mathcal{K}}_x$ (the set of connected components in the predicted segmentation $\hat{\mathcal{S}}_x$). The corresponding class of $k' \in \mathcal{K}_x$ is denoted by $y_z(x) \in \mathcal{C}$ (for any $z \in k'$). For $k \in \hat{\mathcal{K}}_x$ let $\mathcal{K}_x^k = \{k' \in \mathcal{K}_x \,:\, \hat{y}_z(x, w) = y_z(x) \text{ for } z \in k \cap k' \text{ and } k \cap k' \neq \emptyset\}$. For each $k \in \hat{\mathcal{K}}_x$ the IoU is given by

$$IoU(k) = \frac{|k \cap K'|}{|k \cup K'|}\,, \qquad K' = \bigcup_{k' \in \mathcal{K}_x^k} k'. \tag{3.11}$$

In our test, we use a slight modification, i.e., the adjusted IoU

$$IoU_{\text{adj}}(k) = \frac{|k \cap K'|}{|k \cup (K' \setminus Q)|}\,, \qquad Q = \bigcup_{q \in \mathcal{Q}_x^k} q\,, \tag{3.12}$$

with $\mathcal{Q}_x^k = \{q \in \hat{\mathcal{K}}_x \,:\, \hat{y}_z(x, w) = y_z(x),\ z \in q \cap K' \text{ and } q \cap K' \neq \emptyset\}$ which is less prone to fragmented objects. The motivation for adjusting the IoU is

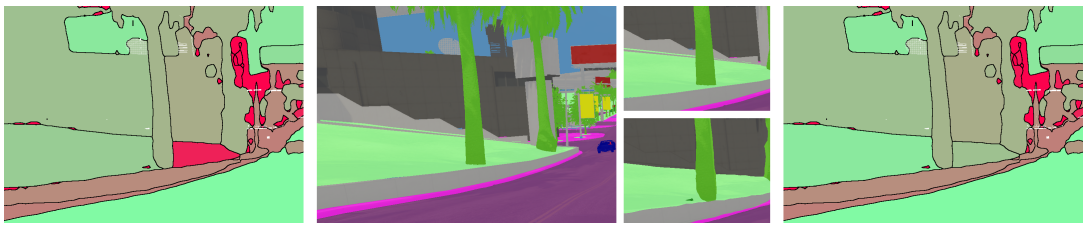


Figure 3.3: Illustration of the different behaviors of calculated IoU and IoU_{adj} per segment. *Left*: Visualization of IoU. *Right*: IoU_{adj}. *Center*: Corresponding ground truth with zooms for the crucial areas. *Center top*: Zoom for ground truth. *Center bottom*: Zoom for predicted segmentation. Green color corresponds to high IoU and IoU_{adj} values, red color to low ones. The predicted segmentation shows the soil (class terrain) divided into two segments by a palm tree (class vegetation). The IoU rates the small segment on the right from the palm tree poor, although the prediction is well. These problems are solved by the IoU_{adj}.

given in Fig. 3.3. It can happen that a ground truth segment is split into two or more segments in the prediction. Even if the segments are well predicted, the IoU values are low. In contrast, the IoU_{adj} does not penalize the prediction of a segment if the remainder of the ground truth is well covered by other predicted segments belonging to the same class.

In this work, we perform segment-wise predictions of the IoU_{adj} (meta regression) comparing different regression approaches and classify between $IoU_{\text{adj}} = 0$ and $IoU_{\text{adj}} > 0$ (meta classification), both for every predicted segment. Both prediction tasks (*meta tasks*) are performed by means of the metrics introduced in Sec. 3.2.2. Note, that these metrics can be computed without the knowledge of the ground truth. Our aim is to analyze to which extent they are suitable for the meta tasks and how much we benefit from using time series. For each segment $k \in \hat{\mathcal{K}}_{x_t}$ in frame t, we have the metrics U_t^k and their information in previous frames due to the segment tracking. Both meta tasks are performed by means of the metrics U_i^k, $i = t - n_c, \ldots, t$, where n_c describes the number of considered frames. An overview of how the metrics as well as their time series are constructed and then, used as input for meta classification and regression is shown in Fig. 3.4. For meta classification, we define $y_k = \{0 \text{ if } IoU_{\text{adj}}(k) = 0, 1 \text{ if } IoU_{\text{adj}}(k) > 0\}$ and we want to predict this value by three different methods. These methods are the logistic regression with ℓ_1-penalty (LASSO [157]), gradient boosting regression [43] and a shallow neural network containing only a single hidden layer with 50 neurons. For meta regression, we compare six different regression methods, this includes plain linear regression, linear regression with ℓ_1- and ℓ_2-penalization, gradient boosting and two shallow neural networks – one with ℓ_1-penalization and one with ℓ_2. An overview of the different methods for regression and classification is given in Table 3.1. Details of these methods are described in Sec. 2.4.

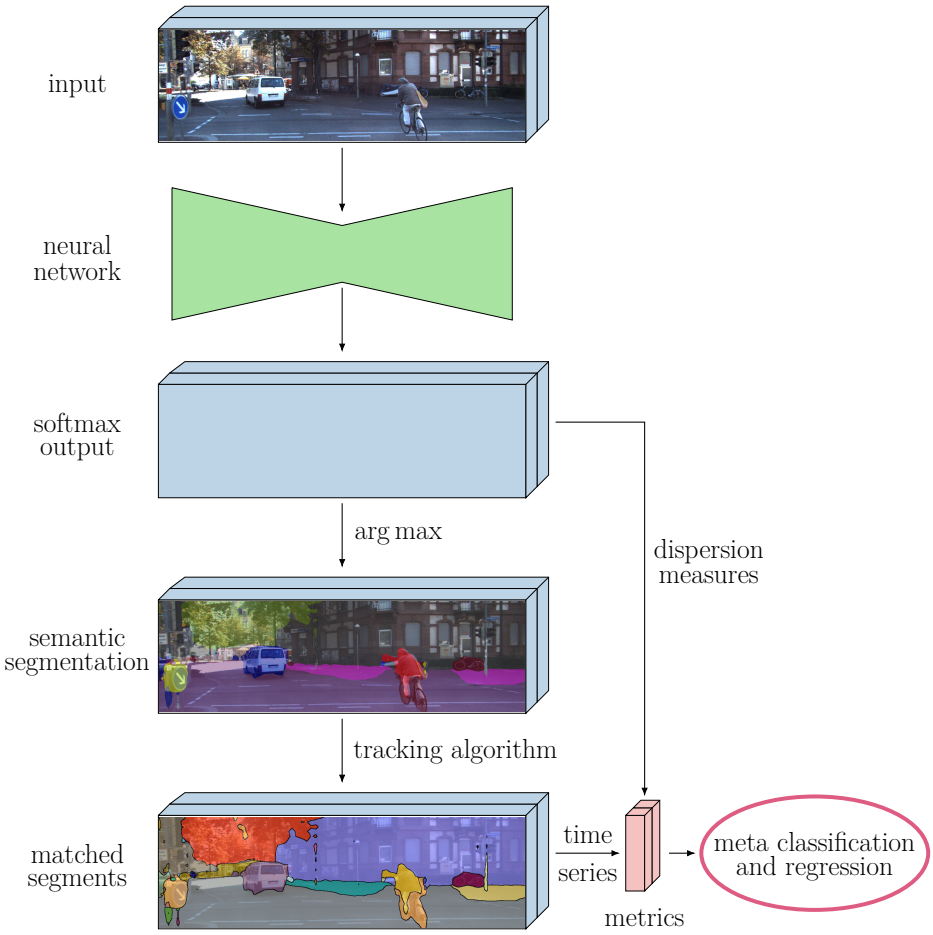


Figure 3.4: Overview of meta classification and regression. A semantic segmentation is predicted by a neural network, and the softmax output is considered for the constructed metrics by means of dispersion measures. By tracking predicted segments, time series of metrics can be obtained. These serve as input for meta classification and regression.

In addition to the presented time-dynamic components, we extend the approach from [136] by incorporating further models for meta tasks, i.e., neural networks and gradient boosting. For this reason, we consider linear models for meta tasks and single-frame ($n_c = 0$) based metrics as used in [136] as baseline. Another approach, presented in [33], also performs meta regression, however, it is only designed for one object per image and on a single-frame basis. As a consequence, we cannot use this approach as a suitable baseline, since ∼150 (#segments/image) CNN inferences per image are unfeasible.

Table 3.1: Overview of meta classification and regression methods. For classification, we consider **LR L1**, **GB**, **NN L2** and for regression all of them.

methods	
LR	linear regression
LR L1	linear / logistic regression with ℓ_1-penalization
LR L2	linear regression with ℓ_2-penalization
GB	gradient boosting
NN L1	neural network with ℓ_1-penalization
NN L2	neural network with ℓ_2-penalization

3.3 Numerical Results

In this section, we investigate the properties of the metrics defined in the previous section and the influence of the length of the time series considered. Moreover, different meta classification and regression methods are studied. We perform our tests on two different datasets (see Sec. 2.5) for the semantic segmentation of street scenes where also image sequences are available, the synthetic VIPER dataset obtained from the computer game GTA V and the KITTI dataset with real street scene images from Karlsruhe, Germany. In all our tests, we consider the DeepLabv3+ network (see Sec. 2.1.3) for semantic segmentation for which we use a reference implementation in Tensorflow [1]. The DeepLabv3+ implementation and weights are available for two network backbones. First, there is the Xception65 network, a modified version of Xception [25]. Next, there is MobilenetV2 [141], a slim architecture designed for mobile devices. Primarily, we use Xception65 for VIPER and MobilenetV2 for KITTI, for the latter we also use Xception65 as a reference network to generate pseudo ground truth for the meta tasks. For tracking segments with our procedure, we assign the parameters defined in Sec. 3.2.1 with the following values: $c_{near} = 10$, $c_{over} = 0.35$, $c_{dist} = 100$ and $c_{lin} = 50$. We study the predictive power of our 22 metrics and segment-wise averaged class probabilities per segment and frame. From our tracking algorithm, we get these metrics additionally from previous frames for every segment.

3.3.1 VIPER Dataset

The VIPER dataset consists of more than 250K 1,920 × 1,080 video frames and for all frames there is ground truth available, consisting of 23 classes. We trained an Xception65 network starting from backbone weights for ImageNet [140]. We choose an output stride of 16 and the input image is evaluated within the framework only on its original scale (DeepLabv3+ allows for evaluation on different scales and averaging the results). For a detailed explanation of the chosen parameters we refer to [24]. We retrain the Xception65 network with 5,147 training

Table 3.2: Correlation coefficients ρ with respect to IoU_{adj} for the VIPER dataset.

S	0.50219	$\bar{E}$	−0.66755	$\bar{V}$	−0.80317	$\bar{M}$	−0.86945	$\bar{k}_v$	0.03538
S_{in}	0.49912	$\bar{E}_{in}$	−0.66908	$\bar{V}_{in}$	−0.77783	$\bar{M}_{in}$	−0.84164	$\bar{k}_h$	−0.04239
S_{bd}	0.55232	$\bar{E}_{bd}$	−0.27683	$\bar{V}_{bd}$	−0.42808	$\bar{M}_{bd}$	−0.61155		
$\tilde{S}$	0.52080	$\tilde{\bar{E}}$	0.42459	$\tilde{\bar{V}}$	0.34224	$\tilde{\bar{M}}$	0.38094		
$\tilde{S}_{in}$	0.52080	$\tilde{\bar{E}}_{in}$	0.46122	$\tilde{\bar{V}}_{in}$	0.37873	$\tilde{\bar{M}}_{in}$	0.39900		

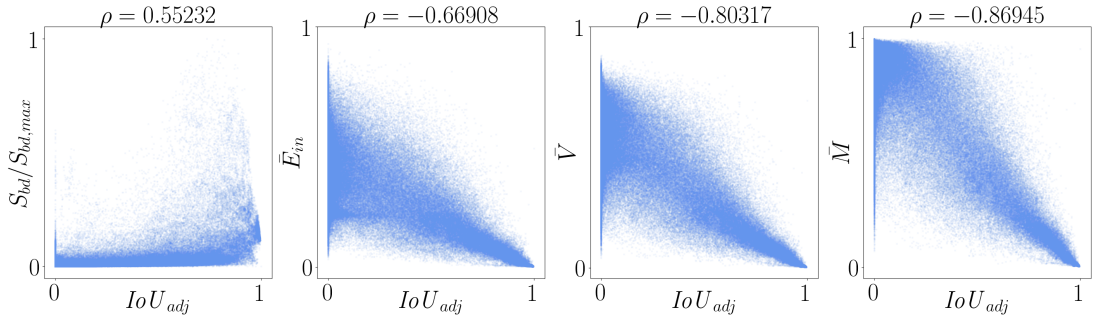


Figure 3.5: Correlations between rescaled metrics and IoU_{adj} for the VIPER dataset.

images and 847 validation images (during training with different numbers of images, we found that this number is sufficient). We only use images from the day category (i.e., bright images, no rain) for training and further processing, achieving a mean IoU of 50.33%. If we remove the classes mobile barrier, chair and van which are also underrepresented in the dataset (yielding IoUs below 10%), the mean IoU rises to 57.38%. The validation set is neither used for early stopping nor for parameter tuning.

For further experiments, we consider the output probabilities and predictions of 13 video sequences consisting of 3,593 images in total. For each segment in the segmentations of these images, we compute the IoU_{adj} and the 22 constructed segment-wise metrics U_t^k for each single frame t. To study the prediction power of the metrics, we compute the Pearson correlation coefficients $\rho \in [-1, 1]$ between IoU_{adj} and each metric. The results are given in Table 3.2. The metrics are standardized to zero mean and unit standard deviation. Furthermore, we provide scatter plots of selected metrics relative to IoU_{adj} in Fig. 3.5. The mean probability margin ($\bar{M}$ and $\bar{M}_{in}$) and variation ratio measures ($\bar{V}$ and $\bar{V}_{in}$) as well as the mean entropies ($\bar{E}$ and $\bar{E}_{in}$) show strong correlations with IoU_{adj}. However, the geometric center and the relative dispersion measures are less correlated with IoU_{adj}.

We perform meta classification and regression in order to investigate the predictive power of the combination of these metrics. From 3,593 images, we obtain roughly 309,874 segments (not yet matched over time) of which 251,368 have non-empty inner. The latter are used in all numerical tests. We study the influence of time-

Table 3.3: Results for meta classification and regression for the VIPER dataset and for the different methods as well as for the naive and entropy baselines. The super script denotes the number of frames where the best performance and in particular, the given values are reached. The best classification and regression results are highlighted.

meta classification $IoU_{\text{adj}} = 0, > 0$			
naive baseline:	$ACC = 66.07\%$	$AUROC = 50.00\%$	
entropy baseline:	$ACC = 68.43\% \pm 0.29\%$	$AUROC = 74.02\% \pm 0.32\%$	
	LR L1	GB	NN L2
ACC	$75.75\% \pm 0.49\%^{8}$	$\mathbf{77.88}\% \pm 0.60\%^{2}$	$76.62\% \pm 0.51\%^{6}$
$AUROC$	$83.44\% \pm 0.47\%^{7}$	$\mathbf{86.01}\% \pm 0.56\%^{4}$	$84.52\% \pm 0.50\%^{11}$
meta regression IoU_{adj}			
entropy baseline:	$\sigma = 0.178 \pm 0.000$	$R^2 = 64.18\% \pm 0.34\%$	
	LR	LR L1	LR L2
σ	0.124 ± 0.002^{6}	0.124 ± 0.002^{7}	0.124 ± 0.002^{5}
R^2	$82.58\% \pm 0.45\%^{6}$	$82.56\% \pm 0.43\%^{7}$	$82.57\% \pm 0.44\%^{5}$
	GB	NN L1	NN L2
σ	$\mathbf{0.112} \pm 0.002^{6}$	0.118 ± 0.002^{4}	0.117 ± 0.002^{2}
R^2	$\mathbf{85.82}\% \pm 0.36\%^{6}$	$84.36\% \pm 0.51\%^{4}$	$84.58\% \pm 0.44\%^{2}$

dynamics on meta tasks, i.e., we firstly only present the segment-wise metrics U_t^k of a single frame t to the meta classifier/regressor, secondly we extend the metrics to time series with a length of up to 10 previous time steps U_i^k, $i = t-10, \ldots, t-1$. We obtain 11 different inputs for the meta tasks models. The presented results are averaged over 10 runs obtained by random sampling of the train/validation/test splitting. In tables and figures, the corresponding standard deviations are given in brackets and by shades, respectively. Out of the 251,368 segments with non-empty inner, 85,291 have an $IoU_{\text{adj}} = 0$.

First, we present results for meta classification, i.e., detection of false positive segments ($IoU_{\text{adj}} = 0$), using 38,000 (randomly sampled) segments that are not presented to the segmentation network during training. We apply a (meta) train/validation/test splitting of 70%/10%/20% and evaluate the performance of different models for meta classification in terms of classification accuracy and $AUROC$ (Sec. 2.4.4). The $AUROC$ is obtained by varying the decision threshold in a binary classification problem, here for the decision between $IoU_{\text{adj}} = 0$ and > 0. We achieve test $AUROC$ values of up to 86.01% and accuracies of up to 77.88%. Table 3.3 shows the best results for different meta classification methods, i.e., logistic regression, a shallow neural network and gradient boosting, see Table 3.1. The super script denotes the number of frames where the best performance and in particular, the given values are reached. We observe that the best results are achieved when considering more than one frame. This observation is confirmed in Fig. 3.6 (left) where the results for meta classification $AUROC$ as functions of the number of frames, i.e., the maximum time series length, are given. Furthermore, significant differences between the methods for meta classification can be

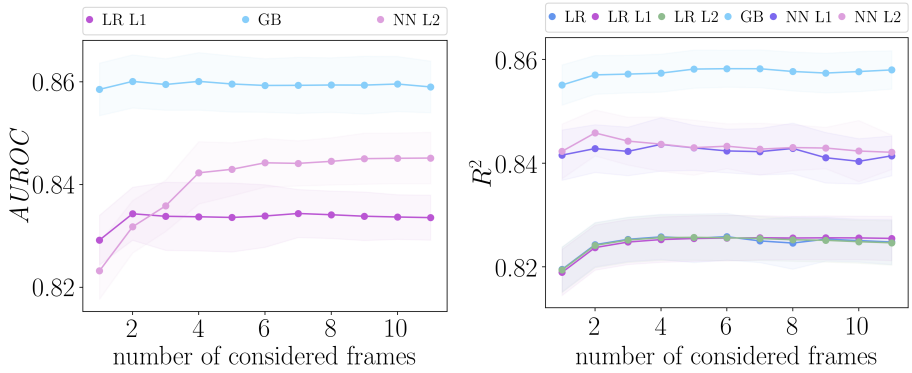


Figure 3.6: Results for meta classification *AUROC* and regression R^2 as functions of the number of frames for the VIPER dataset and different methods. *Left*: Meta classification. *Right*: Meta regression.

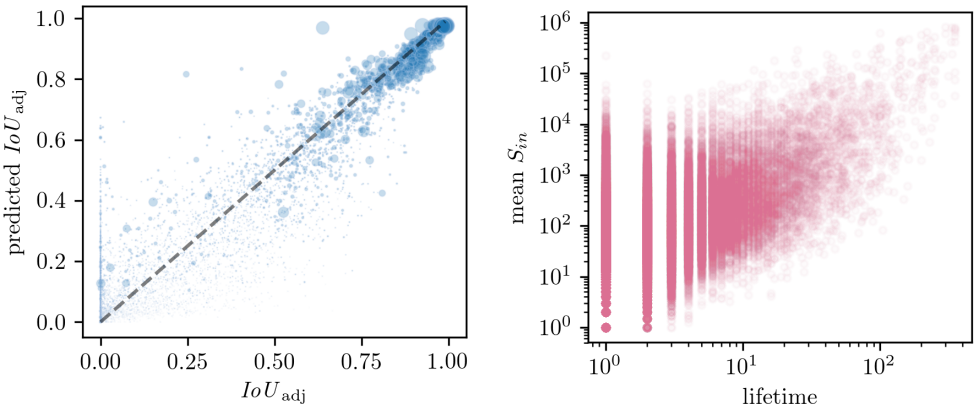


Figure 3.7: *Left*: Predicted IoU_{adj} vs. IoU_{adj} for all non-empty segments. The dot size is proportional to the segment size. *Right*: Segment lifetime (time series length) vs. mean inner segment size, both on log scale.

observed, gradient boosting shows the best performance with respect to accuracy and *AUROC*.

Next, we predict IoU_{adj} values via meta regression to estimate prediction quality. For this task, we state regression standard errors σ and R^2 values (Sec. 2.4.4). We achieve R^2 values of up to 85.82%. This value is obtained by gradient boosting incorporating 5 previous frames. For this particular case, the correlation of the calculated and predicted IoU_{adj} is depicted in Fig. 3.7 (left) and an illustration of the resulting quality estimate is given in Fig. 3.8. We also provide video sequences[1] that visualize the IoU_{adj} prediction and the segment tracking. Results for meta regression are also summarized in Table 3.3, the findings are in analogy to

[1] `http://youtu.be/TQaV5ONCV-Y`

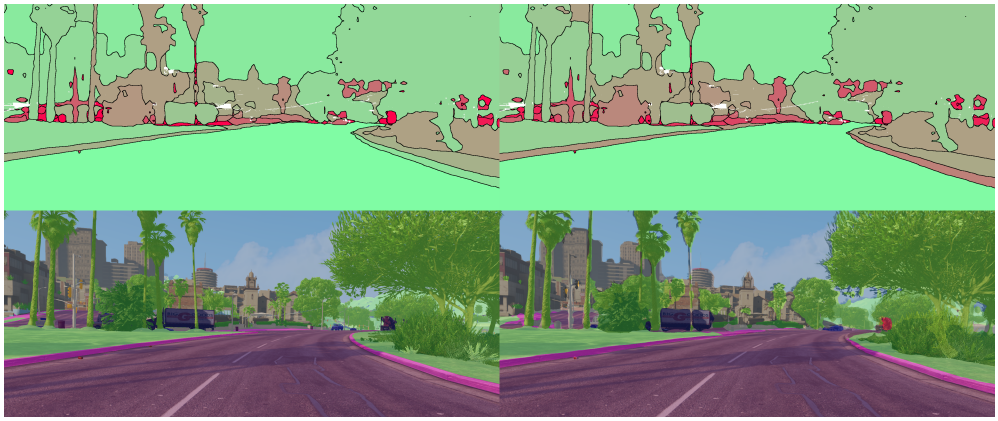


Figure 3.8: *Bottom left*: Ground truth image. *Bottom right*: Semantic segmentation obtained by a neural network. *Top left*: Visualization of the true segment-wise IoU_{adj} of prediction and ground truth. *Top right*: IoU_{adj} prediction obtained from meta regression for the VIPER dataset. Green color corresponds to high IoU_{adj} values and red color to low ones. For the white regions there is no ground truth available, these regions are not included in the statistical evaluation.

those for meta classification. Gradient boosting performs best, and more frames yield better results than a single one. The results for meta regression R^2 as functions of the number of frames are shown in Fig. 3.6 (right). Fig. 3.7 (right) depicts the correlation of the time series length and the mean inner segment size. On average, a predicted segment exists for 4.4 frames, however, when we consider only segments that contain at least 1,000 inner pixels, the average life time increases to 19.9 frames.

The approach in [136] can be considered as a baseline for meta classification and regression, since we extend this single-frame based method by time series and further classification/regression models. The results in [136] have been compared with the mean entropy per segment as a single-metric baseline and with a naive baseline which is given by random guessing (randomly assigning a probability to each segment and then thresholding on it). The classification accuracy is the number of correctly predicted observations divided by the total number of observations. It is maximized for the threshold being either 1 if we have more $IoU_{\text{adj}} > 0$ than $IoU_{\text{adj}} = 0$ segments, or 0 else. The corresponding $AUROC$ value is 50%. For the entropy baseline, we use single-frame gradient boosting. Table 3.3 includes these two baselines, both are clearly outperformed just as the single-frame method using linear models [136] as shown in Fig. 3.6.

Table 3.4: Train/val/test splitting, different compositions of training data and their approximate number of segments.

splitting	types of data / annotation		no. of segments
train	R	real	∼ 3,400
	RA	real and augmented	∼ 27,000
	RAP	real, augmented and pseudo	∼ 27,000
	RP	real and pseudo	∼ 27,000
	P	pseudo	∼ 27,000
val		real	∼ 500
test		real	∼ 1,000

3.3.2 KITTI Dataset

For the KITTI dataset, we use both DeepLabv3+ networks (pre-trained on the Cityscapes dataset [29], available on GitHub), however, for the evaluation, we primarily use MobilenetV2. As parameters for the Xception65 network, we choose an output stride of 8, a decoder output stride of 4 and an evaluation of the input on scales of 0.75, 1.00 and 1.25 (averaging the results). For the MobilenetV2, we use an output stride of 16 and the input image is evaluated within the framework only on its original scale. In our tests, we use 29 street scene videos consisting of 12,223 images with a resolution of 1,242 × 375. Only 142 of these images are labeled. An evaluation of the meta tasks requires a train/val/test splitting. Therefore, the small number of labeled images seems almost insufficient. Hence, we acquire alternative sources of useful information besides the (real) ground truth. First, we utilize the Xception65 network with high predictive performance, its predicted segmentations we term *pseudo ground truth*. We generate pseudo ground truth for all images where ground truth is not available. The mean IoU performance of the Xception65 network for the 142 labeled images is 64.54%, for the MobilenetV2 the mean IoU is 50.48%. In addition, to augment the structured dataset of metrics, we apply a variant of SMOTE for continuous target variables for data augmentation (see [17, 159]). SMOTE is an upsampling procedure which is used in cases of class imbalance. It addresses the problem of imbalance between rare target classes and most frequent ones. The procedure has been developed for nominal target variables and is modified in [159] to work with continuous target variables. Data augmentation is particularly important when only working with the scarce real ground truth as in our tests, we observe that SMOTE prevents overfitting in this scenario. An overview of the different compositions of training data and the train/val/test splitting are given in Table 3.4. The train/val/test splitting of the data with ground truth available is the same as for the VIPER dataset, i.e., 70%/10%/20%. The shorthand "augmented" refers to data obtained from SMOTE, "pseudo" refers to pseudo ground truth obtained from the Xception65 network and "real" refers to ground truth obtained from a human annotator.

Table 3.5: Correlation coefficients ρ with respect to IoU_{adj} for the KITTI dataset.

S	0.61093	$\bar{E}$	−0.74185	$\bar{V}$	−0.85079	$\bar{M}$	−0.90278	$\bar{k}_v$	−0.14323
S_{in}	0.60773	$\bar{E}_{in}$	−0.74343	$\bar{V}_{in}$	−0.83448	$\bar{M}_{in}$	−0.88593	$\bar{k}_h$	0.04568
S_{bd}	0.67728	$\bar{E}_{bd}$	−0.40583	$\bar{V}_{bd}$	−0.53066	$\bar{M}_{bd}$	−0.66180		
$\tilde{S}$	0.72428	$\tilde{\bar{E}}$	0.45695	$\tilde{\bar{V}}$	0.37841	$\tilde{\bar{M}}$	0.43304		
$\tilde{S}_{in}$	0.72428	$\tilde{\bar{E}}_{in}$	0.49676	$\tilde{\bar{V}}_{in}$	0.41446	$\tilde{\bar{M}}_{in}$	0.44651		

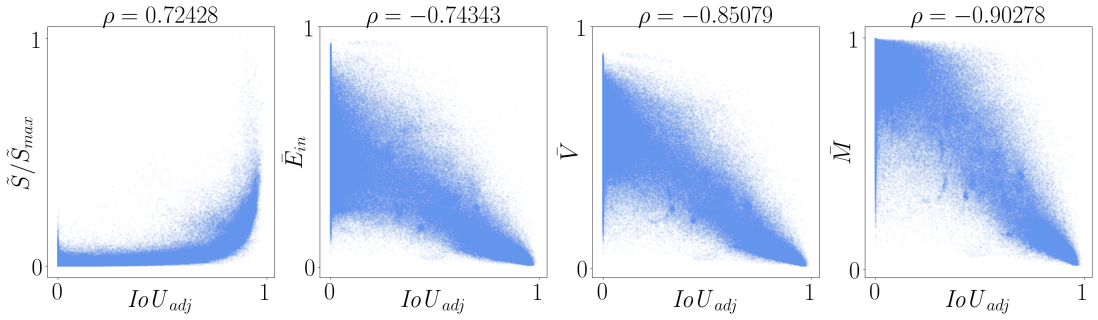


Figure 3.9: Correlations between rescaled metrics and IoU_{adj} for the KITTI dataset.

These additions are only used during training the meta classifier/regressor. We employ the Xception65 network only for the generation of pseudo ground truth, all tests are performed using the MobilenetV2. The KITTI dataset consists of 19 classes (4 classes less than VIPER).

To study the predictive power of the metrics, we compute (as for the VIPER dataset) the Pearson correlation coefficients between IoU_{adj} and each metric, see Table 3.5. Moreover, we provide scatter plots of a selection of metrics relative to IoU_{adj} in Fig. 3.9. Similar to the VIPER dataset, the IoU_{adj} is strongly correlated with the mean dispersion measures, while the correlations between the geometric center and the relative dispersion measures are rather weak. Compared to the VIPER dataset, all correlation values are higher and also the metrics based on the segment's geometry show high correlations. The KITTI dataset also achieves greater values for time series length of the segments obtained by the tracking algorithm compared to the VIPER dataset. On average, a predicted segment exists for 4.7 frames and when we consider only segments that contain at least 1,000 inner pixels, the average life time increases to 20.2 frames.

From the 12,223 chosen images, we obtain 452,287 segments of which 378,984 have non-empty inner. Of these segments, 129,033 have an $IoU_{\text{adj}} = 0$. A selection of results for meta classification $AUROC$ and regression R^2 as functions of the number of frames, i.e., the maximum time series length, is given in Fig. 3.10. The approach of [136] corresponds to the single-frame results in these plots. In contrast to the single-frame approach using only linear models, we increase the accuracy by 6.78 percent points (pp) and the $AUROC$ value by 5.04 pp. The

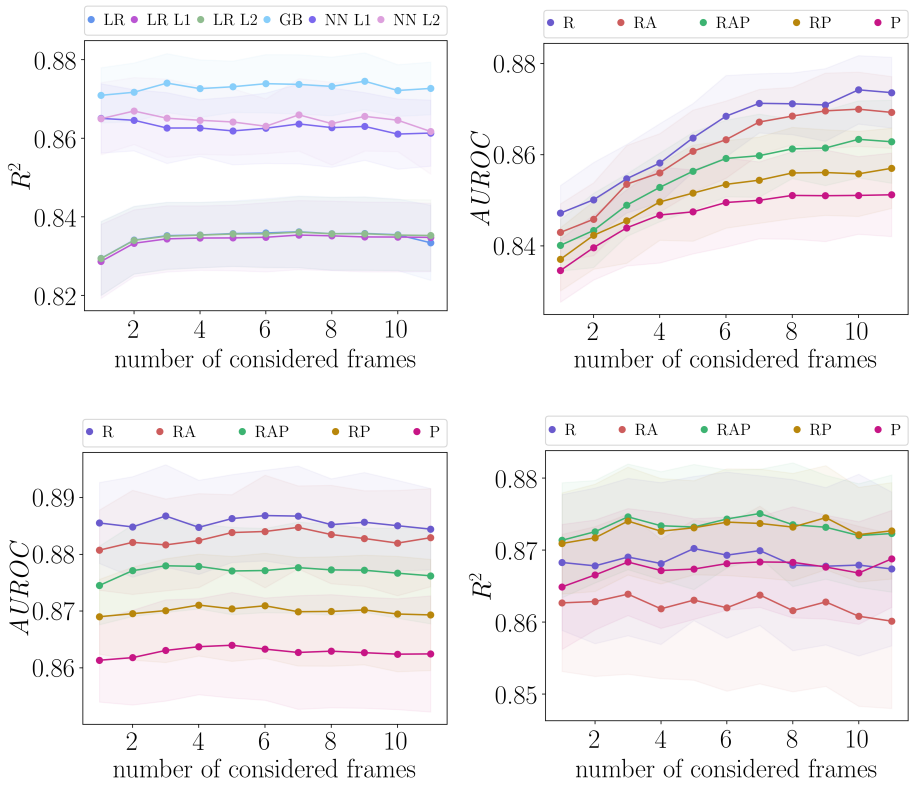


Figure 3.10: A selection of results for meta classification *AUROC* and regression R^2 as functions of the number of frames for the KITTI dataset. *Top left*: Meta regression for different methods. *Top right*: Meta classification via a neural network with ℓ_2-penalization for different compositions of training data (see Table 3.4). *Bottom left*: Meta classification via gradient boosting. *Bottom right*: Meta regression via gradient boosting.

R^2 value for meta regression is increased by 5.63 pp. The meta classification results for neural networks presented in Fig. 3.10 (top right) indeed show, that an increasing length of time series has a positive effect on the performance. On the other hand, the results in Fig. 3.10 (bottom left) show that gradient boosting does not benefit as much from time series. In both cases, augmentation and pseudo ground truth do not improve the models' performance on the test set and although, the neural network benefits a lot from time series, its best performance is still about 1% below the performance of gradient boosting. With respect to the influence of time series length, the results for meta regression with gradient boosting in Fig. 3.10 (bottom right) are qualitatively similar to those in Fig. 3.10 (bottom left). However, we observe in this case that the incorporation of pseudo ground truth slightly increases the performance. Noteworthily, for the R^2 values, we achieve the best results with the training set consisting of real, augmented and pseudo ground truth in two out of six models (see Table 3.6), demonstrating

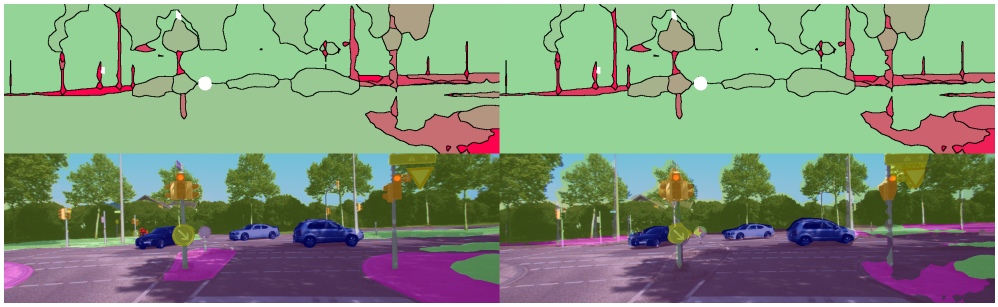


Figure 3.11: *Bottom left*: Ground truth image. *Bottom right*: Semantic segmentation obtained by a neural network. *Top left*: A visualization of the true segment-wise IoU_{adj} of prediction and ground truth. *Top right*: Prediction of IoU_{adj} obtained from meta regression for the KITTI dataset. Green color corresponds to high IoU_{adj} values and red color to low ones. For the white regions, there is no ground truth available, these regions are not included in the statistical evaluation.

that SMOTE helps in this case. Furthermore, gradient boosting trained with real ground truth and gradient boosting trained only with pseudo ground truth perform almost equally well. This shows that meta regression can be learned when there is no ground truth but a strong reference model available. Note, that this (except for the data augmentation part) is in accordance to our findings for the VIPER dataset. Results for a wider range of tests (including those previously discussed) are summarized in Table 3.6. As for the VIPER dataset, significant differences between the methods for meta classification and regression (see Fig. 3.10 (top left)) can be observed, the linear models achieve the lowest results and gradient boosting performs best with respect to all evaluation metrics. For this dataset, video sequences[2] are also provided. For meta classification, we achieve accuracies of up to 81.20% and $AUROC$ values of up to 88.68%. For meta regression, we obtain R^2 values of up to 87.51%. This value is obtained by gradient boosting incorporating 6 previous frames. For this particular case, an illustration of the resulting quality estimate is shown in Fig. 3.11. As for the VIPER dataset, we outperform the analogous baselines (see Table 3.6). Since the labeled 142 images only yield 4,877 segments, we observe overfitting in our tests for all models when increasing the length of the time series. This might serve as an explanation that in some cases, time series do not increase performance. In particular, we observe overfitting in our tests when using gradient boosting, this holds for both datasets, KITTI and VIPER. It is indeed well-known that gradient boosting requires plenty of data.

[2] `http://youtu.be/YcQ-i9cHjLk`

Table 3.6: Results for meta classification and regression for different compositions of training data and methods for the KITTI dataset. The dataset for the entropy baseline is selected such that the baseline performance is maximized. The super script denotes the number of frames where the best performance and thus, the given value is reached. The best results for each data composition are highlighted.

meta classification $IoU_{\text{adj}} = 0, > 0$				
naive baseline:		$ACC = 65.95\%$	$AUROC = 50.00\%$	
entropy baseline for **P**:		$ACC = 68.66\% \pm 1.82\%$	$AUROC = 75.81\% \pm 1.68\%$	
		LR L1	**GB**	**NN L2**
ACC	**R**	**76.69**$\% \pm 1.68\%^{10}$	**81.20**$\% \pm 1.02\%^{4}$	**79.67**$\% \pm 0.93\%^{10}$
	RA	$76.60\% \pm 1.31\%^{7}$	$80.73\% \pm 1.03\%^{9}$	$78.62\% \pm 0.61\%^{11}$
	RAP	$76.18\% \pm 1.22\%^{7}$	$79.64\% \pm 1.03\%^{7}$	$77.08\% \pm 1.05\%^{9}$
	RP	$76.52\% \pm 0.80\%^{8}$	$78.45\% \pm 0.88\%^{8}$	$76.35\% \pm 0.67\%^{9}$
	P	$75.96\% \pm 0.80\%^{11}$	$77.56\% \pm 0.95\%^{5}$	$75.68\% \pm 0.67\%^{11}$
$AUROC$	**R**	$85.13\% \pm 0.84\%^{1}$	**88.68**$\% \pm 0.80\%^{6}$	**87.42**$\% \pm 0.75\%^{10}$
	RA	$85.00\% \pm 1.05\%^{7}$	$88.47\% \pm 0.73\%^{7}$	$87.00\% \pm 0.81\%^{10}$
	RAP	**85.39**$\% \pm 0.97\%^{6}$	$87.80\% \pm 0.82\%^{3}$	$86.34\% \pm 0.84\%^{10}$
	RP	$85.38\% \pm 0.87\%^{8}$	$87.11\% \pm 0.90\%^{4}$	$85.70\% \pm 0.88\%^{11}$
	P	$84.94\% \pm 1.03\%^{6}$	$86.40\% \pm 0.93\%^{5}$	$85.12\% \pm 0.92\%^{11}$
meta regression IoU_{adj}				
entropy baseline for **RP**:		$\sigma = 0.167 \pm 0.006$	$R^2 = 71.05\% \pm 2.58\%$	
		LR	**LR L1**	**LR L2**
σ	**R**	**0.128** $\pm 0.003^{2}$	0.129 ± 0.003^{2}	**0.128** $\pm 0.003^{2}$
	RA	0.134 ± 0.003^{2}	0.134 ± 0.003^{3}	0.134 ± 0.003^{2}
	RAP	0.129 ± 0.003^{7}	0.129 ± 0.003^{7}	0.129 ± 0.003^{7}
	RP	**0.128** $\pm 0.003^{7}$	**0.128** $\pm 0.002^{7}$	**0.128** $\pm 0.003^{7}$
	P	**0.128** $\pm 0.003^{7}$	0.129 ± 0.002^{7}	0.129 ± 0.003^{7}
R^2	**R**	$83.48\% \pm 0.99\%^{2}$	$83.37\% \pm 0.92\%^{2}$	$83.49\% \pm 0.96\%^{2}$
	RA	$82.06\% \pm 0.96\%^{2}$	$82.09\% \pm 0.94\%^{3}$	$82.08\% \pm 0.95\%^{2}$
	RAP	$83.38\% \pm 0.89\%^{7}$	$83.35\% \pm 0.90\%^{7}$	$83.40\% \pm 0.92\%^{7}$
	RP	**83.62**$\% \pm 0.91\%^{7}$	**83.54**$\% \pm 0.88\%^{7}$	**83.61**$\% \pm 0.91\%^{7}$
	P	$83.43\% \pm 0.90\%^{7}$	$83.36\% \pm 0.86\%^{7}$	$83.41\% \pm 0.91\%^{7}$
		GB	**NN L1**	**NN L2**
σ	**R**	0.114 ± 0.004^{5}	**0.114** $\pm 0.005^{1}$	**0.113** $\pm 0.005^{1}$
	RA	0.116 ± 0.004^{3}	0.118 ± 0.007^{1}	0.116 ± 0.005^{1}
	RAP	**0.112** $\pm 0.003^{7}$	**0.114** $\pm 0.003^{1}$	0.114 ± 0.005^{1}
	RP	**0.112** $\pm 0.002^{9}$	0.116 ± 0.004^{1}	0.115 ± 0.003^{2}
	P	0.114 ± 0.002^{11}	0.118 ± 0.004^{1}	0.117 ± 0.004^{3}
R^2	**R**	$87.02\% \pm 1.00\%^{5}$	$86.98\% \pm 1.07\%^{1}$	**87.16**$\% \pm 1.25\%^{1}$
	RA	$86.39\% \pm 1.11\%^{3}$	$85.94\% \pm 1.76\%^{1}$	$86.46\% \pm 1.32\%^{1}$
	RAP	**87.51**$\% \pm 0.61\%^{7}$	**87.03**$\% \pm 0.71\%^{1}$	$86.97\% \pm 1.10\%^{1}$
	RP	$87.45\% \pm 0.72\%^{9}$	$86.51\% \pm 0.88\%^{1}$	$86.69\% \pm 0.85\%^{2}$
	P	$86.88\% \pm 0.67\%^{11}$	$86.13\% \pm 0.95\%^{1}$	$86.24\% \pm 0.99\%^{3}$

3.4 Discussion

In this chapter, we presented our post-processing methods, the meta classification and regression, based on time-dynamics for semantic segmentation networks. To this end, we introduced a light-weight tracking algorithm for semantic segmentation. From matched segments, we generated time series of metrics and used these as inputs for the meta classifiers/regressors. The metrics are based on segments geometry and dispersion measures extracted from the segmentation network's softmax output, such as pixel-wise entropy. In our tests, we studied the influence of the time series length on different models for the meta tasks, i.e., gradient boosting, neural networks and linear ones. Our results showed significant improvements in comparison to those presented in [136]. More precisely, in contrast to the single-frame approach using only linear models, we increased the accuracy by 6.78 pp and the $AUROC$ by 5.04 pp. The R^2 value for meta regression was increased by 5.63 pp. We have shown that the time-dynamic method outperforms the single-frame approach and achieves good results in the meta tasks. For meta classification, classifying between $IoU_{\text{adj}} = 0$ and $IoU_{\text{adj}} > 0$, we achieved $AUROC$ values of up to 88.68% for the KITTI dataset. For meta regression, the direct prediction of the IoU_{adj} as quality estimate, we obtained R^2 values of up to 87.51%. To further increase the rating measures (like R^2 and $AUROC$), truly time-dynamic metrics can be developed, as the presented metrics are still single-frame based.

An idea for a time-dynamic metric is the shape preservation of tracked segments in consecutive frames, such as their size, width or height. Large deformations may indicate poorly predicted segments. Moreover, roughness measures can indicate the consistency of segments over time. To this end, time series of metrics, like size or geometric center, are used to predict the size/geometric center of a segment and compare this with the true value. Large deviations indicate that the segment is not consistent over time. These proposed time-dynamic metrics are based on time series of the segment's geometry. For this, we consider time series of tracked segments in more detail. In Fig. 3.12, a convex hull of a tracked car is shown. This is a typical time series of a moving car that drives faster than the ego car and thus, reduces its size over time. In contrast, the convex hull of a tracked segment related to class car is given in Fig. 3.13. The unstable shape results of segments building. At the beginning of the time series, the segment consists of one car and after further frames, it becomes a large segment of the class car consisting of several cars. If a segment varies over time, as in the example, it contradicts the idea of metrics like shape preservation or consistent time series of object's geometry. In Fig. 3.14, time series for mean entropy, size and IoU_{adj} of a segment of class car are shown. On the one hand, the time series are unstable and metrics vary significantly over time. On the other hand, the cars on the right hand side

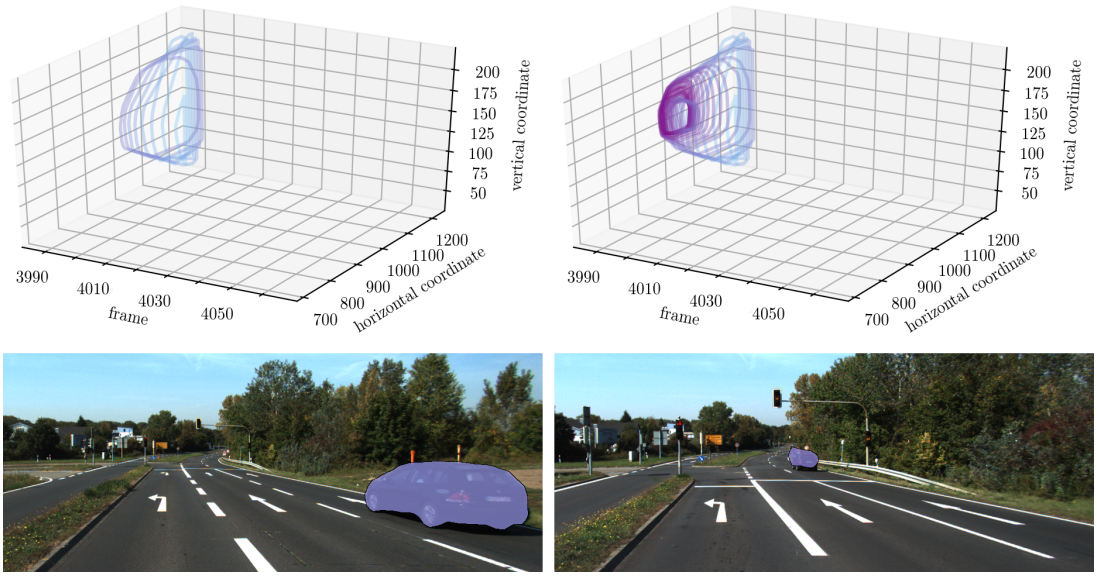


Figure 3.12: Convex hull of a tracked segment of class car in 85 consecutive frames. There are 25 time steps between the two frames shown here.

are predicted poorly and this reduces the IoU_{adj} and increases the entropy value for the whole segment, although all other cars are well predicted.

Another idea for a time-dynamic metric is based on survival analysis [111]. The remaining lifetime of segments can be predicted by information of the lifetimes of corresponding ground truth segments. Very short survival times suggest uncertainty. However, the first problem is that for used datasets (as well as for all other available datasets) ground truth data for multi-object tracking of segments is not available. For this reason, temporal metrics based on time series of ground truth data cannot be generated. Another consequence is that no evaluation of our light-weight tracking algorithm is possible without available ground truth data. These problems and the unstable time series, due to segments building, lead us to investigate instances rather than segments in further experiments. In instance segmentation, phenomena such as those shown in Fig. 3.13 do not appear due to the different definition of instances and segments.

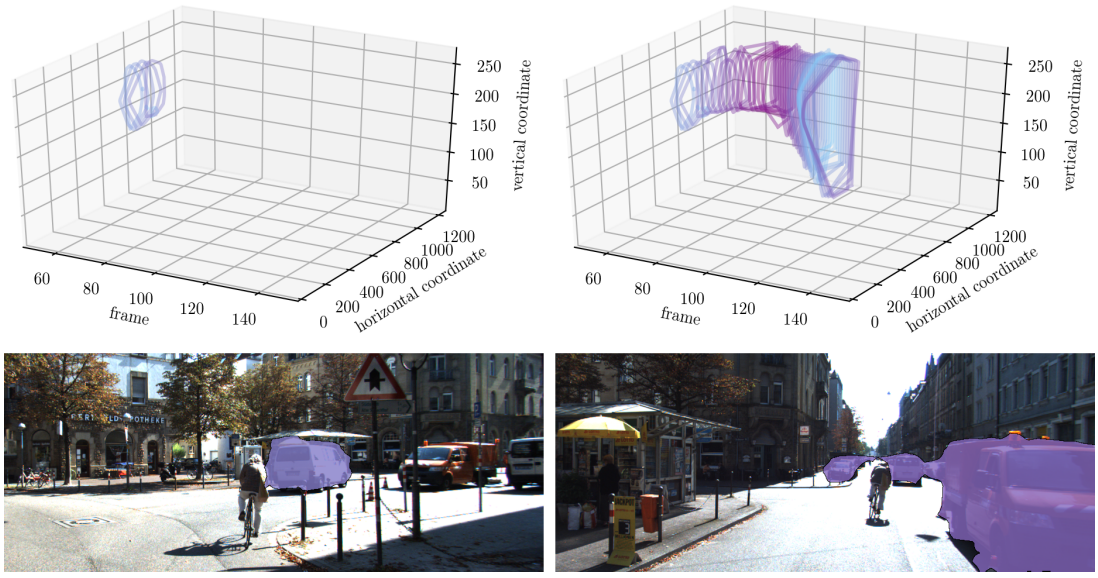


Figure 3.13: Convex hull of a tracked segment of class car in 104 consecutive frames. There are 42 time steps between the two frames shown here.

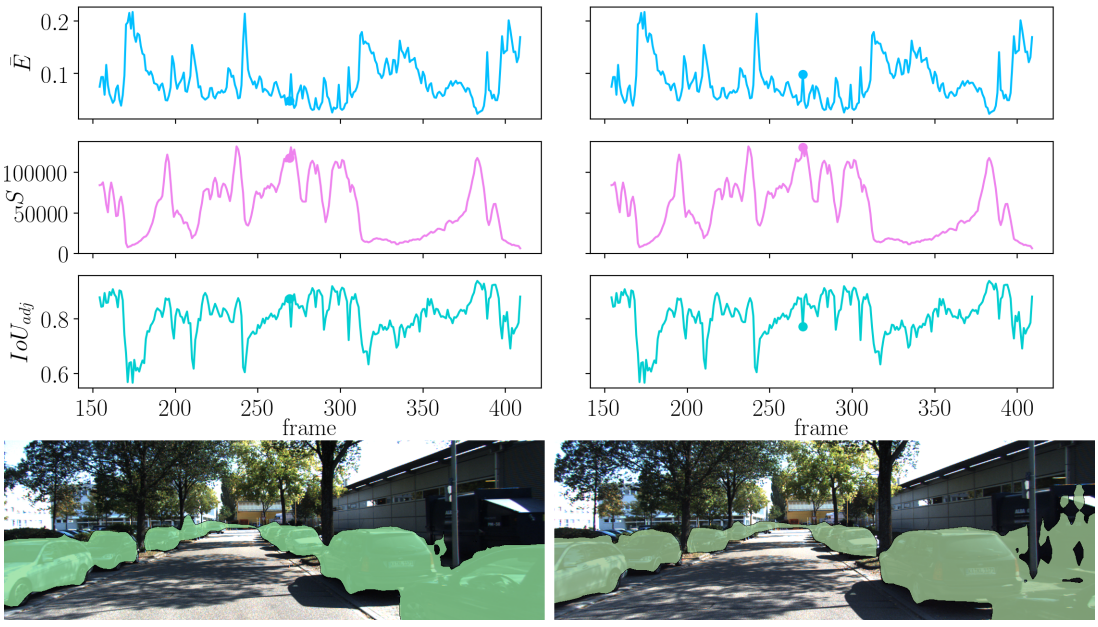


Figure 3.14: Time series of selected metrics $\bar{E}$, S and IoU_{adj}. The time series are constructed by tracking a segment of the class car in a video sequence. This segment is tracked over 255 frames. Here, two consecutive frames of these time series are shown. Green color corresponds to a high IoU_{adj} value.

Chapter 4

Improving Video Instance Segmentation by Light-weight Temporal Uncertainty Estimates

Object detection describes the task of identifying and localizing objects of a set of given classes. With respect to image data, state-of-the-art approaches are mostly based on convolutional neural networks. Localization can be performed, for example, by predicting bounding boxes or predicting each pixel that corresponds to a given instance. The latter is also known as instance segmentation (see Sec. 2.1.4) which is an essential tool for scene understanding and considered throughout this chapter. An example is given in Fig. 4.1. The reliability of neural networks in terms of prediction quality estimation [33, 136] and uncertainty quantification [44] is of highest interest, in particular, in safety critical applications like medical diagnosis [120] and automated driving [87]. However, instance segmentation networks such as YOLACT [13] and Mask R-CNN [59] do not give well adjusted uncertainty estimations [53]. These networks provide a confidence value, the score, for each instance which can have high values for false predictions and low ones for correct predictions. Confidence calibration [53] addresses this problem by adjusting the confidence values to reduce the error between the average precision (as a performance measure) and the confidence values [85]. During inference of instance segmentation networks, all instances with score values below a threshold are removed. It can happen that correctly predicted instances disappear, while many false positives remain. For this reason, we do not use a score threshold during inference and instead present an alternative based on an uncertainty quantification method that gives more accurate information compared to the simple score value. We utilize this uncertainty quantification to improve the networks' performance in terms of accuracy. Another approach to improve the

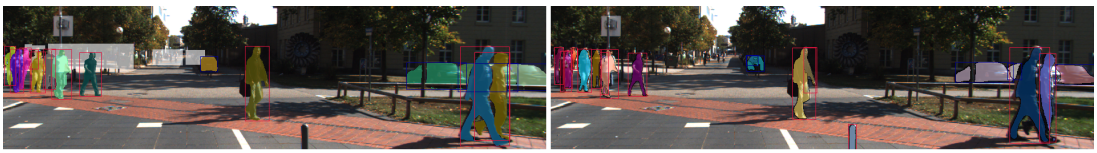


Figure 4.1: *Left*: Ground truth image with ignored regions (white). *Right*: Instance segmentation obtained by the Mask R-CNN network. The bounding boxes drawn around the instances represent the class, blue denotes cars and red pedestrians.

trade-off between false negatives and false positives has been introduced in [23]. Different decision rules are applied by introducing class priors which assign larger weight to underrepresented classes.

In this thesis, for the first time, we introduce the tasks of meta classification and meta regression for instance segmentation, which was previously introduced only for semantic segmentation, see [136] and Sec. 3.2.3. A semantic segmentation prediction is post-processed in order to estimate the quality of each predicted segment. Meta classification refers to the task of predicting whether a predicted instance intersects with the ground truth or not. A commonly used performance measure is the intersection over union which quantifies the degree of overlap of prediction and ground truth. In instance segmentation, a predicted object instance is called false positive if the IoU is less than 0.5. Hence, we consider the task of (meta) classifying between $IoU < 0.5$ and $IoU \geq 0.5$ for every predicted instance. We use meta classification to identify false positive instances and to improve the overall network performance compared to the application of a score threshold during inference. The task of meta regression is the prediction of the IoU for each predicted instance. Both meta classification and regression are able to reliably quantify the quality of an instance segmentation obtained from a neural network. In addition, the prediction of the IoU serves as a performance estimate. For learning both meta tasks, we use instance-wise metrics as input for the classifier/regressor. In Chapter 3, single-frame metrics were introduced which characterize uncertainty and geometry features of a given segment. By tracking these segments over time, time series of single-frame metrics are generated. In addition to those metrics applied to instance segmentation, we extend them by novel and truly time-dynamic metrics. These metrics are based on survival time analysis, on changes in the shape and on expected position of instances in an image sequence. From this information, we estimate the prediction quality on instance-level by means of temporal uncertainties. Additionally, for generating time series, we propose a light-weight tracking approach for predicted instances. Our tracking algorithm matches instances based on their overlap in consecutive frames by shifting instances according to their expected location in the subsequent frame predicted via linear regression.

In this chapter, we present post-processing methods for uncertainty quantification

and performance improvement in terms of accuracy. We only assume that a trained instance segmentation network and image sequences of input data are available. In our tests, we employ two publicly available networks, the YOLACT and the Mask R-CNN network (see Sec. 2.1.4). We apply these networks to the KITTI and the MOT dataset for multi-object tracking and instance segmentation (see Sec. 2.5). The source code of our method is publicly available at `http://github.com/kmaag/Temporal-Uncertainty-Estimates`. Our contributions are summarized as follows:

- We present a light-weight tracking algorithm for instances predicted by a neural network and resulting time-dynamic metrics. These metrics then serve as input for different models for meta classification and meta regression.

- We evaluate our tracking algorithm on the KITTI and the MOT dataset.

- We perform meta classification and regression to evaluate the quality of two state-of-the-art instance segmentation networks, the YOLACT and the Mask R-CNN network. Furthermore, we study different types of models for meta tasks w.r.t. their dependence on the length of time series and compare them with different baselines. For meta regression, we obtain R^2 values of up to 86.85% and for meta classification, $AUROC$ values of up to 98.78% which is clearly superior to the performance of previous approaches.

- For the first time, we demonstrate successfully that time-dynamic meta classification performance can be traded for instance segmentation performance. We compare the meta classification performance with the application of a score threshold during inference. Meta classification reduces the number of false positives by up to 44.03% while maintaining the number of false negatives.

This chapter is structured as follows. In Sec. 4.1, we show the differences to related work on uncertainty estimation and object tracking methods. This is followed by the presentation of our method in Sec. 4.2. First, the tracking algorithm for instances is presented in Sec. 4.2.1. Next, we describe our truly time-dynamic metrics in Sec. 4.2.2 and the meta classification as well as regression models in Sec. 4.2.3. The numerical results are presented in Sec. 4.3. We evaluate our light-weight tracking algorithm and investigate the properties of our temporal metrics. Furthermore, we perform meta regression and classification and study to which extent false positive detection can be traded for additional object detection performance. Finally, we conclude with a discussion in Sec. 4.4.

4.1 Related Work

We presented the related work on uncertainty quantification and confidence calibration in Sec. 2.2. For instance segmentation, only uncertainty estimation methods based on dropout sampling [112] exist so far. While MC Dropout [44, 70, 89, 169] is based on multiple runs of a network, our method can be applied to any network as a post-processing step using only information of the network output. In most confidence calibration works, the score value is adjusted by modifying the network architecture [107, 185] or by using additional information like training sample size [121] or bounding box positions [85]. Our method does not modify the network architecture and uses more information of the network output. In Chapter 3, we have introduced time-dynamic meta classification and regression for semantic segmentation. In this chapter, meta classification and regression are applied to instance segmentation. While we only provide time series of single-frame metrics in Chapter 3, we go beyond that and introduce truly time-dynamic metrics that quantify temporal changes in geometry and expected position complemented with quantities derived from a survival analysis. Furthermore, we demonstrate that time-dynamic meta classification performance can be traded for instance segmentation performance.

Deep learning is widely considered in object tracking approaches as described in Sec. 2.3.1. Methods for tracking instances are often based on the Mask R-CNN network extending this network by a tracking branch [176], by 3D convolutions to incorporate temporal information [162] or by a mask propagation branch [9]. Our tracking method is solely based on the degree of overlap of predicted instances. Following the tracking algorithm for segments (see Sec. 3.2.1), we introduce a method to track instances. Hence, we get away with a simple algorithm and less matching steps. For example, segments of the same class that are close to each other are merged to one segment, while this contradicts the idea of instance segmentation. Due to the lack of data in semantic segmentation, the tracking performance was not evaluated in Chapter 3. For instance segmentation, we evaluate our tracking algorithm and compare it with the deep learning approach presented in [162]. Note, in contrast to tracking methods integrated into object detection or instance segmentation networks, our approach is independent of the network and serves as a post-processing step.

4.2 Method

Instance segmentation is an extension of object detection. In both tasks, multiple bounding boxes with corresponding class affiliations are predicted. In instance segmentation, an additional pixel-wise mask representation is included. In both

tasks, first a score threshold is used to remove objects, and thereafter, a non-maximum suppression is applied to avoid multiple predictions for the same object. Our method is based on temporal information of these remaining instances. We track instances over multiple frames in image sequences and generate time-dynamic metrics. From this information, we estimate the prediction quality (meta regression) on instance-level and predict false positive instances (meta classification), also in order to improve the networks' performance in terms of accuracy. In the following, we describe our tracking approach, the temporal metrics as well as the methods used for meta classification and regression.

4.2.1 Tracking Method for Instances

In this section, we introduce a light-weight tracking algorithm for predicted instances in image sequences, where instance segmentation is available for each frame. Since our method is a post-processing step, the tracking algorithm is independent of the choice of an instance segmentation network. Each instance i of an image x has a label y from a prescribed label space $\mathcal{C}$. In Fig. 4.1 (left) a ground truth image is shown. Therein, the white areas are ignored regions with unlabeled cars and pedestrians. An evaluation for predicted instances in these regions is not possible, therefore all instances where 80% of their number of pixels are inside an ignored region are not considered for tracking and further experiments. Given an image x, $\hat{\mathcal{I}}_x$ denotes the set of predicted instances not covered by ignored regions. Following the procedure of the tracking algorithm for segments in Sec. 3.2.1, we match instances of the same class in consecutive frames as follows: Instances are matched according to their overlap or if their geometric centers are close to each other. For this purpose, we shift instances according to their expected location in the next frame using information from previous frames. We use a linear regression on t_l consecutive frames to match instances that are at least one and at most $t_l - 2$ frames apart in temporal direction in order to account for flashing instances (temporary false negatives or occluded instances). We define the overlap of two instances i and j by

$$\tilde{O}_{i,j} = \frac{|i \cap j|}{|i \cup j|} \tag{4.1}$$

(according to the formula of IoU) and the geometric center of instance i represented as pixel-wise mask in frame t by

$$\bar{i}_t = \frac{1}{|i|} \sum_{(z_v, z_h) \in i} (z_v, z_h) \tag{4.2}$$

where (z_v, z_h) describes the vertical and horizontal coordinate of pixel z. We denote by $\{x_t : t = 1, \ldots, T\}$ the image sequence with a length of T. Our tracking algorithm is applied sequentially to each frame $t = 1, \ldots, T$. The instances in

frame 1 are assigned with random IDs. Then, the ones in frame $t-1$ follow a tracking procedure to match with instances in frame t. To give priorities for matching, the instances are sorted by size and passed in descending order. A detailed description of how an instance $i \in \hat{\mathcal{I}}_{x_{t-1}}$ in frame $t-1$ is matched with an instance $j \in \hat{\mathcal{I}}_{x_t}$ in frame t is described in Algorithm 4.1. This is performed for every instance i. If the instances i and j are matched, the algorithm terminates and instance j is excluded from further considerations in order to preserve the instance uniqueness. When the algorithm has processed all instances $i \in \hat{\mathcal{I}}_{x_{t-1}}$, the instances $j \in \hat{\mathcal{I}}_{x_t}$ that have not been matched (e.g. newly occurring instances) are assigned with new IDs. Within the description of the tracking algorithm, we introduce parameters c_o, c_d and c_l as thresholds for overlap, distance and distance after shifting according to linear regression, respectively.

4.2.2 Temporal Instance-wise Metrics

First, we consider the metrics introduced in Sec. 3.2.2 applied to instances. These metrics are single-frame metrics based on an object's geometry, like instance size and geometric center, as well as dispersion measures extracted from the segmentation network's softmax output, such as an average over an instance's pixel-wise entropy. To calculate instance-wise metrics from any uncertainty heatmap (like pixel-wise entropy), we compute the mean of the pixel-wise uncertainty values of a given instance. In addition, an instance is divided into inner and boundary. The ratio of pixels in the inner and the boundary indicates fractal shaped instances which signals a false prediction. We analogously define uncertainty metrics for the inner and boundary since uncertainties may be higher on an instance's boundary. To this end, for each pixel z, a probability distribution over the class labels $y \in \mathcal{C}$, $\mathcal{C} = \{y_1, \ldots, y_c\}$, is required. In more detail, if the network provides a probability distribution per pixel, the set of metrics is given by

$$\begin{aligned} U^i = &\{S, S_{in}, S_{bd}, \tilde{S}, \tilde{S}_{in}\} \cup \{\bar{i}\} \cup \{P(y|i) \,:\, y = 1, \ldots, c\} \\ &\cup \{\bar{D}, \bar{D}_{in}, \bar{D}_{bd}, \tilde{\bar{D}}, \tilde{\bar{D}}_{in} \,:\, D \in \{E, V, M\}\}\,. \end{aligned} \tag{4.3}$$

If the network does not provide a probability distribution for each pixel, but only for the instance, we only use a smaller set of metrics presented in Sec. 3.2.2 that can be computed from the predicted instance mask, i.e.,

$$U^i = \{S, S_{in}, S_{bd}, \tilde{S}, \tilde{S}_{in}\} \cup \{\bar{i}\} \cup \{P(y|i) \,:\, y = 1, \ldots, c\} \cup \{E, \tilde{E}\}\,. \tag{4.4}$$

Instead of calculating the mean of pixel-wise uncertainty heatmaps, we compute the entropy E of the probability distribution for the instance and the relative entropy $\tilde{E} = E\tilde{S}$. We denote both sets of metrics by U^i independent of the

Algorithm 4.1: Tracking algorithm

/* shift */
if $t > 2$ *and instance* i *exists in frame* $t-2$ **then**
  shift instance i from frame $t-1$ by the vector $(\bar{i}_{t-1} - \bar{i}_{t-2})$
  if $\max_{j \in \hat{\mathcal{I}}_{x_t}} \tilde{O}_{i,j} \geq c_o$ **then**
    match instances i and j
  else if $\min_{j \in \hat{\mathcal{I}}_{x_t}} \|\bar{j}_t - \bar{i}_{t-1}\|_2 + \|(\bar{i}_{t-1} - \bar{i}_{t-2}) - (\bar{j}_t - \bar{i}_{t-1})\|_2 \leq c_d$ **then**
    match instances i and j
end
/* distance */
if $t > 1$ *and* i *does not exist in frame* $t-2$ **then**
  if $\min_{j \in \hat{\mathcal{I}}_{x_t}} \|\bar{j}_t - \bar{i}_{t-1}\|_2 \leq c_d$ **then**
    match instances i and j
  end
end
/* overlap */
if $t > 1$ *and* $\max_{j \in \hat{\mathcal{I}}_{x_t}} \tilde{O}_{i,j} \geq c_o$ **then**
  match instances i and j
end
/* regression */
if $t > 3$ *and instance* i *appears in at least two of the frames* $t - t_l, \ldots, t-1$ **then**
  compute geometric centers of instance i in frames $t - t_l$ to $t-1$ (in case i exists in all these frames)
  perform linear regression to predict the geometric center $(\hat{\bar{i}}_t)$
  if $\min_{j \in \hat{\mathcal{I}}_{x_t}} \|\bar{j}_t - \hat{\bar{i}}_t\|_2 \leq c_l$ **then**
    match instances i and j
  else
    shift instance $i \in \hat{\mathcal{I}}_{x_{t_{max}}}$ by the vector $(\hat{\bar{i}}_t - \bar{i}_{t_{max}})$ where $t_{max} \in \{t - t_l, \ldots, t-1\}$ denotes the frame where i contains the maximum number of pixels
    if $\max_{j \in \hat{\mathcal{I}}_{x_t}} \tilde{O}_{i,j} \geq c_o$ **then**
      match instances i and j
    end
  end
end

network. These metrics serve as a baseline in our tests and as a basis for the following metrics.

Next, we define six additional metrics mostly based on temporal information extracted from the tracking algorithm. Instance segmentation, as an extension of object detection, provides a confidence value for each instance. We add this *score value*, denoted by s, to our set of metrics.

The next metric is based on the variation of instances in consecutive frames. Instance i of frame $t-1$ is shifted such that i and its matched counterpart j in frame t have a common geometric center. Then, we calculate the overlap (4.1) as a measure of *shape preservation* f. Large deformations may indicate poorly predicted instances.

In the following, time series of the previously shown metrics are constructed. For each instance i in frame t, we gather a time series of the geometric centers $\bar{i}_k$, $k = t-5, \ldots, t-1$. If the instance exists in at least two previous frames, the geometric center $\hat{\bar{i}}_t$ in frame t is predicted using linear regression. The *deviation* between geometric center and expected geometric center $\|\hat{\bar{i}}_t - \bar{i}_t\|_2$ is used as a time-dynamic measure denoted by d_c. We proceed analogously with the instance size $S = |i|$ and compute the *size deviation* $|\hat{S} - S| =: d_s$. Small deviations d_c and d_s indicate that the predicted instance is consistent over time.

The following metric is based on a survival analysis [111] of the instances. All predicted instances that are matched in the previous 5 frames with the same ground truth instance are chosen. A predicted instance i and a ground truth instance g are considered as a match if they have an $IoU \geq 0.5$. The associated survival time of instance i in frame t is described by the number of frames in which ground truth instance g appears in consecutive frames. By the proposed tracking method, we obtain time series for each of the previously presented single-frame metrics $U_k^i \cup \{s_k\}$, $k = t-5, \ldots, t$. These serve as input for a Cox regression [30] survival model which predicts the *survival times* v of instances. A higher value of v indicates reliable instances, while a lower value suggests uncertainty.

The final measure is based on the height to width ratio of the instances. To this end, we calculate the average height to width ratio of ground truth instances $g_{\tilde{c}}$ per class $\tilde{c}$ by

$$r_{\tilde{c}}^{\mathrm{GT}} = \frac{1}{|g_{\tilde{c}}|} \sum_{g \in g_{\tilde{c}}} \frac{h_g}{w_g} \tag{4.5}$$

where h_g denotes the height and w_g the width of an instance g. We separate the ratio by class, as for instance cars and pedestrians typically have different ratios of height and width. False predictions can result in deviations of these typical ratios. The *ratio metric* for a predicted instance i is given by $r := (h_i/w_i)/r_{\tilde{c}}^{\mathrm{GT}}$.

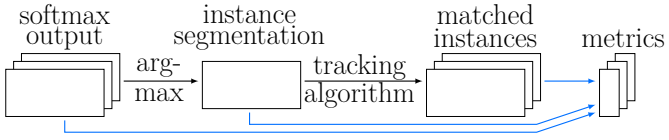


Figure 4.2: Overview of the metrics' construction that only requires a sequence of the softmax outputs. The information is extracted from the softmax output and further downstream from the instance segmentation and the instance tracking (*blue arrows*). The resulting metrics serve as input for the meta classifiers and regressors.

For the calculation of the ratio $r_{\tilde{c}}^{\mathrm{GT}}$ and the training of the survival model, only the ground truth data of the training set is used. Thus, the metrics r and v for instances in the test dataset can be determined without knowledge of ground truth. In summary, we use the following set of metrics

$$V^i = \{U^i\} \cup \{s, f, d_c, d_s, v, r\}\,. \tag{4.6}$$

An overview of the metrics' construction is shown in Fig. 4.2.

4.2.3 IoU Prediction

The intersection over union is a measure to determine the prediction accuracy of segmentation networks with respect to the ground truth. A predicted instance i is matched with a ground truth instance g if its overlap is the highest compared to the other ground truth instances and g is not yet matched with another predicted instance. Each ground truth instance can be matched with at most one predicted instance. The IoU is then calculated between these two instances i and g. In this thesis, we perform instance-wise predictions of the IoU (meta regression) comparing different regression approaches. In addition, we classify between $IoU < 0.5$ and $IoU \geq 0.5$ (meta classification) for all predicted instances. If the IoU of a predicted instance is less than 0.5, this instance is considered as a false positive. The metrics introduced in Sec. 4.2.2 serve as input for both prediction tasks. Just like for the survival model, we compute time series of these metrics. We have for an instance $i \in \hat{\mathcal{I}}_{x_t}$ in frame t the metrics V_t^i, as well as $V_{t'}^i$ from previous frames $t' < t$ due to object tracking. Meta classification and regression are performed by means of the metrics V_k^i, $k = t - n_c, \ldots, t$, where n_c describes the number of considered frames. For regression, we use gradient boosting regression (GB), shallow neural networks containing only a single hidden layer consisting of 50 neurons with ℓ_1/ℓ_2-penalization (NN L1/NN L2) and linear regression with ℓ_1/ℓ_2-penalization (LR L1/LR L2) as well as without penalization (LR). Gradient boosting, a shallow neural network with ℓ_2-penalization and logistic regression with ℓ_1-penalization are considered for classification. Details of the used methods are described in Sec. 2.4. We analyze the benefit from using time series and the extent to which the additional metrics $V^i \setminus U^i$ yield improvements.

4.3 Numerical Results

In this section, we evaluate our light-weight tracking algorithm and investigate the properties of the metrics defined in the previous section. We perform meta regression and classification and study the influence of the length of the time series considered as well as different methods for the meta tasks. Furthermore, we study to which extent false positive detection can be traded for additional object detection performance and thus, serve as an advanced score. To this end, we compare meta classification with ordinary score thresholds in terms of numbers of false positives and false negatives. We perform our tests on the KITTI dataset for multi-object tracking and instance segmentation, which contains 21 street scene videos from Karlsruhe (Germany) consisting of 1,242 $\times$ 375 8,008 images. Additionally, we use the MOT dataset with scenes from pedestrian areas and shopping malls, which consists of 2,862 images with resolutions of 1,920 $\times$ 1,080 (3 videos) and 640 $\times$ 480 (1 video). For both datasets, annotated image sequences are available [162]. In contrast to the KITTI dataset, the MOT dataset only includes labels for the class pedestrian, but not for car. Details of both datasets are shown in Sec. 2.5. In our experiments, we consider two different networks, the YOLACT network and the Mask R-CNN which are introduced in Sec. 2.1.4. The YOLACT network has a slim architecture designed for a single GPU. We retrain the network using a ResNet-50 [61] backbone and starting from backbone weights for ImageNet [140]. We choose 12 image sequences consisting of 5,027 from the KITTI dataset (same splitting as in [162]) and 300 images from the MOT dataset for training. As validation set, we use the remaining 9 sequences of the KITTI dataset, achieving a mean average precision (mAP) of 57.06%. The Mask R-CNN focuses on high-quality instance-wise segmentation masks. As backbone, we use a ResNet-101 and weights for COCO [95]. We choose the same 12 image sequences of the KITTI dataset as training set as well as 9 sequences as validation set achieving a mAP of 89.87%. In the training of both networks, the validation set is neither used for parameter tuning nor early stopping. We choose a score threshold of 0 to use all predicted instances for further experiments. For instance tracking (see Algorithm 4.1), we use the following values: $c_o = 0.35$, $c_d = 100$ and $c_l = 50$. In our experiments, we use metrics extracted from the segmentation network's softmax output. The Mask R-CNN outputs a probability distribution per pixel, while the YOLACT network only returns one probability per instance. For this reason, the set U^i consists of more metrics for the Mask R-CNN network.

4.3.1 Evaluation of our Tracking Algorithm

For the evaluation of our tracking algorithm, we use common object tracking metrics introduced in Sec. 2.3.2. The following metrics precision, recall, F-measure

Table 4.1: Mismatch ratio $\overline{mme}$ and *MOTA* results obtained by TrackR-CNN and by our tracking approach.

	TrackR-CNN		ours	
	$\overline{mme}$	*MOTA*	$\overline{mme}$	*MOTA*
MOT	0.0117	0.6657	0.0185	0.6589
KITTI	0.0153	0.7993	0.0235	0.7911

and *FAR* are based on the errors of the object detection like false positive and false negative objects. Furthermore, the multiple object tracking precision (*MOTP*) corresponds to the averaged distance between geometric centers (geo) or bounding box centers (bb) of matched ground truth and predicted instances. We consider these metrics as performance measures for object tracking methods. Moreover, the multiple object tracking accuracy (*MOTA*) focuses on the tracking of objects and also of the detection. The *MOTA* is based on three error ratios, the ratio of false negatives, false positives and mismatches ($\overline{mme}$). A mismatch error occurs when the ID of a predicted instance that was matched with a ground truth instance changes. In addition, we define by GT all ground truth objects of an image sequence which are identified by different IDs and divide these into three cases. An object is mostly tracked (MT) if it is tracked for at least 80% of frames (out of the total number of frames in which it appears), mostly lost (ML) if it is tracked for less than 20%, else partially tracked (PT).

In [162], the TrackR-CNN is presented which tracks objects and performs instance segmentation, both by means of a single neural network. To this end, the Mask R-CNN network is extended by 3D convolutions to incorporate temporal information. This method is tested on the MOT dataset and the validation dataset of KITTI. To compare our approach with the TrackR-CNN method, we use the instances predicted by this network and apply our tracking algorithm to these instances to assess only the tracking quality (not the instance prediction quality of the network). The performance metrics mainly evaluate the object detection, while object tracking is only evaluated in the ratio of mismatches and *MOTA*. For this reason, we consider the latter two performance metrics for our comparison on the TrackR-CNN instances. The results are given in Table 4.1. We obtain for the MOT dataset a slightly higher mismatch ratio than TrackR-CNN and a *MOTA* value which is only 0.68 pp lower. For the KITTI dataset, the results are similar. In summary, our results are slightly weaker in tracking performance than TrackR-CNN, however, our algorithm does not use deep learning and is independent of the choice of instance segmentation network which enables us to conduct time-dynamic uncertainty quantification for any given instance segmentation network.

For further tests, we use the YOLACT and the Mask R-CNN network for instance prediction. The results of object tracking metrics and performance measures for

Table 4.2: Object tracking results and performance measures for the KITTI dataset.

	precision	recall	F-measure	*FAR*	$MOTP_{\text{bb}}$	$MOTP_{\text{geo}}$
YOLACT	0.3186	0.7211	0.4420	294.20	3.34	2.68
Mask	**0.5546**	**0.9379**	**0.6970**	**143.74**	**2.61**	**2.39**
	GT	*MT*	*PT*	*ML*	$\overline{mme}$	*MOTA*
YOLACT	219	124	75	20	**0.0539**	−0.8746
Mask	219	204	13	2	0.0564	**0.1281**

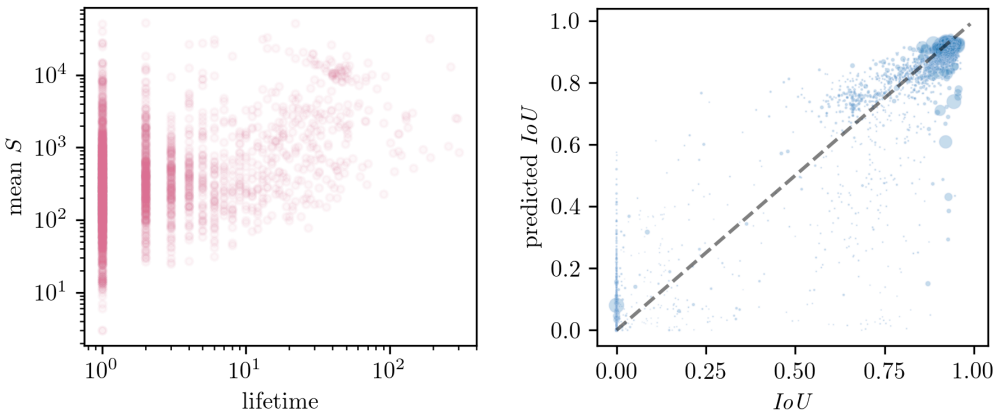


Figure 4.3: *Left*: Instance lifetime (time series length) vs. mean instance size, both on log scale. *Right*: Predicted *IoU* vs. *IoU* for all predicted instances. The dot size is proportional to the instance size.

the KITTI dataset are shown in Table 4.2. Mask R-CNN mainly achieves better results than YOLACT. This can be attributed to the fact that Mask R-CNN achieves higher *mAP* values than YOLACT. In Fig. 4.3 (left) the correlation between the instance lifetime and the mean instance size is depicted for the Mask R-CNN. On average, a predicted instance exists for 6.0 frames and when we consider only instances that contain at least 1,000 pixels, the average life time increases to 16.0 frames. For the YOLACT network, a predicted instance exists for 8.8 frames and this can be increased to 16.4 frames.

4.3.2 Meta Classification and Regression

While the KITTI dataset contains 10 frames per second videos of street scenes, the MOT dataset contains mostly 30 fps videos of pedestrian scenes. Due to slower motions, the data extracted from the images is very redundant and we get fewer different instances. As an example, two consecutive frames with an offset of two seconds are shown in Fig. 4.4. This shows the less variation in the MOT videos and thus, the redundancy of the images. As a consequence, significant

Figure 4.4: Two frames of the MOT dataset. Between both frames is a time gap of two seconds, i.e., 60 frames. *Top*: Ground truth with ignored regions (white). *Bottom*: Instance segmentation obtained by the Mask R-CNN. Each color correspond to an instance ID.

Table 4.3: Correlation coefficients ρ with respect to IoU for the YOLACT network.

s	0.87603	d_s	0.27549	S	0.17161	$\tilde{S}$	0.26096	E	−0.29414
f	0.80331	v	0.81413	S_{in}	0.17042	$\tilde{S}_{in}$	0.26096	$\tilde{E}$	0.25904
d_c	0.41061	r	0.74152	S_{bd}	0.19498	$\bar{i}_v$	0.07208	$\bar{i}_h$	0.02051

overfitting problems occur when applying meta classification and regression. Also downsampling the frame rate is not an option due to the lack of data. Hence, we only consider the KITTI dataset for further experiments. In the following, we study the predictive power of the metrics, present the results of the meta tasks and compare our methods with different baselines.

In order to investigate the predictive power of the metrics, we compute the Pearson correlation coefficients ρ between the metrics V^i and the IoU. The results for the YOLACT network are given in Table 4.3. The score value s, the shape preservation metric f, the survival metric v as well as the ratio r demonstrate a strong correlation with the IoU. The corresponding scatter plots are shown in Fig. 4.5. The YOLACT network only provides a probability distribution per instance, while the Mask R-CNN outputs pixel-wise probability distributions. For this reason, we obtain more metrics for the Mask R-CNN, see Table 4.4. As for the YOLACT network, the Mask R-CNN demonstrates strong correlations between the IoU as well as the score value s and the shape preservation metric f. Moreover, high correlations are shown for the mean entropy $\bar{E}$ and the mean variation ratio $\bar{V}$ as well as for the mean variation ratio on the boundary

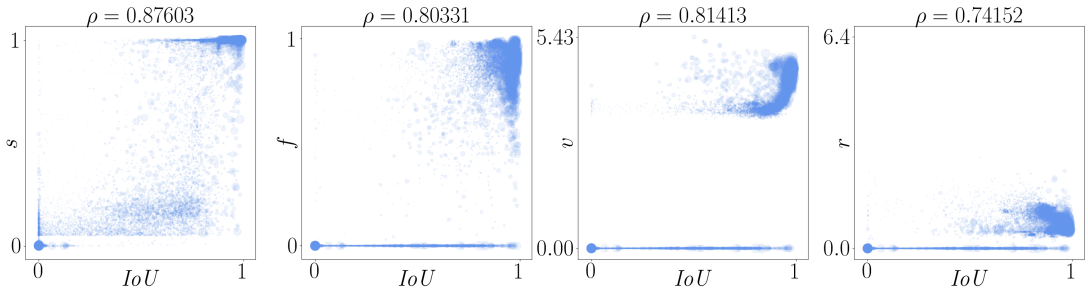


Figure 4.5: Correlations between metrics and IoU for the YOLACT network. The dot size is proportional to the instance size.

Table 4.4: Correlation coefficients ρ with respect to IoU for the Mask R-CNN.

s	0.73026	S	0.29241	$\bar{E}$	−0.70813	$\bar{V}$	−0.61669	$\bar{M}$	−0.32468
f	0.60332	S_{in}	0.28913	$\bar{E}_{in}$	0.00395	$\bar{V}_{in}$	−0.14498	$\bar{M}_{in}$	−0.05736
d_c	−0.10817	S_{bd}	0.37846	$\bar{E}_{bd}$	−0.42140	$\bar{V}_{bd}$	−0.61522	$\bar{M}_{bd}$	−0.64857
d_s	0.11985	$\tilde{S}$	0.45847	$\tilde{\bar{E}}$	0.45809	$\tilde{\bar{V}}$	0.46154	$\tilde{\bar{M}}$	0.46599
v	0.20490	$\tilde{S}_{in}$	0.45847	$\tilde{\bar{E}}_{in}$	0.45835	$\tilde{\bar{V}}_{in}$	0.46349	$\tilde{\bar{M}}_{in}$	0.46821
r	−0.10071	$\bar{i}_v$	0.15245	$\bar{i}_h$	0.05788				

$\bar{V}_{bd}$ and the mean probability margin on the boundary $\bar{M}_{bd}$. As a consequence, the single-frame metrics U^i extracted from the network's output can be used as a baseline. For details of the construction of these metrics see Sec. 3.2.2. In comparison to semantic segmentation where typically a cross-entropy loss (2.14) and a softmax function (2.6) per pixel are used, the Mask R-CNN uses a binary loss and the sigmoid function (2.3) per pixel. This results in a mask generation without competition among classes and high values for the dispersion measures like the pixel-wise entropy (see Fig. 4.6). The differences of the mean entropy between predicted segments and instances for the KITTI dataset are shown in Fig. 4.7. For segments, we obtain mean entropy values between 0 and 1 while for instances these values are greater than 0.9. However, the results are similar in both cases. For high uncertainty, i.e., high entropy values, the IoU/IoU_{adj} values are very low and vice versa.

For meta classification (false positive detection: $IoU < 0.5$ vs. $IoU \geq 0.5$) and meta regression (prediction of the IoU), we use the KITTI validation set consisting of 2,981 images. In our tests, we choose relatively low score thresholds, i.e., 0.1 for YOLACT and 0.4 for Mask R-CNN. This is done to balance the number of false positives, such that the meta tasks obtain enough training data and the tracking performance is not significantly degraded. As input for the meta models, we use a combination of the presented metrics and further, study their predictive power as well as the influence of time series. Firstly, we only present the instance-wise metrics V_t^i of a single frame t to the meta classifier/regressor,

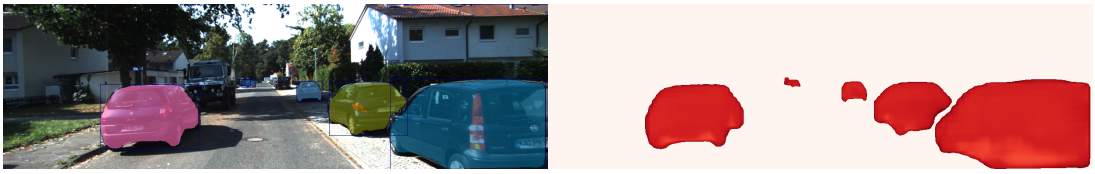


Figure 4.6: *Left*: Instance segmentation predicted by the Mask R-CNN. *Right*: Corresponding pixel-wise entropy. The missing predictions of the cars are located in ignored regions and for this reason, not considered.

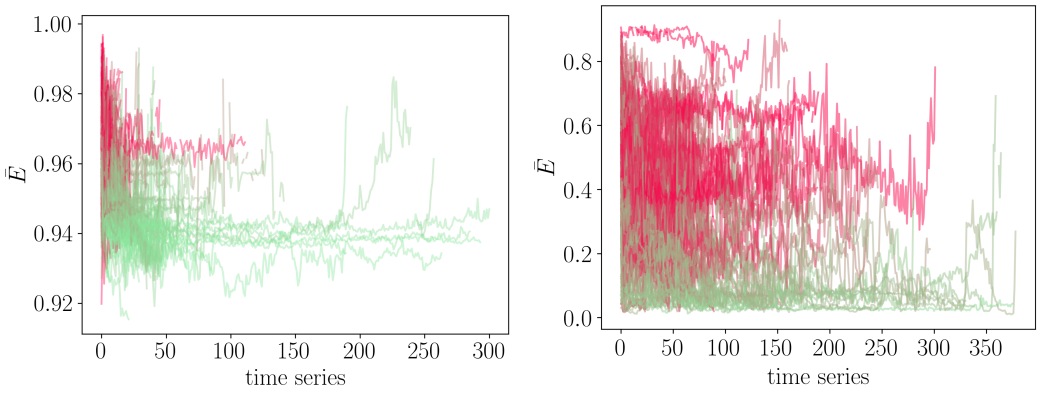


Figure 4.7: *Left*: The mean entropy metric for all tracked instances as time series. *Right*: Time series for all tracked segments where only time series with a length of at least 50 frames are included. The instances are predicted by the Mask R-CNN and the segments by the MobilenetV2 both for the KITTI dataset. The color corresponds to the average IoU/IoU_{adj} value for each instance/segment, green color for high values and red color for low ones.

secondly we extend the metrics to time series with a length of up to 10 previous time steps V_k^i, $k = t-10, \ldots, t-1$. For the presented results, we apply a (meta) train/validation/test splitting of 70%/10%/20% and average the results over 10 runs obtained by randomly sampling the splitting. In tables and figures, the corresponding standard deviations are provided.

For the YOLACT network, we obtain roughly 13,074 instances (not yet matched over time) of which 4,486 have an $IoU < 0.5$. For the Mask R-CNN this ratio is 17,211/6,614. For meta classification, we consider as performance measures the classification accuracy and $AUROC$ (Sec. 2.4.4). For meta regression, we compute the standard errors σ and R^2 values (Sec. 2.4.4). In Fig. 4.8 the results for regression R^2 (right) and for classification $AUROC$ (left) as functions of the number of frames for both networks are given. We observe that the methods benefit from temporal information, although there are significant differences between them. The best results (over the course of time series lengths) for meta classification and meta regression are given in Table 4.5. Gradient boosting shows

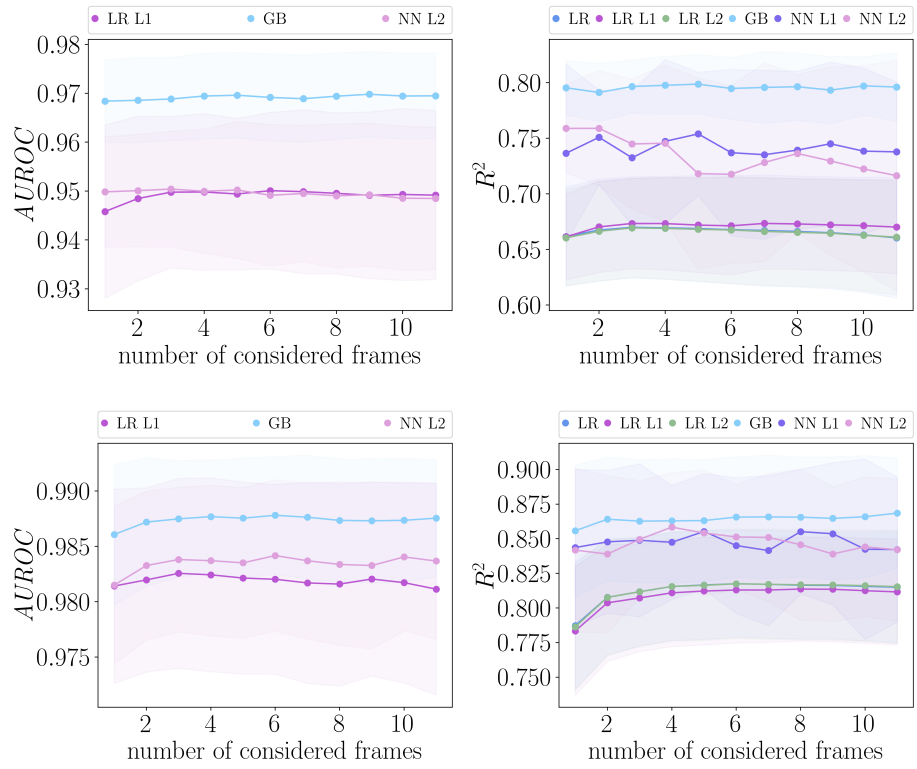


Figure 4.8: *Top left*: Results for meta classification *AUROC* as functions of the number of frames for the YOLACT network. *Bottom left*: Meta classification results for the Mask R-CNN. *Top right*: Results for meta regression R^2 as functions of the number of frames for the YOLACT network. *Bottom right*: Meta regression results for the Mask R-CNN.

the best performance in comparison to linear models and neural networks with respect to all classification and regression measures. For meta classification, we achieve *AUROC* values of up to 98.78%. In Fig. 4.9, the influence of the metrics V^i on the meta classification performance for the YOLACT network is shown. The time series length of used metrics corresponds to the highest *AUROC* value achieved by incorporating 5 previous frames for the linear model and 8 previous frames for gradient boosting. In both plots, the score s, the shape preservation metric f, the relative entropy $\tilde{E}$ and metrics based on instance sizes have the most influence on the performance. Furthermore, the influence gets smaller as the time series length increases. For meta regression, the highest R^2 value of 86.85% for the Mask R-CNN network is obtained by incorporating 10 previous frames. For this specific case, the correlation of the calculated and predicted IoU is depicted in Fig. 4.3 (right) and an illustration of the resulting quality estimate is shown in Fig. 4.10. We also provide video sequences[1] that visualize the IoU prediction

[1] `http://youtu.be/6SoGmsAarTI`

Table 4.5: Results for meta classification and regression for the different meta classifiers and regressors. The super script denotes the number of frames where the best performance and in particular, the given values are reached.

YOLACT			
meta classification $IoU < 0.5, \geq 0.5$			
	LR L1	**GB**	**NN L2**
ACC	$90.22\% \pm 3.06\%^{2}$	$\mathbf{92.62}\% \pm 2.48\%^{9}$	$90.49\% \pm 2.34\%^{1}$
$AUROC$	$95.01\% \pm 1.60\%^{6}$	$\mathbf{96.98}\% \pm 0.87\%^{9}$	$95.04\% \pm 1.19\%^{3}$
meta regression IoU			
	LR	**LR L1**	**LR L2**
σ	0.165 ± 0.017^{3}	0.164 ± 0.013^{7}	0.165 ± 0.017^{3}
R^2	$66.99\% \pm 4.56\%^{3}$	$67.34\% \pm 4.10\%^{7}$	$66.94\% \pm 4.46\%^{3}$
	GB	**NN L1**	**NN L2**
σ	$\mathbf{0.130} \pm 0.018^{5}$	0.142 ± 0.021^{5}	0.141 ± 0.021^{2}
R^2	$\mathbf{79.87}\% \pm 2.66\%^{5}$	$75.40\% \pm 5.55\%^{5}$	$75.89\% \pm 5.19\%^{2}$
Mask R-CNN			
meta classification $IoU < 0.5, \geq 0.5$			
	LR L1	**GB**	**NN L2**
ACC	$93.43\% \pm 1.78\%^{4}$	$\mathbf{95.07}\% \pm 1.24\%^{5}$	$94.31\% \pm 1.65\%^{4}$
$AUROC$	$98.26\% \pm 0.86\%^{3}$	$\mathbf{98.78}\% \pm 0.53\%^{6}$	$98.42\% \pm 0.68\%^{6}$
meta regression IoU			
	LR	**LR L1**	**LR L2**
σ	0.168 ± 0.017^{6}	0.170 ± 0.017^{8}	0.168 ± 0.017^{6}
R^2	$81.75\% \pm 3.87\%^{6}$	$81.36\% \pm 3.76\%^{8}$	$81.75\% \pm 3.95\%^{6}$
	GB	**NN L1**	**NN L2**
σ	$\mathbf{0.142} \pm 0.020^{11}$	0.149 ± 0.020^{5}	0.148 ± 0.025^{4}
R^2	$\mathbf{86.85}\% \pm 3.96\%^{11}$	$85.52\% \pm 4.19\%^{5}$	$85.84\% \pm 3.94\%^{4}$

and instance tracking.

In Fig. 4.11 (right), we compare our best results for meta classification and regression for both networks with the following baselines. Our results in Sec. 3.3 are compared with a single-metric baseline as the mean entropy. We apply as single-metric also the mean entropy per instance as well as the score value using single-frame gradient boosting. In addition, the approach in [136] can be considered as a baseline. For this, we only apply the metrics U^i for a single frame and the linear models. For our time-dynamic extension introduced in Chapter 3 as baseline, the metrics U^i are also used, however, as time series and as input for the best performing meta model. The best results for each method are displayed in Fig. 4.11 (right). We observe that our method using metrics V^i and gradient boosting outperforms all baselines. In Fig. 4.11 (left), a comparison is shown between metrics U^i, all single-frame metrics (U^i plus score and ratio) and all metrics including the temporal ones. As shown before, the metrics U^i are clearly outperformed. The single-frame metrics obtain significantly lower R^2 values compared to all metrics V^i. This difference can be observed when applying linear models

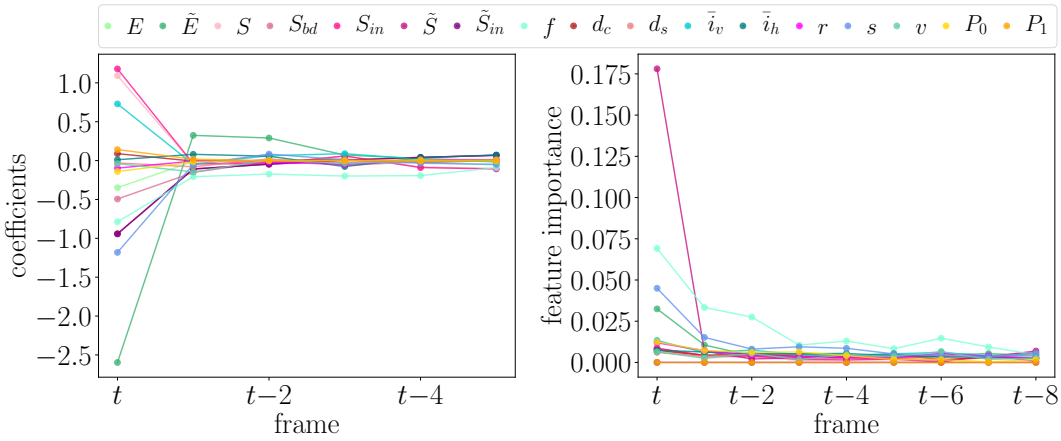


Figure 4.9: *Left*: The weight coefficients for the 15 metrics and predicted class probabilities $P(y|i)$ denoted by P_i, $i = 0, 1$, computed with LASSO fits (linear regression with ℓ_1-penalization). *Right*: Feature importance analysis of these metrics obtained by gradient boosting. For both figures, the time series length corresponds to the number of considered frames used to achieve the meta classification results in Table 4.5 for the YOLACT network.

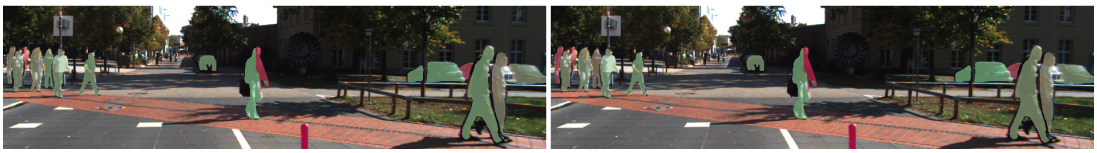


Figure 4.10: *Left*: A visualization of the true instance-wise IoU of prediction and ground truth. Green color corresponds to high IoU values and red color to low ones. *Right*: Instance-wise IoU prediction obtained from meta regression. Corresponding ground truth and instance segmentation are shown in Fig. 4.1.

as well as neural networks, whereas it is rather small when using gradient boosting. Gradient boosting has a strong tendency to overfitting and due to the small amount of data, we suspect that the gap between the performance of different metrics would also increase with more data. In summary, we outperform all baselines by using our time-dynamic metrics as input for meta classifiers/regressors. We reach $AUROC$ values of up to 98.78% for classifying between true and false positives.

4.3.3 Advanced Score Values

In object detection, the score value describes the confidence of the network's prediction. During inference, a score threshold removes false positives. If the

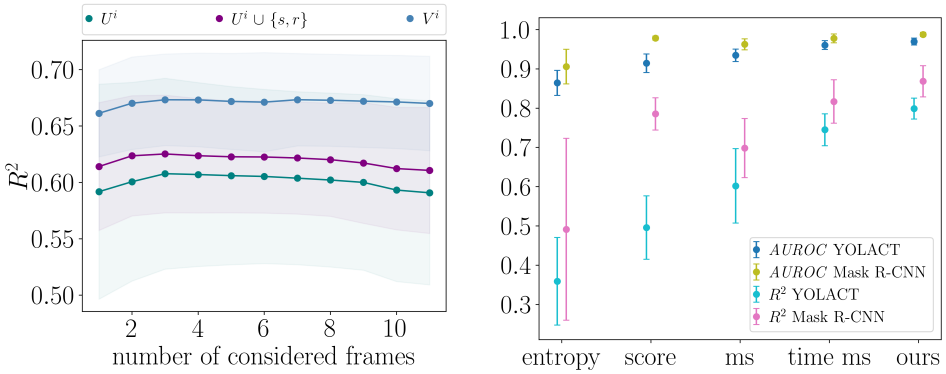


Figure 4.11: *Left*: Meta regression via linear regression with ℓ_1-penalization for various input metrics and for the YOLACT network. *Right*: Different baselines for both networks comparing $AUROC$ and R^2 values. From left to right: mean entropy, score value, single-frame MetaSeg (ms) approach [136] with linear models, our time-dynamic extension presented in Chapter 3 using metrics U^i and the best performing meta model, our method using metrics V^i and also the best performing meta model.

threshold is raised to a higher value, not only false positives are removed, but also true positives. We study the network's detection performance while varying the score threshold. In total, we select 30 different score thresholds from 0.01 to 0.98. Meta classification provides a probability of observing a false positive given a predicted instance. We threshold on this probability also with 30 different thresholds and compare this to ordinary score thresholding. To this end, we feed gradient boosting as meta classifier with all metrics V^i including 5 previous frames. In Fig. 4.12 the performance is stated in terms of the number of remaining false positives and false negatives. Each point represents one of the chosen thresholds. For the YOLACT network, the meta classification achieves a lower number of errors, i.e., less false positives and false negatives. In comparison to score thresholding, we can reduce the number of false positives by up to 44.03% for the approximate same number of false negatives. This performance increase is also reflected in the mAP. The highest mAP value obtained by score thresholding is 57.04% while meta classification achieves 58.22%. In Fig. 4.13, the mAP values for score and meta classification thresholding are given for the YOLACT network. The gap between both methods is shown as well as the performance increase by meta classification. When applying the Mask R-CNN, we can reduce the number of false positives up to 43.33% for the approximate same number of false negatives. The maximum mAP values for both methods are similar. Score thresholding achieves an mAP of up to 89.87% and meta classification of up to 89.83%. For both networks, we can improve the networks' performance by reducing false positives while the number of false negatives remains almost unchanged. Our method can be applied after training an instance segmentation

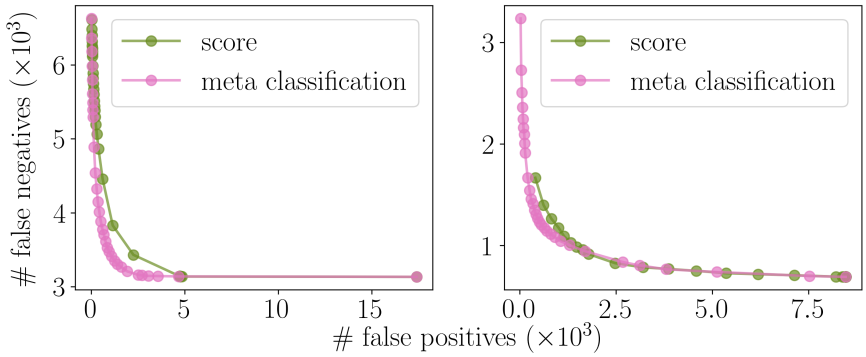


Figure 4.12: *Left*: Number of false positive vs. false negative instances for different thresholds for the YOLACT network. *Right*: Results for the Mask R-CNN.

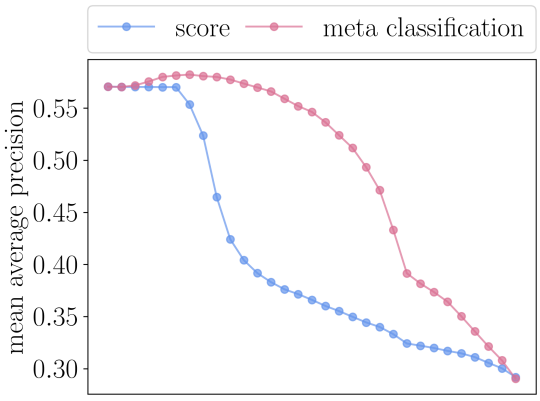


Figure 4.13: Mean average precision values for different thresholds and the YOLACT network.

network and does not increase the network complexity. Up to some adjustment in the considered metrics, this approach is applicable to all instance segmentation networks.

4.4 Discussion

In this chapter, we proposed post-processing methods for instance segmentation networks, namely meta classification and meta regression. These methods are based on temporal information and can be used for both uncertainty quantification and accuracy improvement. We introduced and evaluated our light-weight tracking algorithm for instances, that is independent of the instance segmentation network. From tracked instances, we constructed time-dynamic metrics (time series) and used these as inputs for the meta tasks. On the one hand, these metrics

Figure 4.14: *Top*: Ground truth of two different images with ignored regions (white). *Bottom*: Instance segmentation obtained by the YOLACT network resulting in non-detected ground truth instances, i.e., false negatives (cyan colored bounding box at the top).

characterize uncertainty and geometry of instances. On the other hand, we used truly time-dynamic ones based on survival time analysis, on changes in the shape and on expected position of instances in an image sequence. In our tests, we studied the influence of these metrics, various time series lengths and different models for the meta classification and regression. For meta classification, classifying between $IoU < 0.5$ and $IoU \geq 0.5$, we obtained $AUROC$ values of up to 98.78% and for meta regression, direct prediction of the IoU, R^2 values of up to 86.85% which is clearly superior to the performance of the baseline methods. Using meta classification we also improved the networks' prediction accuracy by replacing the score threshold by the estimated probability of correct classification during inference. We reduced the number of false positives by up to 44.03% while maintaining the number of false negatives.

If we change the point of view in Fig. 4.12, we can reduce the number of false negatives by up to 15.70% for the approximate same number of false positives using meta classification. However, the number of false negatives, especially in the YOLACT network, is still high with about 3K. In Fig. 4.14, two example images of an instance segmentation are shown. In one instance segmentation (left), two pedestrians are not detected. In this frame, both pedestrians are on the sidewalk, although they may cross the street a few seconds later. For this reason, the detection of pedestrians at an early stage is important to prevent hazardous scenarios. In the other instance segmentation (right), the vehicle parking left of the ego car is not detected, even though the vehicle is close. Due to the non-detection, an accident can occur if the ego car is going to park on the left side or moves to the left due to the narrow road. The two examples of the non-detected instances illustrate the significance of minimizing false negative instances.

Up to now, we have focused on the detection of false positive segments and in-

stances. This is important since, for example, in automated driving, the flow of traffic should not be impaired by falsely detected objects. On the other hand, we have demonstrated in Fig. 4.14 that false negative objects can lead to dangerous situations and therefore, should be minimized. In the following chapter, we study the reduction of false negative instances in image sequences using uncertainty information.

Chapter 5

False Negative Reduction in Video Instance Segmentation using Uncertainty Estimates

Instance segmentation combines object detection, which means the task of categorizing as well as localizing objects using bounding boxes, and semantic segmentation, i.e., the pixel-wise classification of image content. In instance segmentation, the localization of objects is performed by predicting each pixel that corresponds to a given instance. Thereby, instance segmentation provides precise localization on instances of important classes. State-of-the-art approaches are mostly based on convolutional neural networks. Neural networks as statistical models produce probabilistic predictions prone to error. For this reason, it is necessary to understand and minimize these errors. In safety critical applications like automated driving [87] and medical diagnosis [120], the reliability of neural networks in terms of uncertainty quantification [44] and prediction quality estimation [33] is of highest interest. The prediction quality estimation is also considered in Chapter 4 for instances. Instance segmentation networks (for example Mask R-CNN [59] and YOLACT [13]) provide for each object a score (confidence) value. However, these scores do not correspond to a well-adjusted uncertainty estimation [53] as they can have low values for correctly predicted instances and high values for false predictions. This problem is addressed by confidence calibration [85] where the confidence values are adjusted to improve the prediction reliability. During inference of an instance segmentation network, a score threshold is applied to remove all instances with low confidences. This is done to balance the number of false positive and false negative instances. Nevertheless, this can result in correctly predicted instances vanishing as well as many false positives remaining. These errors shall be reduced to improve network performance in terms of accuracy.

In applications such as automated driving, the detection of road users, i.e., the reduction of false negative instances, is particularly important. In other words, it is preferable to incorrectly predict the presence of road users than to missing them. For this reason, we use a relatively small score threshold value during inference and apply a light-weight false negative detection algorithm on these remaining instances to find possible overlooked ones. Since the number of predicted instances can be greatly increased by our algorithm to reduce false negatives, we use false positive detection based on uncertainty estimates to prune them. We utilize this fused approach to improve the networks' performance and to reduce false negatives, i.e., attain a high recall rate, compared to using ordinary score thresholds.

In this chapter, we introduce a false negative reduction method based on uncertainty estimates for instance segmentation networks. Our approach serves as a post-processing step applicable to any network. First, the predicted instances obtained by a neural network are tracked in consecutive frames. Next, our light-weight detection algorithm is applied which is based on inconsistencies in the time series of the tracked instances such as a gap in the time series or a sudden end. We detect these cases and construct new instances that the neural network may have missed using the information of previous frames. To this end, we create time series of the instances and shift the pixel-wise mask of a previous frame to the predicted current position of the new instance via linear regression. By this detection method, the number of instances can be greatly increased and thus, we deploy meta classification (see Sec. 4.2.3) to improve also the precision rate. Meta classification addresses the task of predicting if a predicted instance intersects with ground truth or not. To quantify the degree of overlap between prediction and ground truth, we consider the IoU. In object detection, meta classification refers to the task of classifying between $IoU < 0.5$ and $IoU \geq 0.5$ and in semantic segmentation between $IoU = 0$ and $IoU > 0$. Since instance segmentation is a combination of both, we classify between $IoU < \tau$ (false positive) and $IoU \geq \tau$ (true positive) for all predicted instances using different thresholds τ between 0 and 0.5. We use meta classification as false positive pruning after the application of our detection algorithm to improve the overall network performance in comparison to score thresholding during inference. As input for the meta classification model, instance-wise metrics are constructed. These metrics characterize uncertainty and features of a given instance like instance size, geometric center and occlusion level. In addition, we apply a depth estimation network which can infer in parallel to the instance segmentation network. Based on the resulting pixel-wise heatmap (see Fig. 5.1), we construct further metrics aggregated per instance. We complete our set of metrics with measures presented in Chapter 4 that are based on expected position, changes in the shape and survival time analysis of instances in an image sequence.

Figure 5.1: *Top left*: Ground truth image with ignored regions (white). The bounding boxes drawn around the instances represent the class, red denotes pedestrians and blue cars. The cyan colored bounding boxes highlight non-detected instances. *Top right*: Instance segmentation. *Bottom left*: A visualization of the calculated instance-wise IoU of prediction and ground truth. Green color corresponds to high IoU values. *Bottom right*: Depth estimation map.

In this chapter, we present a post-processing method for performance improvement and in particular, for false negative reduction based on uncertainty estimates. We only assume that image sequences of input data and a trained instance segmentation network are available. In our tests, we employ two instance segmentation networks, YOLACT and Mask R-CNN (see Sec. 2.1.4), and deploy these networks to the KITTI and the MOT dataset for multi-object tracking and instance segmentation (see Sec. 2.5). The source code of our method is publicly available at `http://github.com/kmaag/Temporal-False-Negative-Reduction`. Our contributions are summarized as follows:

- We introduce a light-weight detection algorithm applied to tracked (by Algorithm 4.1) predictions of an instance segmentation network to detect possible missed instances of this network. Furthermore, we construct efficient time-dynamic metrics for meta classification.

- We study the properties of the metrics and the influence of different lengths of time series which are used as input for the meta classification model. We perform meta classification to detect false positive instances achieving $AUROC$ values of up to 99.30%.

- For the first time, we demonstrate successfully that a post-processing false negative detection method can be traded for instance segmentation performance. We compare our approach with the application of a score threshold during inference. Our detection algorithm fused with uncertainty based meta classification achieves $AUPRC$ values of up to 98.07%.

The chapter is structured as follows. The related work on false negative reduction methods is discussed in Sec. 5.1. In Sec. 5.2, we introduce our method including the false negative detection algorithm in Sec. 5.2.1, the construction of instance-wise metrics in Sec. 5.2.2 and meta classification in Sec. 5.2.3. In comparison to Sec. 4.2.3, we refine the meta classification task using different thresholds and computation rules. The numerical results are presented in Sec. 5.3. We study the properties of the metrics and the influence of different lengths of the time series which serve as input to the meta classifier. Furthermore, we evaluate to which extent our detection algorithm fused with uncertainty based meta classification can improve an instance segmentation performance and reduce false negative instances. We conclude in Sec. 5.4.

5.1 Related Work

In this section, we present the related work on methods for false negative reduction, i.e., recall rate increase. For semantic segmentation, a method to achieve a higher recall rate is proposed in [171] based on the loss function, classifier and decision rule for a real-time neural network. The similar approach in [172] uses an importance-aware loss function to improve the networks' reliability. In [23], the differences between the maximum likelihood and the Bayes decision rule are considered to reduce false negatives of minority classes by introducing class priors which assign larger weight to underrepresented classes. The following methods address false negative reduction for the object detection task. In [174], a boosting chain for learning successive boosting is presented. Using previous information during cascade training, the model is adjusted to a very high recall rate in each layer of the boosting cascade. An ensemble based method is proposed in [21] where the number of false negatives is reduced by the different predictions of the networks, i.e., some objects that are not detected by one network could be detected by another one. In addition to false negative reduction, the number of false positives is also decreased in [52] by training a neural network with differently labeled images composed of correct and incompletely labeled images. In [164], a set of hypotheses of object locations and figure-ground masks are generated to achieve a high recall rate and thereafter, false positive pruning is applied to obtain a high precision rate as well. For 3D object detection, a single-stage fully-convolutional neural network is introduced in [175]. Instead of using a proposal generation step, the model outputs pixel-wise predictions where each prediction corresponds to a 3D object estimate resulting in a recall rate of 100%. In one work [92] applied to instance segmentation, a high recall rate is ensured by generating 800 object proposals for any given image. This is followed by the application of the maximum a-posteriori probability principle to produce the final set of instances. In [188], a recurrent deep network using pose estimation is considered

to improve instance segmentation in an iterative manner and to obtain a high recall rate. Another approach [94] constructs a variational autoencoder on top of the Mask R-CNN to generate high-quality instance masks and track multiple instances in image sequences. To reduce false negative instance predictions, spatial and motion information shared by all instances is captured.

In comparison to most of the described false negative detection approaches, our method neither modifies the training process nor the network architecture, instead our detection algorithm serves as post-processing step applicable to any instance segmentation network. In Chapter 4, we have compared the ordinary score thresholding during inference of an instance segmentation network with meta classification as a post-processing step. Instead of using a score threshold, meta classification, i.e., false positive detection, is used to improve the trade-off between the number of false positive and false negative instances. However, the number of detected false negative instances is limited to those predicted by the network without using a score threshold, since no other instances are generated. In this chapter, we go beyond this method and present a detection algorithm that generates instances additionally to the predicted ones and thus, instances that have been initially missed by the network can be detected.

5.2 Method

In instance segmentation (as extension of object detection), instances represented as pixel-wise masks are predicted with corresponding class affiliations. During inference, a score threshold is used to remove instances with low scores followed by a non-maximum suppression to avoid multiple predictions for the same instance. On these remaining instances, we apply our tracking Algorithm 4.1 to obtain time series. Our false negative detection method is based on temporal information of tracked instances. We detect inconsistencies in the time series and construct new instances in these cases. As a result, the number of instances can be greatly increased and thus, we deploy meta classification (false positive detection) to filter them out. To this end, we present meta classification, which uses time-dynamic metrics aggregated per predicted instance as input. Our method is intended to improve the networks' performance in terms of accuracy. An overview of the method is shown in Fig. 5.2.

5.2.1 Detection Algorithm

In this section, we introduce our detection method to identify possible non-detected instances in image sequences. We assume that an instance segmentation

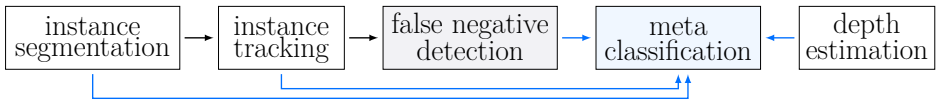


Figure 5.2: Overview of our approach which consists of false negative detection and false positive pruning applied to tracked instances. For metrics construction, information is extracted from instance geometry, instance tracking and depth estimation (*blue arrows*). The resulting metrics serve as input for the meta classification.

is available for each frame, as our method serves as a post-processing step and is independent of the choice of instance segmentation network. An example of an instance segmentation with corresponding ground truth image is shown in Fig. 5.1. The white areas within the ground truth image are ignored regions $\mathcal{R}$ with unlabeled instances, here cars and pedestrians. All predicted instances where 80% of their pixels are inside an ignored region are not considered for detection and further experiments as an evaluation for these instances is not feasible. This overlap $O_{i,j}$ of an instance i with an instance or region j is defined in (3.2). Given an image x, the set of predicted instances not covered by ignored regions is denoted by $\hat{\mathcal{I}}_x$. In addition to instance segmentation, we also assume a tracking of these instances in the image sequences. To this end, we use our tracking algorithm for instances (Algorithm 4.1). Instances are matched according to their overlap $\tilde{O}_{i,j}$ (4.1) in consecutive frames by shifting instances based on their expected location in the subsequent frame. The tracking algorithm is applied sequentially to each frame $\{x_t : t = 1, \ldots, T\}$ of an image sequence of length T. A detailed description of how an instance $i \in \hat{\mathcal{I}}_{x_{t-1}}$ in frame $t-1$ is matched with an instance $j \in \hat{\mathcal{I}}_{x_t}$ in frame t, is given in Sec. 4.2.1.

Our detection method is based on inconsistencies in the time series of the tracked instances such as a gap in the time series or a sudden end. We detect these cases and construct new instances that may have been overlooked by the neural network using the information of previous frames. To this end, time series of the geometric centers (4.2) of the instances are created and the pixel-wise mask of an instance of a previous frame is shifted to the predicted position of the new instance using a linear regression. If $t^{\text{last}} < t$ is the last frame in which instance i occurs, then we adopt this pixel-wise mask as the representation for the new instance in frame t. To avoid false positives, we track a lost instance for at most 10 frames and check if the instance is mostly covered by another one or an ignored region. A detailed description of our detection method is depicted in Algorithm 5.1.
An example how our detection algorithm works is shown in Fig. 5.3 applied to the MOT dataset and predictions of the Mask R-CNN. This network does not predict instances in frames $t+2$ and $t+4$ resulting in non-detected instances. Our algorithm shifts the instance of the previous frame (here $t+1$ and $t+3$) into the following one via linear regression as there is a gap in the time series.

Algorithm 5.1: False negative detection algorithm

$\hat{\mathcal{I}}_{x_t}^{\text{detect}} := \{\}\ \forall\ t = 1, \ldots, T$
/* detect instances */
for $i \in \bigcup_{t=1}^{T} \hat{\mathcal{I}}_{x_t}$ **do**
    $G := \{\}$ /* time series of geometric centers */
    $t^{\text{last}} := 0$ /* last previous frame in which instance i occurs */
    for $t = 1, \ldots, T$ **do**
        if *i exists in frame t* **then**
            $G := G \cup \{\bar{i}_t\}$
            $t^{\text{last}} := t$
        end
        if *i does not exist in frame t and $t - t^{last} \leq 10$ and $|G| \geq 2$* **then**
            perform linear regression to predict the geometric center $(\hat{\bar{i}}_t)$ using the geometric centers of the previous frames G
            shift instance $i \in \hat{\mathcal{I}}_{x_{t^{\text{last}}}}$ by the vector $(\hat{\bar{i}}_t - \bar{i}_{t^{\text{last}}})$ and denote the resulting instance by i^{shift}
            $\hat{\mathcal{I}}_{x_t}^{\text{detect}} := \hat{\mathcal{I}}_{x_t}^{\text{detect}} \cup \{i^{\text{shift}}\}$
        end
    end
end
/* check covering */
for $t = 1, \ldots, T$ **do**
    for $j \in \hat{\mathcal{I}}_{x_t}^{detect}$ **do**
        if $O_{i,\mathcal{R}} < 0.8$ *and* $\max_{k \in \hat{\mathcal{I}}_{x_t}} \tilde{O}_{j,k} \leq 0.95$ **then**
            $\hat{\mathcal{I}}_{x_t} := \hat{\mathcal{I}}_{x_t} \cup \{j\}$
        end
    end
end

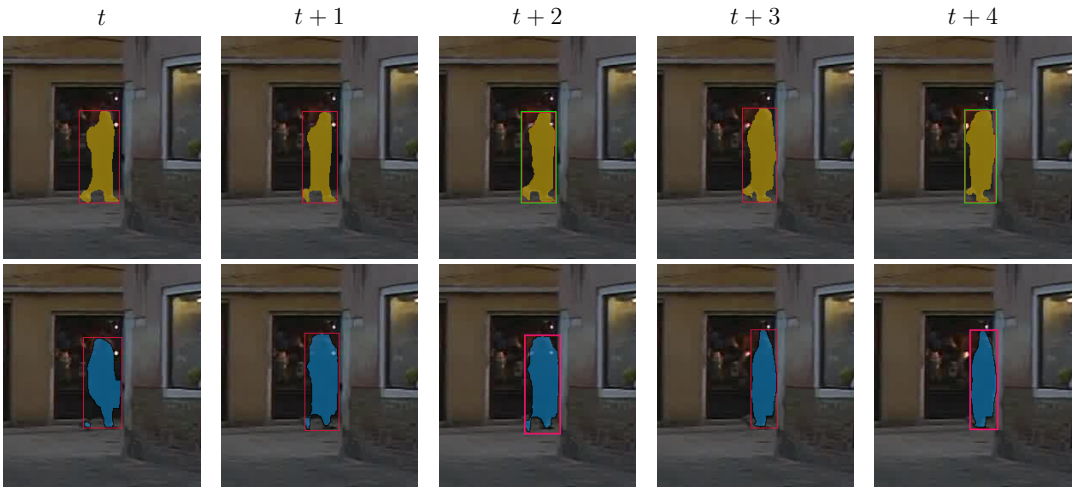


Figure 5.3: Example of our false negative detection algorithm applied to five consecutive images of the MOT dataset. *Top*: Ground truth images with non-detected instances by the neural network (bounded by a green box). *Bottom*: Instance segmentation obtained by the Mask R-CNN. In frames $t+2$ and $t+4$ no instances are predicted, both instances are constructed by our detection algorithm.

5.2.2 Metrics

In instance segmentation, the neural network provides information of the instances like geometric characteristics or their attached class confidences derived from the softmax probabilities. For example, a probability distribution over the classes $y \in \mathcal{C}$ for each pixel z or only for each instance is provided depending on the network architecture. However, a probability distribution is not available for the detected instances and thus, we construct metrics based on information that is also obtainable for the detected ones. To this end, we use selected metrics introduced in Sec. 4.2.2 and extend this set of metrics by new ones.

First, we add measures for an instance i introduced in Sec. 4.2.2 to our set of metrics like the *geometric center* $\bar{i}$ (4.2), the predicted *class* $\hat{c}$ and the *score* value s which is provided by the instance segmentation network. We calculate the score value for the detected instances as the average score value from the previous frames of the respective instance. Furthermore, we use the *size* S of a whole instance as well as of the *inner* S_{in} and *boundary* S_{bd} and the *relative instance sizes* $\tilde{S}$, $\tilde{S}_{in}$. The separate treatment of inner and boundary is motivated by poor or false predictions that often results in fractal instance shapes, i.e., a relatively large amount of boundary pixels, measurable by the relative measures.

Moreover, we define the *occlusion* measure o using instances in two consecutive frames. Instance i of frame $t-1$ is shifted such that instance i and its matched counterpart in frame t have a common geometric center. Then, the occlusion of

instance i in frame t is given by $O_{i,K}$ where $K = \bigcup_{k \in \hat{\mathcal{I}}_{x_t} \setminus \{i\}} k$. Large occlusions may indicate poorly predicted instances.

Besides an instance segmentation network, we consider a monocular depth estimation network, which uses the same input images as the instance segmentation network and can run in parallel. The motivation in using the extra depth estimation network is to acquire additional sources of useful information about the instances. Given an image x, the pixel-wise depth prediction is denoted by $H_z(x)$, see Fig. 5.1 for an example. To obtain metrics per instance, we define the *mean depth* as

$$\bar{H}_* = \frac{1}{S_*} \sum_{z \in i_*} H_z(x), \quad * \in \{\_, in, bd\} \tag{5.1}$$

as an additional metric, we also divide the inner and boundary. The *relative mean depth* measures are calculated as $\tilde{\bar{H}} = \bar{H}\tilde{S}$ and $\tilde{\bar{H}}_{in} = \bar{H}_{in}\tilde{S}_{in}$. A lower value of mean depth indicates reliable instances, while a higher value suggests uncertainty.

Next, for each instance i in frame t, a time series of the mean depth $\bar{H}$ of the 5 previous frames is constructed. Using linear regression, the mean depth $\hat{\bar{H}}$ in frame t is predicted if the instance exists in at least two previous frames. The *deviation* between expected mean depth and mean depth $|\hat{\bar{H}} - \bar{H}|$ is used as a temporal measure denoted by d_h. Small deviations d_h indicate consistency over time of the instance.

Finally, we consider five further metrics explained in Sec. 4.2.2. Analogous to the depth deviation d_h, we calculate the deviation of the instance size d_s and of the geometric center d_c. For the survival analysis metric v, time series of metrics are also considered. More precisely, time series of the previously presented single-frame metrics ($\bar{i}, \hat{c}, s$, size and depth metrics) serve as input for a Cox regression to predict the survival time of an instance i in frame t. A lower value of v indicates uncertainty. The next metric r is based on the height to width ratio of the instances. Deviations from this ratio for an instance i suggest false predictions. The final measure f describes the variation of an instance in two consecutive frames by calculating the overlap (4.1). Poorly predicted instances can result in large deformations.

In summary, we use the following set of metrics

$$\begin{aligned} U^i = &\{\bar{H}, \bar{H}_{in}, \bar{H}_{bd}, \tilde{\bar{H}}, \tilde{\bar{H}}_{in}\} \cup \{o, d_h\} \cup \{S, S_{in}, S_{bd}, \tilde{S}, \tilde{S}_{in}\} \\ &\cup \{\bar{i}, \hat{c}, s, d_s, d_c, v, r, f\}\,. \end{aligned} \tag{5.2}$$

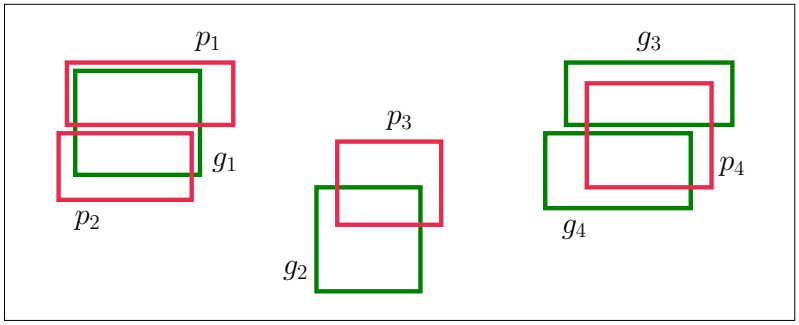


Figure 5.4: Example image for object detection. Green boxes correspond to ground truth objects g_k, $k = 1, \ldots, 4$, and red boxes to predicted objects p_k, $k = 1, \ldots, 4$.

Table 5.1: Results of matched ground truth und predicted objects of Fig. 5.4 for the object detection as well as semantic segmentation perspective. The predicted object/ ground truth is given with matched counterpart, the IoU value and decision of true positiv or false positive/negative.

		object detection			semantic segmentation		
prediction	score	gt	IoU	tp/fp	gt	IoU	tp/fp
p_1	0.95	g_1	≥ 0.5	tp	g_1	> 0	tp
p_2	0.83	g_1	≥ 0.5	fp	g_1	> 0	tp
p_3	0.76	–	< 0.5	fp	g_2	> 0	tp
p_4	0.51	g_4	≥ 0.5	tp	g_4 & g_3	> 0	tp
ground truth		prediction	IoU	tp/fn	prediction	IoU	tp/fn
g_1		p_1	≥ 0.5	tp	p_1 & p_2	> 0	tp
g_2		–	< 0.5	fn	p_3	> 0	tp
g_3		p_4	≥ 0.5	fn	p_4	> 0	tp
g_4		p_4	≥ 0.5	tp	p_4	> 0	tp

5.2.3 Meta Classification

Meta classification is used to identify false positive instances provided by a neural network. A predicted instance is considered as a false positive, if the intersection over union is less than a threshold τ. The IoU is a commonly used performance measure that quantifies the degree of overlap of prediction and ground truth. Meta classification refers to the task of classifying between $IoU < \tau$ and $IoU \geq \tau$ for all predicted instances. Note, if an IoU threshold of $\tau = 0$ is considered, we classify between $IoU = 0$ and $IoU > 0$. In object detection, classification is typically between $IoU < 0.5$ and $IoU \geq 0.5$ and in semantic segmentation between $IoU = 0$ and $IoU > 0$. Besides the IoU threshold, there is another difference in the calculation procedure between object detection and semantic segmentation. In Fig. 5.4 an example of ground truth and predicted objects is shown. The corresponding matches are given in Table 5.1. First, predictions p_1 and p_2 have an $IoU \geq 0.5$ with ground truth object g_1 resulting in a true positive and a false

positive for object detection as a ground truth object can be matched only once (in comparison to semantic segmentation). Moreover, prediction p_3 and ground truth g_2 are matched only for semantic segmentation due to the IoU threshold. In semantic segmentation, the IoU is calculated with all ground truth instances that an object is covering, thus object p_4 is matched with ground truth objects g_3 and g_4. For this reason, g_3 is not considered to be a false negative as it would be the case in object detection (IoU of p_4 with g_4 is higher than with g_3). In summary, we obtain less false positives and false negatives in semantic segmentation. In our experiments, we examine both calculation options for the network' errors and additionally threshold on different IoU values. We denote by *MOD* the one-to-one matching of predicted and ground truth object (object detection perspective) and by *MSS* the many-to-many matching (semantic segmentation perspective). Note, in Chapter 4, we have performed the evaluations according to the object detection principle with an IoU threshold of 0.5.

We perform meta classification using the metrics introduced in Sec. 5.2.2 as input for the classifier, in particular, we apply time series of these metrics. For an instance $i \in \hat{\mathcal{I}}_{x_t}$ in frame t, the metrics U_t^i are obtained as well as $U_{t'}^i$ from previous frames $t' < t$ due to the tracking of the instances. Meta classification is conducted by means of these time series of metrics U_k^i, $k = t - n_c, \ldots, t$ where n_c describes the number of considered frames. As classifier model, we use gradient boosting that performs best in comparison to linear models and shallow neural networks as shown in Sec. 3.3 and Sec. 4.3.2. We study the benefit from using time series and to which extent meta classification along with our false negative detection method can improve the overall network performance compared to the application of a score threshold during inference.

5.3 Numerical Results

In this section, we study the properties of the metrics introduced in the previous section and the influence of different lengths of the time series which serve as input to the meta classifier. Furthermore, we evaluate to which extent our detection algorithm fused with uncertainty based meta classification can improve an instance segmentation performance and reduce false negative instances. To this end, we compare this approach with ordinary score thresholds in terms of numbers of false negatives and false positives, i.e., recall and precision rates.

We perform our tests on two datasets for multi-object tracking and instance segmentation, which we have introduced in Sec. 2.5. The KITTI dataset contains 21 street scene image sequences from Karlsruhe consisting of 8,008 1,242 $\times$ 375 images. The MOT dataset containing 2,862 images with resolutions of 1,920 $\times$ 1,080

(3 image sequences) and 640×480 (1 image sequence) provides scenes from pedestrian areas and shopping malls. For both datasets annotated image sequences, i.e., tracking IDs and instance segmentations, are available [162]. The KITTI dataset includes labels for the classes car and pedestrian, while the MOT dataset only contains labels for the class pedestrian.

In our experiments, we consider the Mask R-CNN and the YOLACT network (see Sec. 2.1.4). The YOLACT network has a slim underlying architecture and is designed for a single GPU. For training this network, we use a ResNet-50 [61] as backbone and pre-trained weights for ImageNet [140]. We choose 12 image sequences from the KITTI dataset consisting of 5,027 images and 300 images (extracted from 2 sequences) from the MOT dataset for training. The remaining 9 sequences of the KITTI dataset serve as validation set (same splitting as in [162] and Sec. 4.3). We achieve a mean average precision of 57.15%. Since we are using 300 images of the MOT dataset for training, we split 2 of the 4 sequences into train/validation and remove 90 frames (equal to three seconds) of each at this splitting point due to redundancy in image sequences. We achieve a mAP of 52.37% on the remaining 2,382 images of the MOT dataset. The Mask R-CNN is focused on high-quality instance-wise segmentation masks representations. For training, we use a ResNet-101 as backbone and start from weights for COCO [95]. As training set, we choose the same 12 image sequences of the KITTI dataset and the validation sets also remain the same. For the KITTI dataset, we obtain a mAP of 89.79% and for the MOT dataset of 78.99%. During training of both networks, the validation set is neither used for early stopping nor for parameter tuning.

For our depth metrics extracted from a depth estimation, we use the network introduced in [90]. This network utilizes local planar guidance layers at multiple stages of the decoding phase for an effective guidance of densely encoded features (see Sec. 2.1.5 for details). As backbone, we consider the DenseNet-161 [69] and pre-trained weights for ImageNet. We train this network over a maximum of 50 epochs on 20,750 RGB images of the KITTI dataset where annotated depth maps are available. The model which achieves the lowest scale invariant logarithmic error (see (2.56)) on 672 validation images is used. This validation silog amounts to 8.493. We also evaluate on the 9 image sequences of the KITTI dataset, that we consider for further experiments, achieving a test silog of 9.459 on these 2,891 images (for 90 images no depth ground truth is available).

During inference, a score threshold is used to remove instances with low score values followed by a non-maximum suppression to avoid multiple predictions for the same instance. Since we compare our detection method with the application of different score values, we apply none or a very low score threshold during the inference. For the KITTI dataset, we choose a score threshold of 0 in both networks to use all predicted instances for further experiments. For the MOT

Table 5.2: A selection of correlation coefficients ρ with respect to IoU. For size and depth measures, the metric with the highest correlation is given.

KITTI	$\tilde{S}_{in}$	0.42316	$\bar{i}_v$	0.08567	o	−0.04165	d_s	0.11309
&	$\tilde{\bar{H}}_{in}$	0.56636	$\hat{c}$	−0.23682	f	0.09413	d_c	−0.04318
Mask R-CNN	s	0.64899	r	−0.08192	v	−0.27352	d_h	−0.19601
KITTI	$\tilde{S}_{in}$	0.28690	$\bar{i}_v$	0.02200	o	0.29603	d_s	0.08056
&	$\tilde{\bar{H}}_{in}$	0.38115	$\hat{c}$	−0.31568	f	0.07016	d_c	−0.07427
YOLACT	s	0.79357	r	−0.17465	v	−0.14472	d_h	−0.16114
MOT	$\tilde{S}_{in}$	0.51436	$\bar{i}_v$	0.01569	o	−0.15981	d_s	0.09812
&	$\tilde{\bar{H}}_{in}$	0.59056	$\bar{i}_h$	−0.04090	f	0.05827	d_c	0.03598
Mask R-CNN	s	0.66106	r	0.00896	v	−0.25305	d_h	−0.04048
MOT	$\tilde{S}_{in}$	0.69746	$\bar{i}_v$	−0.01964	o	−0.21768	d_s	0.13392
&	$\tilde{\bar{H}}_{in}$	0.68968	$\bar{i}_h$	−0.12691	f	0.05930	d_c	−0.01485
YOLACT	s	0.67695	r	−0.22188	v	−0.03406	d_h	−0.13630

dataset, we choose relatively low score thresholds, i.e., 0.3 for the Mask R-CNN and 0.05 for the YOLACT network. In our experiments, we use 9 image sequences of the KITTI dataset consisting of 2,981 images and 4 sequences of the MOT dataset containing 2,382 images (equal to the validation sets used for instance segmentation).

5.3.1 Meta Classification

First, we study the predictive power of the instance-wise metrics, which serve as input for the meta classifier, by computing the Pearson correlation coefficients ρ between selected metrics of U^i and the IoU, see Table 5.2. For both networks and both datasets, the score value s shows a strong correlation with the IoU. The relative instance size $\tilde{S}_{in}$ as well as the relative mean depth $\tilde{\bar{D}}_{in}$ demonstrate high correlations for the MOT dataset (for both networks) and the KITTI dataset in combination with the Mask R-CNN. Selected scatter plots for the MOT dataset and the YOLACT network are shown in Fig. 5.5. For meta classification (false positive detection: $IoU < \tau$ vs. $IoU \geq \tau$), we use the set of metrics U^i as input and investigate the influence of time series. We present this set of metrics U^i_t of a single frame t to the meta classifier and then, the metrics are extended to time series with a length of up to 10 previous time steps U^i_k, $k = t-10, \ldots, t-1$. Furthermore, we use a (meta) train/validation/test splitting of 70%/10%/20% and average the results over 10 runs for the KITTI dataset and 4 runs for the MOT dataset by randomly sampling this splitting. For the presented results, on the one hand, we choose the object detection perspective (MOD) with an IoU

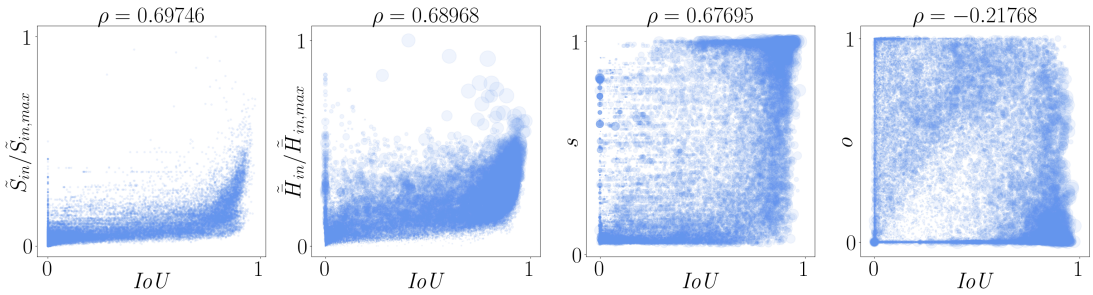


Figure 5.5: Correlations between metrics and IoU for the MOT dataset and the YOLACT network. The dot size is proportional to the instance size.

Table 5.3: Number of instances predicted by a neural network (PI), number of detected instances (DI) using Algorithm 5.1, total number of instances (not yet matched over time), instances with $IoU < 0.5$ and $IoU = 0$.

	PI	DI	PI+DI	$IoU < 0.5$	$IoU = 0$
KITTI & Mask R-CNN	19,239	11,787	31,026	20,337	13,036
KITTI & YOLACT	25,743	14,694	40,437	32,114	24,961
MOT & Mask R-CNN	36,003	27,374	63,377	45,307	20,146
MOT & YOLACT	22,495	16,400	38,895	26,153	3,736

threshold of $\tau = 0.5$ and on the other hand, the semantic segmentation perspective (MSS) classifying between $IoU = 0$ and $IoU > 0$.

We apply our detection algorithm to the predictions of both instance segmentation networks for both datasets. The corresponding numbers of predicted instances and instances obtained by the detection algorithm as well as the number of instances with an $IoU < 0.5$ and $IoU = 0$ are provided in Table 5.3. The relatively high number of false positive instances is the motivation to perform meta classification after the detection algorithm in order to get rid of false positives. As performance measures for the meta classification, we consider the classification accuracy and $AUROC$. The best results are given in Table 5.4. We achieve $AUROC$ values up to 99.30%. Higher values are obtained by classifying between $IoU < 0.5$ and $IoU \geq 0.5$. Furthermore, we notice that gradient boosting benefits from temporal information which can also be observed in Fig. 5.6 where the $AUROC$ results as functions of the number of frames are given for both dataset and the object detection calculation approach. For further experiments, we use the number of considered frames where the best $AUROC$ performance is reached, respectively.

Table 5.4: Meta classification results (with corresponding standard deviations) using IoU thresholds $\tau = 0.5$ (MOD) and $\tau = 0$ (MSS). The super script denotes the number of considered frames where the best performance and in particular, the given values are reached.

MOD	*ACC*	*AUROC*
KITTI & Mask R-CNN	$95.64\% \pm 0.74\%^{7}$	$99.04\% \pm 0.31\%^{5}$
KITTI & YOLACT	$96.22\% \pm 1.15\%^{7}$	$99.30\% \pm 0.34\%^{7}$
MOT & Mask R-CNN	$93.60\% \pm 3.42\%^{9}$	$98.25\% \pm 0.93\%^{9}$
MOT & YOLACT	$87.66\% \pm 5.45\%^{9}$	$96.22\% \pm 1.22\%^{5}$

MSS	*ACC*	*AUROC*
KITTI & Mask R-CNN	$86.57\% \pm 3.64\%^{4}$	$95.38\% \pm 1.57\%^{3}$
KITTI & YOLACT	$90.75\% \pm 4.41\%^{3}$	$96.20\% \pm 1.86\%^{4}$
MOT & Mask R-CNN	$73.55\% \pm 15.07\%^{7}$	$88.74\% \pm 3.33\%^{7}$
MOT & YOLACT	$93.54\% \pm 0.46\%^{10}$	$90.68\% \pm 3.07\%^{2}$

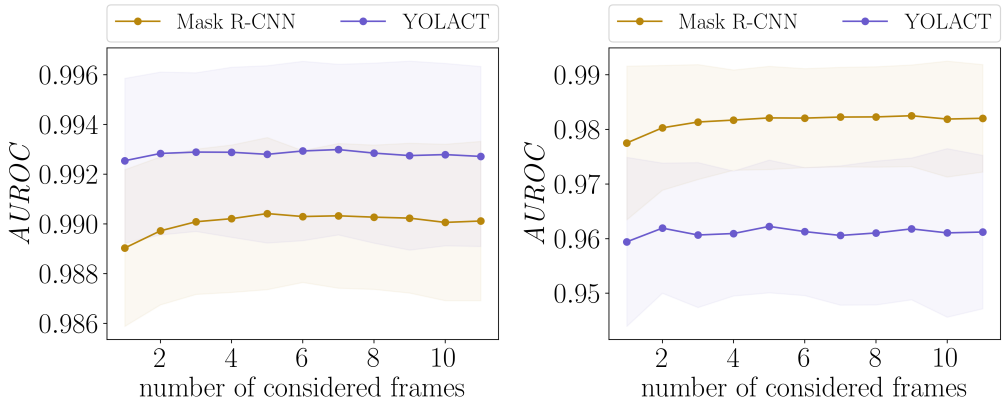


Figure 5.6: *Left*: Results for meta classification $AUROC$ as functions of the number of frames for the KITTI dataset and MOD. *Right*: Results for the MOT dataset.

5.3.2 Evaluation of the Detection Method

Our detection method assumes that the instance segmentation by a neural network and tracking IDs are given. We consider our tracking algorithm for instances introduced in Sec. 4.2.1 where the tracking performance is measured by different object tracking metrics (see Sec. 4.3.1). In the following, we compute object tracking metrics (introduced in Sec. 2.3.2) and apply these to the instances predicted by a network and the additional instances obtained by our detection method. We denote by GT all ground truth instances of an image sequence which are identified by different tracking IDs and divide these into three cases. An instance is called mostly tracked (MT) if it is tracked for at least 80% of frames (out of the total number of frames in which it occurs), mostly lost (ML) if it is tracked for less

Table 5.5: Object tracking results for the instances obtained by our false negative detection method (including those predicted by an instance segmentation network). The values in brackets correspond to the outcomes for the predicted instances without using the detection step.

	KITTI				MOT			
	Mask R-CNN		YOLACT		Mask R-CNN		YOLACT	
GT	219		219		201		201	
MT	205	(204)	129	(124)	123	(120)	45	(42)
PT	12	(13)	71	(75)	72	(75)	92	(92)
ML	2	(2)	19	(20)	6	(6)	64	(67)
smn	186	(189)	352	(379)	797	(824)	552	(597)

than 20%, otherwise partially tracked (PT). In addition, we compute the number of switches between matched and non-matched *smn*, i.e., a switch appears when a ground truth instance is matched with a predicted instance in frame t and non-matched in frame $t+1$ and vice versa. The results are shown in Table 5.5. We observe that our algorithm can increase the number of mostly tracked ground truth instances and reduce the number of switches between matched and non-matched for both networks and datasets. Note, the Mask R-CNN mainly achieves better results than YOLACT which can be observed by Mask R-CNN achieving higher mAP values than YOLACT.

The score value describes the confidence of the network's prediction and is chosen during inference to balance the number of false negatives and false positives. We select 15 different score thresholds between 0 and 1 to study the network's detection performance while varying this threshold. We apply our false negative detection algorithm to the predicted instances and fuse this with uncertainty based meta classification. Meta classification provides a probability of observing a false positive instance and thus, we also threshold on this probability with 15 different values. In our tests, we consider 6 different IoU thresholds τ for meta classification and both calculation options. We feed the meta classifier with all metrics U^i including the number of previous frames given in Table 5.4 independent of the IoU threshold. We compare ordinary score thresholding with our method using the $AUPRC$ as performance measure (see Sec. 2.1.6). As our detection algorithm receives time series of predicted instances as input, certain ground truth instances cannot be found. For this reason, we exclude ground truth instances for further testing if the respective instance defined by its tracking ID is never found by the instance segmentation network and in the frames before the instance is first detected by the network. On the one hand, our method cannot be applied to these instances, and on the other hand, the network shall also be able to predict them, i.e., a ground truth instance defined by its tracking ID should have been detected at least once. The number of ground truth instances depends on the respective IoU threshold, i.e., at higher values, more ground truth

Table 5.6: *AUPRC* results for score thresholding vs. our false negative detection algorithm fused with meta classification thresholding for *MOD* and *MSS* calculation procedure.

KITTI	Mask R-CNN		YOLACT		Mask R-CNN		YOLACT	
τ	score	ours	score	ours	score	ours	score	ours
0.5	91.17%	**92.35%**	78.58%	**81.43%**	91.39%	**92.67%**	79.57%	**82.60%**
0.4	93.19%	**93.81%**	81.03%	**84.17%**	93.53%	**94.28%**	82.64%	**85.95%**
0.3	93.98%	**94.45%**	82.28%	**85.83%**	94.50%	**95.23%**	84.53%	**88.19%**
0.2	94.49%	**94.88%**	81.83%	**85.61%**	95.34%	**96.08%**	85.19%	**89.17%**
0.1	94.67%	**95.08%**	82.19%	**86.06%**	95.84%	**96.74%**	86.57%	**90.29%**
0.0	94.99%	**95.39%**	83.87%	**87.78%**	97.51%	**98.07%**	91.80%	**93.03%**
	MOD				*MSS*			
MOT	Mask R-CNN		YOLACT		Mask R-CNN		YOLACT	
τ	score	ours	score	ours	score	ours	score	ours
0.5	77.45%	**79.05%**	62.32%	**63.73%**	77.56%	**78.50%**	63.72%	**64.29%**
0.4	82.44%	**82.76%**	68.39%	**69.71%**	**82.79%**	82.62%	**71.03%**	71.02%
0.3	**85.17%**	85.02%	71.29%	**72.77%**	**86.17%**	85.26%	**75.26%**	75.11%
0.2	**86.94%**	86.85%	72.31%	**74.22%**	**88.67%**	87.39%	77.31%	**77.99%**
0.1	**88.56%**	88.43%	73.89%	**76.96%**	**91.36%**	89.49%	80.60%	**84.04%**
0.0	**89.85%**	89.48%	76.11%	**81.77%**	**97.27%**	94.40%	91.58%	**94.81%**

instances are not detected. For the KITTI dataset and the Mask R-CNN, at most 0.96% and for the YOLACT network, 13.23% of the ground truth instances are not found and for this reason, not considered for further evaluations. For the MOT dataset and the Mask R-CNN, this holds for at most 2.19% and for the YOLACT network 17.62% of the ground truth instances. The *AUPRC* results are given in Table 5.6. Smaller *IoU* thresholds increase the possibility of matches between ground truth and predicted instances and consequently the *AUPRC* value increases. We observe that our method performs better in most cases compared to the use of a score threshold (or obtains similar values). Furthermore, the *AUPRC* value increases for the *MSS* calculation option as the possibility of matches is more likely. We obtain *AUPRC* values of up to 95.39% for the object detection perspective and up to 98.07% for the semantic segmentation one. An example for the precision-recall curves is shown in Fig. 5.7 using *IoU* thresholds of $\tau = 0.5$ (left) and $\tau = 0$ (right) for *MOD* and for the KITTI dataset in combination with the YOLACT network. Each point represents one of the chosen score or meta classification thresholds. Our detection method achieves a lower number of errors, i.e., higher recall and precision rates. In particular, we can reduce the number of false negative instances.

For further experiments, we bin the ground truth instances into different occlusion levels. To this end, we calculate for each ground truth instance the *IoU* with the other ones represented as bounding boxes denoted by IoU_{bb}. On the one hand, for high IoU_{bb} values, the instance can be partially covered by other

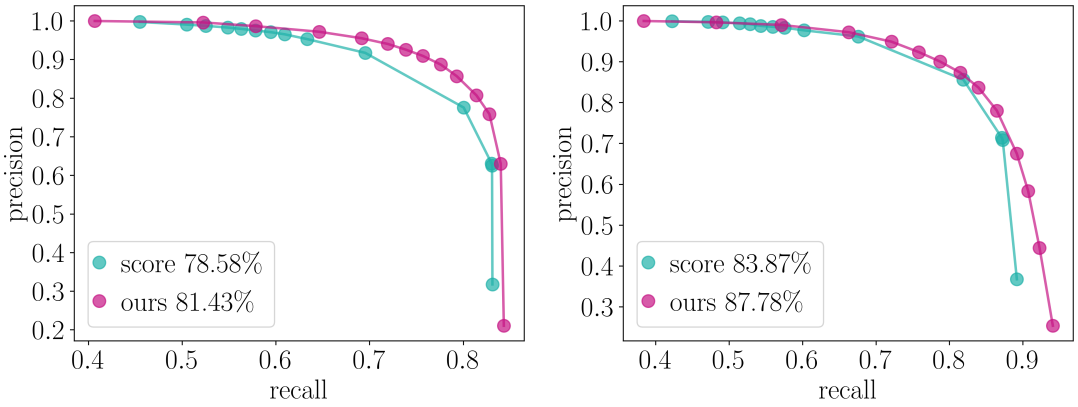


Figure 5.7: Precision-recall curves for different score as well as meta classification thresholds for the YOLACT network and the KITTI dataset. *Left*: An IoU threshold of $\tau = 0.5$ is used. *Right*: IoU threshold $\tau = 0$.

instances or even cover others. On the other hand, for low IoU_{bb} values or a value of zero, detecting the ground truth instance through an instance segmentation network is simpler as if the instance is occluded and located in a crowd of instances. In Table 5.7, the number of ground truth instances and $AUPRC$ results for the different occlusion levels are given for both datasets and the Mask R-CNN considering an IoU threshold of $\tau = 0.4$ and MOD. For the KITTI dataset, most of the ground truth instances have an occlusion level of $IoU_{bb} = 0.0$ and for the MOT dataset of $0.0 < IoU_{bb} \leq 0.1$ as pedestrian areas and shopping malls are provided. Our false negative detection approach outperforms ordinary score thresholding in terms of $AUPRC$ values. Moreover, our method can improve the instance segmentation also in more difficult cases, i.e., at higher occlusion levels. Our findings for the YOLACT network, the other IoU thresholds and MSS are analogous. In Fig. 5.8, the comparison of ordinary score thresholding and our detection method is shown in terms of the number of remaining false negatives and false positives considering an occlusion level of $0.5 < IoU_{bb} \leq 0.6$. As before, each point represents one chosen threshold. We achieve a lower number of false negative and false positive instances. For an IoU threshold of $\tau = 0$, this error reduction is also reflected in the $AUPRC$ values. The $AUPRC$ value obtained by score thresholding is 98.93% while the detection algorithm fused with meta classification achieves 99.86%. Note, in this case, we can detect up to 198 out of 202 ground truth instances. Remember, for lower IoU thresholds more ground truth instances can be found by the neural network and thus, more instances are considered for our evaluation.

In Fig. 5.9, an example of how our detection algorithm works is demonstrated for a ground truth instance of class car and the KITTI dataset. The time series of the instance size S and of the IoU between this ground truth instance and

Table 5.7: Number of ground truth instances and *AUPRC* results for different occlusion levels. For both datasets, the Mask R-CNN network, an IoU threshold of $\tau = 0.4$ and *MOD* are considered.

	KITTI			MOT		
occlusion level	# gt	score	ours	# gt	score	ours
$IoU_{bb} = 0.0$	5,207	97.51%	**98.29%**	5,154	94.66%	**96.28%**
$0.0 < IoU_{bb} \leq 0.1$	2,875	96.96%	**98.04%**	7,997	90.56%	**91.30%**
$0.1 < IoU_{bb} \leq 0.2$	1,848	97.56%	**98.67%**	4,301	94.25%	**95.67%**
$0.2 < IoU_{bb} \leq 0.3$	637	97.96%	**98.92%**	1,842	95.05%	**96.81%**
$0.3 < IoU_{bb} \leq 0.4$	320	98.05%	**99.09%**	976	95.91%	**97.82%**
$0.4 < IoU_{bb} \leq 0.5$	180	98.28%	**99.34%**	609	96.17%	**98.10%**
$0.5 < IoU_{bb} \leq 0.6$	62	98.46%	**99.52%**	194	96.82%	**98.75%**
$0.6 < IoU_{bb} \leq 0.7$	37	98.54%	**99.60%**	106	96.98%	**98.91%**
$0.7 < IoU_{bb} \leq 0.8$	13	98.69%	**99.70%**	90	96.98%	**98.90%**
$0.8 < IoU_{bb} \leq 0.9$	1	99.32%	**99.86%**	6	97.53%	**99.20%**

instances obtained by the YOLACT network are shown. We observe three areas where the instance has an $IoU = 0$ and hence, has not been detected by the network. Our detection algorithm is applied to the predicted instances, which also results in time series of IoU denoted by IoU_d. Since the procedure requires at least two geometric centers of an instance to generate further instances, the first plateau cannot be fixed, although the following two ones can be. Both images, ground truth (top) and instance segmentation (bottom), represent frame 190 where the instance (bounded by a green box) is not detected by the network, but by our method. We construct the car labeled with the pink bounding box in the segmentation image. Noteworthy, though no score threshold is used during inference, the network has not predicted any instance at this position. Also note, both images correspond to the zoomed image sections and the ground truth instance is relatively small which our algorithm can handle. Our false negative reduction method consisting of the detection algorithm and meta classification can be applied to any instance segmentation network after training and thus, does not increase the network complexity.

5.4 Discussion

In this chapter, we proposed a post-processing method applicable to any instance segmentation network to reduce the number of false negative instances by performing a false negative detection and a false positive pruning step. Our lightweight detection algorithm is based on inconsistencies in the time series of tracked instances such as a gap in the time series or a sudden end. We detected these cases and constructed new instances that the neural network may have missed

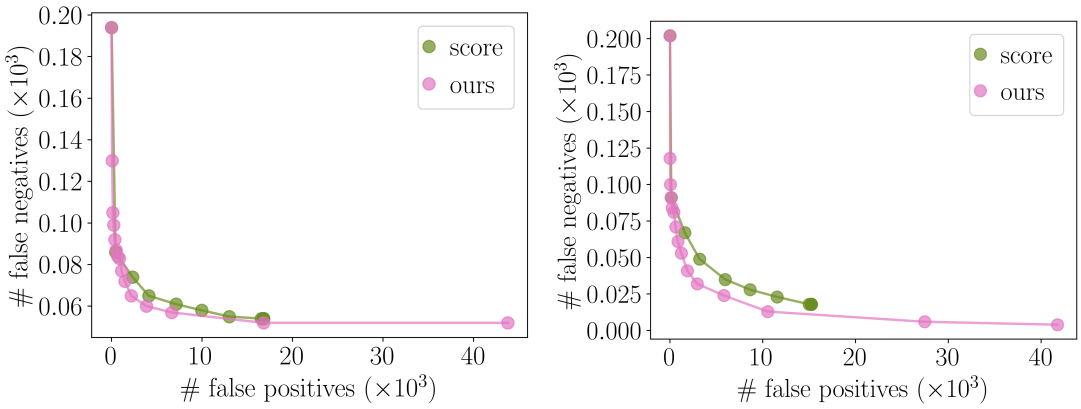


Figure 5.8: Number of false positive vs. false negative instances for different score and meta classification thresholds for the MOT dataset and the Mask R-CNN network. An occlusion level of $0.5 < IoU_{bb} \leq 0.6$ is used. *Left*: IoU threshold $\tau = 0.4$. *Right*: IoU threshold $\tau = 0.0$.

using the information of previous frames. Since the number of instances can be greatly increased, we employed meta classification to reduce false positive instances. As input for the meta classification model, instance-wise metrics were constructed characterizing uncertainty, geometry and depth of a given instance. We studied the properties of the metrics, the influence of different time series lengths on the meta classification model and various IoU thresholds. Furthermore, two different calculation methods for meta classification were considered, i.e., one-to-one matching of predicted and ground truth object (object detection perspective) and many-to-many matching (semantic segmentation perspective). We achieved $AUROC$ values of up to 99.30%. In our tests, we compared our approach to the application of a score threshold during inference and improved the networks' prediction accuracy. We obtained $AUPRC$ values of up to 98.07%. In particular, the number of false negative instances can be reduced. Moreover, we studied occlusion levels for the ground truth instances in order to improve the instance segmentation also in more difficult cases, i.e., we detected instances that were occluded and located in a crowd of instances.

We have demonstrated that our light-weight detection method can improve instance segmentation performance in terms of accuracy in comparison to ordinary score thresholds. Especially, we reduce the number of false negative instances, i.e., achieve higher recall rates. However, our method is limited to image sequences as our false negative detection algorithm is based on inconsistencies in the time series of tracked instances. We use the availability of image sequences in applications such as automated driving, for the detection of potentially missed objects by a neural network and in addition, to enhance the meta classification performance using instance-wise metrics characterizing temporal uncertainties as

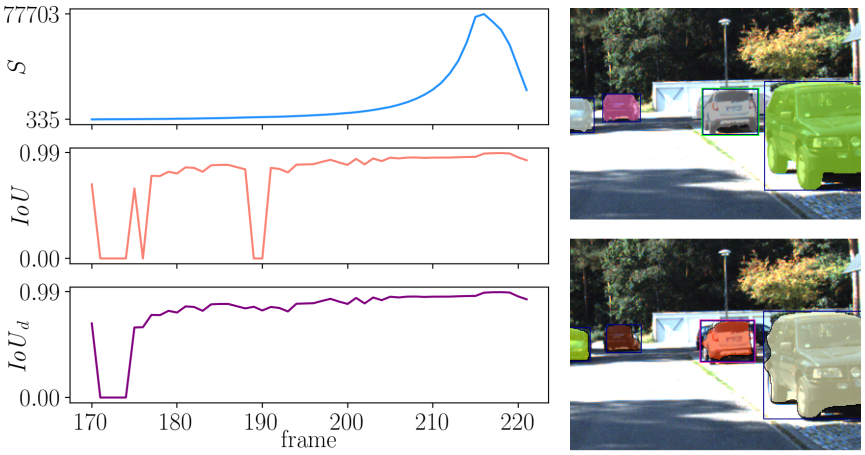


Figure 5.9: *Left*: Time series of size S for a ground truth instance of the KITTI dataset, the calculated IoU between this ground truth instance and instances predicted by the YOLACT network as well as the IoU_d after the application of our detection method. *Top right*: Corresponding ground truth image in frame 190 including the considered instance (bounded by a green box). *Bottom right*: Instance segmentation followed by our detection algorithm which constructs the car instance with the pink bounding box.

input. In applications, like medical diagnosis, image sequences are not necessarily obtainable and thus, our method is not applicable.

Chapter 6

Conclusions & Outlook

In this thesis, we presented methods for the prediction quality rating and performance improvements in terms of accuracy for semantic and instance segmentation networks using time-dynamic uncertainty estimates. In particular, we increased the recall rates, i.e., reduced the number of non-detected objects of the network. Our methods serve as post-processing steps applicable to any semantic or instance segmentation network and were tested on different datasets and networks demonstrating their efficiency and improved performance. Furthermore, we used the availability of image sequences to obtain more information of the neural networks without additional computational overhead for enhancing our methods.

For semantic segmentation networks, we proposed a time-dynamic variant of meta classification and meta regression. The first one tackles the task of false positive detection and the second one serves as prediction quality estimate for neural networks. To this end, we constructed single-frame metrics based on dispersion and geometry properties of segments extracted from the segmentation network's softmax output, such as pixel-wise entropy. To obtain time series of these metrics, we introduced a light-weight tracking algorithm for segments of the same semantic class. Note, this tracking approach is also independent of the semantic segmentation network. From matched segments over time, we generated time series of metrics which served as inputs for the meta classifiers/regressors. In our tests, we studied the influence of the time series length as well as different models for meta tasks like gradient boosting, neural networks and linear ones. We observed that gradient boosting performs best in comparison to the other methods and that our results benefit from time series. More precisely, in contrast to the single-frame approach [136] using only linear models, we increased the accuracy by 6.78 pp and the $AUROC$ by 5.04 pp for classification. For meta regression, the R^2 value was increased by 5.63 pp. Our time-dynamic method showed significant improvements in comparison to the single-frame approach achieving improved results in

the meta tasks. For meta regression, the direct prediction of the IoU_{adj} as quality estimate, we obtained R^2 values of up to 87.51% for the KITTI dataset. For meta classification, classifying between $IoU_{\text{adj}} = 0$ and $IoU_{\text{adj}} > 0$, we achieved $AUROC$ values of up to 88.68%.

In a next step, our intention was to develop further metrics, truly time-dynamic ones, to increase the rating measures since the metrics for segments are still single-frame based. Due to the properties of segments (connected components of a certain class), we obtained unstable time series, for example, in cases where a segment size increases over time since it contains more and more objects. This unstable preservation of shape due to enlarging segments contradicts the idea of constructing metrics based on shape preservation and object characteristics over time. In addition, ground truth data for multi-object tracking of segments is not available. For this reason, the construction of metrics based on ground truth data and an evaluation of our light-weight tracking algorithm are not possible. This led us to instance segmentation.

We transferred the tasks of meta classification and meta regression to the predictions obtained by an instance segmentation network. As input for the meta models, we used metrics characterizing uncertainty and geometry of instances (similar to those for segments). We adapted our light-weight tracking algorithm to instances for temporal metrics and evaluated our method by different object tracking metrics. From tracked instances, we obtained time series of single-frame metrics as well as truly time-dynamic ones based on survival time analysis, on changes in the shape and on expected position of instances in an image sequence. In our tests, we studied the influence of these metrics on the meta tasks, various time series lengths and different meta models. For meta regression, direct prediction of the IoU, we achieved R^2 values of up to 86.85% and for meta classification, classifying between $IoU < 0.5$ and $IoU \geq 0.5$, $AUROC$ values of up to 98.78% which is clearly superior over the performance of the baseline methods (like mean entropy or score as single-metric and the single-frame approach of [136]). Moreover, we used meta classification as advanced score value improving the networks' prediction accuracy by replacing the score threshold by the estimated probability of correct classification during inference. We reduced the number of false positives by up to 44.03% while maintaining a constant number of false negatives.

Changing the perspective, we decreased the number of false negatives by up to 15.70% for the approximate same number of false positives using meta classification. However, the number of non-detected instances in the considered datasets is still high which may cause safety issues. For this reason, the detection of road users, i.e., the reduction of false negative instances, is of highest interest and we also focused on this problem.

We proposed a temporal post-processing method applicable to any instance seg-

mentation network to attain a high recall rate (low number of false negatives). First, we used our light-weight false negative detection algorithm which is based on inconsistencies in the time series of tracked instances such as a gap in the time series or a sudden end. Using the information of previous frames, these cases were detected and new instances were constructed that the neural network may have initially missed. Our false negative detection method increases the sensitivity towards the prediction of instances, possibly yielding an increased number of false positives. In order to prune those mentioned false positives, we employed meta classification. As input for the meta classifier, we used selected metrics characterizing uncertainty and geometry as well as the depth of a given instance. We studied the properties of the metrics, the influence of different time series lengths on the meta classification model (here gradient boosting) and various IoU thresholds considered for the evaluation. In our tests, we obtained $AUROC$ values of up to 99.30%. Furthermore, we compared our approach consisting of the detection and false positive pruning step with the application of a score threshold during inference improving the networks' prediction accuracy. We achieved $AUPRC$ values of up to 98.07% and in particular, reduced the number of false negative instances. In addition, we studied different occlusion levels for the ground truth instances observing that our approach can enhance the instance segmentation also in more difficult cases, e.g. we detected also instances that were occluded and located in a crowd of instances.

As an extension of our work, it might be interesting to develop further metrics for meta classification and meta regression based on the respective network structure. For example, we consider the YOLACT instance segmentation network [13] where the instance masks are formed by prototypes and associated coefficients. We did not use this provided information, instead we only considered the softmax probability distribution per instance given by the network. Based on the information typical of the network, new metrics could be created which might improve the performance of our meta tasks which are able to reliably evaluate the quality of a semantic or instance segmentation obtained from a neural network. Moreover, we focused on street scene images for automated driving in view and on scenes from pedestrian areas and shopping malls. It could also be of interest to investigate other applications, e.g. medical diagnosis. The Multimodal Brain Tumor Image Segmentation (BRATS) dataset [108] consists of multi-contrast MR scans of glioma patients and provides image sequences of these scans. On such a dataset our time-dynamic methods for prediction quality rating and performance improvements (in terms of false positives and false negatives detection) could be tested.

List of Figures

List of Tables

List of Algorithms

List of Notations

Throughout this thesis, the following abbreviations and notations are used across all chapters:

x	input (image)
y	label of input x
$\mathcal{C}$	label space
c	number of classes
w	learned weights of a neural network
z	pixel
$f_z(y\|x,w)$	probability distribution per pixel z
$\hat{y}_z(x,w)$	predicted class per pixel z
$\hat{\mathcal{S}}_x$	predicted segmentation
$\hat{\mathcal{K}}_x$ / $\hat{\mathcal{I}}_x$	the set of predicted segments/instances
$\mathcal{S}_x$	ground truth segmentation
$\mathcal{K}_x$	the set of segments in the ground truth
T	number of frames
x_t	t^{th} image
$O_{j,k}$	overlap of object k with object j
$\tilde{O}_{j,k}$	overlap of two objects j and k
$\bar{k}_t$ / $\bar{i}_t$	geometric center of segment k/instance i in frame t
$E_z(x,w)$	pixel-wise entropy
$V_z(x,w)$	pixel-wise variation ratio
$M_z(x,w)$	pixel-wise probability margin
$\bar{D}$	mean dispersion
$H_z(x)$	pixel-wise depth estimation
$\bar{H}$	mean depth
S	segment/instance size
k_{in}	the inner of a segment
k_{bd}	the boundary of a segment
$P(y\|k)$ / $P(y\|i)$	mean class probability
$\hat{c}$	class
s	score value

f	shape preservation
o	occlusion metric
d_c	geometric center deviation
d_s	size deviation
d_h	depth deviation
v	survival time
r	ratio metric
U, V	set of metrics
IoU	intersection over union
IoU_{adj}	adjusted IoU
mAP	mean average precision
ACC	accuracy
$AUROC$	area under receiver operating characteristics curve
$AUPRC$	area under precision-recall curve
σ	standard error
R^2	coefficient of determination
τ	IoU threshold for meta classification
$MOTP$	multiple object tracking precision
$MOTA$	multiple object tracking accuracy
$\overline{mme}$	mismatch ratio
smn	number of switches
GT	number of ground truth objects
MT	number of mostly tracked objects
PT	number of partially tracked objects
ML	number of mostly lost objects

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, (2015).

[2] C. Aeschliman, J. Park, and A. C. Kak, *A probabilistic framework for joint segmentation and tracking*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (2010), pp. 1371–1378.

[3] J. Aldrich, *R.A. Fisher and the making of maximum likelihood 1912-1922*, Statistical Science, 12 (1997), pp. 162 – 176.

[4] H. Attias, *A variational bayesian framework for graphical models*, Advances in Neural Information Processing Systems 12, (2000), pp. 209–215.

[5] B. Babenko, M. Yang, and S. Belongie, *Visual tracking with online multiple instance learning*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2009), pp. 983–990.

[6] V. Belagiannis, F. Schubert, N. Navab, and S. Ilic, *Segmentation based particle filtering for real-time 2d object tracking*, European Conference on Computer Vision (ECCV), (2012), pp. 842–855.

[7] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE transactions on neural networks, 5 (1994), pp. 157–66.

[8] K. Bernardin and R. Stiefelhagen, *Evaluating multiple object tracking performance: The clear mot metrics*, EURASIP Journal on Image and Video Processing, (2008).

[9] G. Bertasius and L. Torresani, *Classifying, segmenting, and tracking object instances in video with mask propagation*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2020), pp. 9736–9745.

[10] L. Bertinetto, J. Valmadre, J. Henriques, A. Vedaldi, and P. Torr, *Fully-convolutional siamese networks for object tracking*, (2016).

[11] C. Bishop, *Regularization and complexity control in feed-forward networks*, Proceedings International Conference on Artificial Neural Networks (ICANN), (1995), pp. 141–148.

[12] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, *Visual object tracking using adaptive correlation filters*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (2010), pp. 2544–2550.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, *Yolact: Real-time instance segmentation*, IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 9156–9165.

[14] L. Bottou, *Online algorithms and stochastic approximations*, (1998).

[15] L. Bottou and O. Bousquet, *The tradeoffs of large scale learning.*, Optimization for Machine Learning, 20 (2007).

[16] Y.-L. Boureau, J. Ponce, and Y. Lecun, *A theoretical analysis of feature pooling in visual recognition*, International Conference on Machine Learning (ICML), (2010), pp. 111–118.

[17] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, *Smote: Synthetic minority over-sampling technique*, Journal of Artificial Intelligence Research (JAIR), 16 (2002), pp. 321–357.

[18] R. Bracewell, *The fourier transform and its applications*, McGraw-Hill Book, (1965).

[19] J. Bridle, *Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters*, (1989).

[20] S. Bullinger, C. Bodensteiner, and M. Arens, *Instance flow based online multiple object tracking*, IEEE International Conference on Image Processing (ICIP), (2017), pp. 785–789.

[21] Á. Casado-García and J. Heras, *Ensemble methods for object detection*, European Conference on Artificial Intelligence (ECAI), (2020).

[22] L. Cehovin Zajc, A. Leonardis, and M. Kristan, *Visual object tracking performance measures revisited*, IEEE Transactions on Image Processing, 25 (2015).

[23] R. Chan, Matthias, Rottmann, F. Hüeger, P. Schlicht, and H. Gottschalk, *Metafusion: Controlled false-negative reduction of minority classes in semantic segmentation*, IEEE International Joint Conference on Neural Networks (IJCNN), (2020).

[24] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, European Conference on Computer Vision (ECCV), (2018).

[25] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 1800–1807.

[26] H.-R. Chou, J.-H. Lee, Y.-M. Chan, and C.-S. Chen, *Data-specific adaptive threshold for face recognition and authentication*, IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), (2019), pp. 153–156.

[27] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. P. Zafeiriou, *Deep polynomial neural networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2021), pp. 1–1.

[28] D. Comaniciu, V. Ramesh, and P. Meer, *Real-time tracking of non-rigid objects using mean shift*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2 (2000), pp. 142–149 vol.2.

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

[30] D. R. Cox, *Regression models and life-tables*, Journal of the Royal Statistical Society. Series B (Methodological), 34 (1972), pp. 187–220.

[31] G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function*, Springer-Verlag New York inc., 1989.

[32] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, *Learning spatially regularized correlation filters for visual tracking*, IEEE International Conference on Computer Vision (ICCV), (2015), pp. 4310–4318.

[33] T. DeVries and G. W. Taylor, *Leveraging uncertainty estimates for predicting segmentation quality*, (2018).

[34] S. C. Diamantas, A. Oikonomidis, and R. M. Crowder, *Depth estimation for autonomous robot navigation: A comparative approach*, IEEE International Conference on Imaging Systems and Techniques, (2010), pp. 426–430.

[35] S. Duffner and C. Garcia, *Pixeltrack: A fast adaptive algorithm for tracking non-rigid objects*, IEEE International Conference on Computer Vision (ICCV), (2013), pp. 2480–2487.

[36] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, (2016).

[37] D. Duvenaud, D. Maclaurin, and R. Adams, *Early stopping as nonparametric variational inference*, International Conference on Artificial Intelligence and Statistics (AISTATS), 51 (2016), pp. 1070–1077.

[38] D. Eigen, C. Puhrsch, and R. Fergus, *Depth map prediction from a single image using a multi-scale deep network*, Conference on Neural Information Processing Systems (NIPS), (2014).

[39] C. Erdem, B. Sankur, and A. Tekalp, *Performance measures for video object segmentation and tracking*, IEEE Transactions on Image Processing, 13 (2004).

[40] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The pascal visual object classes challenge (voc2012)*, (2012).

[41] T. Fawcett, *Roc graphs: Notes and practical considerations for researchers*, Machine Learning, 31 (2004), pp. 1–38.

[42] J. Friedman, *Greedy function approximation: A gradient boosting machine*, The Annals of Statistics, 29 (2000).

[43] J. H. Friedman, *Stochastic gradient boosting*, Computational Statistics and Data Analysis, 38 (2002), pp. 367–378.

[44] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, International Conference on Machine Learning (ICML), (2016), pp. 1050–1059.

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, *Vision meets robotics: The kitti dataset*, The International Journal of Robotics Research, 32 (2013), pp. 1231–1237.

[46] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite*, Conference on Computer Vision and Pattern Recognition (CVPR), (2012).

[47] S. Geman, E. Bienenstock, and R. Doursat, *Neural networks and the bias/variance dilemma*, Neural Computation, 4 (1992), pp. 1–58.

[48] R. Girshick, *Fast r-cnn*, (2015).

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2013).

[50] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, Journal of Machine Learning Research, 15 (2010).

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[52] S. Goswami, *False detection (positives and negatives) in object detection*, (2020).

[53] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, International Conference on Machine Learning (ICML), (2017), p. 1321–1330.

[54] S. Hanson and L. Pratt, *Comparing biases for minimal network construction with back-propagation*, International Conference on Neural Information Processing Systems (NIPS), (1988).

[55] K. Hariharakrishnan and D. Schonfeld, *Fast object tracking using adaptive block matching*, IEEE Transactions on Multimedia, 7 (2005), pp. 853–859.

[56] V. Harisankar., V. V. Sajith, and K. P. Soman, *Unsupervised depth estimation from monocular images for autonomous vehicles*, Fourth International Conference on Computing Methodologies and Communication (ICCMC), (2020), pp. 904–909.

[57] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, Springer, 2009.

[58] A. He, C. Luo, X. Tian, and W. Zeng, *Towards a better match in siamese network based visual object tracker*, European Conference on Computer Vision (ECCV) Workshops, (2018).

[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, IEEE International Conference on Computer Vision (ICCV), (2017), pp. 2980–2988.

[60] K. He and J. Sun, *Convolutional neural networks at constrained time cost*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2015), pp. 5353–5360.

[61] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 770–778.

[62] D. Held, S. Thrun, and S. Savarese, *Learning to track at 100 FPS with deep regression networks*, Lecture Notes in Computer Science, (2016).

[63] D. Hendrycks and K. Gimpel, *A baseline for detecting misclassified and out-of-distribution examples in neural networks*, (2016).

[64] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, (2012).

[65] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, A Field Guide to Dynamical Recurrent Neural Networks, (2003).

[66] K. Hoebel, V. Andrearczyk, A. Beers, J. Patel, K. Chang, A. Depeursinge, H. Müller, and J. Kalpathy-Cramer, *An exploration of uncertainty information for segmentation quality assessment*, (2020).

[67] K. Hornik, M. Stinchcombe, and H. White, *Multilayer Feedforward Networks are Universal Approximators*, vol. 2, 1989.

[68] C. Huang, Q. Wu, and F. Meng, *Qualitynet: Segmentation quality evaluation with deep convolutional networks*, Visual Communications and Image Processing (VCIP), (2016), pp. 1–4.

[69] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 2261–2269.

[70] P. Huang, W. T. Hsu, C. Chiu, T. Wu, and M. Sun, *Efficient uncertainty estimation for semantic segmentation in videos*, European Conference on Computer Vision (ECCV), (2018).

[71] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, (2020).

[72] P. Jaccard, *The distribution of the flora in the alpine zone*, New Phytologist, 11 (1912), pp. 37–50.

[73] K. Janocha and W. Czarnecki, *On loss functions for deep neural networks in classification*, Schedae Informaticae, 25 (2017).

[74] B. A. Jilani, T. Rabie, and M. Baziyad, *Autonomous motion tracking for dynamic objects using a temporal quad-tree algorithm*, Advances in Science and Engineering Technology International Conferences (ASET), (2019), pp. 1–5.

[75] A. Jung, *Explainable empirical risk minimization*, (2020).

[76] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, *Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, (2016), pp. 680–688.

[77] J. Kang and J. Gwak, *Ensemble of instance segmentation models for polyp segmentation in colonoscopy images*, IEEE Access, 7 (2019), pp. 26440–26447.

[78] A. Kendall, V. Badrinarayanan, and R. Cipolla, *Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding*, (2015).

[79] C. Kim, F. Li, and J. M. Rehg, *Multi-object tracking with neural gating using bilinear lstm*, European Conference on Computer Vision (ECCV), (2018).

[80] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, International Conference on Learning Representations, (2014).

[81] F. Kraus and K. Dietmayer, *Uncertainty estimation in one-stage object detection*, IEEE Intelligent Transportation Systems Conference (ITSC), (2019), p. 53–60.

[82] M. Kristan, J. Matas, A. Leonardis, T. Vojír, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Cehovin, *A novel performance evaluation methodology for single-target trackers*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 38 (2016), pp. 2137–2155.

[83] A. Krizhevsky, I. Sutskever, and G. Hinton, *Imagenet classification with deep convolutional neural networks*, Neural Information Processing Systems, 25 (2012).

[84] S. Kullback and R. A. Leibler, *On Information and Sufficiency*, The Annals of Mathematical Statistics, 22 (1951), pp. 79 – 86.

[85] F. Küppers, J. Kronenberger, A. Shantia, and A. Haselhoff, *Multivariate confidence calibration for object detection*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, (2020), pp. 1322–1330.

[86] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, International Conference on Neural Information Processing Systems (NIPS), (2017), pp. 6405–6416.

[87] M. T. Le, F. Diehl, T. Brunner, and A. Knol, *Uncertainty estimation for deep neural object detectors in safety-critical applications*, International Conference on Intelligent Transportation Systems (ITSC), (2018), pp. 3873–3878.

[88] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278 – 2324.

[89] H. Lee, S. T. Kim, N. Navab, and Y. Ro, *Efficient ensemble model generation for uncertainty estimation with bayesian approximation in segmentation*, (2020).

[90] J. H. Lee, M.-K. Han, D. Ko, and I. H. Suh, *From big to small: Multi-scale local planar guidance for monocular depth estimation*, (2019).

[91] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, *High performance visual tracking with siamese region proposal network*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2018), pp. 8971–8980.

[92] G. Li, Y. Xie, L. Lin, and Y. Yu, *Instance-level salient object segmentation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017).

[93] S. Liang, Y. Li, and R. Srikant, *Principled detection of out-of-distribution examples in neural networks*, (2017).

[94] C.-C. Lin, Y. Hung, R. Feris, and L. He, *Video instance segmentation tracking with a modified vae architecture*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2020).

[95] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context*, (2014).

[96] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2016), pp. 1–1.

[97] J. Luiten, P. Torr, and B. Leibe, *Video instance segmentation 2019: A winning approach for combined detection, segmentation, classification and tracking.*, IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), (2019), pp. 709–712.

[98] K. Maag, *False negative reduction in video instance segmentation using uncertainty estimates*, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), (2021).

[99] K. Maag, M. Rottmann, and H. Gottschalk, *Time-dynamic estimates of the reliability of deep semantic segmentation networks*, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), (2020), pp. 502–509.

[100] K. Maag, M. Rottmann, S. Varghese, F. Hüger, P. Schlicht, and H. Gottschalk, *Improving video instance segmentation by light-weight temporal uncertainty estimates*, IEEE International Joint Conference on Neural Networks (IJCNN), (2021).

[101] A. L. Maas, A. Y. Hannun, and Y. N. Andrew, *Rectifier nonlinearities improve neural network acoustic models*, International Conference on Machine Learning (ICML) Workshop, (2013).

[102] W. Maass, *Networks of spiking neurons: The third generation of neural network models*, Neural Networks, 10 (1996), pp. 1659–1671.

[103] D. J. C. MacKay, *A practical bayesian framework for backpropagation networks*, Neural Computation, 4 (1992), pp. 448–472.

[104] D. Mandic and J. Chambers, *Recurrent neural networks for prediction: Learning algorithms,architectures and stability*, (2001).

[105] L. Mason, J. Baxter, P. Bartlett, and M. Frean, *Boosting algorithms as gradient descent*, International Conference on Neural Information Processing Systems (NIPS), (1999), pp. 512–518.

[106] W. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, 5 (1943), p. 115–133.

[107] A. Mehrtash, W. M. Wells, C. M. Tempany, P. Abolmaesumi, and T. Kapur, *Confidence calibration and predictive uncertainty estimation for deep medical image segmentation*, IEEE Transactions on Medical Imaging, 39 (2020), pp. 3868–3878.

[108] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al., *The multimodal brain tumor image segmentation benchmark (brats)*, IEEE Transactions on Medical Imaging, 34 (2015), pp. 1993–2024.

[109] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, *Mot16: A benchmark for multi-object tracking*, (2016).

[110] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf, *Dropout sampling for robust object detection in open-set conditions*, (2017).

[111] D. Moore, *Applied Survival Analysis Using R*, 2016.

[112] D. Morrison, A. Milan, and E. Antonakos, *Uncertainty-aware instance segmentation using dropout sampling*, (2019).

[113] X. Mu, J. Che, T. Hu, and Z. Wang, *A video object tracking algorithm combined kalman filter and adaptive least squares under occlusion*, International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), (2016), pp. 6–10.

[114] I. Namatevs, L. Aleksejeva, and I. Polaka, *Neural network modelling for sports performance classification as a complex socio-technical system*, Information Technology and Management Science, 19 (2016).

[115] National Transportation Safety Board, *Collision between vehicle controlled by developmental automated driving system and pedestrian, tempe, arizona, march 18, 2018*, Highway Accident Report, (2019).

[116] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman, *Applied Linear Statistical Models*, Irwin, Chicago, 1996.

[117] L. Neumann, A. Zisserman, and A. Vedaldi, *Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection*, (2018).

[118] P. Niyogi and F. Girosi, *On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions*, Neural Computation, 8 (1995).

[119] P. Oberdiek, M. Rottmann, and H. Gottschalk, *Classification uncertainty of deep neural networks based on gradient information*, Artificial Neural networks and Pattern Recognition (ANNPR), (2018).

[120] O. Ozdemir, B. Woodward, and A. A. Berlin, *Propagating uncertainty in multi-stage bayesian convolutional neural networks with application to pulmonary nodule detection*, (2017).

[121] T.-Y. Pan, C. Zhang, Y. Li, H. Hu, D. Xuan, S. Changpinyo, B. Gong, and W.-L. Chao, *On model calibration for long-tailed object detection and instance segmentation*, (2021).

[122] T. H. Park and G. Casella, *The bayesian lasso*, Journal of the American Statistical Association, 103 (2008), pp. 681 – 686.

[123] C. Payer, D. Stern, T. Neff, H. Bischof, and M. Urschler, *Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks*, Medical Image Computing and Computer Assisted Intervention (MICCAI), (2018).

[124] B. Phan, R. Salay, K. Czarnecki, V. Abdelzad, T. Denouden, and S. Vernekar, *Calibrating uncertainties in object localization task*, (2018).

[125] J. Platt, *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods*, Advances in Large Margin Classifiers, 10 (2000).

[126] B. Polyak, *Some methods of speeding up the convergence of iteration methods*, Ussr Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17.

[127] L. Prechelt, *Early Stopping - But When?*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 55–69.

[128] J. Ranstam and J. Cook, *Lasso regression*, British Journal of Surgery, 105 (2018), pp. 1348–1348.

[129] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 779–788.

[130] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2015).

[131] S. Riad, *The deconvolution problem: An overview*, Proceedings of the IEEE, 74 (1986), pp. 82–85.

[132] S. R. Richter, Z. Hayder, and V. Koltun, *Playing for benchmarks*, IEEE International Conference on Computer Vision, ICCV, (2017), pp. 2232–2241.

[133] H. Robbins and S. Monro, *A Stochastic Approximation Method*, The Annals of Mathematical Statistics, 22 (1951), pp. 400 – 407.

[134] R. Rosales, M. Schmidt, and G. Fung, *Fast optimization methods for l1 regularization: A comparative study and two new approaches*, Science, (2007).

[135] V. Roth, *The generalized lasso*, IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 15 (2004), pp. 16–28.

[136] M. Rottmann, P. Colling, T. Hack, F. Hüger, P. Schlicht, and H. Gottschalk, *Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities*, IEEE International Joint Conference on Neural Networks (IJCNN), (2020).

[137] M. Rottmann and M. Schubert, *Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, (2019).

[138] A. G. Roy, S. Conjeti, N. Navab, and C. Wachinger, *Inherent brain segmentation quality control from fully convnet monte carlo sampling*, (2018).

[139] D. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536.

[140] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision (IJCV), 115 (2015), pp. 211–252.

[141] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation*, (2018).

[142] A. Saxena, S. Chung, and A. Ng, *Learning depth from single monocular images*, Advances in Neural Information Processing Systems, 18 (2006).

[143] T. Schaul, I. Antonoglou, and D. Silver, *Unit tests for stochastic optimization*, International Conference on Learning Representations, (2014).

[144] M. Schubert, K. Kahl, and M. Rottmann, *Metadetect: Uncertainty quantification and prediction quality estimates for object detection*, IEEE International Joint Conference on Neural Networks (IJCNN), (2021).

[145] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, *Deep network flow for multi-object tracking*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017).

[146] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning. from theory to algorithms*, (2013).

[147] C. E. Shannon, *A mathematical theory of communication*, The Bell System Technical Journal, 27 (1948), pp. 379–423.

[148] A. N. Shiryayev, *On The Law of Large Numbers*, Springer Netherlands, Dordrecht, 1992, pp. 43–47.

[149] J. Son, M. Baek, M. Cho, and B. Han, *Multi-object tracking with quadruplet convolutional neural networks*, Conference on Computer Vision and Pattern Recognition (CVPR), (2017).

[150] J. Son, I. Jung, K. Park, and B. Han, *Tracking-by-segmentation with online gradient boosting decision tree*, IEEE International Conference on Computer Vision (ICCV), (2015), pp. 3056–3064.

[151] S. Sonoda and N. Murata, *Neural network with unbounded activation functions is universal approximator*, Applied and Computational Harmonic Analysis, 43 (2015).

[152] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research, 15 (2014), pp. 1929–1958.

[153] P. Srivastava, K. Mishra, V. Awasthi, V. Sahu, and P. K. Pal, *Plant disease detection using convolutional neural network*, International Journal of Advanced Research, 09 (2021), pp. 691–698.

[154] F. Sultana, A. Sufian, and P. Dutta, *Advancements in image classification using convolutional neural network*, Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), (2018), pp. 122–129.

[155] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, *On the importance of initialization and momentum in deep learning*, (2013).

[156] L. Thai, T. S. Hai, and N. Thuy, *Image classification using support vector machine and artificial neural network*, International Journal of Information Technology and Computer Science, 4 (2012), pp. 32–38.

[157] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B, 58 (1996), pp. 267–288.

[158] R. Tibshirani, *The lasso problem and uniqueness*, Electronic Journal of Statistics, 7 (2012), pp. 1456–1490.

[159] L. Torgo, R. Ribeiro, B. Pfahringer, and P. Branco, *Smote for regression*, Progress in Artificial Intelligence, (2013), pp. 378–389.

[160] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, *End-to-end representation learning for correlation filter based tracking*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017).

[161] V. Vapnik, *Principles of risk minimization for learning theory*, 4 (1992).

[162] P. Voigtlaender, M. Krause, A. Ošep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, *MOTS: Multi-object tracking and segmentation*, Conference on Computer Vision and Pattern Recognition (CVPR), (2019).

[163] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, *Phoneme recognition using time-delay neural networks*, IEEE Transactions on Acoustics, Speech and Signal Processing, 37 (1989), pp. 328 – 339.

[164] L. Wang, J. Shi, G. Song, and I.-f. Shen, *Object detection combining recognition and segmentation*, Computer Vision (ACCV), (2007).

[165] L. Wang, J. Zhang, O. Wang, Z. L. Lin, and H. Lu, *Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2020), pp. 538–547.

[166] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, *Fast online object tracking and segmentation: A unifying approach*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2019), pp. 1328–1338.

[167] S. Wang, B. Nan, S. Rosset, and J. Zhu, *Random lasso*, The annals of applied statistics, 5 (2011), pp. 468–485.

[168] Z. Wang, L. Zheng, Y. Liu, and S. Wang, *Towards real-time multi-object tracking*, (2019).

[169] K. Wickstrøm, M. Kampffmeyer, and R. Jenssen, *Uncertainty and interpretability in convolutional neural networks for semantic segmentation of colorectal polyps*, Medical Image Analysis, 60 (2019).

[170] Y. Wu, J. Lim, and M. Yang, *Online object tracking: A benchmark*, IEEE Conference on Computer Vision and Pattern Recognition, (2013), pp. 2411–2418.

[171] K. Xiang, K. Wang, and K. Yang, *A comparative study of high-recall real-time semantic segmentation based on swift factorized network*, Security + Defence, (2019).

[172] K. Xiang, K. Wang, and K. Yang, *Importance-aware semantic segmentation with efficient pyramidal context network for navigational assistant systems*, IEEE Intelligent Transportation Systems Conference (ITSC), (2019).

[173] Y. Xiang, A. Alahi, and S. Savarese, *Learning to track: Online multi-object tracking by decision making*, IEEE International Conference on Computer Vision (ICCV), (2015), pp. 4705–4713.

[174] R. Xiao, L. Zhu, and H.-J. Zhang, *Boosting chain learning for object detection*, IEEE International Conference on Computer Vision (ICCV), (2003).

[175] B. Yang, W. Luo, and R. Urtasun, *Pixor: Real-time 3d object detection from point clouds*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2018).

[176] L. Yang, Y. Fan, and N. Xu, *Video instance segmentation*, IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 5187–5196.

[177] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, *Denseaspp for semantic segmentation in street scenes*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2018), pp. 3684–3692.

[178] T. Yang and A. B. Chan, *Learning dynamic memory networks for object tracking*, (2018).

[179] R. Yao, G. Lin, C. Shen, Y. Zhang, and Q. Shi, *Semantics-aware visual object tracking*, IEEE Transactions on Circuits and Systems for Video Technology, 29 (2019), pp. 1687–1700.

[180] D. Yarotsky, *Universal approximations of invariant maps by neural networks*, (2018).

[181] D. Yeo, J. Son, B. Han, and J. H. Han, *Superpixel-based tracking-by-segmentation using markov chains*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 511–520.

[182] A. Yilmaz, O. Javed, and M. Shah, *Object tracking: A survey*, ACM Computing Surveys, 38 (2006), pp. 1–45.

[183] F. Yu and V. Koltun, *Multi-scale context aggregation by dilated convolutions*, (2015).

[184] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, European Conference on Computer Vision (ECCV), (2014).

[185] J. Zhang, Y. Dai, X. Yu, M. Harandi, N. Barnes, and R. Hartley, *Uncertainty-aware deep calibrated salient object detection*, (2020).

[186] X. Zhang, X. Zhou, M. Lin, and J. Sun, *Shufflenet: An extremely efficient convolutional neural network for mobile devices*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2018), pp. 6848–6856.

[187] D. Zhao, H. Fu, L. Xiao, T. Wu, and B. Dai, *Multi-object tracking with correlation filter for autonomous vehicle*, Sensors, 18 (2018), p. 2004.

[188] D. Zhou and Q. He, *Poseg: Pose-aware refinement network for human instance segmentation*, IEEE Access, 8 (2020), pp. 15007–15016.

[189] D.-X. Zhou, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis, 48 (2019).

[190] Y. Zhou and R. Chellappa, *Computation of optical flow using a neural network*, IEEE 1988 International Conference on Neural Networks, (1988), pp. 71–78 vol.2.

[191] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, *Online multi-object tracking with dual matching attention networks*, European Conference on Computer Vision (ECCV), (2018).