Unsupervised Open World Recognition in Computer Vision

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Dissertation

Bergische Universität Wuppertal
Fakultät für Mathematik und Naturwissenschaften

eingereicht von
Svenja Uhlemeyer, M. Sc.
zur Erlangung des Grades eines Doktors der Naturwissenschaften

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

Wuppertal, 29.03.2023

Acknowledgments

A special “thank you” goes to Hanno Gottschalk and Matthias Rottmann for introducing me to the world of machine learning and for the extensive mentoring over the last three years. Considering that I startet my PhD without any *prior knowledge* about neural networks, it seems that your training methods were successful nevertheless.

The research was funded by the German Federal Ministry for Economic Affairs and Energy, within the project “KI Delta Learning”, grant no. 19A19013Q. I would like to thank the consortium and all the partners for the successful cooperation.

Furthermore, I would like to thank the entire team for the great time and the informative exchanges. In particular, I would like to thank my office colleagues Annika Mütze, Antonia van Betteray and Chewie for the great time together in the office, on business trips, and also outside the office. Especially Chewie, in his role as OoD dog, has improved a lot of works in the field of OoD detection.

Finally, I would like to thank Kamil Kowol for always providing me with coffee and OoD data, for the joint writing sessions, for proofreading this dissertation, and for his constant motivation.

Foreword

Parts of the work presented in this thesis were already published in the following publications:

- S. Uhlemeyer, M. Rottmann, and H. Gottschalk, *Towards unsupervised open world semantic segmentation*, in The 38th Conference on Uncertainty in Artificial Intelligence, 2022
  - the whole paper was mostly written by myself
  - all experiments were implemented and conducted by myself
- R. Chan, K. Lis, S. Uhlemeyer, H. Blum, S. Honari, R. Siegwart, P. Fua, M. Salzmann, and M. Rottmann, *SegmentMeIfYouCan: A Benchmark for Anomaly Segmentation*, in Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2021
  - major contribution to dataset statistics described in Table 1, subsection 2.1 and Figure 2
  - most of the figures and plots were created by myself
  - sections D, F 4 and 5, G and H in the Appendix were mostly written by myself
  - major contribution to creating the datasets in terms of recording, web sourcing, licenses and annotation
  - major contribution to the experimental results
  - major contribution to the design and content of the benchmark webpage https://segmentmeifyoucan.com

- K. Maag, R. Chan, S. Uhlemeyer, K. Kowol, and H. Gottschalk, *Two Video Data Sets for Tracking and Retrieval of Out of Distribution Objects*, in Asian Conference on Computer Vision (ACCV), 2023
  - the paragraph about *Object Retrieval* in section 2, subsection 4.3, the paragraphs about *OoD Object Retrieval* in section 5 were mostly written by myself
  - implementation and visualization of OoD retrieval and clustering https://github.com/kmaag/OOD-Tracking
- D. Bogdoll, S. Uhlemeyer, K. Kowol, and J. M. Zollner, *Perception datasets for anomaly detection in autonomous driving: A survey*, ArXiv, abs/2302.02790 (2023)
  - abstract, subsections II B and II F and the discussion topics *Definition of Normality, Domain Shift, Size, Similarity* in section III were mostly written by myself
  - general editing of section II A-H
  - major contribution to the structure and content of Table 1
  - implementation of code for visualizations (except for CODA) https://github.com/daniel-bogdoll/anomaly_datasets
- S. Uhlemeyer, J. Lienen, E. Hüllermeier, and H. Gottschalk, *Detecting novelties with empty classes*, to appear on ArXiv, (2023)
  - major contribution to the idea
  - sections 3,4 and 5 were mostly written by myself
  - all experiments were implemented and conducted by myself

Contents

Chapter 1

Introduction

The development and improvement of deep convolutional neural networks (DNNs) for complex tasks such as detecting diseases in medical images or operating as perception systems in automated vehicles is a rapidly evolving research area. To obtain information about an entire scene, semantic segmentation DNNs classify image data on pixel level, providing further information such as location, shape or context of an object. Restricted to a closed semantic world, which assumes that all objects can be assigned to one of the predefined classes, state-of-the-art DNNs as for instance the DeepLabV3+ [25] or the PSPNet [155] already achieve high segmentation accuracy. However, these DNNs are ill-equipped to operate in an open world, where they will inevitably be confronted with domain shifts or novel classes. The first refers to a shift of the data generating distribution, which can be due to different day times, weather conditions or countries. This means a changed appearance of known classes, causing a performance drop of the segmentation DNN. Furthermore, novel classes may appear, regardless of whether the domain is shifted or not. For example, this happens due to technical innovations such as e-scooters, which have been approved for road traffic in Germany since June 2019, or to a change of location, *e.g.* from urban to coastal street scenes. The semantic segmentation DNN employed in Figure 1.1 was trained on the urban Cityscapes dataset [29], which *e.g.* does not include any boats. However, near the coast, boats may appear frequently, either as trailer, or, as depicted in the example, carried or dragged by pedestrians. Urged to make a decision, the DNN incorrectly assigns the boat pixels to known classes such as *fence, pedestrian* or *car*. This issue of selecting a class despite having a low confidence is addressed by adding an *I don't know* output class.

Open World Open world approaches embrace the detection and learning of unknown classes. However, there exist many notions of anomaly detection (or

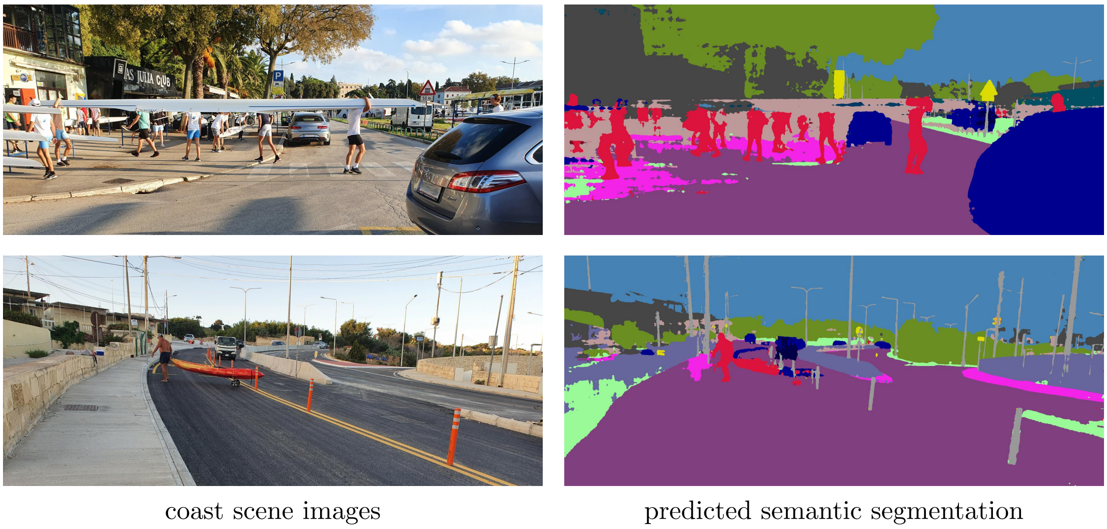


Figure 1.1: A semantic segmentation DNN trained on German urban street scenes, applied to scenes near the coast of Croatia (*top*) and Malta (*bottom*). Pedestrians are crossing the street with a boat, which is not included in the model's underlying classes.

segmentation), which are defined inconsistently in literature. The following taxonomy has been introduced in [147]. *Anomaly detection* describes the task of identifying all instances, either in training or test data, that do not fit the model's underlying distribution. If this task is applied during inference of the test data, *i.e.*, without any knowledge of labels or the data generating distribution, it is referred to as *out-of-distribution* (OoD) detection. The intention of open world applications is not only to detect anomalous objects, but novel classes. This is denoted as *novelty detection*, *i.e.*, the detection of objects which correspond to a new semantic class. *Open set recognition* describes the task of fitting a machine learning model to work well on in-distribution data while recognizing anomalies. When the model is further extended by novel classes, the task is termed *open world recognition*. This is, while open set recognition only demands the model to classify anomalous or novel objects as *none-of-the-knowns*, open world recognition requires a distinction of different novel semantic classes as well as corresponding labels. Besides, there is also the task of *outlier detection*, which seeks to detect covariate or label values that are extreme and therefore distort the fitting of a machine learning model. For example, this could be a consequence of a defect sensor or direct sun exposure. Thus, outlier are extreme or rare observations, drawn from the data generating distribution, while anomalies are drawn from a different distribution as depicted in Figure 1.2.

Solutions of open world recognition problems are defined as a tuple [6], which includes a model, a novelty detector, a labeling process and an incremental learning

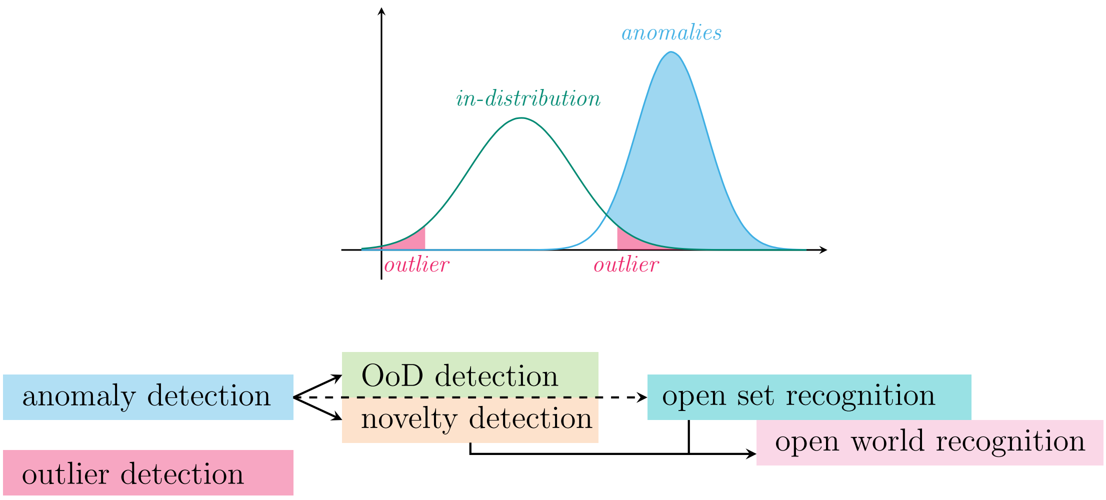


Figure 1.2: Open world recognition builds upon novelty detection, which can be understood as detecting anomalies from another distribution and equip them with labels.

function. The most basic concept is to collect anomalous data samples belonging to novel classes. In order to incrementally train a model to adapt to these novel classes, annotated data, called *ground truth* (GT), is required. In supervised learning approaches, the data is annotated by humans. There are well-established methods for supervised incremental learning that address the issue of *catastrophic forgetting* [94], *i.e.*, the performance drop on previously acquired knowledge while learning the novel classes. Although originally developed for image classification or object detection tasks, some incremental learning approaches are also applicable to semantic segmentation [98]. Since human annotators are expensive, semi- and weakly supervised approaches were developed, which utilize (few) labeled and unlabeled data. Few-shot learning approaches require at most 5 labeled samples of a novel class. Even more sophisticated zero-shot approaches learn novel classes without any image GT, but with information from different modalities, *e.g.* from text about an object's appearance, properties or functionalities. Finally, unsupervised approaches create pseudo-labels by grouping the anomalies into visually related clusters [50, 134]. Each cluster thus constitutes a novel, but unlabeled, semantic class. While this is rather basic for image classification, semantic segmentation requires pixel-level pseudo-labels.

Contribution This work addresses the problem of learning novel classes in an unsupervised manner, with focus on the semantic segmentation of street scenes. Currently, research in this area suffers from a lack of datasets for anomaly detection, and in particular for novelty detection. Anomaly datasets require a wide variety of anomalous object types to ensure that methods can generalize well.

However, most of the anomalies such as *bottles*, *balls* or *tree stumps* are negligible when learning new classes, hence it is sufficient to recognize them as anomalies or obstacles on the road. For novelty detection, datasets are needed which contain frequently occuring novelties such as *animals, boats* or *e-scooters* in the test data. Ideally, these objects are not present in the training images. A survey [12] of the existing datasets for anomaly segmentation in street scenes provides a structured overview, discusses the different strategies and issues regarding dataset creation, and the general problem of not having a universal definition of normality. Several datasets have been created in the context of this thesis. The RoadAnomaly21 and RoadObstacle21 datasets [21] are part of the *SegmentMeIfYouCan*[1] [21] benchmark, which provides a comparative overview of existing anomaly segmentation approaches. The Wuppertal OoD tracking datasets [88] contain video sequences for the detection and tracking of anomalies over multiple frames, followed by clustering, where novel classes are identified by clustering feature embeddings of image patches that are tailored to the proposed anomalies.

The first proposed approach [134] on open world recognition extends a semantic segmentation DNN by novel classes, using anomaly segmentation along with clustering as labeling process. Next, pseudo-labels are generated by mapping the cluster labels back to a pixel-level. The DNN is fine-tuned on the pseudo-labeled data by incremental learning, adapting to the novel classes without any human-annotated GT. As the first attempt to extend a semantic segmentation DNN by novel classes using clustering, it is considered as a baseline for further approaches in the field of unsupervised open world semantic segmentation.

Since the baseline approach suffers from noisy pseudo-labels and from clustering methods that are highly sensitive to the choice of hyperparameters, the question arises, whether the clustering of anomalies can be incorporated into the fine-tuning of the model. In a more sophisticated approach [133], the anomalies are not clustered and pseudo-labeled in a preprocessing step, but a matrix of pairwise distances between them is computed. Then the model is extended by *empty classes* and trained with a modified loss function. This loss pushes the model to concentrate the softmax scores for anomalies in the empty classes, and based on the distance matrix, it distributes the anomalies to the new classes.

In summary, this thesis encompasses several works on anomaly datasets, such as a benchmark for evaluating anomaly segmentation approaches and an anomaly retrieval and clustering approach tailored to video sequences, and finally two pioneering approaches in the field of unsupervised open world recognition for semantic segmentation.

[1] https://segmentmeifyoucan.com

Structure The thesis is organized as follows. The fundamentals in Chapter 2 give an overview of the theory behind deep learning, providing the basis for the methods developed in the main chapters. Section 2.1 covers learning theory for supervised learning, including learnability and sources of error and uncertainty in machine learning. Next, Section 2.2 provides an overview of deep (convolutional) neural networks, giving a mathematical description of feed forward networks in general and going into more detail on convolutional neural networks as well as on optimization algorithms applied to neural networks. DNNs can be used in computer vision to interpret and understand visual data such as videos or images. More specifically, they can learn to identify, classify, or localize objects in the visual world. The computer vision tasks image classification and semantic segmentation are considered in Section 2.3, including a review of state-of-the-art DNNs and common evaluation metrics. While the previous sections were mostly tailored to supervised learning approaches, there is a wide variety of machine learning paradigms that address learning with few labeled or even unlabeled data samples, some of which are presented in Section 2.4.

Chapter 3 deals with anomalies in street scenes, including datasets and applications, starting with a survey of datasets in Section 3.1, that discusses approaches to generating anomalous datasets. It further gives a definition of normality, and provides an overview of existing datasets. Then, Section 3.2 provides an overview of several anomaly segmentation methods and introduces an approach for anomaly clustering and retrieval, along with appropriate evaluation metrics.

The open world recognition approach for semantic segmentation of street scene images with pseudo-labels is presented in Chapter 4. It starts with an introduction to the topic in Section 4.1, outlining the main steps of the method and describing the experimental setup, followed by the related works in Section 4.2. The discovery of novel classes and the pseudo-labeling process are presented in Section 4.3, describing anomaly detection by meta regression, clustering of image patch embeddings and novelty segmentation. Then, detailed information on the extension of the DNN's semantic space by class-incremental learning is given in Section 4.4, followed by experiments and results in Section 4.5 and a conclusion in Section 4.6.

Next, the approach of novelty detection with empty classes is presented in Chapter 5. Starting with an introduction in Section 5.1 and the related works in Section 5.2, the method for image classification is introduced in Section 5.3. It is divided into four building blocks, which are the learning model, the OoD detection approach, the computation of a distance matrix, and the class-incremental learning with the proposed loss functions. In Section 5.4, the approach is adapted to the task of semantic segmentation. The experiments and results are provided in Section 5.5. Finally, Section 5.6 discusses possibilities for further improvements.

This thesis concludes in Chapter 6 with a summary of the main contributions along with a discussion of the inherent difficulties and challenges of the proposed open world approaches, and an outlook on future work raising possible improvements.

Chapter 2

Foundations

To enable machines like cars or robots to interact with the world, they need to be aware of their environment. Cameras or different sensors such as radar or LiDAR are used to create a representation of the surrounding area, which is then interpreted by a parameterized statistical model. Deep learning employs artificial neural networks which are capable of solving highly complex perception tasks. However, these neural networks require a huge amount of labeled data to *learn* the best parameterization, which is time-consuming and expensive. This chapter introduces the theoretical foundations of statistical learning and deep neural networks for image data, followed by an overview of learning approaches which leverage unlabeled data to improve the performance of a neural network.

2.1 Statistical Learning Theory for Supervised Learning

In the context of statistical learning with parameterized models, *learning* refers to optimizing the model parameters. To this end, the performance of a model is measured by computing a loss function over labeled training samples. Usually, the parameter choice is restricted to a hypothesis space, which is a predefined set of feasible models. This chapter introduces the main learnability concepts for hypothesis spaces. Furthermore, the error of a statistical model is decomposed into different error types, showing that the error can be minimized if each of the errors is controllable.

2.1.1 Learning Models and Learnability

Supervised machine learning approaches employ parameterized statistical models $f_\theta : \mathcal{Y} \times \mathcal{X} \to (0, 1)$ with

$$\int_{\mathcal{Y}} f_\theta(y|x)dy = 1 \quad \text{or} \quad \sum_{y \in \mathcal{Y}} f_\theta(y|x) = 1 \ ,$$

for a continuous or a discrete target space $\mathcal{Y}$, respectively, to estimate a data generating distribution $P_{X,Y}$ in terms of a conditional probability density function $p_{Y|X}$. This is done by observing a training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m \sim P_{X,Y}^{\otimes m}$, which is assumed to be independent and identically distributed, to *learn* the parameters $\theta \in \Theta$ of the statistical model, such that $f_\theta \approx p_{Y|X}$. Thereby, the learning algorithm can be restricted to a predefined set of functions, called the hypothesis space $\mathcal{H} \subseteq \{f_\theta : \ \theta \in \Theta\}$.

A popular statistical method for estimating the model parameters is maximum likelihood estimation (MLE) [9, 127], yielding a parameter estimate $\hat{\theta}$, which maximizes the likelihood, *i.e.*, which satisfies the equation

$$\hat{\theta} = \operatorname*{argmax}_{\theta \in \Theta:\ f_\theta \in \mathcal{H}} \prod_{i=1}^m f_\theta(y_i|x_i) \ . \tag{2.1}$$

This is done by solving the first-order optimality condition $\nabla_\theta \ell(\mathcal{S}|f_\theta) = 0$ of the negative log-likelihood function

$$\ell(\mathcal{S}|f_\theta) = -\log(\prod_{i=1}^m f_\theta(y_i|x_i)) = -\sum_{i=1}^m \log(f_\theta(y_i|x_i)) \ . \tag{2.2}$$

The true loss of a statistical model f_θ in terms of the negative log-likelihood is defined as

$$\mathcal{L}(f_\theta) = \mathbb{E}_{(x,y)\sim P_{X,Y}}[-\log(f_\theta(y|x))] \ . \tag{2.3}$$

Since the data generating distribution $P_{X,Y}$ is unknown, a learning algorithm can only estimate the true loss in terms of the empirical loss function

$$\mathcal{L}_\mathcal{S}(f_\theta) = \frac{1}{m} \sum_{i=1}^m - \log(f_\theta(y_i|x_i)) \ , \tag{2.4}$$

which averages the negative log-likelihood over the examples in $\mathcal{S}$. A learning algorithm is called an empirical risk minimizer (ERM) [127], if it yields a parameterization which satisfies

$$f_{\hat{\theta}} \in \operatorname*{argmin}_{f_\theta \in \mathcal{H}} \mathcal{L}_\mathcal{S}(f_\theta) \ . \tag{2.5}$$

It follows, that with the negative log-likelihood as empirical loss function, the ERM coincides with MLE.

For a classification task with a finite set of classes $\mathcal{Y} = \{1, \dots, Q\}$, the cross-entropy loss is employed as

$$\ell_{\mathrm{CE}}(y, f_\theta) = -\sum_{q=1}^{Q} \mathbb{1}_{\{q=y\}} \log(f_\theta(q|x)) = -\log(f_\theta(y|x)) \; \forall (x, y) \in \mathcal{S} \ , \tag{2.6}$$

which is just the same as the negative log-likelihood. Therefore, the conditional probability $p_{Y|X}$ is represented by a so-called one-hot encoding with $p(y|x) = 1$ for $(x, y) \in \mathcal{S}$ and $p(y'|x) = 0$ for all $y' \neq y, \ (x, y) \in \mathcal{S}$.

Furthermore, the negative log-likelihood is directly related to the Kullback-Leibler (KL) divergence [72] between the data and the model distributions

$$\begin{aligned} \mathrm{KL}(p_{Y|X}||f_\theta) &= \mathbb{E}_{(x,y)\sim P_{X,Y}}[\log(\frac{p_{Y|X}(y|x)}{f_\theta(y|x)})] & (2.7) \\ &= -\mathbb{E}_{(x,y)\sim P_{X,Y}}[\log(f_\theta(y|x))] + \mathbb{E}_{(x,y)\sim P_{X,Y}}[\log(p_{Y|X}(y|x))] & (2.8) \\ &= \mathcal{L}(f_\theta) - \mathcal{L}(p_{Y|X}) \ , \end{aligned}$$

which satisfies $\mathrm{KL}(p_{Y|X}||f_\theta) \geq 0$ with $\mathrm{KL}(p_{Y|X}||f_\theta) = 0$ if and only if $p_{Y|X} = f_\theta$ [9], but is neither symmetric, nor does it satisfy the triangle equality [102]. With that, learning can be understood as minimization of the KL divergence between the true and the estimated distributions.

Assuming that the realizability assumption $p_{Y|X} \in \mathcal{H}$ is satisfied, a hypothesis space $\mathcal{H}$ is called probably approximately correct (PAC) learnable [127], if there exists a sample complexity function $m_\mathcal{H} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm A, which yields a function $f_{\hat{\theta}} = A(\mathcal{S})$ so that for all $\varepsilon, \delta \in (0, 1)$, $P_{X,Y}$ and $m \geq m_\mathcal{H}(\varepsilon, \delta)$, it holds that

$$\mathbb{P}_{(x,y)\sim P_{X,Y}}([\mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}(p_{Y|X})] > \varepsilon) \leq \delta \ . \tag{2.9}$$

Whenever the realizability assumption is not satisfied, a learner cannot eliminate the model error

$$\varepsilon_{\mathrm{model}} = \min_{f_\theta \in \mathcal{H}} \mathcal{L}(f_\theta) - \mathcal{L}(p_{Y|X}) \ . \tag{2.10}$$

Agnostic PAC learnability therefore only requires that the KL divergence converges to this model error, *i.e.*,

$$\mathbb{P}_{(x,y)\sim P_{X,Y}}([\mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}(p_{Y|X})] > \varepsilon + \varepsilon_{\mathrm{model}}) \leq \delta \ . \tag{2.11}$$

That is, a hypothesis space $\mathcal{H}$ is (agnostic) PAC learnable [127], if there exists a learning algorithm, which *probably* – with confidence $(1 - \delta)$ – yields an *approximately* – up to an error of ε $(+ \ \varepsilon_{\mathrm{model}})$ – correct function if the sample size m is sufficiently large.

Furthermore, a hypothesis space $\mathcal{H}$ is agnostically PAC learnable, whenever it has the uniform convergence property [127], *i.e.*, whenever there exists a sample complexity function $m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$, so that for all $\varepsilon, \delta \in (0,1)$, $P_{X,Y}$ and $m \geq m_{\mathcal{H}}^{UC}(\varepsilon, \delta)$, $\mathcal{S} \sim P_{X,Y}^{\otimes m}$ is ε-representative with a probability of at least $1-\delta$, or formally,

$$\mathbb{P}_{(x,y)\sim P_{X,Y}}(\{\exists\ f_\theta \in \mathcal{H} :\ |\mathcal{L}_{\mathcal{S}}(f_\theta) - \mathcal{L}(f_\theta)| > \varepsilon\}) \leq \delta\ . \tag{2.12}$$

2.1.2 Sources of Error and Uncertainty

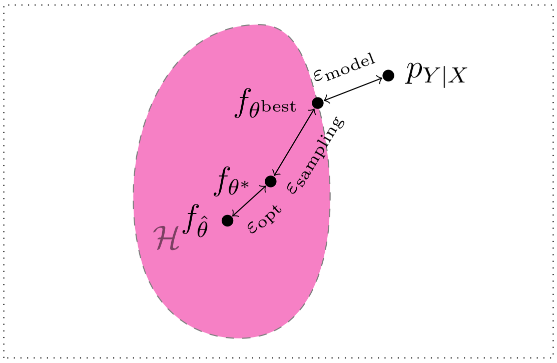

$\varepsilon_{\text{model}}$
$p_{Y|X}$
$f_{\theta^{\text{best}}}$
$\varepsilon_{\text{sampling}}$
f_{θ^*}
$\mathcal{H}$
$f_{\hat{\theta}}$
ε_{opt}

Figure 2.1: Hypothesis space $\mathcal{H}$ which does not include the true probability distribution $p_{Y|X}$, illustrating the sources of model, sampling and optimization errors. Here, $f_{\theta^{\text{best}}} = \operatorname{argmin}_{f_\theta \in \mathcal{H}} \mathcal{L}(f_\theta)$ denotes the best model in $\mathcal{H}$, $f_{\theta^*} = \operatorname{argmin}_{f_\theta \in \mathcal{H}} \mathcal{L}_{\mathcal{S}}(f_\theta)$ an ERM and $f_{\hat{\theta}} = A(\mathcal{S})$ the model obtained by an algorithm A.

The true error of a learned model $A(\mathcal{S}) = f_{\hat{\theta}}$ regarding the true conditional probability density function $p_{Y|X}$ can be expressed as the KL divergence $\mathrm{KL}(p_{Y|X}||f_{\hat{\theta}})$, which can be upper bounded by a telescopic sum decomposing into the *model*, *optimization* and *sampling error* [127], Figure 2.1. With $f_{\theta^*} \in \operatorname*{argmin}_{f_\theta \in \mathcal{H}} \mathcal{L}_{\mathcal{S}}(f_\theta)$, the error decomposition is stated as

$$\begin{aligned}
\mathrm{KL}(p_{Y|X}||f_{\hat{\theta}}) &= \mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}(p_{Y|X}) && \\
&\leq \inf_{f_\theta \in \mathcal{H}} \mathcal{L}(f_\theta) - \mathcal{L}(p_{Y|X}) && \text{model error} \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) && \text{optimization error} \\
&+ 2 \sup_{f_\theta \in \mathcal{H}} |\mathcal{L}(f_\theta) - \mathcal{L}_{\mathcal{S}}(f_\theta)| && \text{sampling error}
\end{aligned}$$

which is proved as follows[1].

[1] Private discussion with H. Gottschalk

Proof.

$$
\begin{aligned}
\mathrm{KL}(p_{Y|X}||f_{\hat{\theta}}) &= \mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}(p_{Y|X}) \\
&= \mathcal{L}(f_{\theta}) - \mathcal{L}(p_{Y|X}) && \text{for any } f_\theta \in \mathcal{H} \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta}) \\
&+ \mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\theta}) - \mathcal{L}(f_{\theta}) \\
&\leq \mathcal{L}(f_{\theta}) - \mathcal{L}(p_{Y|X}) \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) && \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) \leq \mathcal{L}_{\mathcal{S}}(f_{\theta}) \\
&+ \mathcal{L}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\theta}) - \mathcal{L}(f_{\theta}) \\
&\leq \mathcal{L}(f_{\theta}) - \mathcal{L}(p_{Y|X}) \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) \\
&+ 2 \sup_{f_{\theta'} \in \mathcal{H}} |\mathcal{L}(f_{\theta'}) - \mathcal{L}_{\mathcal{S}}(f_{\theta'})| \\
&\leq \inf_{f'_{\theta} \in \mathcal{H}} \mathcal{L}(f'_{\theta}) - \mathcal{L}(p_{Y|X}) && \inf_{f'_{\theta} \in \mathcal{H}} \mathcal{L}(f'_{\theta}) \leq \mathcal{L}(f_{\theta}) \\
&+ \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) \\
&+ 2 \sup_{f_{\theta} \in \mathcal{H}} |\mathcal{L}(f_{\theta}) - \mathcal{L}_{\mathcal{S}}(f_{\theta})|
\end{aligned}
$$

□

The model error

$$\varepsilon_{\text{model}} = \inf_{f_\theta \in \mathcal{H}} \mathcal{L}(f_\theta) - \mathcal{L}(p_{Y|X}) \tag{2.13}$$

arises from restricting the learner to a predefined hypothesis space $\mathcal{H}$, which was already discussed in the context of agnostic PAC learning, Equation 2.10. It can be reduced by enlarging the hypothesis space at the cost of increased complexity. Obviously, this error disappears whenever the realizability assumption is satisfied. In terms of neural networks, the model error can be controlled due to their universal approximation property.

Definition 2.1 (Universal Approximation [15])**.** Consider a function space $\mathcal{F}$, endowed with a distance $\mathrm{d} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$, and let $\mathcal{H}$ be a subset of $\mathcal{F}$ including functions which can be represented by a neural network. Then, $\mathcal{H}$ is a *universal approximator* for the space $\mathcal{F}$, if $\mathcal{H}$ is d-dense in $\mathcal{F}$, *i.e.*,

$$\forall f^* \in \mathcal{F},\ \forall \varepsilon > 0,\ \exists f \in \mathcal{H} :\ \mathrm{d}(f^*, f) < \varepsilon\ . \tag{2.14}$$

The optimization error

$$\varepsilon_{\text{opt}} = \mathcal{L}_{\mathcal{S}}(f_{\hat{\theta}}) - \mathcal{L}_{\mathcal{S}}(f_{\theta^*}) \tag{2.15}$$

occurs when the computation of the ERM is NP-hard and thus the learning algorithm A, *e.g.* a heuristic, only returns a proxy for f_{θ^*}. Consequently, it is reducible by employing a proper optimization algorithm such as *stochastic gradient descent*. Finally, the sampling error

$$\varepsilon_{\text{sampling}} = \sup_{f_\theta \in \mathcal{H}} |\mathcal{L}(f_\theta) - \mathcal{L}_{\mathcal{S}}(f_\theta)| \tag{2.16}$$

can be controlled if $\mathcal{H}$ has the uniform convergence property, Equation 2.12, stating that the sampling error vanishes for a sufficiently large sample size.

Instead of the general error inherent in a model, one is often interested in the *predictive uncertainty* for a concrete input $x \in \mathcal{X}$. It can be quantified *e.g.* by computing the *entropy* [102]

$$H(f_\theta(\cdot|x)) = \sum_{q \in \mathcal{Y}} f_\theta(q|x) \log(f_\theta(q|x)) \ , \tag{2.17}$$

which measures the variance of $f_\theta(\cdot|x)$ or the prediction confidence. Therefore, the entropy score is misleading for uncalibrated confidence. In [53], it has been shown that the class of neural networks with *rectified linear unit* (ReLU) activations produces predictions with arbitrarily high confindece for far away from the training data. Especially in saftey-critical applications such as automated driving, neural networks should *know when they don't know* [96].

Similar to the model error, the predictive uncertainty can also be decomposed into different sources, which are *epistemic* and *aleatoric* uncertainty [60]. Epistemic uncertainty includes the reducible part of the uncertainty. It is related to the model and the optimization error and thus, reducible by minimizing the respective errors. Aleatoric uncertainty corresponds to the irreducible part, explaining the variability in outcome due to randomness, caused by a non-deterministic dependency between $\mathcal{X}$ and $\mathcal{Y}$, *e.g.* when prediciting the outcome in coin flipping. That is, aleatoric uncertainty is tailored to the data, while epistemic uncertainty denotes the uncertainty about the choice of the model parameters θ, see Figure 2.2.

2.2 Deep Neural Networks

Neural networks are statistical models, which, in the context of supervised learning, approximate a probability density function $p_{Y|X}$ over Q classes by fitting a parameterized statistical model $f_\theta : \mathcal{X} \to (0,1)^Q$ to a labeled dataset $\mathcal{S} \sim P_{X,Y}^{\otimes m}$. A popular type of neural networks is the *feedforward neural network*, which is a chain of functions or layers that are connected by weight matrices. The input

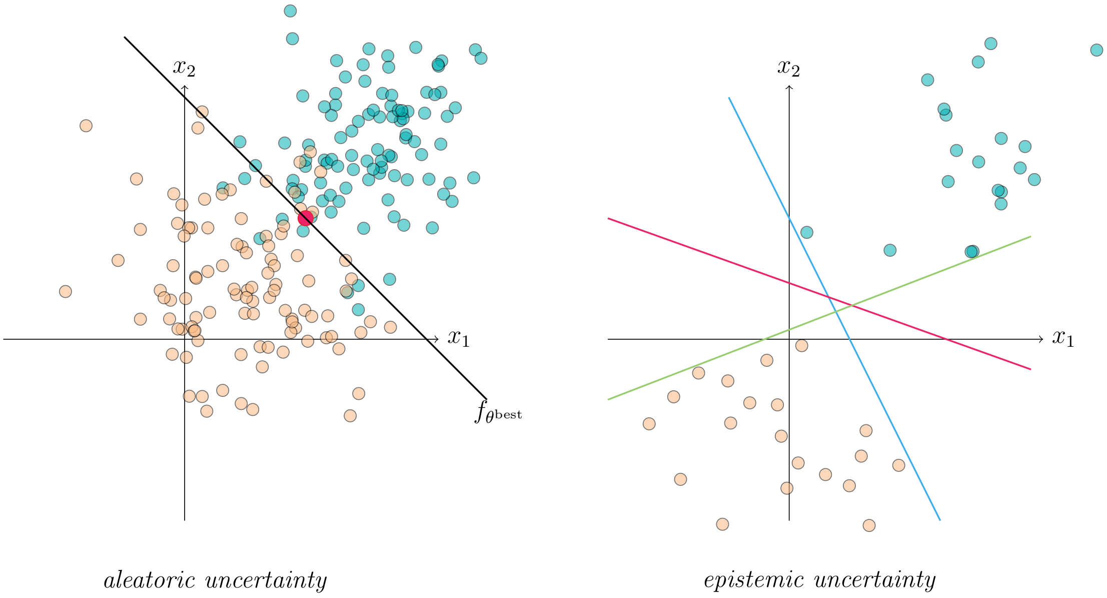


Figure 2.2: (*Left*): since the classes are not separable, the model exhibits high aleatoric uncertainty for the red exampel, although $p_{Y|X}$ is known. (*Right*): the small amount of data causes high epistemic uncertainty regarding the parameter choice.

flows from one to the next layer by matrix multiplications. In order to solve tasks with high complexity, the number of layers increases, meaning that a neural network becomes *deeper*. However, the capacity of a neural network is limited by computational constraints. Convolutional neural networks reduce the parameters of a neural network by replacing matrix multiplications with convolution operations. This chapter introduces feedforward and convolutional neural networks and explains how to optimize the parameters of neural networks.

2.2.1 Feedforward Neural Networks

Feedforward neural networks are a specific type of neural networks, which can be described by an acyclic or *feedforward* graph, *i.e.*, the information flows only in one direction. In its simplest form, a feedforward neural network consists of one *perceptron* neuron.

Definition 2.2 (Perceptron Neuron [121])**.** Let $x \in \mathcal{X}$, $\dim(\mathcal{X}) = n$ be an input vector. A perceptron neuron is a mapping $f_\theta : \mathcal{X} \to \{0, 1\}$ with

$$f_\theta(x) = \begin{cases} 1 & \omega^\intercal x + b > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.18}$$

where the parameters $\theta = \{\omega, b\}$ consist of a weight vector $\omega \in \mathbb{R}^n$ and a bias $b \in \mathbb{R}$.

The perceptron neuron is a deterministic model, capable of solving any binary classification problem for a linearly separable training dataset [9] by thresholding on the weighted sum of the input with threshold $-b$. In order to solve more complex tasks than binary classification, the perceptron neuron has been extended to the multilayer perceptron (MLP) [44] or multilayer neural network, which is a composition of multiple functions or *layers*. According to the universal approximation theorem, they can approximate any continuos function.

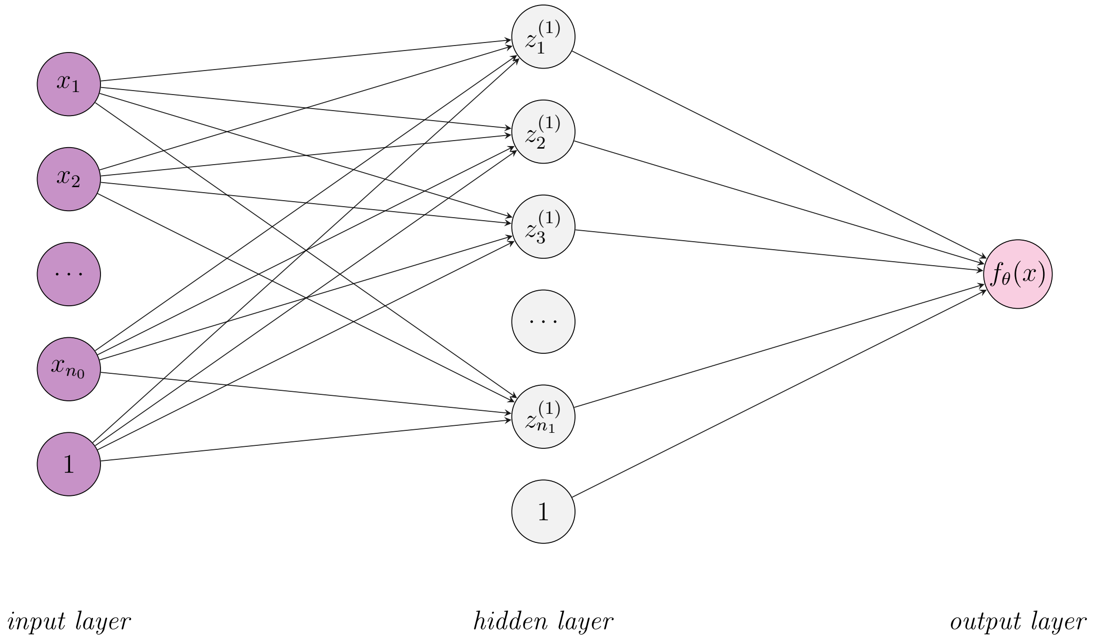


Figure 2.3: A multilayer neural network with one hidden layer and one output neuron. The weights and biases are omitted in this visualization.

Definition 2.3 (Multilayer Neural Network [44])**.** Let $x \in \mathcal{X}$ be an input vector and $f_{\theta^{(0)}}(x) = x$ the input layer. A multilayer neural network is a mapping $f_\theta : \mathcal{X} \to \mathbb{R}^Q, Q \in \mathbb{N}$, which is a composition of $L \in \mathbb{N}, L \geq 2$ layers

$$f_\theta(x) = (f_{\theta^{(L)}} \circ f_{\theta^{(L-1)}} \circ \ldots \circ f_{\theta^{(1)}})(x) \ , \tag{2.19}$$

where $f_{\theta^{(1)}}, \ldots, f_{\theta^{(L-1)}}$ are the hidden and $f_{\theta^{(L)}}$ the output layer. Let $n_l \in \mathbb{N}, l \in \{0, \ldots, L\}$ denote the number of neurons in layer $f_{\theta^{(l)}}$. The set of parameters $\theta = \{\theta^{(l)}\}_{l=1}^{L}$ with $\theta^{(l)} = \{\omega^{(l)}, b^{(l)}\}$ consists of weight matrices $\omega^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, linking each layer to the preceding layer, and bias vectors $b^{(l)} \in \mathbb{R}^{n_l}$. The output per layer is defined as

$$z^{(l)} = f_{\theta^{(l)}}(z^{(l-1)}) = \phi(\omega^{(l)} z^{(l-1)} + b^{(l)}) \ , \ l \in \{1, \ldots, L\}, \ z^{(0)} = x \ , \tag{2.20}$$

where $\phi : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ denotes an activation function.

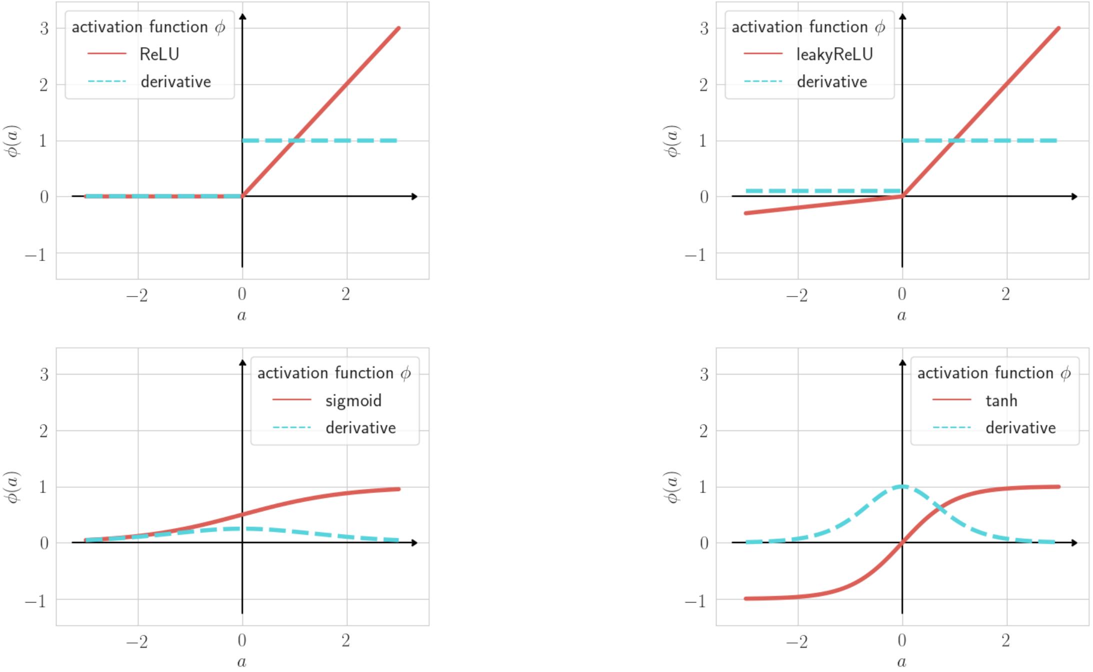


Figure 2.4: Graphs of the ReLU, leaky ReLU, sigmoid and tanh activation functions and their respective derivatives for an interval from −3 to 3.

A multilayer neural network consisting of one hidden layer and one output neuron is illustrated in Figure 2.3.

In order to obtain the universal approximation property, activation functions, *cf.* Figure 2.4, perform non-linear transformations to the input. One of the most widely used activation functions is the sigmoid or logistic activation function,

$$\phi_{\text{sigmoid}}(a) = \frac{1}{1+e^{-a}} \quad \in [0,1] \ , \tag{2.21}$$

which takes real values in $[0,1]$ and is thus commonly used to predict probabilities. It is differentiable and provides a smooth gradient, however, the derivative saturates for large positive or negative numbers, which causes vanishing gradients. The same holds for the tanh activation function

$$\phi_{\text{tanh}}(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}} \quad \in [-1,1] \ , \tag{2.22}$$

however, it is zero centered, which simplifies learning for the next layer. Thus, it is usually preferred to the sigmoid activation function. The rectified linear unit (ReLU) activation function

$$\phi_{\text{ReLU}}(a) = \max(0,a) \quad \in [0,\infty) \tag{2.23}$$

is the default activation function recommended for use with most MLPs. As it is a piece-wise linear function with two linear pieces, it preserves many of the properties which simplify the optimization with gradient-based methods and which make the model generalize well. Furthermore, it does not require floating point operations, allowing a fast computation. However, it causes the dying ReLU problem, since the gradient becomes zero on the negative half-axis. Consequently, the weights and biases of some neurons won't be updated, which can create "dead" neurons, *i.e.*, neurons which never get activated. Hence, the leaky ReLU

$$\phi_{\text{leakyReLU}}(a) = \max(0.1 \cdot a, a) \quad \in (-\infty, \infty) \ , \tag{2.24}$$

adds a small positive slope. Although not zero, gradients on the negative half-axis are still small values which makes learning time-consuming. Further modifications of the ReLU function are not discussed in this work.

For classification problems with $\mathcal{Y} = \{1, \dots, Q\}$, the values of the output neurons should represent estimates of the probabilities $p_{Y|X}(y|x)$ for an input $x \in \mathcal{X}$ to have class affiliation $y \in \mathcal{Y}$. Therefore, the softmax function

$$\sigma(z^{(L)})_i = \frac{e^{z_i^{(L)}}}{\sum_{j=1}^{Q} e^{z_j^{(L)}}} \quad \in [0,1], \ i = 1 \dots, Q \tag{2.25}$$

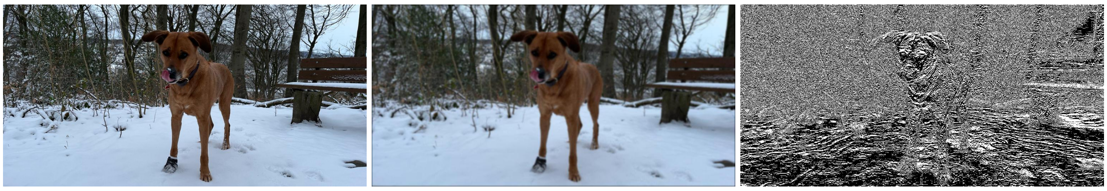
is applied in order to convert the output of the neural network into a probability distribution of q possible outcomes. The final class prediction for an input $x \in \mathcal{X}$ is then obtained by applying a decision rule, which is usually the *maximum a-posteriori principle* [9], yielding

$$\hat{y} = \operatorname*{argmax}_{q \in \mathcal{Y}} f_\theta(x)_q \ . \tag{2.26}$$

While multilayer neural networks perform well on many tasks where the input space $\mathcal{X}$ is low-dimensional, they are limited by the curse of dimensionality [5, 9].

2.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are feedforward neural networks with at least one convolutional layer, in which the general matrix multiplication is replaced by the convolution operation. Convolution layers reduce the amount of parameters by *parameter sharing*. Instead of a weight matrix, where each weight is applied to the input once, convolution layers consist of a set of parameters, called *kernel*, which slides over the input, using the same parameters multiple times. In contrast to fully connected neural networks, where each neuron is connected to every neuron in the next layer, CNNs have *sparse interactions*. Furthermore, they are capable of learning *translation invariances*. In particular in computer vision, where the model inputs are images, neural networks need to be invariant to translations in order to recognize instances of the same class, regardless of *e.g.* their size, rotation or position in the image. Convolution layers are usually followed by a pooling layer to reduce redundancy, and a final fully connected layer which maps the extracted features onto the output classes. Furthermore, normalization layers can be added to increase the stability of the CNN.



original image blurred image horizontal edges

Figure 2.5: Application of convolutions in image processing, using kernels for Gaussian image blurring and horizontal edge detection, respectively.

The idea of transforming an image by convolutions is also exploited in image processing, where fixed kernels instead of learnable parameters are employed to perform operations on an image, *e.g.* for edge detection, image sharpening or blurring. In Figure 2.5, the kernels for horizontal edge detection and Gaussian image blurring,

$$k_{\text{blur}} = \frac{1}{273}\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \quad \& \quad k_{\text{edge}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

are applied to each color channel of an RGB image separately, and the resulting feature maps are averaged or stacked, respectively.

Definition 2.4 (Convolution Operation [44])**.** In general, a convolution is an operation on two real-valued integrable functions $x : \mathbb{R} \to \mathbb{R}$ and $k : \mathbb{R} \to \mathbb{R}$, denoted by $*$ and defined as

$$(x * k)(i) = \int_{\mathbb{R}} x(j)k(i-j)\mathrm{d}j \ . \tag{2.27}$$

In machine learning applications, these functions correspond to the discrete input and kernel vectors. Therefore, we introduce the discrete convolution operation.

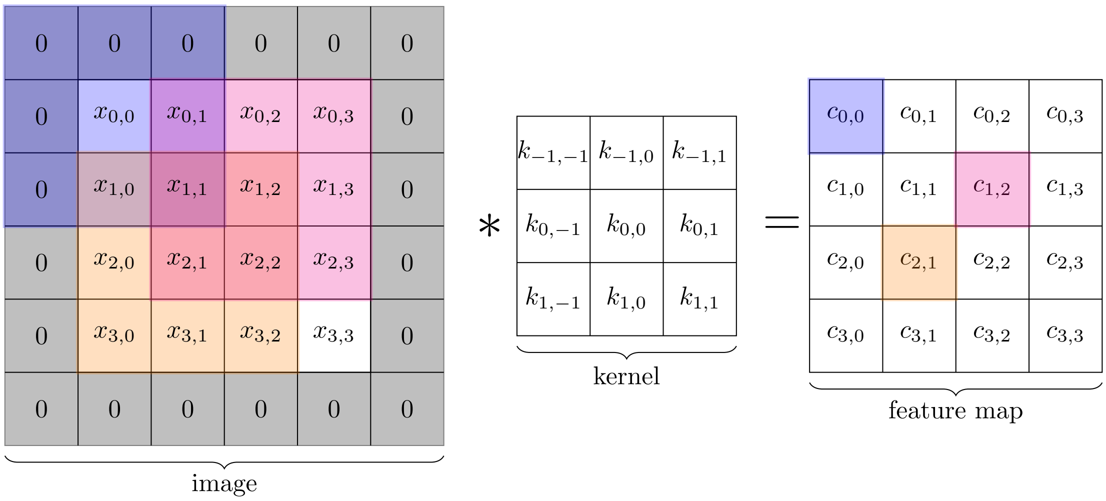

0 0 0 0 0 0
0 $x_{0,0}$ $x_{0,1}$ $x_{0,2}$ $x_{0,3}$ 0
0 $x_{1,0}$ $x_{1,1}$ $x_{1,2}$ $x_{1,3}$ 0
0 $x_{2,0}$ $x_{2,1}$ $x_{2,2}$ $x_{2,3}$ 0
0 $x_{3,0}$ $x_{3,1}$ $x_{3,2}$ $x_{3,3}$ 0
0 0 0 0 0 0
image
$*$
$k_{-1,-1}$ $k_{-1,0}$ $k_{-1,1}$
$k_{0,-1}$ $k_{0,0}$ $k_{0,1}$
$k_{1,-1}$ $k_{1,0}$ $k_{1,1}$
kernel
$=$
$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ $c_{0,3}$
$c_{1,0}$ $c_{1,1}$ $c_{1,2}$ $c_{1,3}$
$c_{2,0}$ $c_{2,1}$ $c_{2,2}$ $c_{2,3}$
$c_{3,0}$ $c_{3,1}$ $c_{3,2}$ $c_{3,3}$
feature map

Figure 2.6: Application of a 2D discrete convolution (or cross correlation) with zero padding.

Definition 2.5 (Discrete Convolution Operation [44])**.** Let $x = (x_i)_{i=1}^n \in \mathcal{X}$ be an input vector, $k = (k_i)_{i=-K}^K \in \mathbb{R}^{2K+1}$ a kernel. The discrete convolution operation is defined as

$$c_i = (x * k)_i = \sum_{j \in \mathcal{Z}} x_j k_{i-j} \ , i \in \{1, \dots, n\} \ , \tag{2.28}$$

where $x_i = 0 \ \forall i \notin \{1, \dots, n\}$, $k_i = 0 \ \forall i \notin \{-K, \dots, K\}$.

The discrete convolution can also be applied to two-dimensional input data, *e.g.* to a grayscale image.

Definition 2.6 (2D Discrete Convolution Operation [44])**.** Let $x = (x_{i,j})_{i,j=1}^{H,W} \in \mathcal{X}$ be a two-dimensional input, $k = (k_{i,j})_{i,j=-K}^K \in \mathbb{R}^{2K+1 \times 2K+1}$ a quadratic kernel. The two-dimensional discrete convolution operation is defined as

$$c_{i,j} = (x * k)_{i,j} = \sum_{r \in \mathcal{Z}} \sum_{s \in \mathcal{Z}} x_{r,s} k_{i-r,j-s} \ , i \in \{1, \dots, H\}, j \in \{1, \dots, W\} \ , \tag{2.29}$$

where $x_{i,j} = 0 \ \forall i \notin \{1, \dots, H\}, j \notin \{1, \dots, W\}$, $k_{i,j} = 0 \ \forall i, j \notin \{-K, \dots, K\}$. The discrete convolution operation can be extended to even higher dimensional data and for non-quadratic kernels.

Usually, kernels are chosen to be significantly smaller than the input data. Hence, summing over the kernel indices reduces the number of operations and is therefore computationally more efficient, which is why the cross-correlation function is usually implemented instead.

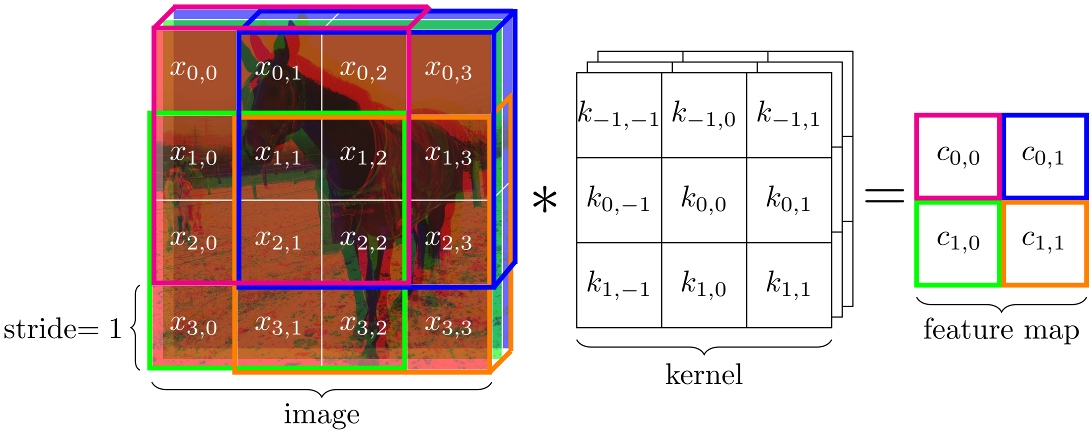


Figure 2.7: Application of a depth-wise convolution (or cross correlation) with three kernels to an RGB image. Each 3×3 kernel slides across one image channel with stride $\delta = 1$, producing three feature maps which are fused, for example by averaging.

Definition 2.7 (Cross-Correlation [44]). Let $x = (x_{i,j})_{i,j=1}^{H,W} \in \mathcal{X}$ be a two-dimensional input, $k = (k_{i,j})_{i,j=-K}^{K} \in \mathbb{R}^{2K+1 \times 2K+1}$ a quadratic kernel. The cross-correlation is defined as

$$c_{i,j} = (x * k)_{i,j} = \sum_{r \in \mathcal{Z}} \sum_{s \in \mathcal{Z}} x_{i+r,j+s} k_{r,s} \\ = \sum_{r=-K}^{K} \sum_{s=-K}^{K} x_{i+r,j+s} k_{r,s} \tag{2.30}$$

where $x_{i,j} = 0 \; \forall i \notin \{1, \ldots, H\}, j \notin \{1, \ldots, W\}$.

The cross-correlation is the same as the convolution with the kernel being flipped on both axes. Adding zero rows and columns around the input matrix x is referred to as *zero padding*, allowing the computation of a feature map $c \in \mathbb{R}^{H \times W}$ of the same size as the input and the retainment of boundary information. The computation of the feature map is visualized in Figure 2.6 as a kernel, which slides across an image. The step size $\delta \in \mathbb{N}$, by which the kernel moves in each convolution, called *stride*, can be incorporated into the cross-correlation as follows.

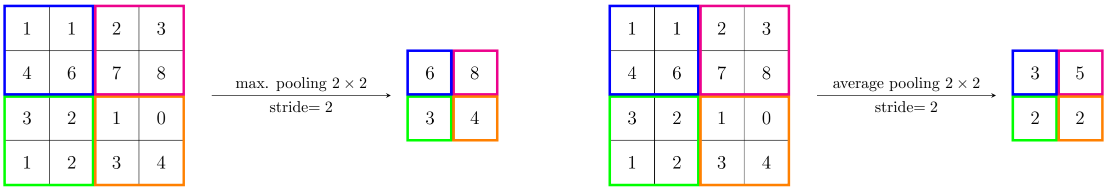
Definition 2.8 (Cross-Correlation with Stride [44]). Let $x = (x_{i,j})_{i,j=1}^{m,n} \in \mathcal{X}$ be a two-dimensional input, $k = (k_{i,j})_{i,j=-K}^{K} \in \mathbb{R}^{2K+1 \times 2K+1}$ a quadratic kernel and $\delta \in \mathbb{N}$ the stride. The cross-correlation with stride is defined as

$$c_{i,j} = (x * k)_{i,j} = \sum_{r \in \mathcal{Z}} \sum_{s \in \mathcal{Z}} x_{i+r,j+s} k_{r,s} \\ = \sum_{r=-K}^{K} \sum_{s=-K}^{K} x_{\delta i+r,\delta j+s} k_{r,s} \tag{2.31}$$

where $x_{i,j} = 0 \ \forall i \notin \{1, \dots, m\}, j \notin \{1, \dots, n\}$.

Commonly, a convolution layer consists of multiple kernels to learn different visual features. Besides the standard point-wise convolution, different types of convolution layers like the transposed or the dilated convolution have been devised. For three dimensional inputs as in Figure 2.7, depth-wise convolutions apply separated 2D kernels to each color or depth channel of the input.

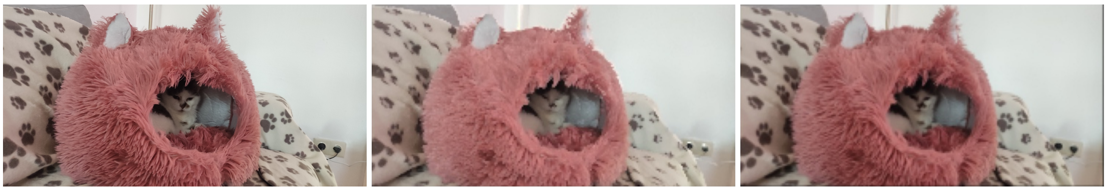
Pooling functions are applied in neural networks to replace the output of the preceding layers by a summary statistic of the surrounding values like the maximal or the average value as visualized in Figure 2.8 and Figure 2.9. Since pooling consolidates information of a neighborhood, it can be applied to reduce the dimension of an input feature map by using non-overlapping pooling regions [44], *i.e.*, using a stride $\delta = p, \ p \in \mathbb{N}$ for a $p \times p$ pooling region *cf.* Figure 2.8.



maximum pooling

average pooling

Figure 2.8: Example for applying maximum and average pooling, respectively.



original image

max pooling

average pooling

Figure 2.9: Illustration of maximum and average pooling applied to an RGB image, using 9 × 9 non-overlapping pooling regions.

Definition 2.9 (Maximum and Average Pooling)**.** Let $x = (x_i)_{i,j=1}^{H,W} \in \mathcal{X}$ be an input, $p \times p, p \in \mathbb{N}$ the size of the pooling region and $\delta \in \mathbb{N}$ the stride. Maximum pooling is defined as

$$c_{i,j} = \max_{1 \leq r,s \leq p} x_{\delta(i-1)+r, \delta(j-1)+s} \ . \tag{2.32}$$

Average pooling is defined as

$$c_{i,j} = \frac{1}{p^2} \sum_{r=1}^{p} \sum_{s=1}^{p} x_{\delta(i-1)+r, \delta(j-1)+s} \ . \tag{2.33}$$

A CNN can be divided into two main parts, which are feature extraction and classification. The features are extracted by the convolution and pooling layers. These are then mapped to the output classes by fully connected layers, where each neuron is connected to every neuron in the next layer by a weight vector, to learn non-linear combinations of the features.

Finally, batch normalization layer [61] are usually included in CNNs to adaptively reparameterize DNNs during training. They are applied to the input and hidden layers of a DNN in order to standardize the layer inputs.

Definition 2.10 (Batch Normalization [44])**.** Let $A^{(l)} = (a_i^{(l)})_{i=1}^{m_{\text{batch}}} \in \mathbb{R}^{m_{\text{batch}} \times n_l}$, $l \in \{0, \dots, L-1\}$ denote the batch activations, where $a^{(l)} = \omega^{(l)} z^{(l-1)} + b^{(l)}$. Then, the mean and standard deviation are computed by

$$\hat{\mu} = \frac{1}{m_{\text{batch}}} \sum_{i=1}^{m_{\text{batch}}} A_{i,\bullet}^{(l)} \tag{2.34}$$

and

$$\hat{\sigma} = \sqrt{\zeta + \frac{1}{m_{\text{batch}}} \sum_{i=1}^{m_{\text{batch}}} (A^{(l)} - \hat{\mu})_i^2}\ , \tag{2.35}$$

respectively, where ζ is a very small number to ensures that $\hat{\sigma} > 0$. With that, the activations are replaced by

$$\bar{A}^{(l)} = \frac{A^{(l)} - \hat{\mu}}{\hat{\sigma}}\ . \tag{2.36}$$

During training, the mean and standard deviation are computed by back propagating through the operations in Equation 2.34 and Equation 2.35, while at test time, they are replaced by running averages which where collected during the training. In convolution layers, the same mean and standard deviation is applied to every spatial location within the feature map. Usually, two parameters $\gamma, \beta \in \mathbb{R}$ are included in Equation 2.36, replacing the activations $A^{(l)}$ by $\gamma \bar{A}^{(l)} + \beta, l \in \{0, \dots, L-1\}$. These parameters allow the neurons to have any mean and standard deviation, which are not determined by complex interactions with other layers, but simply by the learnable parameters γ and β.

2.2.3 Training of Neural Networks

The estimated class-probabilities for some input $x \in \mathcal{X}$ are obtained by *propagating* the input forwards through the neural network. During the training process of a neural network, a loss function is computed based on the final output. In order to update the parameters θ of all layers, the information flows backwards from the output to the input layer. *Backpropagation* [31] is an algorithm, which computes the gradient of the loss function with respect to all parameters of the neural network, which are then used by a *gradient descent* optimization algorithm.

Backpropagation A multilayer neural network is defined as a composition of functions

$$f_\theta(x) = (f_{\theta^{(L)}} \circ f_{\theta^{(L-1)}} \circ \ldots \circ f_{\theta^{(1)}})(x) \ ,$$

cf. Equation 2.19, with parameters $\theta^{(l)} = \{\omega^{(l)}, b^{(l)}\}$ for $l = 1, \ldots, L$. Using the *chain rule*, the gradient of a loss function $\mathcal{L}_\mathcal{S}$ with respect to the model parameters $\theta^{(l)}$ is computed by

$$\begin{aligned} \nabla_{\theta^{(l)}} \mathcal{L}_\mathcal{S} \circ f_\theta &= \nabla_{\theta_l} (\mathcal{L}_\mathcal{S} \circ f_{\theta^{(L)}} \circ \ldots \circ f_{\theta^{(l)}}) \\ &= \nabla_{f_{\theta^{(L)}}} \mathcal{L}_\mathcal{S} J_{f_{\theta^{(L-1)}}} f_{\theta^{(L)}} \ldots J_{f_{\theta^{(l+1)}}} f_{\theta^{(l+2)}} J_{f_{\theta^{(l)}}} f_{\theta^{(l+1)}} J_{\theta^{(l)}} f_{\theta^{(l)}} \ , \end{aligned} \tag{2.37}$$

where ∇ denotes the gradient, J the Jacobian of a function. Since the gradient of a deeper layer

$$\nabla_{\theta^{(l+1)}} \mathcal{L}_\mathcal{S} \circ f_\theta = \nabla_{f_{\theta^{(L)}}} \mathcal{L}_\mathcal{S} J_{f_{\theta^{(L-1)}}} f_{\theta^{(L)}} \ldots J_{f_{\theta^{(l+1)}}} f_{\theta^{(l+2)}} J_{\theta^{(l+1)}} f_{\theta^{(l+1)}}$$

shares the most computations with $\nabla_{\theta^{(l)}} \mathcal{L}_\mathcal{S} \circ f_\theta$ and using the associativity of matrix multiplication, backpropagation recursively computes the gradients, starting from the last layer and storing only vectors from previous iterations instead of big matrices. In this way, backpropagation allows efficient computation of gradients, which is particularly useful when training deep neural networks with up to several million parameters.

Definition 2.11 (Local Gradient [127])**.** Let $f_\theta : \mathcal{X} \to \mathbb{R}^q$, $q \in \mathbb{N}$ be a neural network as defined in Equation 2.19 with $L \in \mathbb{N}$ layers and let $\mathcal{L}_\mathcal{S} : \mathbb{R}^q \to [0, \infty)$ denote a loss function. The local gradient of the l-th layer is defined as

$$\delta^{(l)} = \nabla_{a^{(l)}} \mathcal{L}_\mathcal{S} \circ f_\theta \ \forall l = 1, \ldots, L \ , \tag{2.38}$$

where $a^{(l)} = (\omega^{(l)} z^{(l-1)} + b^{(l)})$.

Definition 2.12 (Backpropagation [31, 127])**.** Let $f_\theta : \mathcal{X} \to \mathbb{R}^q$, $q \in \mathbb{N}$ be a neural network as defined in Equation 2.19 with $L \in \mathbb{N}$ layers and activation functions $\phi^{(l)}, l = 1, \ldots, L$. The local gradients regarding a loss function $\mathcal{L}_\mathcal{S} : \mathbb{R}^q \to [0, \infty)$ are obtained by

$$\delta^{(L)} = \operatorname{diag}({\phi'}^{(L)}(a^{(L)})) \nabla_{f_\theta} \mathcal{L}_\mathcal{S} \tag{2.39}$$

for the last layer, and for the previous layers by backpropagation

$$\delta^{(l)} = \operatorname{diag}({\phi'}^{(l)}(a^{(l)})) \omega^{\intercal(l+1)} \delta^{(l+1)} \ l = 1, \ldots, L-1 \ . \tag{2.40}$$

The gradient of $\mathcal{L}_{\mathcal{S}}$ with respect to the weights $\omega^{(l)}$ in the l-th layer is obtained by

$$\nabla_{\omega^{(l)}_{\bullet,i}} \mathcal{L}_{\mathcal{S}} \circ f_\theta = z_j^{(l-1)} \delta^{(l)} \ \forall i = 1, \ldots, n_l \ l = 1, \ldots, L \tag{2.41}$$

and by

$$\nabla_{b^{(l)}} \mathcal{L}_{\mathcal{S}} \circ f_\theta = \delta^{(l)} \ l = 1, \ldots, L \tag{2.42}$$

with respect to the biases.

Optimizer Since backpropagation allows to efficiently compute the gradients of a loss function with respect to the parameters in a neural network, these can be optimized by gradient descent algorithms, which iteratively update the model parameters θ to

$$\theta(t+1) = \theta(t) - \eta \cdot \nabla_\theta (\mathcal{L}_{\mathcal{S}} \circ f_\theta)_{|\theta=\theta(t)}, \quad t \in \mathbb{N} \ , \tag{2.43}$$

where $\theta(t)$ denotes the parameters after t iterations.

```
Input: learning rate $\eta$, initial parameters $\theta(0)$, training set $\mathcal{S}$, loss
       function $\ell$
$t = 0$
while stopping criterion not met do
    sample a batch $\{(x_1, y_1), \ldots, (x_{\bar{m}}, y_{\bar{m}})\} \subset \mathcal{S}$
    compute gradient estimate as $\hat{g} := \frac{1}{\bar{m}} \nabla_\theta \big( \sum_{i=1}^{\bar{m}} \ell(f_\theta(x_i), y_i) \big)_{|\theta=\theta(t)}$
    update parameters: $\theta(t+1) = \theta(t) - \eta \cdot \hat{g}$
    $t = t + 1$
$t^* = t$
Output: optimized parameters $\hat{\theta} = \theta(t^*)$
```

Algorithm 2.1: Stochastic gradient descent [44, 118].

The most common optimization algorithms for neural networks are the *stochastic gradient descent* (SGD) [118] and the *Adam* [66] optimizers. In each training step, the model receives a batch $\{(x_i, y_i)\}_{i=1}^{\bar{m}} \subseteq \mathcal{S}$. It can be shown that the average gradient on this batch is an unbiased estimate of the gradient [44]. The SGD algorithm updates the parameters of a neural network according to Equation 2.43, but replacing the gradient by an estimate as shown in Algorithm 2.1.

```
Input: learning rate $\eta$, initial parameters $\theta(0)$, initial velocity $\nu(0)$,
       training set $\mathcal{S}$, loss function $\ell$, weight decay $\alpha$
$t = 0$
while stopping criterion not met do
    sample a batch $\{(x_1, y_1), \dots, (x_{\bar{m}}, y_{\bar{m}})\} \subset \mathcal{S}$
    compute gradient estimate as $\hat{g} := \frac{1}{\bar{m}} \nabla_\theta \big( \sum_{i=1}^{\bar{m}} \ell(f_\theta(x_i), y_i) \big)_{|\theta=\theta(t)}$
    update velocity: $\nu(t+1) = \alpha\nu(t) - \eta \cdot \hat{g}$
    update parameters: $\theta(t+1) = \theta(t) + \nu(t+1)$
    $t = t + 1$
$t^* = t$
Output: optimized parameters $\hat{\theta} = \theta(t^*)$
```

Algorithm 2.2: Stochastic gradient descent with momentum [44, 114].

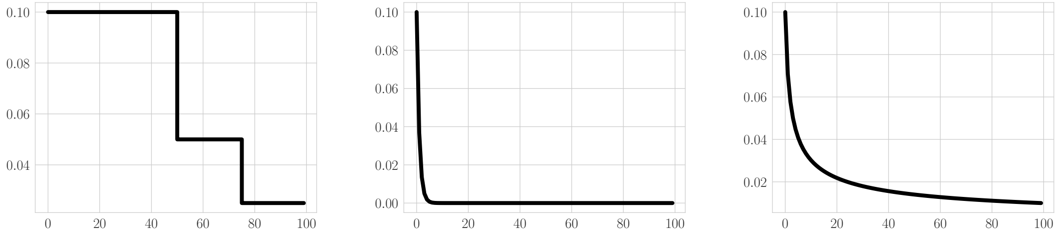
Usually, the learning rate is not fixed throughout the training, but gradually decreases by the application of learning rate schedulers. Furthermore, learning can be accelerated by the method of momentum. This algorithm considers an exponentially decreasing moving average of past gradients and continues to move in their direction. It changes the update rule in Algorithm 2.1 by adding a velocity ν, which denotes the direction and speed at which the parameters θ move through the parameter space Θ. Further, a weight decay hyperparameter $\alpha \in [0, 1)$ adjusts the speed at which the effects of previous gradients diminish. The SGD algorithm with momentum is provided in Algorithm 2.2.

```
Input: learning rate $\eta$, initial parameters $\theta(0)$, training set $\mathcal{S}$, loss
       function $\ell$, weight decay rates $\beta_1, \beta_2 \in [0, 1)$
$t = 0$, $\mu(0) = 0$, $\nu(0) = 0$
while stopping criterion not met do
    sample a batch $\{(x_1, y_1), \dots, (x_{\bar{m}}, y_{\bar{m}})\} \subset \mathcal{S}$
    compute gradient estimate as $\hat{g} := \frac{1}{\bar{m}} \nabla_\theta \big( \sum_{i=1}^{\bar{m}} \ell(f_\theta(x_i), y_i) \big)_{|\theta=\theta(t)}$
    update first moment estimate: $\mu(t+1) = \beta_1 \mu(t) + (1 - \beta_1)\hat{g}$
    update second moment estimate: $\nu(t+1) = \beta_2 \nu(t) + (1 - \beta_2)\hat{g}^2$
    compute bias-corrected first moment estimate:
    $\hat{\mu}(t+1) = \mu(t+1)/(1 - \beta_1^{t+1})$
    compute bias-corrected second moment estimate:
    $\hat{\nu}(t+1) = \nu(t+1)/(1 - \beta_2^{t+1})$
    update parameters: $\theta(t+1) = \theta(t) - \eta \cdot \hat{\mu}(t+1)/(\sqrt{\hat{\nu}(t+1)} + \zeta)$
    $t = t + 1$
$t^* = t$
Output: optimized parameters $\hat{\theta} = \theta(t^*)$
```

Algorithm 2.3: Adam optimizer [66].

The Adam [66] optimizer computes individual adaptive learning rates for different parameters from estimates of the gradients' first and second moments, which we denote as μ and ν, respectively. The pseudo-code is provided in Algorithm 2.3. The parameter ζ prevents division by zero and is usually very small, *e.g.* $\zeta = 10^{-8}$.

Scheduler While the Adam optimizer automatically adapts the learning rate η, the SGD algorithm uses a constant learning rate, which needs to be carefully chosen such that the algorithm converges to a local optimum. Instead of an exhaustive grid search to fit η for each training, a learning rate scheduler can be employed to decrease η over time. If the scheduler satisfies the Robbins-Monro [118] conditions



step decay exponential decay polynomial decay

Figure 2.10: Examples of some common learning rate schedules.

$$\eta_t \to 0, t \to \infty, \qquad \frac{\sum_{t=1}^{\infty} \eta_t^2}{\sum_{t=1}^{\infty} \eta_t} \to 0 \tag{2.44}$$

the SGD theoretically converges to a local optimum [102]. Common examples of learning rate schedulers are illustrated in Figure 2.10.

Definition 2.13 (Learning Rate Scheduler [102])**.** At given milestones t_i with $i \in \mathbb{N}$, the *piece-wise constant scheduler* adjusts the learning rate to specified values $\eta_i \in (0,1)$. Then, the learning rate in iteration $t \in \mathbb{N}$ is obtained by

$$\eta_t = \eta_i : \; t \in [t_i, t_{i+1}) \; . \tag{2.45}$$

Using a *step decay* with reduction factor $\gamma \in (0,1)$, the learning rate is adjusted to values

$$\eta_i = \eta_0 \cdot \gamma^i \; . \tag{2.46}$$

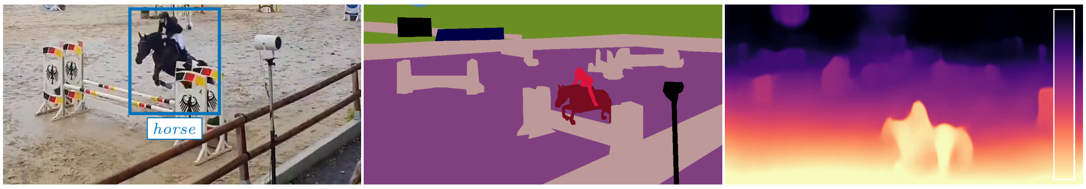
Alternatives to piece-wise constant schedulers are the *exponential decay*

$$\eta_t = \eta_0 e^{-\lambda t} \; , \lambda \in \mathbb{R} \; , \tag{2.47}$$

or the *polynomial decay*

$$\eta_t = \eta_0 (\beta t + 1)^{-\alpha} \; , \alpha, \beta \in \mathbb{R} \; . \tag{2.48}$$

2.3 Deep Learning for Computer Vision



object detection semantic segmentation depth estimation

Figure 2.11: Visualization of different computer vision tasks for three consecutive frames of a video, including object detection and classification (*left*), semantic segmentation (*middle*) and depth estimation (*right*).

Computer vision encompasses all approaches which enable machines to gain a high level of understanding from image and video data. In computer vision, understanding means transforming visual data into numerical or symbolic information. Computer vision tasks include *e.g. object classification*, *detection* and *segmentation*, *video tracking*, *motion* or *depth estimation*. Some of these applications are visualized in Figure 2.11. Roughly speaking, object detection is the task of classifying and localizing objects which belong to an underlying set of semantic classes by tranforming an image into a set of bounding box coordinates and class-probabilities. Semantic segmentation and depth estimation models provide class-probability or depth estimates for each image pixel, respectively. While this work focusses on image classification and in particular semantic segmentation tasks, open world applications can leverage further information *e.g.* using depth or tracking information to improve anomaly detection or retrieval results. In this section, we introduce image classification and semantic segmentation, including some popular CNN architectures and the main evaluation metrics. Both tasks operate on an image input space $\mathcal{X} = [0, 255]^{H\times W\times C}$ and yield class-probability estimates over classes $\mathcal{Y} = \{1, \dots, Q\}$.

2.3.1 Image Classification

An image classification model with softmax activation

$$f_\theta : \mathcal{X} \to (0, 1)^Q \tag{2.49}$$

estimates the class-probabilities that an image $x \in \mathcal{X}$ has class affiliation $y \in \mathcal{Y}$. The input images are usually tailored to one or more objects stemming from the same class. Consequently, image classification is not suitable for understanding entire scenes with multiple classes, nor for object localization. In the context

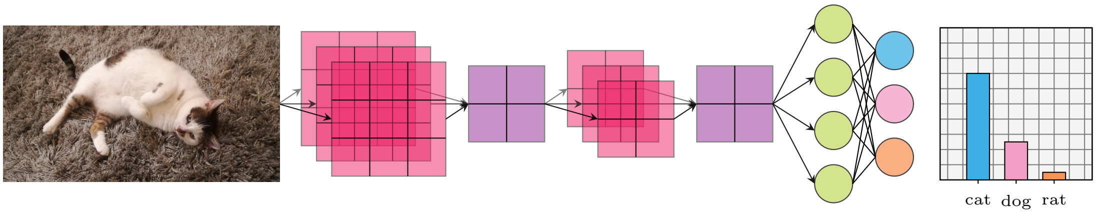


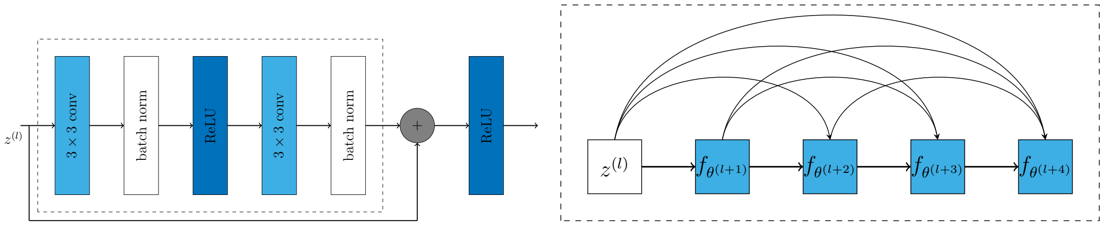
Figure 2.12: A shallow image classification CNN with underlying classes $0 \mathrel{\widehat{=}} \mathit{cat}$, $1 \mathrel{\widehat{=}} \mathit{dog}$ and $2 \mathrel{\widehat{=}} \mathit{rat}$. The model consists of two convolution layers *(pink)*, each followed by a pooling layer *(purple)*, a flattening *(green)* and a fully connected layer with three neurons, yielding class-probability estimates for an image of a cat.

of this work, image classification is mainly used to identify anomaly clusters by tailoring images to single anomalies.

In Figure 2.12, a shallow CNN produces class-probability estimates over $\mathcal{Y} = \{0, 1, 2\} \mathrel{\widehat{=}} \{\mathit{cat}, \mathit{dog}, \mathit{rat}\}$ for an image of a cat. The final prediction following Equation 2.26 is obtained by

$$\hat{y} = \operatorname*{argmax}_{q \in \{0,1,2\}} f_\theta(x)_q = 0 \mathrel{\widehat{=}} \mathit{cat} \ .$$

The LeNet [74] is one of the earliest CNNs, possessing convolution, pooling and fully connected layers as introduced in Subsection 2.2.2. It is capable of solving simple image classification tasks such as the classification of handwritten digits from the MNIST [75]. For tasks with an increased complexity, more sophisticated CNN architectures like ResNet [51] or DenseNet [59] have been developed.



ResNet block

DenseNet block

Figure 2.13: Building blocks of the CNN architectures ResNet and DenseNet.

The ResNet architecture has been developed in 2015, extending the simple CNN structure by residual blocks, *cf.* Figure 2.13 (*left*).

Definition 2.14 (Residual Block [102])**.** Let $\mathcal{F}$ be a block of stacked layers $f_{\theta^{(l)}} \circ \ldots \circ f_{\theta^{(l+s)}}$, $s \in \mathbb{N}$, where the input flows straight forward through the layers, yielding $z^{(l+s)} = \mathcal{F}(z^{(l)})$. Additionally, a skip connection adds the input $z^{(l)}$ to the output $z^{(l+s)}$, yielding

$$z^{(l+s)} = \phi(z^{(l)} + \mathcal{F}(z^{(l)})) \ , \tag{2.50}$$

hence, the model only needs to learn the residual between the input and output of a residual block.

Using residual blocks allows us to train very deep models, since the gradient can flow directly from the output to the previous layers via the skip connections [102].

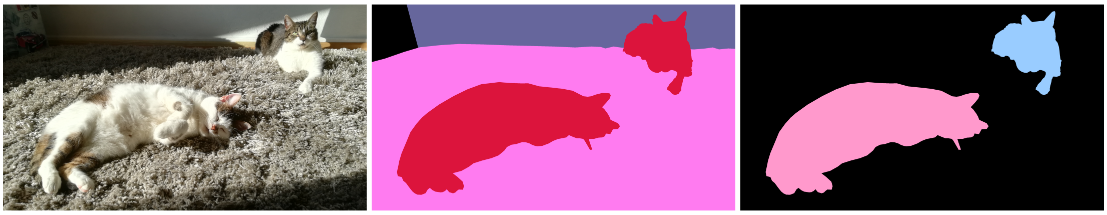
Instead of employing addition, the DenseNet (2016) uses skip connections to concatenate the in- and output of each layer, *cf.* Figure 2.13 (*right*).

Definition 2.15 (DenseNet Block [102])**.** Let $\mathcal{F}$ be a block of stacked layers $f_{\theta^{(l+1)}} \circ \ldots \circ f_{\theta^{(l+s)}}, s \in \mathbb{N}$, and let all layers be connected by a skip connection. The output of a DenseNet block is obtained by

$$z^{(l+s)} = [z^{(l)}, f_{\theta^{(l+1)}}(z^{(l)}), f_{\theta^{(l+2)}}(z^{(l)}, f_{\theta^{(l+1)}}(z^{(l)})), \ldots] \ . \tag{2.51}$$

Due to the dense connectivity, the output layer has access to all previous features, which can enhance the performance of a CNN at the expense of increased computational cost.

2.3.2 Semantic Segmentation



image

semantic segmentation

instance segmentation

Figure 2.14: Semantic segmentation is the pixel-wise classification into semantic classes, while instance segmentation is the pixel-wise classification into instances of object classes like *cat* and a background class, including *e.g. wall* or *ground*.

Instead of estimating the probability that an entire image has one class affiliation, a semantic segmentation network yields class-probabilities per pixel. Besides localizing and classifying objects in an image, semantic segmentation also considers background classes, provides context and shape information. However, it does not distinguish between instances of the same class. Instance segmentation allows the pixel-wise classification of object instances instead of classes while treating any non-object classes such as *sky, road, building* or *nature* as background. An example for both, semantic and instance segmentation, is provided in Figure 2.14. A combination of semantic and instance segmentation to enrich the instance labels with background classes is referred to as panoptic segmentation [67].

Let the pixel positions of the images $x \in \mathcal{X}$ be denoted as

$$z \in \mathcal{Z} = \{(h, w) : \ h \in \{0, \dots, H-1\}, w \in \{0, \dots, W-1\}\} \ ,$$

hence, a pixel of $x \in \mathcal{X}$ is defined as $x_z, \ z \in \mathcal{Z}$. A semantic segmentation model with softmax activation

$$f_\theta : \mathcal{X} \to (0,1)^{H \times W \times Q} \tag{2.52}$$

estimates class-probabilities for each pixel over the classes $\mathcal{Y} = \{1, \dots, Q\}$. Again, following Equation 2.26, the final class prediction at pixel position z is obtained by

$$\hat{y}_z = \operatorname*{argmax}_{q \in \mathcal{Y}} f(x)_{z,q} \ \forall z \in \mathcal{Z} \ . \tag{2.53}$$

Since semantic segmentation is far more complex than image classification, more sophisticated model architectures have been developed.

A popular architecture for semantic segmentation neural networks is the encoder-decoder architecture. The encoder reduces the dimension of an input into a bottleneck using standard convolution computed by Equation 2.30. This bottleneck captures high-level properties of the input images, which are then mapped back to the input resolution by the decoder. Adding skip connections from the input to the output layer mitigates the information loss in the bottleneck.

To upsample the feature map, the decoder employs transposed convolutions. These are also known as deconvolutions, however, this is not appropriate, since deconvolutions remove the effect of a convolution to reconstruct the original input. Transposed convolutions are easier to understand by transforming the standard convolution into a matrix vector multiplication first. To this end, consider the following example.

Example. Let $x \in \mathbb{R}^{4 \times 4}$ denote an input matrix, which can be flattened into a vector $\tilde{x} = (x_{0,\bullet}, x_{1,\bullet}, x_{2,\bullet}, x_{3,\bullet})^\intercal$, $k \in \mathbb{R}^{3 \times 3}$ a kernel. Then, a convolution without padding and stride $\delta = 1$ yields a feature map $c \in \mathbb{R}^{2\times 2}$, where

$$
\begin{aligned}
c_{0,0} &= k_{-1,-1}x_{0,0} + k_{-1,0}x_{0,1} + k_{-1,1}x_{0,2} \\
&\quad + k_{0,-1}x_{1,0} + k_{0,0}x_{1,1} + k_{0,1}x_{1,2} \\
&\quad + k_{1,-1}x_{2,0} + k_{1,0}x_{2,1} + k_{1,1}x_{2,2} \\
c_{0,1} &= k_{-1,-1}x_{0,1} + k_{-1,0}x_{0,2} + k_{-1,1}x_{0,3} \\
&\quad + k_{0,-1}x_{1,1} + k_{0,0}x_{1,2} + k_{0,1}x_{1,3} \\
&\quad + k_{1,-1}x_{2,1} + k_{1,0}x_{2,2} + k_{1,1}x_{2,3} \\
c_{1,0} &= k_{-1,-1}x_{1,0} + k_{-1,0}x_{1,1} + k_{-1,1}x_{1,2} \\
&\quad + k_{0,-1}x_{2,0} + k_{0,0}x_{2,1} + k_{0,1}x_{2,2} \\
&\quad + k_{1,-1}x_{3,0} + k_{1,0}x_{3,1} + k_{1,1}x_{3,2} \\
c_{1,1} &= k_{-1,-1}x_{1,1} + k_{-1,0}x_{1,2} + k_{-1,1}x_{1,3} \\
&\quad + k_{0,-1}x_{2,1} + k_{0,0}x_{2,2} + k_{0,1}x_{2,3} \\
&\quad + k_{1,-1}x_{3,1} + k_{1,0}x_{3,2} + k_{1,1}x_{3,3} \ .
\end{aligned}
$$

This is equivalent to compute

$$(c_{0,\bullet}, c_{1,\bullet}) := \tilde{k}\tilde{x}$$

with

$$
\tilde{k} = \begin{pmatrix}
k_{-1,\bullet} & 0 & k_{0,\bullet} & 0 & k_{1,\bullet} & 0 & & \\
0 & k_{-1,\bullet} & 0 & k_{0,\bullet} & 0 & k_{1,\bullet} & & \text{\Large 0} \\
 & & k_{-1,\bullet} & 0 & k_{0,\bullet} & 0 & k_{1,\bullet} & 0 \\
\text{\Large 0} & & 0 & k_{-1,\bullet} & 0 & k_{0,\bullet} & 0 & k_{1,\bullet}
\end{pmatrix} ,
$$

followed by reshaping into $c = \begin{pmatrix} c_{0,\bullet} \\ c_{1,\bullet} \end{pmatrix}$.

Definition 2.16 (Transposed Convolution[102])**.** Let $c \in \mathbb{R}^{r\times s}$, $r, s \in \mathbb{N}$ be a feature map, $k \in \mathbb{R}^{2K+1\times 2K+1}$, $K \in \mathbb{N}$ a kernel, and $m \times n$ the desired output shape. By flattening, we obtain $(c_{0,\bullet}, \dots, c_{r,\bullet}) \in \mathbb{R}^{r\cdot s}$, and transforming the kernel like in Example 2.3.2 yields $\tilde{k} \in \mathbb{R}^{r\cdot s\times m\cdot n}$. The transposed convolution is obtained by computing

$$(u_{0,\bullet}, \dots, u_{m,\bullet}) := \tilde{k}^\intercal \tilde{c} \ , \tag{2.54}$$

followed by reshaping into $u = \begin{pmatrix} u_{0,\bullet} \\ \vdots \\ u_{m,\bullet} \end{pmatrix}$.

An illustrative example showing the idea of the encoder-decoder architecture is provided in Figure 2.15, where the encoder transforms an RGB input image into a

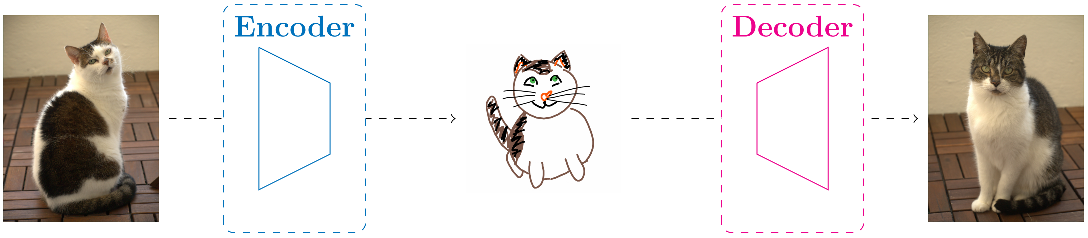


Figure 2.15: Simplified illustration of an encoder-decoder architecture: An RGB image of a cat is transformed into a sketch by the encoder. The decoder transforms the sketch back into an RGB image using transposed convolutions.

sketch of a cat, and the decoder tries to reconstruct the image with the comprised information. By adding a reconstruction loss, DNNs can be trained to improve the bottleneck features, *i.e.*, to capture as much information as possible. The more informative the features are, the more similar the input and output will be. These models are called *autoencoders*. However, instead of reconstructing the input image, the decoder can also produce dense predictions such as a pixel-wise semantic segmentation mask [107], *cf.* Figure 2.16 and Figure 2.18.

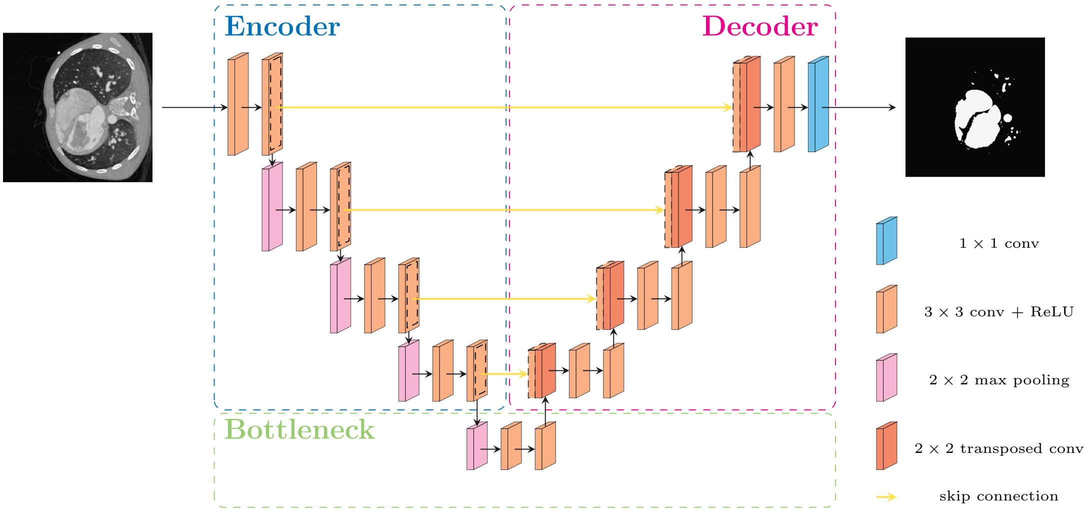


Figure 2.16: U-Net architecture for biomedical segmentation, *e.g.* for images from the ImageCHD dataset [146].

One of the first encoder-decoder networks for the task of biomedical segmentation was the U-Net [120], which is illustrated in Figure 2.16. It consists of a contracting path (encoder) and an expansive path (decoder). The encoder consists of blocks of two unpadded 3×3 convolutions, each followed by a ReLU function, and a 2×2 max pooling with stride $\delta = 2$ for downsampling. The decoder consists of blocks of a transposed 2×2 convolution for upsampling, which is concatenated with the output of the skip connection, followed by two 3×3 convolutions and ReLU functions. In the final layer, a 1 × 1 convolution is applied to map the features to the number of output classes. As border pixels get lost due to the unpadded convolutions, cropping is applied to the features before the skip connections, input and output are not of the same size.

The repeated combination of max-pooling and striding in convolutional layers reduces the dimension of the resulting feature maps, which can be remedied by transposed convolutions like in the U-Net. However, this requires additional memory and time. The DeepLab [23] uses atrous convolutions instead, which adds a rate or dilation factor $\rho \in \mathbb{N}$ to the standard convolution. For a stride $\delta = 1$, the atrous convolution, similar to Equation 2.31, is computed as

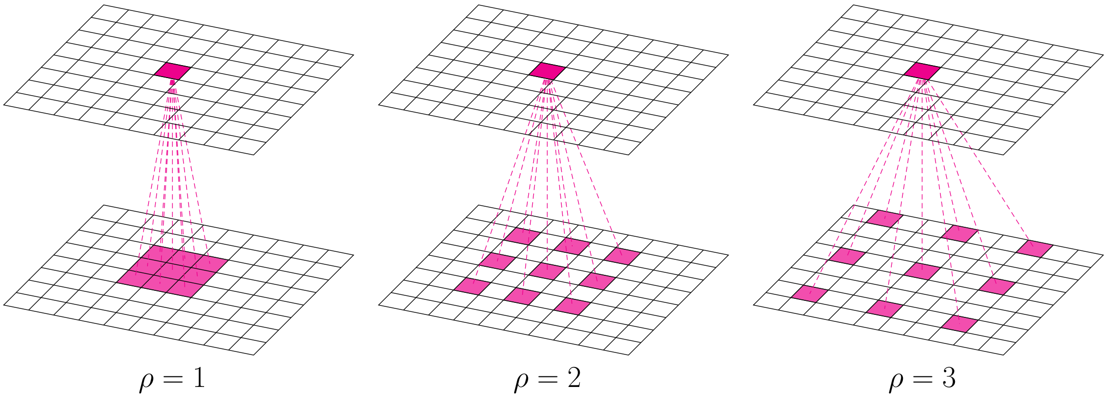


Figure 2.17: Atrous convolution using a 3×3 kernel, a stride $\delta = 1$ and dilation factors $1, 2$ and 3.

Definition 2.17 (Atrous Convolution [102])**.** Let $x = (x_{i,j})_{i,j=1}^{H,W} \in \mathcal{X}$ be a two-dimensional input, $k = (k_{i,j})_{i,j=-K}^{K} \in \mathbb{R}^{2K+1 \times 2K+1}$ a quadratic kernel and $\rho \in \mathbb{N}$ the dilation factor. The atrous convolution is defined as

$$c_{i,j} = (x * k)_{i,j} = \sum_{r=-K}^{K} \sum_{s=-K}^{K} x_{i+\rho r, j+\rho s} k_{r,s} \ , \tag{2.55}$$

where $x_{i,j} = 0 \ \forall i \notin \{1, \dots, H\}, j \notin \{1, \dots, W\}$.

The atrous convolution with $\rho = 1$ is identical to the standard convolution. For $\rho \geq 2$, the atrous convolution skips every ρ-th input element, so it is also called *convolution with holes* [92]. Visually, the atrous convolution applies zero padding to the kernel, adding $\rho - 1$ rows and columns between those of the kernel, *cf.* Figure 2.17.

Example. Let $k = (k_{i,j})_{i,j=-1}^{1}$ be a kernel, $\rho = 2$. The atrous convolution applies zero padding to the kernel, yielding

$$\begin{pmatrix} k_{-1,-1} & k_{-1,0} & k_{-1,1} \\ k_{0,-1} & k_{0,0} & k_{0,1} \\ k_{1,-1} & k_{1,0} & k_{1,1} \end{pmatrix} \overset{\text{pad}}{\mapsto} \begin{pmatrix} k_{-1,-1} & 0 & k_{-1,0} & 0 & k_{-1,1} \\ 0 & 0 & 0 & 0 & 0 \\ k_{0,-1} & 0 & k_{0,0} & 0 & k_{0,1} \\ 0 & 0 & 0 & 0 & 0 \\ k_{1,-1} & 0 & k_{1,0} & 0 & k_{1,1} \end{pmatrix} .$$

The advantages of atrous convolutions are that they convert image classification networks into dense feature extractors without including any additional learnable parameters like the transposed convolutions do. Thus, they speed up the training of the DNN. Furthermore, they allow us to arbitrarily enlarge the *field-of-view* of kernels at any layer, which offers to control the trade-off between accurate localization and context assimilation [102].

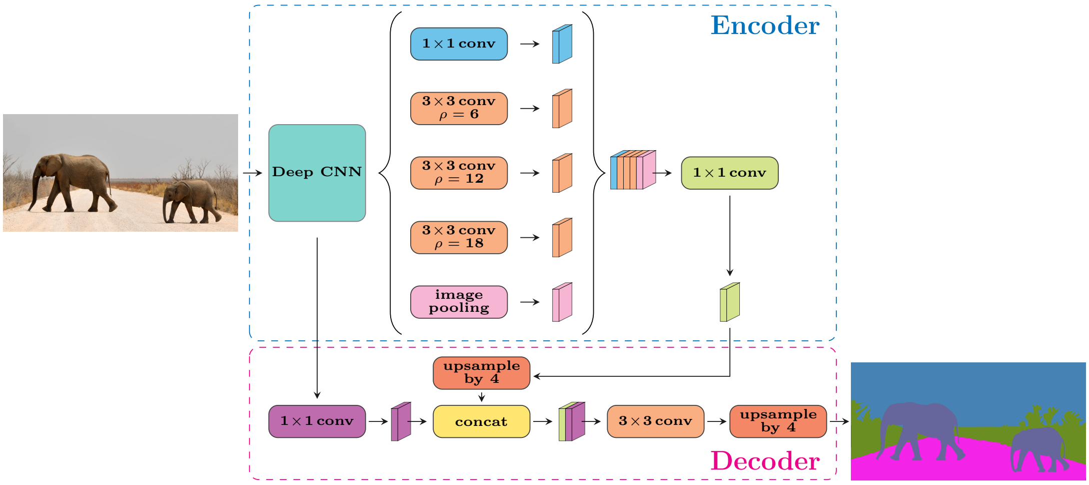


Figure 2.18: Architecture of the DeepLabv3+ model with encoder-decoder structure for the task of semantic segmentation: The features obtained by a deep CNN are passed into the decoder twice, once straight and once after application of the atrous spatial pyramid pooling module (ASPP), which applies atrous convolutions with different dilation factors.

Using the DeepLabv3 [24] as encoder, the DeepLabv3+ [25], which is illustrated in Figure 2.18, adds a decoder module which upsamples the encoder features by a factor of 4 and then concatenates them with the low-level features from the backbone, whose channels are reduced first by a 1 × 1 convolution. After that, a few 3 × 3 convolutions are applied for feature refinement, followed by another upsampling by a factor of 4.

The PSPNet [155] is not a full semantic segmentation network itself. As the DeepLabv3, it can be employed as encoder and extended by an arbitrary decoder. The PSPNet shown in Figure 2.19 uses a deep CNN equipped with atrous

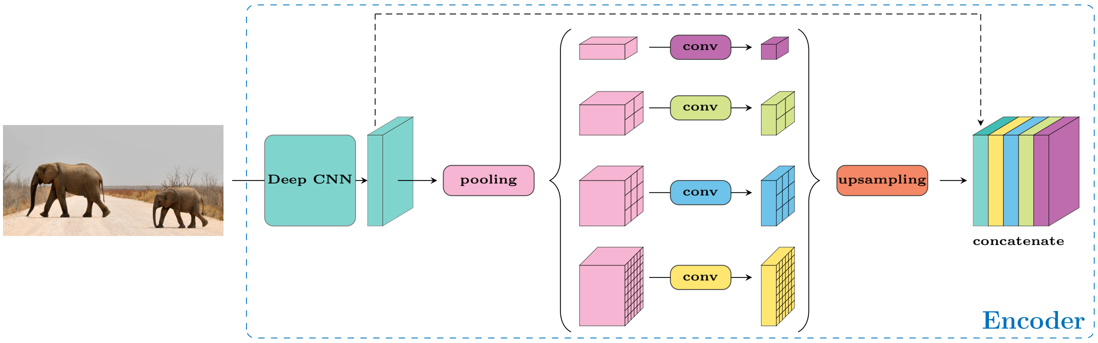


Figure 2.19: Architecture of the PSPNet: The feature map obtained by a deep CNN is passed into the pyramid pooling module, which produces four feature maps at different scales. These are upsampled and concatenated with the initial feature map. These fused features can then be passed into any decoder model.

convolutions as backbone for feature extraction. The resulting feature map is fed into the pyramid pooling module, where it is pooled at different sizes and then passed through a convolution layer. Finally, all feature maps are upsampled and concatenated with the original feature map which was input to the pooling module. Fusing features at different scales enables the model to capture both, high and low resolution features. Usually, the decoder consists of a convolution layer, followed by a 8× bilinear upsampling.

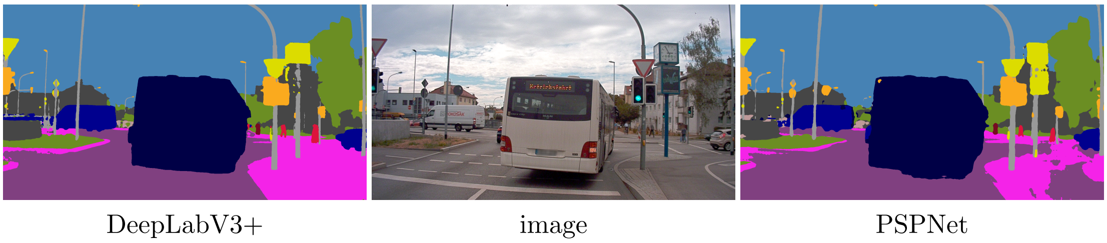


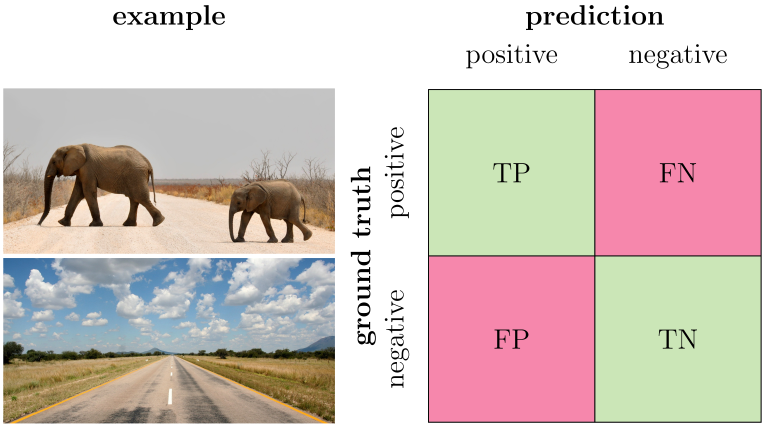
Figure 2.20: Comparison of the semantic segmentation predictions of a street scene obtained by a DeepLabV3+ and a PSPNet, respectively.

Both, the DeepLab and the PSPNet perform spatial pyramid pooling and thus, exploit multiscale information. However, the DeepLabV3+ outperforms the PSPNet with upsampling decoder, as it is capable of capturing high resolution information and thus, more accurate to detail. A comparison of both architectures is provided for a street scene image from the A2D2 [42] dataset in Figure 2.20.

2.3.3 Evaluation Metrics

Image classification as well as semantic segmentation models can be evaluated by computing a *confusion matrix* [49], summarizing the numbers of correct and incorrect predictions. For binary classification with a positive ($y = 1$) and a negative ($y = 0$) class as in Figure 2.21, the confusion matrix includes the quantities

- true positives (TP): number of correct positive predictions ($\hat{y} = y = 1$)
- false positives (FP): number of incorrect positive predictions ($\hat{y} = 1,\ y = 0$)
- true negatives (TN): number of correct negative predictions ($\hat{y} = y = 0$)
- false negatives (FN): number of incorrect negative predictions ($\hat{y} = 0,\ y = 1$)



		prediction	
		positive	negative
ground truth	positive	TP	FN
	negative	FP	TN

Figure 2.21: Confusion matrix for binary classification: $\mathcal{Y} \mathrel{\widehat{=}} \{elephant, no\ elephant\}$.

The confusion matrix can also be extended to a $Q \times Q$ matrix for multi-class classification. With respect to class $q \in \mathcal{Y} = \{1, \ldots, Q\}$, the quantities are obtained by

$$\mathrm{TP}_q = \sum_{(y,\hat{y})} \mathbb{1}_{\{\hat{y}=q\}} \mathbb{1}_{\{y=q\}} \ , \tag{2.56}$$

$$\mathrm{FP}_q = \sum_{(y,\hat{y})} \mathbb{1}_{\{\hat{y}=q\}} \mathbb{1}_{\{y\neq q\}} \ , \tag{2.57}$$

$$\mathrm{TN}_q = \sum_{(y,\hat{y})} \mathbb{1}_{\{\hat{y}\neq q\}} \mathbb{1}_{\{y\neq q\}} \ , \tag{2.58}$$

$$\mathrm{FN}_q = \sum_{(y,\hat{y})} \mathbb{1}_{\{\hat{y}\neq q\}} \mathbb{1}_{\{y=q\}} \ , \tag{2.59}$$

which are adapted to semantic segmentation by counting over pixels instead of images. Using these quantities, several evaluation metrics can be computed to measure the performance of a model. The per-class *accuracy* is obtained by

$$\mathrm{accuracy}_q = \frac{\mathrm{TP}_q + \mathrm{TN}_q}{\mathrm{TP}_q + \mathrm{TN}_q + \mathrm{FP}_q + \mathrm{FN}_q} \ \forall q \in \mathcal{Y} \ , \tag{2.60}$$

measuring the percentage of images or pixels, which are correctly classified with respect to a class q. Accuracy is misleading for underrepresented classes, since it is biased to the TN predictions. Hence, a model which never predicts a rare class q may still achieve high accuracy scores. Separate metrics to quantify the different error types are *precision* and *recall*, obtained by

$$\mathrm{precision}_q = \frac{\mathrm{TP}_q}{\mathrm{TP}_q + \mathrm{FP}_q}, \qquad \mathrm{recall}_q = \frac{\mathrm{TP}_q}{\mathrm{TP}_q + \mathrm{FN}_q} \ \forall q \in \mathcal{Y} \ . \tag{2.61}$$

The precision measures the percentage of correctly classified images or pixels which are predicted to have class affiliation q, the recall the percentage of correctly classified images or pixels which have GT q. Hence, precision quantifies FPs (type 1 errors), recall FNs (type 2 errors). The F_1-score provides the harmonic mean of precision and recall, *i.e.*, it is obtained by

$$F_{1_q} = \frac{2 \cdot \mathrm{TP}_q}{2 \cdot \mathrm{TP}_q + \mathrm{FN}_q + \mathrm{FP}_q} \ \forall q \in \mathcal{Y} \ . \tag{2.62}$$

Measuring the error types separately is particularly significant when one type of error is more serious than the other. For example, overlooking an obstacle on the road ahead is likely to lead to an accident, while predicting a non-existent road hazard leads to unnecessary braking of the vehicle in the worst case.

Pixel-wise Evaluation A popular evaluation metric for semantic segmentation is the *Intersection over Union* (IoU) or *Jaccard index*

$$\mathrm{IoU}_q = \frac{\sum_{(y,\hat{y})} \sum_{z \in \mathcal{Z}} \mathbb{1}_{\{\hat{y}_z = q \wedge y_z = q\}}}{\sum_{(y,\hat{y})} \sum_{z \in \mathcal{Z}} \mathbb{1}_{\{\hat{y}_z = q \vee y_z = q\}}} \ \forall q \in \mathcal{Y} \ , \tag{2.63}$$

measuring the relative number of matching predicted and GT pixels with respect to a class $q \in \mathcal{Y}$. The computation of the IoU is visualized in Figure 2.22 for the *pedestrian* class in one frame.

To obtain an overall quality measure, the class-wise IoU scores are usually averaged over all classes, yielding

$$\text{mean IoU} = \frac{1}{Q} \sum_{q=1}^{Q} \mathrm{IoU}_q \ . \tag{2.64}$$

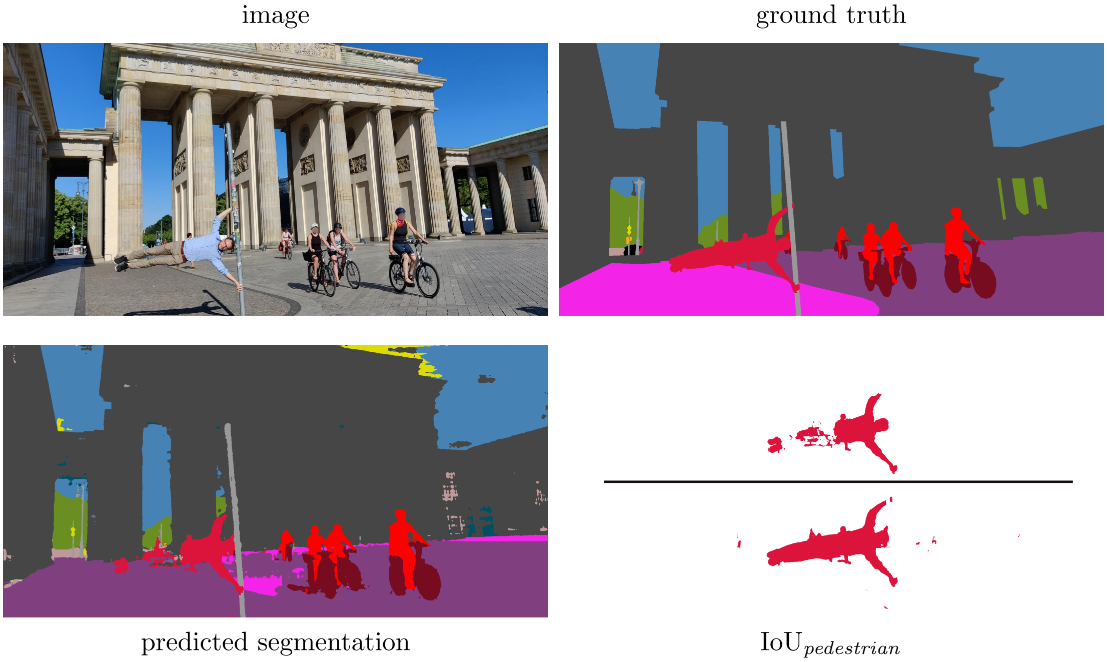


Figure 2.22: Computation of the IoU with respect to the *pedestrian* class.

Segment-wise Evaluation Pixel-wise evaluation metrics report the averaged performance of the DNN over all pixels in the evaluation data, not providing any pixel- or instance-specific quality scores. To compute the IoU for a specific GT (or predicted) segment, which is a connected component of pixels that share the same GT (or predicted) class, the union involves only predicted (or GT) segments which intersect with the segment of interest. In the case that several GT segments which are close together are covered by one predicted segment, or that one GT segment is covered by several predicted segments, the IoU is over-pessimistic, *cf.* Figure 2.23. Hence, an adjusted version of the IoU [122] further occludes pixels from the union which are covered by another segment of the respective class.

Depending on a predefined threshold $\tau \in [0, 1)$, a GT segment s is a true positive, if $\mathrm{IoU}(s) > \tau$, a false negative else. Similar, a predicted segment $\hat{s}$ is a false positive, if $\mathrm{IoU}(s) \leq \tau$.

2.4 Learning from Unlabeled Data

Supervised learning approaches require lots of labeled training data. While unlabeled data exists in quantity, image annotation is time-consuming and expensive, in particular for semantic segmentation. The question arises, whether machine

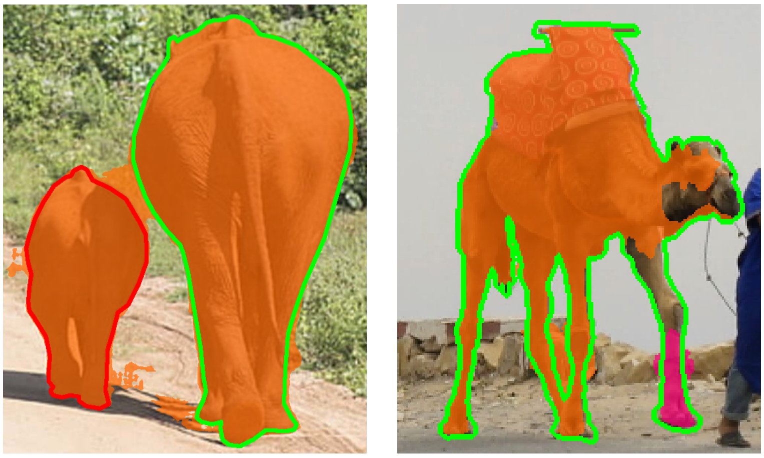


Figure 2.23: (*Left*): Two GT segments, outlined in green & red, intersect with one predicted segment (orange). Green: IoU 68.18% vs. sIoU 87.01%; red: IoU 21.68% vs. sIoU 68.44%. (*Right*): Two predicted segments (orange & pink) intersect with one GT segment, outlined in green. Orange: IoU 78.97% vs. sIoU 81.69%; pink: IoU 03.44% vs. sIoU 18.91% [21].

learning algorithms can benefit from the unlabeled data. Common tasks in unsupervised learning are clustering and dimensionality reduction, which are solely based on patterns in the input data. More sophisticated learning approaches for DNNs leverage unlabeled data for downstream tasks like semantic segmentation.

2.4.1 Unsupervised Learning Algorithms

While in supervised learning, a model is trained on a dataset $\mathcal{S}$ consisting of labeled samples (x, y) to learn a mapping from the input space $\mathcal{X}$ to the target space $\mathcal{Y}$, unsupervised learning describes algorithms where the model observes only the inputs $\mathcal{U} = \{x_i\}_{i=1}^{m} \subseteq \mathcal{X}$ without information about the corresponding labels $\{y_i\}_{i=1}^{m}$ $y_i \in \mathcal{Y}$ $\forall i = 1, \ldots, m$. In the context of neural networks, unsupervised learning approaches fit an unconditional model $p(x)$ instead of a conditional model $p(y|x)$, to detect patterns in the data. Usually, clustering and dimensionality reduction algorithms seek a partition or projection of given data samples, *e.g.* by optimizing loss functions or performing neighborhood search. Since they do not fit a model, these methods are not suitable for integrating new data samples into existing clusters or low-dimensional feature spaces.

Clustering Clustering is an effective approach to identify patterns in unlabeled data. It is a relevant tool in the context of open world recognition, as it can

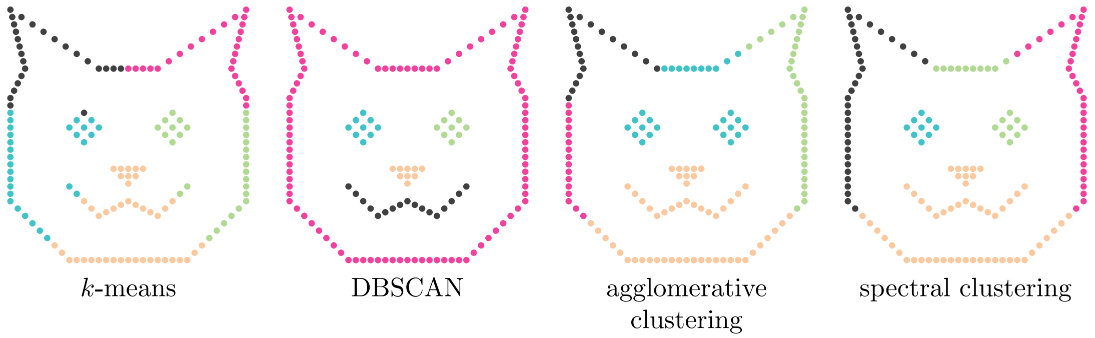


Figure 2.24: Comparison of common clustering methods for a two-dimensional example.

be employed to identify anomalies which stem from the same (unknown) class. Following [145], clustering algorithms can be divided into nine categories, *e.g.* into *partition-based*, *density-based* or *hierarchical* clustering approaches. Some selected approaches are introduced as follows, *cf.* Figure 2.24.

Definition 2.18 (**k**-means [39, 90, 127]). Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of n-dimensional inputs, D a distance metric and $k \in \mathbb{N}$ a given number of clusters. The k-means algortihm partitions the input data $\mathcal{U}$ into k clusters $\mathcal{U}_1, \ldots, \mathcal{U}_k$, which are represented by their centroids

$$\tilde{x}_i = \underset{\tilde{x} \in \mathcal{X}}{\operatorname{argmin}} \sum_{x \in \mathcal{U}_i} D(x, \tilde{x})^2 \,, \tag{2.65}$$

minimizing the cost function

$$\min \sum_{i=1}^{k} \sum_{x \in \mathcal{U}_i} D(x, \tilde{x}_i)^2 \,. \tag{2.66}$$

Since quadratic programming is NP-hard, *i.e.*, not solvable in polynomial time, heuristics are applied to approximate the optimal solution. The k-means algorithm randomly initializes $k \in \mathbb{N}$ centroids and allocates each x_i, $i = 1, \ldots, m$, to the cluster whose centroid is nearest. Next, the centroids are updated in each dimension to

$$\tilde{x}_{i,j} = \frac{1}{|\mathcal{U}_i|} \sum_{x \in \mathcal{U}_i} x_j \ \forall j = 1, \ldots, n, \ i = 1, \ldots, k \,, \tag{2.67}$$

which is repeated until some stopping criterion is satisfied.

In k-means, clusters are assumed to be compact in terms of the sum of squared distances to the centroids. Hence, it is not capable of handling data, where clusters are of varying size or densities. Furthermore, outliers have a huge impact on the cluster centroids. The *Density-Based Spatial Clustering of Applications with Noise* [36] (DBSCAN) algorithm detects clusters of varying shapes and sizes, without a specified number of clusters. Furthermore, it is able to identify noise.

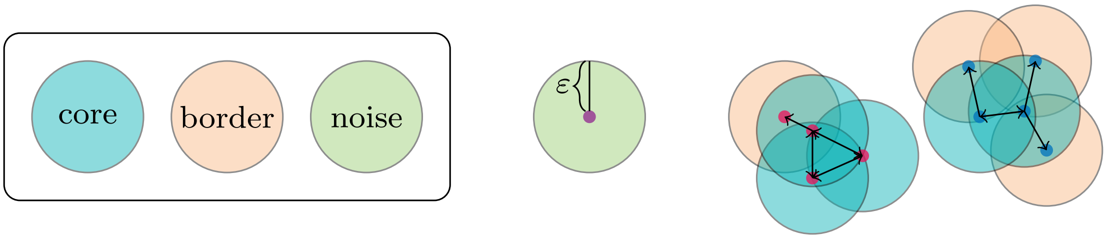


Figure 2.25: Visualization of the DBSCAN algorithm, resulting in two clusters and one noise observation. Here, a data point is considered a core point, if its ε-neighborhood contains at least $\delta = 3$ observed data points.

Definition 2.19 (DBSCAN [36])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of inputs. DBSCAN differentiates between *core points*, *border points* and *noise*, *cf.* Figure 2.25. For given parameters $\varepsilon \in (0, \infty)$, $\delta \in \mathbb{N}$, an input $x_i,\ i \in \{1, \dots, m\}$ is a core point, if its ε-neighborhood $\mathbb{B}_\varepsilon(x_i)$ contains at least δ inputs $x_j,\ j = 1, \dots, m$. Two core points $x_i, x_j,\ i \neq j$ are *connected*, if $\exists x_k \in \mathbb{B}_\varepsilon(x_i) \cap \mathbb{B}_\varepsilon(x_j),\ k \in \{1, \dots, m\}$. An input x_i, which is not a core point itself, is called border point, if there is a core point x_j, such that $x_i \in \mathbb{B}_\varepsilon(x_j)$. Otherwise, x_i is considered noise. Finally, each chain of connected core points including their respective border points constitutes a cluster.

However, DBSCAN fails if the cluster densities deviate too much. Furthermore, it is highly sensitive to the choice of the parameters ε and δ, which is overcome in an improved version called OPTICS [2].

Hierarchical clustering methods produce a hierarchical structure of the data, allowing clusters to be selected at different hierarchical levels. To this end, *agglomerative clustering* or *divisive clustering* algorithms are employed, which recursively merge or split clusters, respectively.

Definition 2.20 (Agglomerative Clustering [39])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of inputs, D a distance metric. Agglomerative clustering initializes m clusters $\mathcal{U}_i = \{x_i\},\ i = 1, \dots, m$. There are different possibilities to measure the distance between two clusters $\mathcal{U}_i$ and $\mathcal{U}_j$. *Single-link* clustering considers the closest distance between $\mathcal{U}_i$ and $\mathcal{U}_j$, yielding

$$D_{\mathrm{sl}}(\mathcal{U}_i, \mathcal{U}_j) = \min_{x \in \mathcal{U}_i, x' \in \mathcal{U}_j} D(x, x') \ , \tag{2.68}$$

complete-link clustering the largest distance

$$D_{\mathrm{cl}}(\mathcal{U}_i, \mathcal{U}_j) = \max_{x \in \mathcal{U}_i, x' \in \mathcal{U}_j} D(x, x') \ , \tag{2.69}$$

and *group average* clustering takes the average distance

$$D_{\mathrm{ac}}(\mathcal{U}_i, \mathcal{U}_j) = \frac{1}{|\mathcal{U}_i| \cdot |\mathcal{U}_j|} \sum_{x \in \mathcal{U}_i} \sum_{x' \in \mathcal{U}_j} D(x, x') \tag{2.70}$$

over all pairs $(x, x') \in \mathcal{U}_i \times \mathcal{U}_j$. In every iteration, agglomerative clustering merges the two clusters with the smallest intra-cluster distance.

Since most clustering methods perform poorly or are not applicable to high dimensional data, dimensionality reduction is usually applied as a preprocessing step, projecting the input data into a low-dimensional space while retaining the most important features.

Definition 2.21 (Spectral Clustering [49])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of n-dimensional inputs. Spectral clustering computes an undirected similarity graph, where each x_i is represented by a node $v_i \in \mathcal{V},\ i = 1, \ldots, m$, and each pair of nodes $(v_i, v_j),\ i \neq j$ is connected by a weighted edge. The weights $\omega_{i,j} \geq 0$ are computed by the Gaussian similarity function

$$\omega_{i,j} = \exp(-\|x_i - x_j\|^2/2\sigma^2)\ \forall i, j \in \{1, \ldots, m\}\ , \tag{2.71}$$

where the parameter σ controls the width of the neighborhoods. Next, the eigenvectors $z_{\bullet,1}, \ldots, z_{\bullet,m}$ of the graph Laplacian L, where

$$L_{i,j} = \begin{cases} \sum_{j=1}^n \omega_{i,j} & i = j \\ -\omega_{i,j} & i \neq j \end{cases} \forall i, j \in \{1, \ldots, m\}\ , \tag{2.72}$$

are computed and sorted by increasing eigenvalues. Finally, the k-means algorithm is applied to cluster the rows of the matrix $\begin{pmatrix} z_{\bullet,1} & \ldots, z_{\bullet,d} \end{pmatrix} \in \mathbb{R}^{m\times d},\ d < n$, where $z_{i,\bullet}$ is the d-dimensional representation of $x_i,\ i = 1, \ldots, m$.

Further linear [127] and nonlinear [49] dimensionality reduction approaches can be applied to simplify not only clustering, but also the training of predictive models by tackeling the curse of dimensionality.

Dimensionality Reduction Similar to spectral clustering, *Principal Component Analysis* (PCA) [112] performs dimensionality reduction on the basis of eigenvectors and -values to find linear transformation matrices W, U for dimensionality reduction and reconstruction, respectively, *cf.* Figure 2.26.

Definition 2.22 (PCA [112, 127])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of n-dimensional inputs, which should be mapped into a d-dimensional space, $d < n$. The goal of PCA is to find linear transformation matrices $W \in \mathbb{R}^{d\times n}$ and $U \in \mathbb{R}^{n\times d}$ which minimize the total squared distance between x and UWx, *i.e.*, which solve the objective

$$\operatorname*{argmin}_{W,U} \sum_{i=1}^m \|x_i - UWx_i\|_2^2\ . \tag{2.73}$$

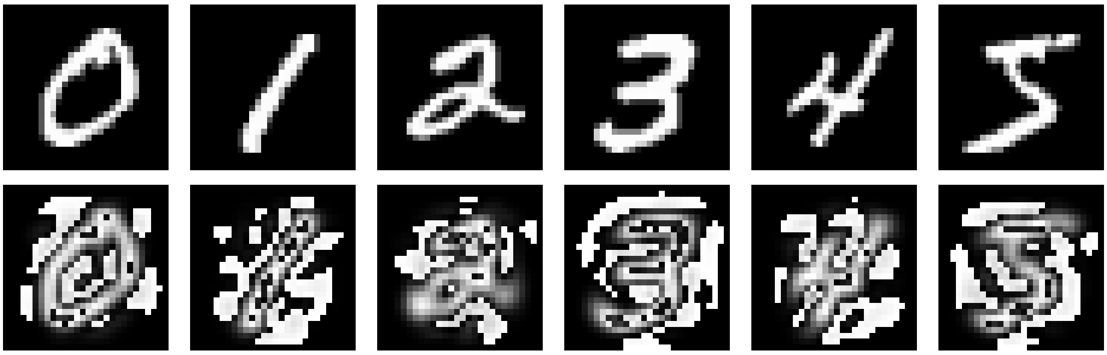
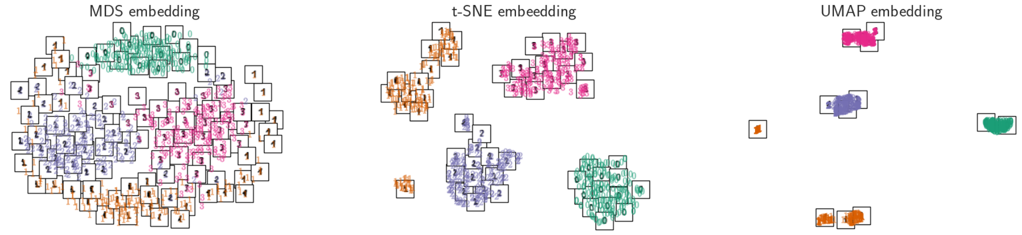


Figure 2.26: Reconstructed samples of the MNIST dataset (*bottom row*) after reducing the dimension of the original data (*top row*) from 784 to 50.

By computing the eigenvectors $v_{\bullet,1}, \ldots, v_{\bullet,n}$ of the covariance matrix $\sum_{i=1}^{m} x_i x_i^\intercal \in \mathbb{R}^{n \times n}$ with corresponding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$, the solution to Equation 2.73 is obtained by

$$U = \begin{pmatrix} v_{\bullet,1} & \ldots & v_{\bullet,d} \end{pmatrix} \; W = U^\intercal \, . \tag{2.74}$$

Instead of applying linear transformations, *manifold learning* approaches perform nonlinear dimensionality reduction, thus, they can generalize to all structures of data. The basic idea of manifold learning is that the data samples are actually samples from a low-dimensional manifold $\mathcal{M}$, which is embedded in a high-dimensional space $\mathcal{X}$. In the following, the three approaches *Multidimensional Scaling* (MDS) [70, 71], *t-Distributed Stochastic Neighbor Embedding* (t-SNE) [135] and *Uniform Manifold Approximation and Projection* (UMAP) [95] are introduced, *cf.* Figure 2.27.



MDS t-SNE UMAP

Figure 2.27: Comparison of manifold learning approaches, employed to the Digits dataset for the digits $0, 1, 2, 3$.

Definition 2.23 (MDS [49, 70, 71])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^{m} \subseteq \mathcal{X}$ be a set of n-dimensional inputs, $\mathcal{M}$ a d-dimensional manifold, $d \leq n$, and D an arbitrary distance metric. The goal of MDS is to find latent vectors $\{z_i\}_{i=1}^m \in \mathcal{M}$, which have approximately the same pair-wise distances as the input vectors, *i.e.*, which satisfy

$$D(z_i, z_j) \approx D(x_i, x_j) \; \forall i, j = 1, \ldots, m \; . \tag{2.75}$$

The stress function

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \left(D(x_i, x_j) - D(z_i, z_j)\right)^2 \tag{2.76}$$

is minimized by employing gradient descent.

Least squares scaling emphasises large distances, which do not capture the local structure of the data. Hence, *Sammons Mapping* [124] extends the stress function by a factor, yielding the non-convex Sammon's stress

$$\sum_{i=1}^{m} \sum_{j=1 j \neq i}^{m} \frac{\left(D(x_i, x_j) - D(z_i, z_j)\right)^2}{D(x_i, x_j)} \; , \tag{2.77}$$

which upweights the small distances. With t-SNE and UMAP, more spohisticated methods to capture local structure have been developed.

Definition 2.24 (t-SNE [135])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^m \subseteq \mathcal{X}$ be a set of n-dimensional inputs and $\mathcal{M}$ a d-dimensional manifold, $d \leq n$. Assuming that neighbors are chosen proportionally to their probability density under a Gaussian with center at x_i and adjustable variance σ_i, stochastic neighbor embedding computes conditional probabilities

$$p(x_j | x_i) = \frac{\exp(-\frac{1}{2\sigma_i^2} \|x_i - x_j\|^2)}{\sum_{k \neq i} \exp(-\frac{1}{2\sigma_i^2} \|x_i - x_k\|^2)} \; , \tag{2.78}$$

which represent the likelihood of $x_j \in \mathcal{U}$ being a neighbor of $x_i \in \mathcal{U}$. In the d-dimensional manifold $\mathcal{M}$, the Gaussian distribution is replaced by a Student's-t distribution, yielding conditional probabilities

$$p'(z_j | z_i) = \frac{(1 + \|z_i - z_j\|^2)^{-1}}{\sum_{k < l} (1 + \|z_k - z_l\|^2)^{-1}} \tag{2.79}$$

for all $z_i, z_j \in \mathcal{M}$. Gradient descent is applied to minimize the KL divergence

$$\sum_{i=1}^{m} \sum_{j=1}^{m} p(x_j | x_i) \log\left(\frac{p(x_j | x_i)}{p'(z_j | z_i)}\right) \tag{2.80}$$

between the conditional distributions.

Hence, t-SNE captures local structures since it preferably pulls distant samples together rather than pushing nearby samples apart. However, the implementation of t-SNE takes $\mathcal{O}(N^2)$ time. UMAP is a much faster manifold learning algorithm, which furthermore preserves global structure better than t-SNE.

Definition 2.25 (UMAP [95])**.** Let $\mathcal{U} = \{x_i\}_{i=1}^{m} \subseteq \mathcal{X}$ be a set of n-dimensional inputs and $\mathcal{M}$ a d-dimensional manifold, $d \leq n$. First, UMAP constructs a graph whose nodes represent the input data $x \in \mathcal{U}$. A weighted edge indicates the likelihood that two samples are connected in terms of Euclidean distances. To this end, each node is extended by a variable radius based on the distance to its k-nearest neighbor. Two inputs are connected if their radii are overlapping. Since the total distances between connected nodes are replaced by probabilities, the connectedness in UMAP is called *fuzzy.* The goal of UMAP is to fit a d-dimensional graph which retains most of the structure and characteristics of the input data. Representing both graphs by their common set of edges $\mathcal{E}$ and their respective edge weight functions $p, p' : \mathcal{E} \to [0, 1]$, UMAP employs gradient descent to optimize the cross entropy loss

$$\sum_{e \in \mathcal{E}} \left(p(e) \log\left(\frac{p(e)}{p'(e)} \right) + (1 - p(e)) \log\left(\frac{1 - p(e)}{1 - p'(e)} \right) \right) . \tag{2.81}$$

In the field of deep learning, unlabeled data can also be leveraged to improve the performance of DNNs.

2.4.2 Unlabeled Data in Semantic Segmentation

While unsupervised learning usually does not include class predictions, several learning paradigms like *self-supervised*, *semi-supervised* or *few-shot learning* leverage unlabeled data to improve the performance of a classification or segmentation DNN. These paradigms do not contradict each other and can be summarized as follows. Self-supervised learning uses unlabeled data to create proxy supervised tasks, semi-supervised learning takes advantage of unlabeled data to enhance training on a small labeled training set and few-shot learning comprises all approaches which can handle very small sample sizes $m \leq 5$, *e.g.* by leveraging prior knowledge.

Self-Supervised Learning Self-supervised learning extracts labels from the input data to solve a proxy-supervised task with the goal of learning useful feature representations. That is, self-supervised learning can be used to train an encoder, whose knowledge can later be transferred to a downstream task such as image classification or semantic segmentation. For example, in [35], spatial context is used as supervisory signal. Therefore, a random image patch as well as one of its eight possible neighbor patches are sampled, and the DNN is trained to predict the localization of the neighbor patch, *cf.* Figure 2.28.

Context encoders [111] generate the pixel values of a missing image region based on the surrounding pixels, *cf.* Figure 2.29. The authors suggest three different

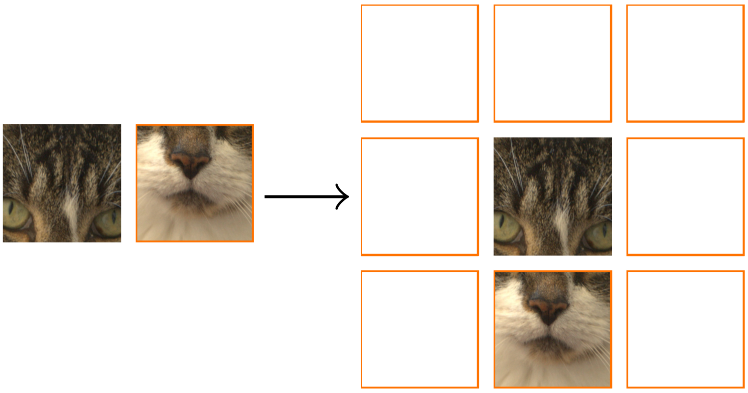


Figure 2.28: Given the center patch of an image, the model learns the relative position of a randomly sampled neighbor patch.

strategies for selecting the masked regions. The simplest shape is the central square patch, which works well for the inpainting task. However, the learned features focus on the constant border pixels and thus do not generalize well. This is mitigated by sampling multiple, possibly overlapping blocks per image, so that the pixel positions of the region boundaries vary significantly. However, these boundaries are still very sharp. Therefore, as a third approach, arbitrary shapes are extracted from the GT masks of another dataset and randomly inserted into the images.

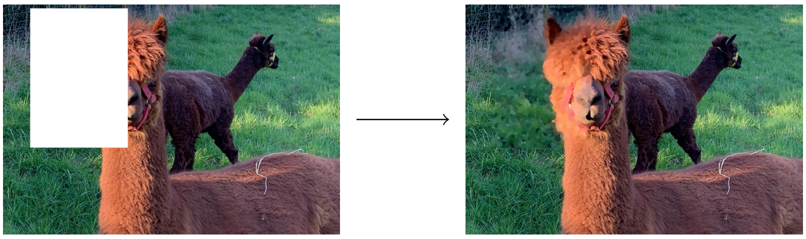
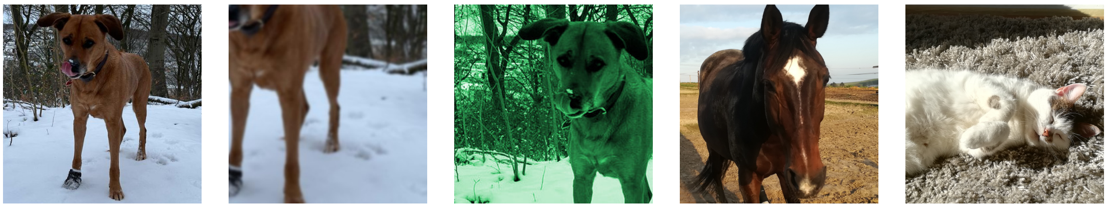


Figure 2.29: The context encoder learns to inpaint missing regions in an image.

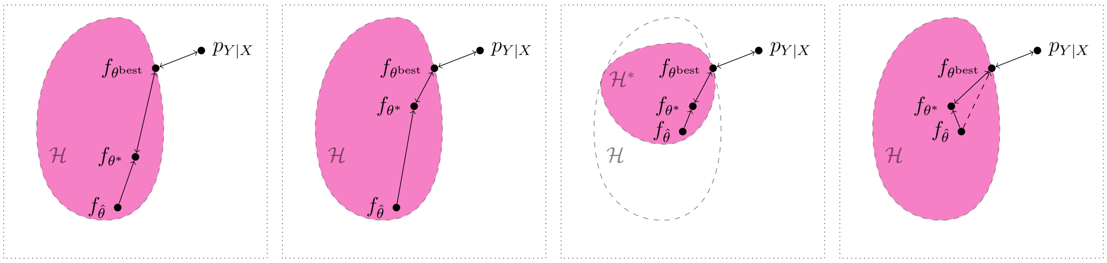
Another approach [26] uses contrastive learning together with image augmentations such as cropping, color distortion or Gaussian blurring. More specifically, for a given anchor image, data augmentation methods are applied to create *positive* images, while the rest of the images contained in the dataset or batch are considered *negative* images, see Figure 2.30 for an example. Then, the DNN is trained with a contrastive loss to align the feature embeddings of positive pairs while pulling negative images apart. An overview of further approaches is provided in [1].



anchor positive positive negative negative

Figure 2.30: Examples for positive and negative images for contrastive learning.

Semi-Supervised Learning Semi-supervised learning includes all approaches, where a model is trained on both, labeled and unlabeled data. An overview of different approaches is provided in [109]. For example, self-supervised approaches are employed to learn high-level features of unlabeled data, followed by supervised fine-tuning for a specific task. Furthermore, DNNs are trained on a small labeled dataset, which is continuously enriched by self-labeled samples [76]. These *pseudo-labels* for unlabeled samples are obtained by the DNNs predictions. Usually, only reliable predictions with low softmax entropy are considered. However, unreliable predictions are used as *negative* samples to occlude classes which are not likely [139]. Another line of work is consistency regularization, including *e.g.* approaches which push the decision boundaries into low-density regions by entropy minimization [45], or enforce consistency between two segmentation networks perturbed with different initialization on the same input image [28].



large estimation error due to small sample

1. data augmentation

2. restricted hypothesis space

3. improved search strategies

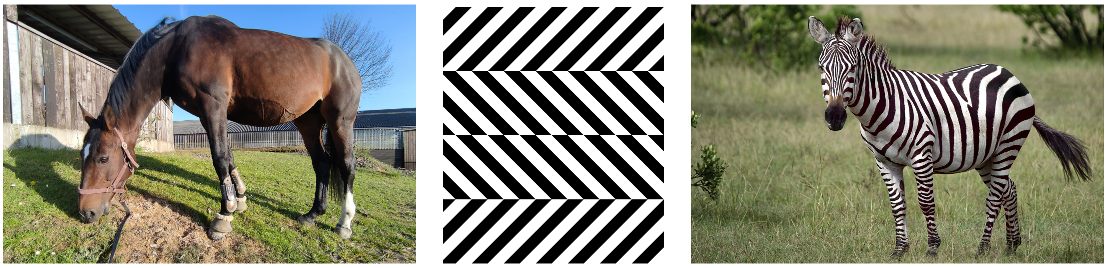
Figure 2.31: FSL strategies to leverage prior knowledge.

Few-Shot Learning Few-shot learning (FSL) describes all learning approaches where a DNN is trained on only few labeled data, *i.e.*, usually having access to less than five labeled samples. As discussed in Subsection 2.1.2, the estimation error can be reduced by increasing the sample size. Thus, the empirical risk $\mathcal{L}_\mathcal{S}$ in FSL may be a bad approximation of the expected risk $\mathcal{L}$, making the empirical risk minimizer unreliable. This problem is alleviated by including prior knowledge in order to

1. increase the sample size using data augmentation
2. constrain the complexity of the hypothesis space by discarding hypotheses that are unlikely to be optimal
3. or to alter the search strategy by providing a good initialization or by guiding the search steps.

The effects of all three strategies are illustrated in Figure 2.31 [140]. For example, the PANet [138] learns and aligns robust prototypes for each semantic class in the feature space. Then, each pixel is labeled as the class of the nearest prototype. The PFENet [131] uses a pretrained image classification CNN to produce *prior masks* which indicate the probability of each pixel belonging to some target class.

Zero-shot learning (ZSL) is a special case of FSL, where novel classes are learned without labeled training data for a specific task. ZSL approaches leverage information from other modalities and transfer the information to the current task. Although this approach is already close to unsupervised learning, it requires adequate knowledge about the image features of unseen classes. This is inspired by how people learn. Consider *e.g.* a child that knows horses but not zebras. Having the prior knowledge that a zebra is very similar to a horse, but having black and white stripes, it will be able to recognize a zebra when seeing it for the first time, *cf.* Figure 2.32.



known class *horse*

known concept *stripes*

unknown class *zebra*

Figure 2.32: The basic idea of zero-shot-learning is to leverage prior knowledge, *e.g.* about the class *horse* and the concept *stripes*, to recognize a previously-unseen class like the class *zebra*.

For example, ZSL leverages word features to get an idea of the corresponding visual features. For this purose, the ZS3Net [14] employs a semantic segmentation DNN together with a generator to align text with visual embeddings. The segmentation DNN is trained on a closed set of semantic classes, and ZSL is applied to incrementally extend it by novel classes. Therefore, the generator is trained on word2vec [99] embeddings of the known class labels to generate synthetic features, which match the features produced by the segmentation DNN. This generator is utilized to produce synthetic features for novel classes. Then, the classifier of the segmentation DNN is fine-tuned on these synthetic features for novel along with real features for known classes.

Chapter 3

Road Anomalies

Being capable of recognizing anomalies is of utmost importance for safety-critical applications such as automated driving. To this end, datasets are required which allow for extensive testing and evaluation of proposed anomaly segmentation approaches along with appropriate evaluation metrics. In the context of open world recognition, the anomalies are not only recognized but also classified in terms of clustering.

3.1 Perception Datasets for Autonomous Driving

DNNs used in perception systems for autonomous driving require a huge amount of data to train on, as they must reliably achieve high performance in all kinds of situations. However, these DNNs are usually limited to a closed set of semantic classes available in their training data and are therefore unreliable when confronted with previously unseen instances. Therefore, several perception datasets have been created for the evaluation of anomaly detection methods, which can be categorized into three groups: real anomalies in real-world, synthetic anomalies augmented into real-world and fully synthetic scenes. This section further divides anomaly datasets into different anomaly sources, followed by a definition of normality and an overview of existing datasets, which is summarized in Table 3.1. The section concludes with a discussion of existing challenges in generating anomaly datasets for semantic segmentation.

dataset	year	sensors	size (test/val)	resolution	anomaly source	#OoD Labels	ground truth
Fishyscapes [11]							
FS Lost and Found	2019	Camera	275 / 100	2048 × 1024	Recording	1	Semantic Mask
FS Static	2019	Camera	1,000 / 30	2048 × 1024	Data Augmentation	1	Semantic Mask
CAOS [54]							
StreetHazards	2019	Camera	1,500	1280 × 720	Simulation	1	Semantic Mask
BDD-Anomaly	2019	Camera	810	1280 × 720	Class Exclusion	3	Semantic Mask
SegmentMeIfYouCan [21]							
RoadAnomaly21	2021	Camera	100 / 10	2048 × 1024 1280 × 720	Web Sourcing	1	Semantic Mask
RoadObstacle21	2021	Camera	327 (+55) / 30	1920 × 1080	Recording	1	Semantic Mask
Wuppertal OoD Tracking [88]							
Street Obstacle Sequences (SOS)	2022	Camera, Depth	1,129	1920 × 1080	Recording	13	Instance Mask
CARLA-WildLife (CWL)	2022	Camera, Depth	1,210	1920 × 1080	Simulation	18	Instance Mask
Misc							
Lost and Found [113]	2016	Stereo Cameras	2,104	2048 × 1024	Recording	42	Semantic Mask
WD-Pascal [8]	2019	Camera	70	1920 × 1080	Data Augmentation	1	Semantic Mask
Vistas-NP [46]	2020	Camera	11,167	Varying	Class Exclusion	4	Semantic Mask

Table 3.1: Overview of anomaly datasets for semantic segmentation of street scenes, clustered by the benchmark in which they were presented.

3.1.1 Anomaly Sources

Anomaly datasets can be divided into different anomaly sources. Some of them utilize existing datasets, such as automated OoD proposal, misc classes, class exclusion and data augmentation, while web sourcing, recording and simulation create completely new data.

- *Automated OoD proposal* allows for the utilization of large, unlabeled datasets. Here, an automated proposal method is used to generate first anomaly proposals. This can be done with any anomaly detection approach, *e.g.* uncertainty, intermediate detections, geometric priors, or model contradictions. Subsequently, human experts take care of false positives and refine the proposals.

- The *misc classes* approach is based on a labeled dataset. All regions which are either labeled with *void* or *misc* can be examined further. These terms are often used interchangeably and mostly refer to uncommon objects or irrelevant areas. Human experts then relabel those classes as anomalies, if appropriate.

- *Class exclusion* is based on a labeled dataset. Hypothetical anomalies are created by excluding frames with known classes from the train and validation splits. A novel test split is created with these, treating the selected classes as anomalies.

- In *web sourcing*, human experts actively search for images that include atypical classes.

- In *recording* and *simulation*, anomalies are recorded through data collection by driving in the real or synthetic world.

- For *data augmentation*, any unlabeled dataset can be used as a baseline. By synthetic manipulation of scenes, anomalies are pasted onto the original image and can be labeled accordingly.

3.1.2 Definition of Normality

There is no clear definition of whether an object is anomalous or not. However, a common approach is to define anomalies as *none-of-the-known classes* with respect to the 19 Cityscapes [29] evaluation classes. Most anomaly techniques adhere to the definition of *Cityscapes as normality*: For web sourcing, simulation, data augmentation and recording, the anomalous objects are selected to fit into this definition. For the void class approaches, the definition of anomalous objects depends on the respective underlying dataset, as the *void* or *misc* category isn't clearly defined, either. Also, for the automated OoD proposal technique, the definition of normality strongly depends on the underlying classes of the employed detector(s). Finally, for class exclusion, normality depends on the choice of excluded classes. The anomalies labeled by this approach are usually not anomalous with respect to Cityscapes and as such do not represent anomalies which would be rare in the real world.

Cityscapes The Cityscapes dataset [29] consists of urban street scene images, which were acquired from a moving vehicle during spring to fall, showing 50 different, mostly German cities in good weather conditions. In total, 5,000 images of resolution 2,048 × 1,024 are provided with dense pixel-level annotations. These are partitioned into a training, test and validation split of sizes 2,975, 1,525 and 500, respectively. For the test data, the labels are withheld for benchmarking purposes. The ground truth differentiates between 30 visual classes which can be grouped into the eight categories *flat, construction, nature, vehicle, sky, object, human* and *void.* Classes that are too rare were excluded from the benchmark, leaving the 19 evaluation classes *road, sidewalk, building, wall, fence, pole, traffic sign, traffic light, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle* and *bicycle.* These 19 classes are often considered as normality.

3.1.3 Anomaly Datasets

This subsection provides an overview of anomaly datasets, some of which have been created in the context of this thesis. Other anomaly datasets which are not considered here are *e.g.* the StreetHazards and the BDD-Anomaly datasets included in the CAOS benchmark [54], the WD-Pascal [8] and the Vistas-NP [46] datasets.

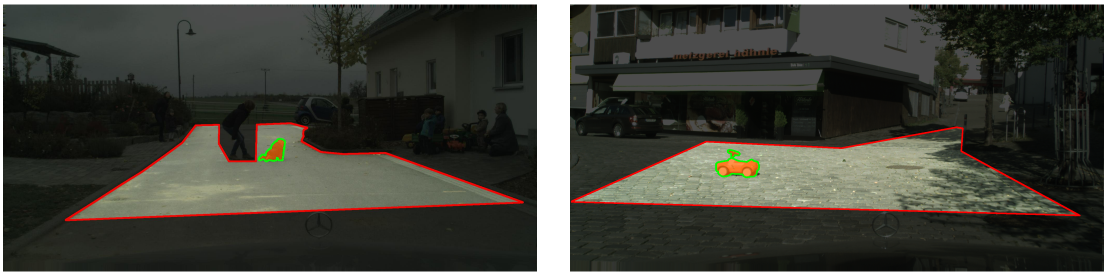
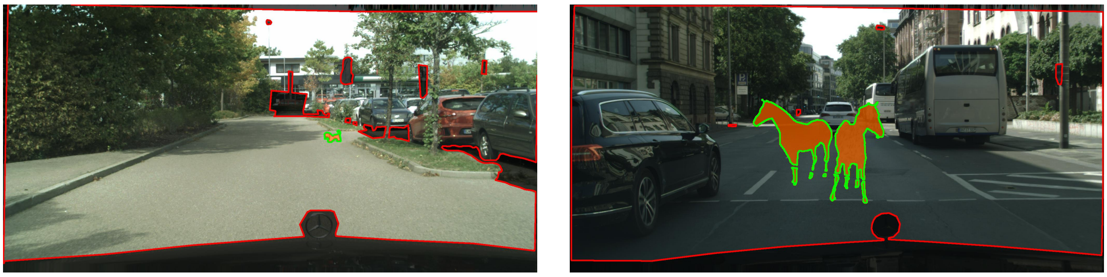


Figure 3.1: Lost and Found: Visualization of exemplary real anomaly types in real-world scenes.

Lost and Found The Lost and Found dataset [113] was introduced in 2016 by Pinggera et al., being the first dataset with a focus on the detection of small road hazards, as shown in Figure 3.1.

The provided stereo masks for the task of semantic segmentation allow for pixel- and instance-level evaluation, as proposed by the authors. Their instance-level approach is based on a 3D stixel representation, which is very method-specific. As the dataset provides data from stereo cameras, geometric methods can be applied. The anomalies include 42 individual object types that can realistically be found in a street environment. The objects are categorized into *standard objects*, *random hazards*, *emotional hazards* as animals or toys, *random non-hazards*, and *humans* and include both static and dynamic obstacles.

The data was collected in the greater Stuttgart area, Germany. It includes irregular road surfaces, large object distances, and illumination changes [113]. Typical environments include housing areas, parking lots, or industrial areas [10].



FS Lost and Found

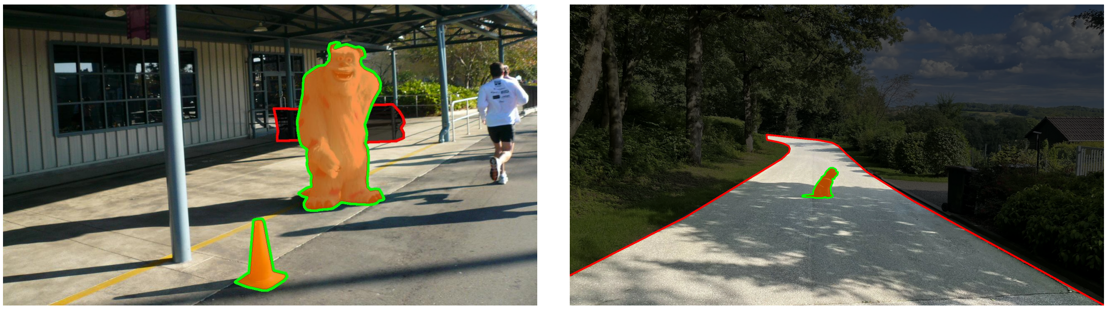
FS Static

Figure 3.2: Fishyscapes: Samples from the val splits, showing real-world scenes with real (left) and synthetic (right) anomalies.

Fishyscapes The Fishyscapes (FS) benchmark [11] was introduced in 2019 by Blum et al. for the evaluation of anomaly detection methods in semantic segmentation. While most of the data is withheld for evaluation, the authors provide validation sets for the different datasets FS Lost and Found and FS Static. A third FS Web dataset is completely withheld. The first dataset is a subset of the Lost and Found dataset [113]. The others are based on the Cityscapes [29] validation data, overlayed with anomalous objects which are either extracted from the generic Pascal VOC [37] dataset or crawled from the internet. The FS Static validation frames are automatically generated from the Cityscapes dataset.

The FS datasets are designed for the task of semantic segmentation. FS Lost and Found is enriched with fine-grained binary semantic masks. The refined annotation of the background is shown by comparing Figure 3.2 with Figure 3.1. Furthermore, sequences, where the anomalous objects are bicycles or children, are filtered out, as they can be assigned to one of the Cityscapes classes. For FS Static and FS Web, novel objects are blended into already annotated scenes from Cityscapes, resulting in fully annotated semantic masks. The anomalies extracted from the Pascal VOC dataset belong to the classes *airplane, bird, boat, bottle, cat, chair, cow, dog, horse, sheep, sofa,* and *tvmonitor*.

As all images originate from the Cityscapes or the Lost and Found datasets, they are recorded at daytime under clear weather conditions. For the augmented Cityscapes data, this also entails that the street scenes include instances from known classes, such as humans or other vehicles. Depending on the anomaly type, the anomalies have a higher probability to appear either on the lower- or the upper half.



RoadAnomaly21

RoadObstacle21

Figure 3.3: SegmentMeIfYouCan: Real-world examples from the RoadAnomaly21 test and the RoadObstacle21 val splits.

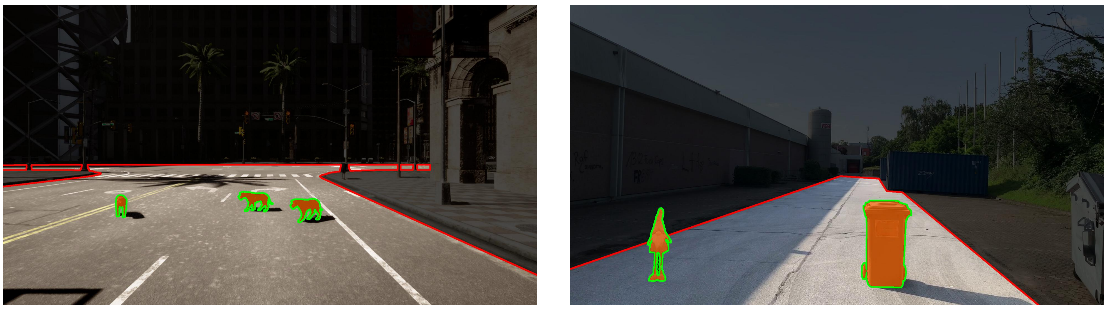
SegmentMeIfYouCan The SegmentMeIfYouCan benchmark [21] was developed in 2021 by Chan, Lis, Uhlemeyer, Blum[1] et al. along with two real-world datasets. A previous version of the RoadAnomaly21 dataset was already published in 2019 by Lis et al. [84]. The current version was both refined and extended. It consists

[1] Equal contribution

of images collected from the internet, which show anomalous objects on or near the road. The RoadObstacle21 dataset was recorded by the authors and includes anomalous objects placed on the road ahead. Similar to FS Lost and Found, which is also included in the benchmark, these datasets only contain real anomalies.

Both datasets are designed for the task of semantic segmentation, the semantic masks include binary anomaly labels. RoadAnomaly21 is designed for general anomaly detection in full street scenes, whereas in RoadObstacle21, the road is considered the region of interest, *i.e.*, the *not anomaly* class. Thus, everything not included in this region is assigned to the *void* class, which is represented in Figure 3.3. The anomalies in RoadAnomaly21 can be categorized into animals, *e.g. elephant, cow, horse*, unknown vehicles, *e.g. airplane, boat trailer, tractor*, and others, such as *tent, piano*, or *cones*. In RoadObstacle21, each object on the road ahead is considered an obstacle. However, all obstacles in this dataset also fit the definition of anomaly as objects which cannot be assigned to the Cityscapes classes. Semantic masks are in both cases only published for small validation sets.

In RoadAnomaly21, images are collected from web resources and thus depict a wide variety of environments and settings. All images are recorded during daytime and in clear weather. The anomalies can appear anywhere in the image, even in the sky. Therefore, they are not necessarily street hazards. The images of RoadObstacle21 are recorded in Germany and Switzerland on seven different road types, also during daytime and in clear weather. Additionally, there are 55 annotated frames by night and in snowy weather conditions.



CARLA-WildLife

Street Obstacle Sequences

Figure 3.4: Wuppertal OoD Tracking: Two examples showing simulated (left) and real (right) anomalies.

Wuppertal OoD Tracking Datasets for OoD tracking [88] were introduced in 2022 by Maag et al.[2], enabling OoD detection and tracking over video sequences. The Street Obstacle Sequences (SOS) dataset contains annotated real-

[2] With contribution of the author of this thesis

world scenes with real anomalies. The CARLA-WildLife (CWL) dataset is a synthetic dataset similar to StreetHazards [54], where freely available assets were inserted as anomalies. A third dataset, Wuppertal Obstacle Sequences (WOS), consists of real-world, but unlabeled sequences.

SOS and CWL provide labels for the tasks of semantic segmentation, instance segmentation and depth estimation. The datasets include semantic masks with binary as well as class-specific anomaly labels. Analogously to RoadObstacle21, the road represents the region of interest, thus, everything besides the road is assigned to the *void* class. Furthermore, both datasets include instance and depth masks. For SOS, 1,129 out of the 8,994 total frames were manually labeled. For CWL, also pixel-wise distance masks and fully annotated semantic masks are available. The anomalies in SOS belong to 13 anomaly types, *e.g. bag, umbrella* or *toy*, the anomalies in CWL to 18 anomaly types, *e.g. dogs, pylons* or *bags.*

3.1.4 Discussion

Besides the inconsistent definition of normality, which was already discussed in Subsection 3.1.2, existing anomaly datasets suffer from several issues, involving realism and domain shifts, a small dataset size as well as missing labels for the regular task.

The recording of anomalies in the real world is time-consuming, as they are rarely present in ordinary street scenes and thus, have to be selected and placed manually. Furthermore, anomalies that would lead to dangerous driving situations cannot be captured. Consequently, anomaly techniques such as data augmentation and simulation emerged to tackle these issues. Simulation has the advantage of having full control. Thus, anomalies certainly do not appear in the training data. However, a natural domain gap to reality exists, so anomaly detection methods that perform well on synthetic data are not implicitly reliable on real-world data. The same holds for data augmentation, which mixes two domains, leading to unrealistic results. For example, anomalies in WD-Pascal and StreetHazards are often placed in implausible locations or scaled in unrealistic ways. To ensure that methods really detect the anomalous objects that are pasted into the images, Fishyscapes pursues two strategies: Augmenting the Cityscapes images and pasting in objects from known classes. The first strategy prevents a method from only detecting pixels that differ from the non-augmented image, the second indicates whether only the domain shift is identified.

Datasets which include real-world scenes showing anomalies that fit the definition of normality, either recorded or collected from web resources, are usually very small. Lost and Found as the largest of these datasets only provides coarse annotations, followed by the Street Obstacle Sequences dataset, which however is highly redundant as the frames are extracted from 20 video sequences. These datasets are mainly for evaluation purposes and not for training. They also provide a wide variety of different anomaly types, which is beneficial for evaluating anomaly detection but is a hindrance for further processing of these anomalies, *e.g.* in terms of image retrieval, clustering and incremental learning. While Vistas-NP is a comparably large dataset, it is still not comparable to regular perception datasets with hundreds of thousands of frames. Generating larger datasets as a combination of similar datasets requires that those have a proper anomaly technique, definition of normality and labeling policy. Such datasets include FS Lost and Found, RoadAnomaly21, RoadObstacle21 and SOS. In particular, it is not possible to combine other datasets in a meaningful way due to different labeling policies.

3.2 Applications

Anomaly segmentation methods can be evaluated on the introduced datasets. The SegmentMeIfYouCan benchmark [21] provides a comparative overview of existing approaches by evaluating them on a pixel- and a segment-level. Usually, anomaly segmentation approaches yield per-pixel anomaly scores $u(x)_z$ for an image $x \in \mathcal{X}$, and the final anomaly prediction is obtained by thresholding on these scores. In [122, 123] a meta regression model has been developed which provides prediction quality scores on a segment-level, yielding anomaly segments as segments with poor prediction quality. On top of that, novel classes along the proposed anomalies are discovered by clustering in the feature space.

3.2.1 Anomaly Segmentation

As anomalous images or image regions are expected to correlate with high uncertainty, uncertainty estimation can be employed to compute anomaly scores. Some metrics for uncertainty quantification for deep neural networks, which derive directly from the softmax probabilities, are *e.g.* the maximum softmax probability [55], its extension ODIN [81] or the softmax entropy [55]. Another possibility for uncertainty quantification is to compute uncertainty scores in previous layers, for example by computing the Mahalanobis distance [77] of the latent features in the penultimate layer. Furthermore, approximations to Bayesian inference such as ensembles [73] or Monte Carlo dropout [40] treat model parameters as distributions and quantify the discrepancies in the softmax confidences of different models or inferences, respectively, *e.g.* by computing the mutual information score [100]. Other anomaly detection approaches leverage modifications of the model architecture [33], or exploit additional data, *e.g.* by incorporating an anomaly dataset during training, both for image classification [56, 96] and semantic segmentation [8, 22, 63]. Recent anomaly segmentation methods employ generative models to reconstruct or resynthesize the original input image and then identify the discrepancies between the original and reconstructed image [30, 34, 83, 84, 101, 142].

Pixel-Level Anomaly Segmentation Let $f_\theta : \mathcal{X} \to (0,1)^{H\times W\times Q}$ denote a semantic segmentation DNN as defined in Subsection 2.3.2 with underlying classes $\mathcal{Y} = \{1,\dots,Q\}$, but before the softmax activation $\sigma(\cdot) : \mathbb{R}^Q \to (0,1)^Q$. The *maximum softmax probability* (MSP) is a commonly-used baseline for OoD detection at image level [55]. It computes an anomaly score for each pixel $z \in \mathcal{Z}$ as

$$u(x)_z = 1 - \max_{q\in\mathcal{Y}} \sigma(f_\theta(x)_{z,q}),\ x \in \mathcal{X}\,. \tag{3.1}$$

Let $t \in \mathbb{R} \setminus \{0\}$ be a temperature scaling parameter and $\varepsilon \in \mathbb{R}$ a perturbation magnitude. *ODIN* [81] adds small perturbations to every pixel $z \in \mathcal{Z}$ of image x by

$$\tilde{x}_z = x_z - \varepsilon \mathrm{sign}\left(-\frac{\partial}{\partial x_z} \log \max_{q\in\mathcal{Y}} \sigma(f_\theta(x)_{z,q}/t)\right)\,. \tag{3.2}$$

Then, an anomaly score is obtained analogously to Equation 3.1 via the MSP as

$$u(x)_z = 1 - \max_{q\in\mathcal{Y}} \sigma(f_\theta(\tilde{x})_{z,q}/t)\,. \tag{3.3}$$

The *Mahalanobis distance* is computed for the features produced by the penultimate layer. Thus, let $f_{\theta^{(L-1)}}$ denote the penultimate layer of f_θ. Under the assumption that

$$P(f_{\theta^{(L-1)}}(x)_z | y_z = q) = \mathsf{N}(f_{\theta^{(L-1)}}(x)_z | \mu^q, \Sigma^q)\,, \tag{3.4}$$

an anomaly score for each pixel z can be computed as the Mahalanobis distance [77]

$$u(x)_z = \min_{q\in\mathcal{Y}}\ (f_{\theta^{(L-1)}}(x)_z - \hat{\mu}^q)^\intercal \hat{\Sigma}^{q^{-1}} (f_{\theta^{(L-1)}}(x)_z - \hat{\mu}^q)\,, \tag{3.5}$$

where $\hat{\mu}^q$ and $\hat{\Sigma}^q$ are estimates of the class mean μ^q and class covariance Σ^q, respectively, of the latent features in the penultimate layer. This Mahalanobis distance yields an estimate of the likelihood of a test sample with respect to the closest class distribution in the training data, which are assumed to be class-conditional Gaussians.

Anomaly segmentation via *Monte Carlo dropout* (MC dropout) considers discrepancies over multiple samplings. Let $M \in \mathbb{N}$ denote the number of Monte Carlo samplings and $\hat{q}_m := \sigma(f_\theta(x)_{z,q})$ the softmax probability of class $q \in \mathcal{Y}$ for sample $m \in \{1, \dots, M\}$. One can compute the predictive entropy as

$$\hat{E}(f_\theta(x)) = -\sum_{q \in \mathcal{Y}} \left(\frac{1}{M} \sum_{m=1}^{M} \hat{q}_m \right) \log \left(\frac{1}{M} \sum_{m=1}^{M} \hat{q}_m \right) . \tag{3.6}$$

As suggested in [100], the mutual information can then be used to define an anomaly score

$$u(x)_z = \hat{E}(f_\theta(x)) - \frac{1}{M} \sum_{q \in \mathcal{Y}} \sum_{m=1}^{M} \hat{q}_m \log \left(\hat{q}_m \right) . \tag{3.7}$$

In [33], an approach to learning the confidence with respect to the presence of anomalies was proposed. The *void classifier* approach adapts this by using the Cityscapes void class to approximate the anomaly distribution. A Cityscapes DNN $f_\theta : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times (Q+1)}$ is trained with an additional class, *i.e.*, a dustbin [154]. Then, the anomaly score for each pixel $z \in \mathcal{Z}$ is computed as the softmax score for the void class, which yields

$$u(x)_z = \sigma(f_\theta(x)_{z,Q+1}), \ x \in \mathcal{X} . \tag{3.8}$$

Starting from a pretrained DNN, *entropy maximization* introduces a second training objective to maximize the softmax entropy on OoD pixels [22, 56, 63]. This yields the multi-criteria loss function

$$\begin{aligned} (1-\lambda)\mathbb{E}_{(x,y)\sim\mathcal{X}_{in}} \left[\ell_{in}(\sigma(f_\theta(x)_z), y_z) \right] + \\ \lambda\mathbb{E}_{x'\sim\mathcal{X}_{out}} \left[\ell_{out}(\sigma(f_\theta(x')_z)) \right], \ \lambda \in [0, 1] , \end{aligned} \tag{3.9}$$

where ℓ_{in} is the empirical cross entropy and ℓ_{out} the averaged negative log-likelihood over all classes for the in-distribution data $\mathcal{X}_{in}$ and the out-distribution data $\mathcal{X}_{out}$, respectively. To approximate $\mathcal{X}_{out}$, a subset of the COCO dataset [82] is used whose images do not depict any object classes also available in $\mathcal{X}_{in}$, which is the Cityscapes dataset [29]. The COCO subset together with the Cityscapes training data are then included into a tender retraining of the pretrained Cityscapes model. The anomaly score is then computed via the softmax entropy as

$$u(x)_z = -\sum_{q \in \mathcal{Y}} \sigma(f_\theta(x)_{z,q}) \log \left(\sigma(f_\theta(x)_{z,q}) \right) . \tag{3.10}$$

Segment-Level Anomaly Segmentation Given that the GT is available, the prediction quality of a semantic segmentation DNN can be easily quantified by computing the IoU, *cf.* Equation 2.63. Meta regression [122, 123] can be applied as a post-processing tool to estimate the prediction quality for unlabeled data. Therefore, the predicted segmentation mask for an image $x \in \mathcal{X}$ is broken down into its segments $s \subseteq x$, which are connected pixels $x_z,\ z \in \mathcal{Z}$ that share the same class prediction. Next, pixel-wise anomaly scores are computed and aggregated over each component, yielding

$$u(s) = \frac{1}{|s|} \sum_{z:\, x_z \in s} u(x)_z,\ \forall s \subseteq x \ . \tag{3.11}$$

In addition, geometrical information about the components are considered, including *e.g.* the segment size $|s|$, its location in the image or the class predictions for neighboring pixels. Altogether, structured arrays for each segment in an image (batch) are obtained as input for the regression model. Given a labeled training dataset, the adjusted IoU is computed for all segments, composing the target value for the training of the meta regressor. The inversed meta regression output at test time can be used as an anomaly score

$$\hat{u}(s),\ s \subseteq x \ , \tag{3.12}$$

analogously to the pixel-level approaches. A visual comparison of the introduced anomaly scores is provided in Figure 3.5.

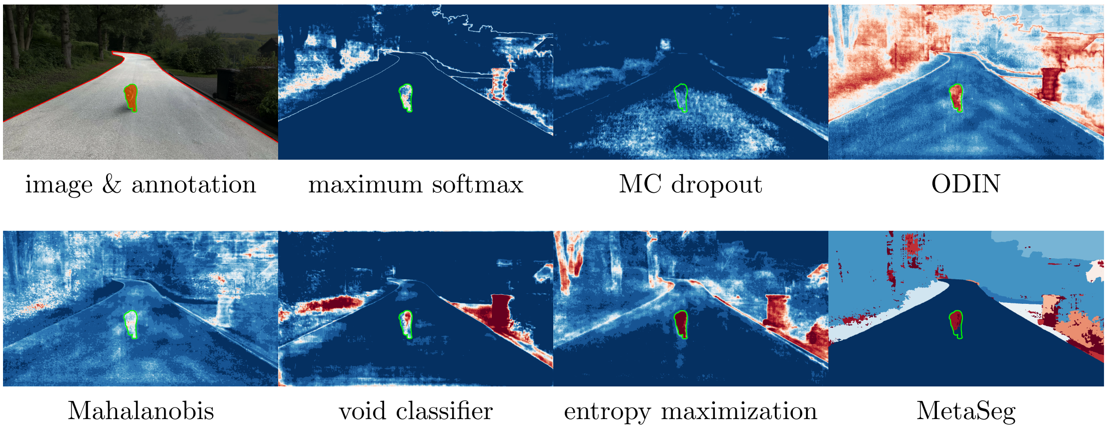


image & annotation

maximum softmax

MC dropout

ODIN

Mahalanobis

void classifier

entropy maximization

MetaSeg

Figure 3.5: Collection and visualization of anomaly segmentation methods.

Evaluation Metrics Anomaly segmentation methods yield per-pixel anomaly scores $u(x) \in \mathbb{R}^{|\mathcal{Z}|}$ for an image $x \in \mathcal{X}$ to discriminate between the two classes $1 \mathrel{\widehat{=}}$ *anomaly* and $0 \mathrel{\widehat{=}}$ *non-anomaly*. The final class prediction is obtained by

$$\hat{y}(\tau)_z = \mathbb{1}_{\{u(x)_z \geq \tau\}},\ \tau \in \mathbb{R},\ \forall z \in \mathcal{Z} \ . \tag{3.13}$$

We evaluate the separability of the pixel-wise anomaly scores via the area under the precision-recall curve (AuPRC).

Then, for the anomaly class $q \mathrel{\widehat{=}}$ *anomaly* we compute the precision and recall scores as defined in Equation 2.61, but considered as functions of τ. The AuPRC approximates $\int \text{precision}(\tau)\, d\text{recall}(\tau)$ and is threshold independent [13]. It also puts emphasis on detecting the minority class, making it particularly well suited as our main evaluation metric since the pixel-wise class distributions of RoadAnomaly21 and RoadObstacle21 are considerably unbalanced.

To consider the safety point of view, we also include the false positive rate at 95% true positive rate (FPR_{95}) in our evaluation, where the true positive rate (TPR) is equal to the recall of the anomaly class. The false positive rate (FPR) is the number of pixels falsely predicted as anomaly over the number of all non-anomaly pixels. Hence, for the anomaly class q we compute

$$\text{FPR}_{95} = \frac{\text{FP}_q}{\text{FP}_q + \text{TN}_q} \quad \text{s.t.} \quad \text{TPR} = 0.95 \ , \tag{3.14}$$

also depending on the threshold $\tau \in \mathbb{R}$. The metric FPR_{95} indicates how many false positive predictions are necessary to guarantee a desired true positive rate. Note that, any prediction which is contained in a ground truth labeled region of class void is not counted as false positive. In particular for the RoadObstacle21 dataset the evaluation is therefore restricted to the road area.

3.2.2 Anomaly Clustering

Anomaly segmentation can be also used as a pre-processing step for further tasks, *e.g.* for tracking, clustering or retrieval. Retrieval methods in general tackle the task of seeking related samples from a large database corresponding to a given query. Early works in this context aim to retrieve images that match best a query text or vice versa [3, 48, 58, 93]. Another subtask deals with content-based image retrieval, which can be sub-categorized into instance- and category level retrieval. This is, given a query image depicting an object or scene, retrieving images representing the same object/scene or objects/scenes of the same category, respectively. To this end, these images must satisfy some similarity criteria based on some abstract description. In a first approach called QBIC [38], images are retrieved based on (global) low level features such as color, texture or shape. More advanced approaches utilize local level features [4, 87], still they cannot fully address the problem of semantic gap [129], which describes the disparity between different representation systems [52]. Recent methods such as [91, 103] apply machine/deep learning to learn visual features directly from the images instead of using hand-crafted features.

In the following, images are not directly retrieved for some given query image. Instead, all objects/images that are contained in the database are clustered based on their visual similarity, as it has been proposed in [108]. This particularly includes OoD objects. Furthermore, the single frame based approach is extended to video sequences, enhancing the effectiveness by incorporating tracking information over multiple frames. Therefore, the detected OoD objects are passed into a feature extractor, resulting in a time series of feature vectors on which we employ a low dimensional embedding via the t-SNE algorithm, *cf.* Subsection 2.4.1. Here the time series viewpoint makes it easy to clean the data and avoid false positives, *e.g.* by setting a filter to the minimum length. Clustering of similar objects, either on the basis of frames or on time series meta-clusters enables the retrieval of previously unseen objects [108, 134] which enables novelty detection. In the following, we apply this on the SOS and the CWL datasets as well as on self-recorded unlabeled data that contains OoD road obstacles. This provides a first method that enables the unsupervised detection of potentially critical situations or corner cases related to OoD objects from video data.

Evaluation Metrics The evaluation of OoD object clusters $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$, which contain the two-dimensional representatives of the segments s of OoD object predictions, depends on the differentiation level of these objects. We consider an instance level and a semantic level based on object classes. Let $\mathcal{Y} = \{1, \ldots, Q\}$ and $\mathcal{Y}^{\mathrm{ID}} = \{1, \ldots, P\}$ denote the set of semantic class and instance IDs, respectively. For some proposed OoD segment s, y_s and y_s^{ID} correspond to the ground truth class and instance ID with which s has the highest overlap. On instance level, we aspire that OoD objects which belong to the same instance in an image sequence are contained in the same cluster. This is, we compute the relative amount of OoD objects per instance in the same cluster,

$$CS_{\mathrm{inst}} = \frac{1}{P} \sum_{i=1}^{P} \frac{\max\limits_{\mathcal{U}_j \in \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}} |\{s \in \mathcal{U}_j \mid y_s^{\mathrm{ID}} = i\}|}{\sum\limits_{\mathcal{U}_j \in \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}} |\{s \in \mathcal{U}_j \mid y_s^{\mathrm{ID}} = i\}|} \in [0, 1] \ , \tag{3.15}$$

averaged over all instances. On a semantic level, we pursue two objectives. The first concerns the semantic class impurity of the clusters,

$$CS_{\mathrm{imp}} = \frac{1}{n} \sum_{i=1}^{n} |\{y_s | s \in \mathcal{U}_i\}| \in [1, Q] \ , \tag{3.16}$$

averaged over all clusters $\mathcal{U}_i \in \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$. Secondly, we aspire a low fragmentation of classes into different clusters

$$CS_{\text{frag}} = \frac{1}{Q} \sum_{i=1}^{Q} \left| \{ \mathcal{U}_j \in \{\mathcal{U}_1, \ldots, \mathcal{U}_n\} | \exists s \in \mathcal{U}_j : y_s = i \} \right| , \qquad (3.17)$$

i.e., ideally, each class constitutes exactly one cluster. Here, we average over the semantic classes in $\mathcal{Y}$.

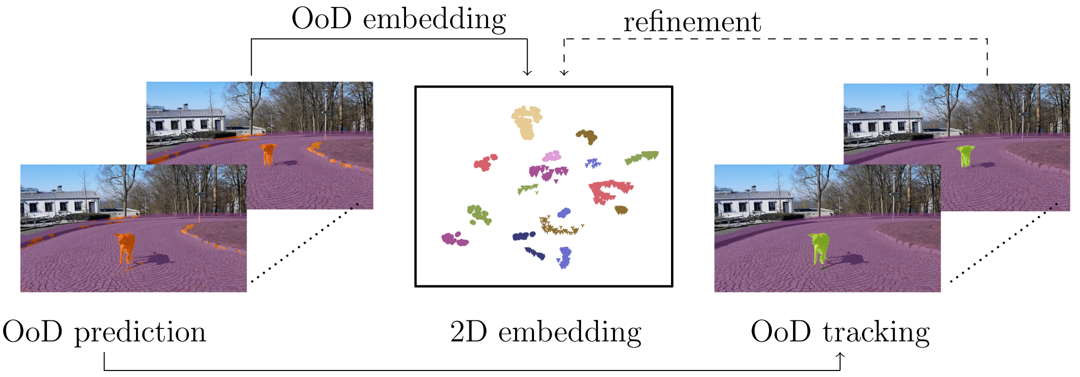


Figure 3.6: Method overview: The OoD prediction is used to produce the tracking IDs and a 2D embedding. The tracking information can be used for a refinement of the embedding space.

Method On top of segmentation and tracking of OoD objects, we perform a method similar to content-based image retrieval in order to form clusters of the OoD objects that constitute novel semantic concepts. To this end we adapt an existing approach [108, 134] to video sequences by incorporating the tracking information which we obtain *e.g.* as described in [88]. That is, we require the tracking information to be available for each frame x and apply OoD object retrieval as a post-processing step which does not depend on the underlying semantic segmentation network nor on the OoD segmentation method but on given OoD segmentation masks, *cf.* Figure 3.6.

For each frame x and OoD segment $s \subseteq x$, let $\hat{y}_s^{ID}$ denote the predicted tracking ID. To diminish the number of the false positives, we only cluster predicted segments that are tracked over multiple frames of an image sequence $\{x_t\}_{t=1}^{T}$, based on some length parameter $\iota \in \mathbb{N}$. Further, each frame x is tailored to boxes around the remaining OoD segments s, which are vertically bounded by the pixel locations $\min_{(h,w):x_{(h,w)}\in s} h$ and $\max_{(h,w):x_{(h,w)}\in s} h$, horizontally by $\min_{(h,w):x_{(h,w)}\in s} w$ and $\max_{(h,w):x_{(h,w)}\in s} w$. Image clustering usually takes place in a lower dimensional latent space due to the curse of dimensionality. To this end, the image patches are fed into an image classification ResNet152 [51] (without its final classification layer) trained on ImageNet [32], which produces feature vectors of equal size regardless of the input dimension. These features are projected into a low-dimensional space by successively applying two dimensionality reduction techniques, namely PCA and t-SNE. As final step, the retrieved OoD object predictions are clustered in the low-dimensional space, *e.g.* via the DBSCAN clustering algorithm, *cf.* Subsection 2.4.1.

Experiments Next, we evaluate the clustering of OoD segments obtained by some OoD object segmentation method. In Table 3.2, we report the clustering metrics CS_{inst}, CS_{imp} and CS_{frag} with ($\iota = 10$) and without ($\iota = 0$) incorporating the OoD tracking information, respectively. For both, the CWL and the SOS dataset, all clustering metrics improve when applying the OoD tracking as a pre-processing step. A reason for this is, that the tracking information "tidies up" the embedding space, *e.g.* by removing noise, which enhances the performance of the clustering algorithm. For CWL (with 18 object types), 1266/1026 OoD segments are clustered into 22/23 clusters without/with using tracking results, for SOS (with 13 object types), we obtain 23/24 clusters which contain 1437/888 OoD segments in total. For the clustering, we applied the DBSCAN algorithm with hyperparameters $\varepsilon = 4.0$ and $\delta = 15$.

	without tracking			with tracking ($\iota = 10$)		
dataset	$CS_{\text{inst}} \uparrow$	$CS_{\text{imp}} \downarrow$	$CS_{\text{frag}} \downarrow$	$CS_{\text{inst}} \uparrow$	$CS_{\text{imp}} \downarrow$	$CS_{\text{frag}} \downarrow$
SOS	0.8652	2.5217	2.8182	0.8955	1.7917	1.9091
CWL	0.8637	2.8181	2.2500	0.8977	2.1739	1.8000

Table 3.2: Object clustering results for the SOS and the CWL dataset. We report results for clustering with and without incorporating the object tracking information.

Retrieval of OoD Objects for WOS In addition to the labeled datasets SOS and CWL, we applied our toolchain to another dataset which we abbreviate as WOS. As this dataset does not include any annotated data, it serves as a test scenario, only. This is, we do not provide any evaluation results, but some visualizations of the retrieved clusters. We trained two meta classifiers on SOS and CWL, respectively. Since the results for both meta classification models are similar and the domain shift between SOS and WOS is less, we limit our visualizations onto this respective meta model, while increasing the minimal tracking length to $\iota = 25$.

As illustrated in Figure 3.7, we are able to retrieve clusters constituted of OoD objects such as dogs, *cf.* Figure 3.8. Our dataset includes three different dogs, that are visible in multiple scenes. We observe that these three dogs do not constitute one overall dog cluster, however, each of them forms a cluster containing multiple sequences, as well as different postures, sizes/distances, backgrounds and

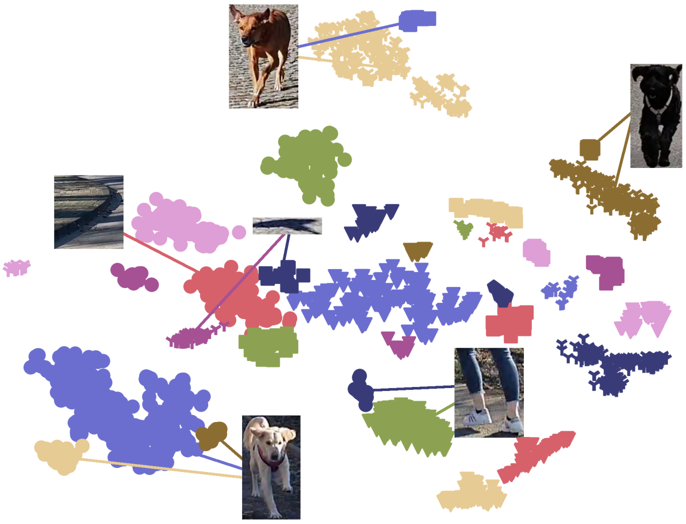


Figure 3.7: Clustering of proposed OoD segments (plus some example images) via DBSCAN in the feature space for a minimum tracking length $\iota = 25$.

perspectives. Moreover, some retrieved clusters represent OoD objects like balls, bags or skateboards. Further, we discover many false positive OoD predictions such as humans, sidewalks, manhole covers or shadows.

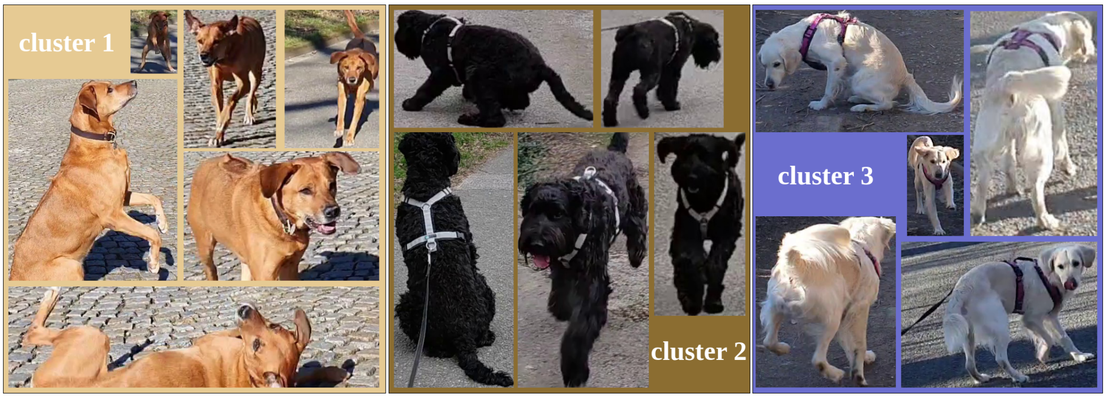


Figure 3.8: Example images taken from three different clusters, all representing the overall category *dog*.

Chapter 4

Towards Unsupervised Open World Semantic Segmentation

For the semantic segmentation of images, state-of-the-art deep neural networks (DNNs) achieve high segmentation accuracy if that task is restricted to a closed set of classes. However, as of now DNNs have limited ability to operate in an open world, where they are tasked to identify pixels belonging to unknown objects and eventually to learn novel classes, incrementally. Humans have the capability to say: “I don’t know what that is, but I’ve already seen something like that”. Therefore, it is desirable to perform such an incremental learning task in an unsupervised fashion. We introduce a method where unknown objects are clustered based on visual similarity. Those clusters are utilized to define new classes and serve as training data for unsupervised incremental learning. More precisely, the connected components of a predicted semantic segmentation are assessed by a segmentation quality estimate. Connected components with a low estimated prediction quality are candidates for a subsequent clustering. Additionally, the component-wise quality assessment allows for obtaining predicted segmentation masks for the image regions potentially containing unknown objects. The respective pixels of such masks are pseudo-labeled and afterwards used for re-training the DNN, *i.e.*, without the use of ground truth generated by humans. In our experiments we demonstrate that, without access to ground truth and even with few data, a DNN’s class space can be extended by a novel class, achieving considerable segmentation accuracy.

source code & data: https://github.com/SUhlemeyer/novelty-learning

4.1 Introduction

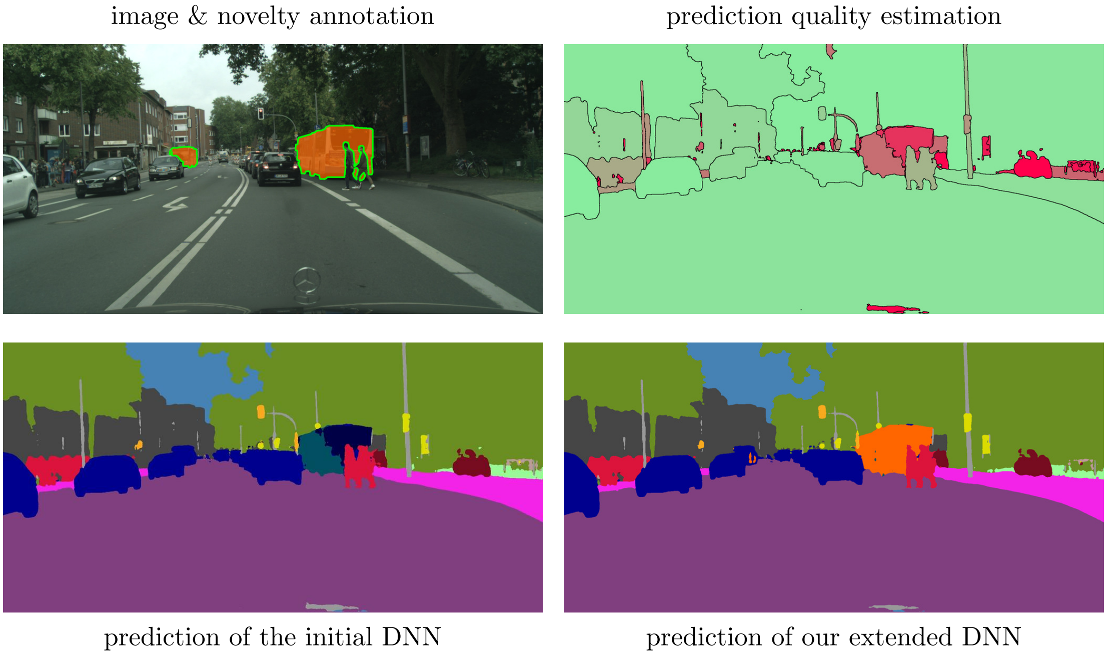


image & novelty annotation
prediction quality estimation
prediction of the initial DNN
prediction of our extended DNN

Figure 4.1: Comparison of the semantic segmentation predictions of an initial DNN (bottom left) whose semantic space does not include the category *bus* and a DNN which is incrementally extended by this novel class (bottom right, novel class in orange) for an image from the Cityscapes dataset. The novel class is highlighted in orange (top left). Further, the initial prediction exhibits a low prediction quality (top right) on pixels belonging to the novel objects, which is indicated by red color.

Semantic segmentation is a computer vision task that terms the classification of image data on pixel level, *cf.* Subsection 2.3.2. State-of-the-art approaches are based on deep convolutional neural networks (DNNs) [25, 137, 155] introduced in Subsection 2.2.2, benefiting from finely annotated datasets, *e.g.* for automated driving [29, 42, 105, 150]. However, DNNs for semantic segmentation are usually trained on a predefined, closed set of classes. This closed world setting assumes, that all classes present during testing were already included in the training set. In an open world setting, this assumption does not hold. In particular for safety-critical open-world applications like perception systems for automated driving, it is indispensable that neural networks recognize previously unseen objects instead of wrongly assigning them to *one-of-the-known* classes. In addition, they must constantly adapt to evolving environments.

Some terms often used interchangeably for anomaly are *outlier*, *out-of-distribution* (OoD) object and *novelty*. As there is no clear convention on how to distinguish these terms, we define them as subcategories of anomalies: outliers and OoD objects denote noise or samples drawn from another distribution than the model was trained on, respectively. In this work, we are seeking novelties, which we define as previously-unseen objects that constitute a new concept, *i.e.*, objects of the same category appear frequently. In automated driving, detecting and learning those novel classes becomes necessary, *e.g.* due to new appearances like e-scooters or due to local specialities like boat trailers near the sea. The concept of detecting and learning novelties was first introduced in [6] as *open world recognition*. Open world recognition for different computer vision tasks is an emerging research area [6, 19, 62, 128], still only little explored for unsupervised methods [50, 104], yet.

We propose a new and modular procedure for learning new classes of novel objects without any handcrafted annotation:

1. Anomaly segmentation to detect suspicious objects,
2. clustering of potentially novel objects,
3. creation of so-called *pseudo-labels*, and
4. incremental learning of novel classes.

In the following, we will outline each of these four steps in more detail.

For the first step, we post-process the predictions of an underlying semantic segmentation DNN via a *meta regressor*, that estimates the quality of the predicted segments, similar as proposed in [89, 122, 123]. In the following, the term **segment** will always refer to connected components of pixels in the semantic segmentation prediction. The segment-wise quality score is obtained on the basis of aggregated dispersion measures and geometrical information, *i.e.*, without requiring ground truth. The output of the semantic segmentation DNN on anomalous objects is often split into several segments. To this end, we first aggregate neighboring segments, *i.e.*, segments that have at least one adjacent pixel each, with quality estimates below some threshold, into (potentially) anomalous objects, termed **suspicious objects**.

For the second step, we adapt the idea introduced in [108] and presented in Subsection 3.2.2 to gather segments with poor prediction quality and to cluster them into visually related neighborhoods. Therefore, all suspicious objects (of sufficient size) are cropped out in the RGB images and the resulting image patches are fed into a convolutional neural network (CNN), *e.g.* for image classification. Whether an image patch is sufficiently large depends on the minimum input size required by this CNN. To obtain comparable information about the suspicious objects, we then extract the features provided by the penultimate layer of the CNN, *i.e.*, right before the final classification layer. By reducing the dimensionality of these features up to two, we enable the use of low-dimensional, unsupervised clustering techniques, such as [36, 90].

As third, we obtain pseudo-labels for novel classes in an automated manner: each (large / dense enough) cluster constitutes a novel category, and each pixel belonging to a clustered object is assigned to the appropriate (not necessarily named) class. More precisely, the prediction of the segmentation model is updated at those pixel positions to the next "free" label ID.

Finally, the segmentation network is incrementally extended by these novel classes, *cf.* Figure 4.1. To this end, we apply established incremental learning methods [57, 119]. However, these are mainly examined for supervised learning tasks, while we do not include any hand-labeled new data. This last two steps were never done in literature so far.

We perform five experiments, following a hierarchical structure of complexity. For the first three experiments, the initial segmentation network is trained on the Cityscapes dataset, but on different subsets of the available training classes. Here, we do not change the data itself, but the training IDs of the Cityscapes classes, *i.e.*, we create artificial anomalies by class exclusion, *cf.* Subsection 3.1.1. For the other experiments, we start with an initial segmentation network that is trained on Cityscapes and test our method on the A2D2 dataset. For those, we have a mapping between the Cityscapes and the A2D2 classes. For most Cityscapes classes, there is a matching class in A2D2. In some cases, A2D2 has coarser classes, *e.g.* we map the Cityscapes classes *vegetation* and *terrain* to the A2D2 class *nature*.

To outline our contributions, we demonstrate in our experiments that our method is able to incrementally extend a neural network by novel classes without collecting or annotating novelties manually. To the best of our knowledge, we are the first to introduce an unsupervised approach for open world semantic segmentation with DNNs. Fine-tuning neural networks on automatically created pseudo-labels instead of human-made annotations is economically valuable. We observe in all experiments, that even a poor labeling quality is sufficient to learn novel classes, achieving IoU values around 40%. Further, the amount of new data was less mostly than 100 images, respectively. Unsupervised open world semantic segmentation therefore is a powerful tool for open world applications, that provides an enormous potential for future improvement.

4.2 Related Works

In this section, we first review anomaly detection methods and briefly go into class discovery approaches. Then we describe different strategies for class-incremental learning. Finally, we give an overview of existing work on open world computer vision tasks.

Novelty Detection The detection of anomalous objects in general is a key task in many machine learning applications. Early works estimate the prediction uncertainty, *e.g.* by uncertainty measures derived from the softmax probability [55, 81]. Uncertainty-based approaches can be further improved by integrating anomalous data into the training procedure [22, 33]. Another line of works employs generative models such as autoencoders (AEs) or generative adversarial models (GANs) to reconstruct or synthesize images and measure the reconstruction quality. Various of those novelty detection methods are described in [136], not only reconstruction-, but also density- or distance-based. A benchmark for anomaly segmentation, *i.e.*, anomaly detection methods for semantic segmentation, was recently published in [21], providing a cleaner comparison of proposed methods. Given a set of anomalies, the prevailing approach for class discovery is to form clusters based on some similarity measure or intrinsic features with traditional clustering methods. A detailed survey of image clustering has been published in [85].

Class-Incremental Learning Class-incremental learning refers to the extension of a neural network's semantic space by further, previously unknown, classes. This extension is achieved by fine-tuning a model on additional, usually human-annotated data [64, 68, 80, 97], whereas in this work we only provide pseudo-labels for these new images. The primary issue to tackle when re-training a neural network is to mitigate the performance loss on previously learned classes, commonly known as catastrophic forgetting [94]. To this end, we employ two different strategies: first, we penalize large variations of the softmax output (compared to the one of the original network) [57], second we utilize a subset of the previously-seen training data [119].

The first strategy belongs to the category of regularization based approaches, or more specifically to knowledge distillation methods. These were originally developed to distill knowledge from sophisticated into simpler models [57], *i.e.*, for model compression. Thereupon, distillation methods have evolved for incremental learning in image classification [64, 65, 78, 80, 148], some of which were later adapted to semantic segmentation [68, 97, 130].

The second approach belongs to so-called rehearsal methods [119], where old training data is included in the re-training process [17, 117].

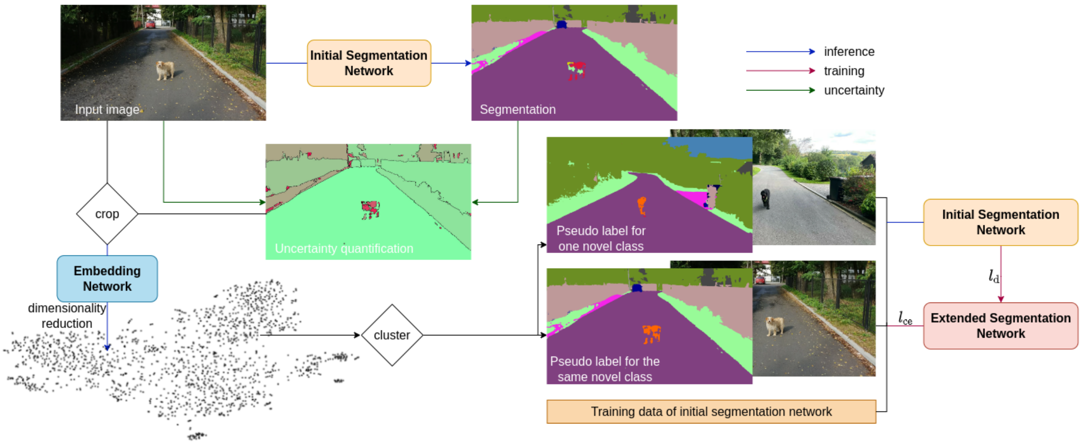

Input image
Initial Segmentation Network
Segmentation
inference
training
uncertainty
crop
Uncertainty quantification
Pseudo label for one novel class
Initial Segmentation Network
l_d
Embedding Network
dimensionality reduction
cluster
l_{ce}
Extended Segmentation Network
Pseudo label for the same novel class
Training data of initial segmentation network

Figure 4.2: Illustration of the overall framework.

Open World The open world setting was first introduced in [6] for image classification. The authors formally define the solution of open world recognition problems as a tuple, consisting of a recognition function, a novelty detector, a labeling process and an incremental learning function. Ideally, these steps should be automated, however, most approaches presume a supervised setting, *i.e.*, they require ground truth for detected novelties. In summary, open world recognition covers the entire process from discovering up to learning novel classes.

A supervised solution for open world object detection is presented in [62], based on contrastive clustering, an unknown-aware proposal network and energy based unknown identification. A similar approach was proposed in [19] for open world semantic segmentation, where novel classes are learned via few-shot learning. In [50], an unsupervised method to obtain pseudo-labels for image classification based on cluster assignments is introduced. There exists also some prior work for unsupervised open world semantic segmentation [104], however, the segmentation mask is obtained via agglomerative clustering of superpixels and there is no update of the neural network at all. While it is capable of creating ad hoc novel classes unsupervised on given images, it does not create a consistent semantic category over multiple images.

Our work introduces an open world semantic segmentation framework, where a neural network is incrementally extended by novel classes. These classes are discovered **and** labeled without any human effort. Therefore, our work goes beyond all existing approaches in this research area.

4.3 Discovery of Unknown Semantic Classes

Whether a class is novel or not depends on the neural network's underlying set of known classes $\mathcal{Y} = \{1, \dots, Q\}$. Let $f_\theta : \mathcal{X} \to (0,1)^{|\mathcal{Z}| \times Q}$ be a semantic segmentation DNN which is trained on the classes in $\mathcal{Y}$, mapping an image $x \in \mathcal{X} \subseteq [0,1]^{|\mathcal{Z}| \times 3}$ onto its softmax probabilities for each pixel $z \in \mathcal{Z}$. Then, $f_\theta(x)_{z,q} \in (0,1)$ denotes the probability with which the model f_θ assigns some pixel x_z, $z \in \mathcal{Z}$ to a class $q \in \mathcal{Y}$. As decision rule, we apply the argmax function, *i.e.*, we obtain the semantic segmentation mask $\hat{m}(x) \in \mathcal{Y}^{|\mathcal{Z}|}$ with $\hat{m}(x)_z = \mathrm{argmax}_{q \in \mathcal{Y}} f_\theta(x)_{z,q}$. In the following, we will estimate the prediction quality on a segment-level instead of pixel-wise, employing a meta regression approach that was first introduced in [122]. On that account, we denote a segment, *i.e.*, a connected component of pixels that share the same class in $\hat{m}(x)$, as $s \subseteq x$.

4.3.1 Meta Regressor

As model for the meta regressor we apply the gradient boosting from the `scikit-learn v.0.24.2` library using the standard settings. The training datasets contain from 67 to 75 uncertainty metrics depending on the number of classes. We train on $313,720$ to $946,318$ segments. Further details on the definition of the segment-wise metrics, the exact size of the training data and the tree models obtained are provided in the Appendix. For any predicted segment k, the gradient boosting regressor, via clipping, outputs a value between 0 and 1, where a value close to 0 expresses low, a value close to 1 high prediction quality.

The motivation to use a segment-wise meta regression framework is to identify segments with low predicted IoU as candidate segments that potentially stem from OoD objects.

4.3.2 Uncertainty Metrics and Prediction Quality Estimation

We consider novelties as *none-of-the-known* objects, *i.e.*, they differ semantically from the model's training data. Assuming that the segmentation DNN produces unstable predictions on these unexplored entities, various measurable phenomena occur. For instance, the model exhibits a high prediction uncertainty. This is quantified by dispersion measures as the softmax entropy, probability margin or variation ratio, which we compute pixel-wise via

$$E_z(f_\theta(x)) = -\frac{1}{\log(Q)} \sum_{q \in \mathcal{Y}} f_\theta(x)_{z,q} \log(f_\theta(x) z, q) \;, \tag{4.1}$$

$$D_z(f_\theta(x)) = 1 - \max_{q\in\mathcal{Y}} f_\theta(x)_{z,q} + \max_{q\in\mathcal{Y}\setminus\{m(x)_z\}} f_\theta(x)_{z,q} \ , \tag{4.2}$$

$$V_z(f_\theta(x)) = 1 - \max_{q\in\mathcal{Y}} f_\theta(x)_{z,q} \ , \tag{4.3}$$

respectively. These are then averaged over the segments s or over the segment boundary. Moreover, we examine some geometrical properties of the segments, such as their size, *i.e.*, the number of pixels $|s|$ contained in s, their shape or their position in the image. For in-depth details on the constructed metrics, we refer to [122]. By feeding these metrics into a meta regression model, we obtain prediction quality estimates for each segment s, which we denote by $1 - \hat{u}(s) \in [0,1]$, *cf.* Equation 3.12. These quality estimates approach the true segment-wise *Intersection over Union* (IoU) with reasonably high accuracy [122]. To fit the meta regressor, we compute the metrics plus the true IoU values of all segments included in the training data of the segmentation network. This meta model is then applied to unseen data, *i.e.*, data that was not included in the training of f_θ, for the purpose of anomaly segmentation. Here, we consider a segment s to be anomalous, if its quality score is below some predefined threshold $\tau \in [0,1]$, *i.e.*, if $\hat{u}(s) \geq \tau$. By that, we identify individual segments as unknown, however, the semantic segmentation of unknown objects usually consists of several segments, *i.e.*, of different predicted classes. As we can uniquely assign each pixel x_z to a segment s, we obtain a binary pixel-wise classification mask $\hat{a} \in \{0,1\}^{|\mathcal{Z}|}$ by

$$\hat{a}_z = \mathbb{1}_{\{\hat{u}(s)\geq\tau\}}, \ x_z \in s \ \forall z \in \mathcal{Z} \ , \tag{4.4}$$

where the class prediction $\hat{a}_z = 1$ indicates anomalous pixels. Finally, the connected components in the anomaly mask $\hat{a}$ merge adjacent anomalous segments into suspicious objects. Under ideal conditions,

1. the semantic segmentation network performs perfectly on in-distribution data,
2. the meta model detects all (but only) unknowns, and
3. novel objects of different classes are separable.

4.3.3 Embedding and Clustering of Image Patches

Image clustering usually takes place in a lower dimensional latent space due to the curse of dimensionality. To this end, we feed image patches tailored to the suspicious objects into an image classification DenseNet201 [59], which is trained on the ImageNet dataset [32] with 1000 classes. The patches are not equally sized. That nevertheless the DenseNet feature extractor returns features of equal size $(1,920)$ for each patch is a consequence of the application of the AdaptiveAvgPool2d layer that is applied as the last layer after the fully convolutional and

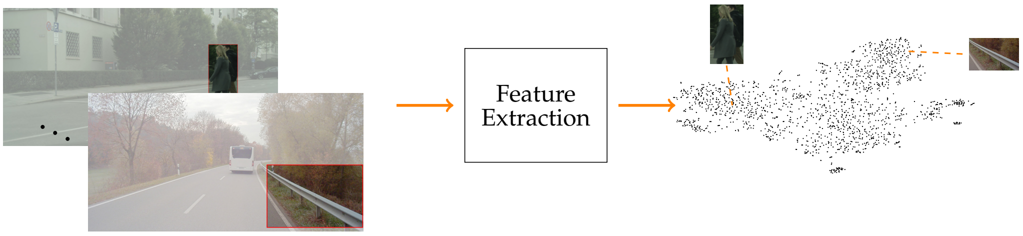


Figure 4.3: Coarse illustration of the feature extraction process. Detected unknown objects, *e.g.* humans and guardrails, are cropped out (indicated by the red box). The image patches are passed to an encoder, producing feature vectors which are projected into a two-dimensional space.

depth-wise interconnected layers of the DenseNet. Put shortly, this last layer pools over both spatial dimension of the feature maps and thereby the output is not dependent on the size of the input, that is transported through the fully convolutional layers. Their feature representations are further compressed, resulting in a two-dimensional embedding space as illustrated in Figure 4.2. We apply two commonly used dimensionality reduction techniques. For complexity reasons, we compute the first 50 principal components [112] before deploying the better performing *t-SNE* method [135] with Euclidean distance as similarity measure.

This procedure for image embedding is adopted from [108], where the authors evaluated several feature extractors, distance metrics and feature dimensions. We employ the best performing setup in this quantitative analysis to obtain clusters of visually related image patches. Beyond that, we identify these clusters using the *DBSCAN* algorithm, *cf.* Subsection 2.4.1. The cluster with the most remaining core points (or all clusters that involve "enough" core points) will be used to extend the segmentation network by new classes.

4.3.4 Novelty Segmentation

Using pseudo-labels instead of manually annotated targets is a cost-efficient (in the sense of human effort) method of training neural networks on unlabeled data. For the sake of simplicity we assume that exactly one cluster is returned by the aforementioned procedure. For some image $x \in \mathcal{X}$, we denote the predicted segmentation mask by $\hat{m}(x)$ and the respective segments by s. Let $\mathcal{U}_1, \dots, \mathcal{U}_k$ denote the proposed clusters. If $\exists s \subseteq x$ and $i \in \{1, \dots, k\}$, so that $s \in \mathcal{U}_i$, *i.e.*, if image x (probably) contains a novel class, we include the tuple $(x, \tilde{y}) \in \mathcal{X} \times \{1, \dots, Q, Q+1, \dots, Q+k\}^{|\mathcal{Z}|}$ into the re-training data $\mathcal{S}^{\text{novel}}$ for learning

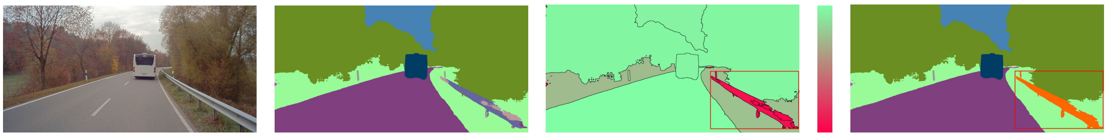


image from A2D2

semantic segmentation prediction

prediction quality estimation from 0 (red) to 1 (green)

pseudo ground truth

Figure 4.4: Novelty segmentation: example for obtaining pseudo ground truth with regard to some image patch (outlined in red) of image x. If segments inside the red box exhibit quality estimates below some predefined threshold, they are "re-labeled" in the segmentation mask $m(x)$.

the novel classes $Q+1,\dots,Q+k$. Here, $\tilde{y}$ denotes the pseudo-label, where

$$\tilde{y}_z = \begin{cases} Q+i & \text{, if } s \in \mathcal{U}_i,\ x_z \in s,\ i \in \{1,\dots,k\} \\ \hat{m}(x)_z & \text{, otherwise} \end{cases} , \tag{4.5}$$

i.e., a pixel z is either assigned to a novel class ID $Q+1,\dots,Q+k$, or to the class $q \in \mathcal{Y}$ that was predicted by the initial model f_θ. An example for acquiring pseudo ground truth for one image is given in Figure 4.4. In the following section we extend the segmentation DNN f_θ by fine-tuning it on $\mathcal{S}^{\text{novel}}$.

4.4 Extension of the Model's Semantic Space

In this section we describe our approach to semantic incremental learning with the pseudo ground truth acquired by novelty segmentation. Starting from our initial segmentation model f_θ, we seek an extended model $f_{\theta^+} : \mathcal{X} \to (0,1)^{|\mathcal{Z}|\times(Q+k)}$ that retains the knowledge of f_θ while additionally learning the novel classes $Q+1,\dots,Q+k$. Denote the extended semantic space by $\mathcal{Y}^+ = \mathcal{Y} \cup \{Q+1,\dots,Q+k\}$. In more detail, we replace the ultimate layer $f_{\theta^{(L)}}$ and reinitialize only the affected weights to obtain the initial model f_{θ^+} for re-training, *i.e.*, the model we train on the newly collected data $\mathcal{S}^{\text{novel}}$. As loss function we apply a weighted cross entropy loss [149], denoted by $\ell_{\text{ce},\omega}$. The class-wise weights $\omega_q \in (0,1]$, $q \in \mathcal{Y}^+$, are recalculated for each batch based on the inverse class frequency to alleviate class imbalances.

To mitigate the problem of catastrophic forgetting [94], we pursue two strategies, namely knowledge distillation [57] and rehearsal [119].

Knowledge distillation in class-incremental learning aims at minimizing variations of the softmax output restricted to only the old classes $q \in \mathcal{Y}$. This is realized by

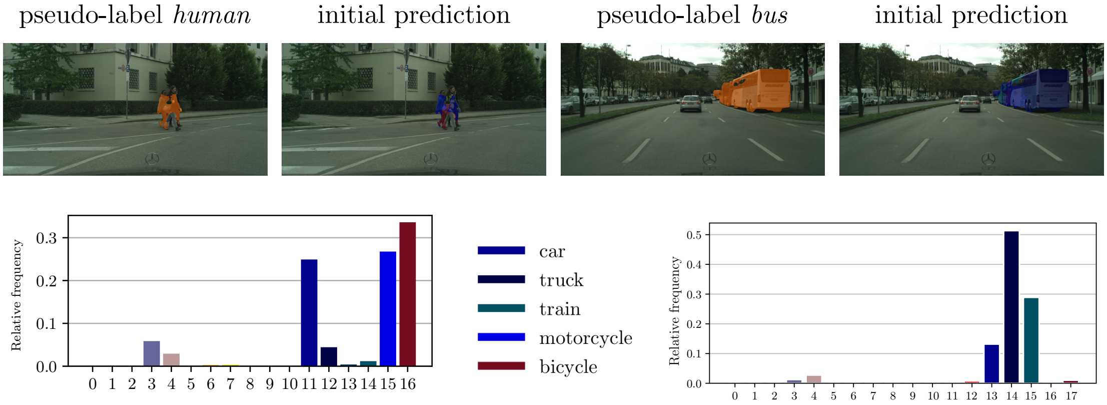


Figure 4.5: Bar plots showing the relative frequencies of predicted classes for instances of the corresponding novel classes *human* and *bus*, each with an exemplary image. The novel pixels in the images are overlayed with the pseudo-label and the initially predicted classes, respectively.

an additional distillation loss function [98] ℓ_{d}, where

$$\ell_{\mathrm{d}}(f_{\theta^+}(x), f_\theta(x)) := -\frac{1}{|\mathcal{Z}|} \sum_{z\in\mathcal{Z}} \sum_{q\in\mathcal{Y}} f_\theta(x)_{z,q} \log(f_{\theta^+}(x)_{z,q}) \, . \tag{4.6}$$

Overall, we aim at minimizing the objective

$$\mathcal{L} := \; \lambda \, \mathbb{E}[\ell_{\mathrm{ce},\omega}(f_{\theta^+}(x), \tilde{y})] + (1-\lambda) \, \mathbb{E}[\ell_{\mathrm{d}}(f_{\theta^+}(x), f_\theta(x))], \quad \lambda \in [0,1] \tag{4.7}$$

with λ regulating the impact of the distillation loss.

Rehearsal methods propose to replay (some of) the data $\mathcal{S}^{\mathrm{train}} \subset \mathcal{X} \times \mathcal{Y}^{|\mathcal{Z}|}$ seen during the training of the initial model f_θ. We select a subset $\mathcal{S}^{\mathrm{known}} \subseteq \mathcal{S}^{\mathrm{train}}$ that contains as much data as $\mathcal{S}^{\mathrm{novel}}$. This subset is chosen largely at random, but in such a way that it involves classes, that are

1. not or rarely present in $\mathcal{S}^{\mathrm{novel}}$ (class frequency), or
2. similar or related to the novel class.

As there is no measure for the second case, we identify those classes by considering the frequency, with which a class is predicted by f_θ on pixels assigned to the novel class. This is, for all data $(x, \tilde{y}) \in \mathcal{S}^{\mathrm{novel}}$, known classes $q \in \mathcal{Y}$ and novel classes $q' \in \{Q+1, \dots, Q+k\}$, we sum up the number of pixels $z \in \mathcal{Z}$ where $\tilde{y}_z = q' \wedge \hat{m}(x)_z = q$. An example is given in Figure 4.5, where the classes *truck*, *train* and *car* are the most frequently predicted classes for instances of the novel class *bus*.

4.5 Experiments

We evaluate our approach on the task of detecting and incrementally learning novel classes in traffic scenes, for which there exist large datasets such as Cityscapes [29] and A2D2 [42]. To this end, all evaluated segmentation DNN's were trained on a training split and only on a subset of all available classes. We then perform our experiments on a test split of the same dataset on which the DNN was trained in order to extend it by exactly one or even multiple novel classes. We measure the performance of the extended models computing the evaluation metrics *intersection over union* (IoU), *precision* and *recall* for a validation set.

4.5.1 Experimental Setup

As segmentation DNNs we employ the DeepLabV3+ [25] and the PSPNet [155]. The first is trained for different subsets of known classes on the Cityscapes dataset. Moreover, both models are pre-trained on Cityscapes with all 19 classes and then fine-tuned on the A2D2 dataset. Here we use a label mapping between both datasets through which 14 classes remain.

We perform five experiments: For the first three experiments, a DeepLabv3+ with a WideResNet38 backbone is trained on the Cityscapes dataset, where 1) the classes *person* & *rider*, 2) the class *bus* and 3) the classes *person* & *rider*, *bus* and *car* are excluded. In a fourth experiment, a DeepLabv3+ as well as a PSPNet based on a ResNet50 backbone are fine-tuned on the A2D2 dataset, for which we specified subsets for training, testing and validation, including 2975, 1355 and 451 annotated images, respectively. Then, we also apply our method to the A2D2 dataset without prior fine-tuning, *i.e.*, under a domain shift, employing a DeepLabV3+ trained on Cityscapes. Our experiments follow a hierarchical structure with increasing complexity:

1. Construction of a “well” separated category (*human*),
2. Construction of a category in the midst of known similar categories (*bus*),
3. Construction of multiple novel categories (*human*, *bus* and *car*),
4. Construction of a new category under domain shift with ground truth for known classes (*guardrail*, with fine-tuning),
5. Construction of a new category under domain shift without ground truth (*guardrail*, without fine-tuning).

Each of those initial DNNs is employed to predict the semantic segmentation masks for the images contained in the respective test set. For the segment-wise prediction quality estimation, we apply a gradient boosting model to obtain the uncertainty scores $u(s) \in [0, 1]$ for each segment $s \subseteq x$ and image x in the test set. The threshold in Equation 4.4 is set to $\tau = 0.5$, *i.e.*, a segment s is considered as anomalous, if $u(s) \geq 0.5$. To extract features of the suspicious objects, we employ a DenseNet201 [59], trained on the ImageNet dataset [32] with 1,000 classes. Note that the DBSCAN hyperparameters have to be selected dependent on the density of the desired clusters.

For the class-incremental extension of an initial DNN f_θ, we replace its final layer to obtain a larger DNN f_{θ^+}. Only the decoder of this model is trained for 70 epochs on the newly collected data $\mathcal{S}^{\text{novel}}$ together with the replayed data $\mathcal{S}^{\text{known}}$. We use random crops of size 1,000 × 1,000 pixels, the Adam optimizer with a learning rate of $5 \cdot 10^{-5}$ and a weight decay of 10^{-4}. Further, the learning rate is adjusted after every iteration via a polynomial learning rate policy [23]. The distillation loss and the cross-entropy loss are weighted equally in the overall loss function defined in Equation 4.7, *i.e.*, $\lambda = 0.5$ (analogously to [97]).

As the five experiments struggle with different issues, the experimental setup slightly differs. For the first case, we construct the novel category *human*, which is "well" separable from all known classes, to enhance the purity of the "human cluster" and to simplify the learning of novel objects. However, we observe that the DNN tends to "overlook" many humans, *i.e.*, they are assigned to the class predicted in the background, *e.g.* to the *road* class. As a consequence, the segment-wise anomaly detection fails to detect such persons, which is why these will be assigned to other classes in our acquired pseudo ground truth. To not distract the extended segmentation network, we modify the pseudo-labels by ignoring all known classes $q \in \mathcal{Y}$ during the incremental training procedure. The *bus* class added in the second experiment is closely related to other classes in the vehicle category, such as *truck*, *train* and *car*, which complicates the construction of pure clusters. We mitigate the impact of objects from similar classes by discarding all objects from the cluster that consist of only one segment in the predicted segmentation. Experiment three extends the previous ones by facing multiple unknown classes, namely *human*, *bus* and *car*. The last two experiments deal with an additional domain shift from urban street scenes in Cityscapes to countryside and highway scenes in A2D2. To bridge this gap, we fine-tune the initial DNN on our A2D2 training set, which, however, requires A2D2 ground truth for the known classes. Without fine-tuning, the prediction quality and thereby the quality of our pseudo ground truth suffers. On that account, we discard images that are generally rated as badly predicted, *i.e.*, where the relative amount of pixels with a low quality estimate exceeds 1/3 of the image in total. Moreover, we renounce the replay of previously-seen data, since this prevents the DNN from adapting to the new domain.

4.5.2 Evaluation of Results

	mIoU$_{\mathcal{Y}}$	**IoU$_{\text{novelty}}$**	**mIoU$_{\mathcal{Y}^+}$**
1. experiment: Cityscapes, human	DeepLabV3+		
initial DNN	68.63	00.00	64.82
extended DNN (ours)	68.53	39.80	66.94
extended DNN (supervised)	69.43	59.33	68.87
oracle	71.05	72.85	71.15
2. experiment: Cityscapes, bus	DeepLabV3+		
initial DNN	66.94	00.00	63.42
extended DNN (ours)	67.07	44.73	65.89
extended DNN (supervised)	66.74	41.40	65.41
oracle	69.48	76.66	69.86
3. experiment: Cityscapes, multi	DeepLabV3+		
initial DNN	56.99	00.00 & 00.00	50.29
extended DNN (ours)	57.52	40.22 & 81.27	57.90
oracle	77.28	81.90 & 94.94	78.59
4. experiment (a): A2D2, guardrail	DeepLabV3+ (fine-tuned)		
initial DNN	75.77	00.00	70.72
extended DNN (ours)	72.07	46.10	70.34
oracle	75.23	74.58	75.19
4. experiment (b): A2D2, guardrail	PSPNet (fine-tuned)		
initial DNN	68.77	00.00	64.19
extended DNN (ours)	64.54	32.79	62.42
oracle	67.71	69.08	67.80
5. experiment: A2D2, guardrail	DeepLabV3+ (not fine-tuned)		
initial DNN	59.38	00.00	55.42
extended DNN (ours)	60.48	20.90	57.84

Table 4.1: Comparing overview of all evaluated models, where the results for our extended DNNs are highlighted in gray. As performance metrics, we provide the mean IoU over the old and new classes, denoted by mIoU$_{\mathcal{Y}}$ and mIoU$_{\mathcal{Y}^+}$, respectively, and the IoU value of the novel class(es), IoU$_{\text{novelty}}$.

In the following, all evaluation values belonging to our extended models are averaged over five runs of the respective experiment. We provide a qualitative comparison of different models for all conducted experiments in Table 4.1, reporting the mean IoU over the known classes and over the extended class set, denoted as mIoU$_{\mathcal{Y}}$ and mIoU$_{\mathcal{Y}^+}$, respectively, as well as the IoU value of the novel classes (IoU$_{\text{novelty}}$). The models considered in this comparison are the initial and the extended DNN, where the class space is extended via our method. For the first and second experiment we further compare our approach with a baseline, where a DNN is extended using a self-training approach. That is, we employ a so-called teacher network, which is already trained on the extended semantic space $\mathcal{Y}^+$, to produce pseudo-labels for some student network. Thereby, we obtain a high quality pseudo ground truth. Apart from this, the baseline DNN is extended analogously to ours. In addition, for the first four experiments we provide results of an *oracle*, *i.e.*, a DNN, that is initially trained on the extended class set $\mathcal{Y}^+$ and only with human-annotated ground truth. In the fifth experiment, we extend

	IoU	**precision**	**recall**	**IoU**	**precision**	**recall**
1. experiment:	DeepLabV3+					
Cityscapes, human	initial			extended		
human	00.00	00.00	00.00	39.80	60.60	53.72
mean over $\mathcal{Y}$	68.63	79.79	80.94	68.53	83.32	77.17
mean over $\mathcal{Y}^+$	64.82	75.36	76.44	66.94	82.05	75.86
2. experiment:	DeepLabV3+					
Cityscapes, bus	initial			extended		
bus	00.00	00.00	00.00	44.73	58.33	66.15
mean over $\mathcal{Y}$	66.94	79.32	79.55	67.07	82.46	76.31
mean over $\mathcal{Y}^+$	63.42	75.15	75.36	65.89	81.19	75.78
3. experiment:	DeepLabV3+					
Cityscapes, multi	initial			extended		
human	00.00	00.00	00.00	40.22	68.74	49.65
car	00.00	00.00	00.00	81.27	86.56	93.05
mean over $\mathcal{Y}$	56.99	65.75	80.88	57.52	78.53	65.77
mean over $\mathcal{Y}^+$	50.29	58.01	71.37	57.90	78.43	66.43
4. experiment (a):	DeepLabV3+					
A2D2, guardrail	initial			extended		
guardrail	00.00	00.00	00.00	46.10	80.41	52.09
mean over $\mathcal{Y}$	75.77	87.86	83.47	72.07	89.01	78.44
mean over $\mathcal{Y}^+$	70.72	82.00	77.90	70.34	88.44	76.69
4. experiment (b):	PSPNet					
A2D2, guardrail	initial			extended		
guardrail	00.00	00.00	00.00	32.79	70.75	38.04
mean over $\mathcal{Y}$	68.77	84.57	76.79	64.54	86.41	71.22
mean over $\mathcal{Y}^+$	64.19	78.93	71.67	62.42	85.36	69.01
5. experiment:	DeepLabV3+					
A2D2, guardrail	initial			extended		
guardrail	00.00	00.00	00.00	20.90	77.12	22.32
mean over $\mathcal{Y}$	59.38	79.50	68.14	60.48	84.08	66.61
mean over $\mathcal{Y}^+$	55.42	74.20	63.60	57.84	83.61	63.66

Table 4.2: Direct comparison of the initial and the extended DNNs for all conducted experiments. We report the IoU, precision and recall values for the novel class (highlighted with gray rows), respectively, as well as averaged over the previously-known and the extended class spaces $\mathcal{Y}$ and $\mathcal{Y}^+$.

the initial DNN by a novel class derived from a different dataset. To some extent, the oracle from experiment 4(a) can serve as a coarse reference for experiment 5. In Table 4.2 we give a more detailed overview about all experiments, reporting not only the IoU, but also the precision and recall values of the novel class as well as averaged over $\mathcal{Y}$ and $\mathcal{Y}^+$. Note that the fourth experiment is evaluated twice, once for (a) the DeepLabV3+ and once for (b) the PSPNet. In general, we observe that our approach succeeds in incrementally extending a DNN by a novel class, while the performance on previously-known classes remains stable. On Cityscapes, we achieve IoU values for the novel classes human and bus of $\mathrm{IoU}_{\mathrm{human}} = 39.80 \pm 0.73\%$ and $\mathrm{IoU}_{\mathrm{bus}} = 44.73 \pm 1.46\%$, respectively. For the third experiment with two novel classes, we obtain similar results for the *human* class with $\mathrm{IoU}_{\mathrm{human}} = 40.22 \pm 1.77\%$ and for the *car* class even $\mathrm{IoU}_{\mathrm{car}} = 81.27 \pm 1.16\%$. While these IoU values are a considerable achievement for a method working without ground truth, the distinct gaps to the oracle's IoU values still leave room for further improvement. Compared to the baseline DNN, we do not achieve competitive performance in the first experiment, while in the second experiment, our approach actually performs slightly better. This is explained by the fact, that the pseudo ground truth for the *human* class incorporates much more noise than that for the *bus* class. In the fourth experiment we mitigate the domain shift from Cityscapes to A2D2 by prior fine-tuning of the networks, using A2D2 ground truth. By that, we obtain IoU values of $\mathrm{IoU}_{\mathrm{guardrail}} = 46.10 \pm 4.8\%$ for the DeepLabV3+ and $\mathrm{IoU}_{\mathrm{guardrail}} = 32.79 \pm 3.48\%$ for the PSPNet. We conclude, that our approach achieves better results for models which are initially better-performing. Without fine-tuning the DeepLabV3+ on A2D2, we obtain $\mathrm{IoU}_{\mathrm{guardrail}} = 20.90 \pm 1.73\%$, while the mean IoU over the previously-known classes $\mathcal{Y}$ slightly increases from 59.38% to $60.48 \pm 0.47\%$.

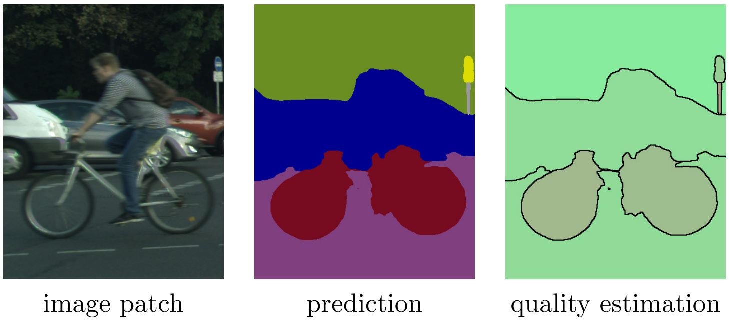


Figure 4.6: Image patch, semantic segmentation and prediction quality estimation for a scene, where a cyclist is overlooked by the initial DNN.

Experiment 1 For the first experiment, we trained a DeepLabV3+ on the Cityscapes dataset, excluding the classes *pedestrian* and *rider*, both together constituting the class *human*. This novelty is well separable from all the known classes as these belong to different, non-organic categories. As there are no similar classes, humans are either totally "overlooked" by the segmentation DNN, *i.e.*, assigned to the class predicted in their background, or predicted as related classes, *e.g.* as *bicycle*, *motorcycle* or *car*, *cf.* Figure 4.5. Since our anomaly detection method fails to spot overlooked persons, these remain mislabeled even in the pseudo ground truth, thus negatively affecting the incremental training procedure. For an example, we refer to Figure 4.6, where a cyclist is assigned to the background classes *road* and *car*. To prevent this issue, we ignore all known classes $q \in \mathcal{Y}$ present in the pseudo-labels. Our newly collected data $\mathcal{S}^{\text{novel}}$ contains 76 pseudo-labeled images. The replayed training data is selected such that at least 25% - 35% of the images contain cars, motorcycles and bicycles, respectively. We evaluated the initial and the extended DNN on the Cityscapes validation data. Class-wise results are provided in Table 4.3.

1. experiment	DeepLabV3+					
Cityscapes, human	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	97.34	98.35	98.96	97.43 ± 0.05	98.54 ± 0.12	98.86 ± 0.08
sidewalk	80.63	89.39	89.16	80.51 ± 0.23	89.50 ± 0.50	88.91 ± 0.67
building	88.91	92.80	95.50	89.40 ± 0.05	93.42 ± 0.20	95.42 ± 0.24
wall	47.24	74.57	56.32	47.74 ± 0.57	78.92 ± 0.49	54.71 ± 0.77
fence	51.03	66.76	68.41	49.20 ± 0.44	70.06 ± 1.55	62.33 ± 1.26
pole	52.90	72.68	66.02	53.30 ± 0.39	74.42 ± 1.41	65.31 ± 1.64
traffic light	55.44	75.04	67.98	55.33 ± 0.19	75.49 ± 1.24	67.47 ± 1.21
traffic sign	66.66	86.22	74.61	66.32 ± 0.62	87.54 ± 1.41	73.27 ± 1.67
vegetation	89.95	93.60	95.85	90.15 ± 0.03	94.01 ± 0.22	95.65 ± 0.22
terrain	56.29	77.66	67.17	55.29 ± 0.47	75.88 ± 1.67	67.14 ± 1.77
sky	93.76	96.38	97.18	93.60 ± 0.11	96.01 ± 0.26	97.39 ± 0.19
human	00.00	00.00	00.00	39.80 ± 0.73	60.60 ± 1.20	53.72 ± 1.42
car	90.61	92.97	97.27	91.16 ± 0.21	95.25 ± 0.50	95.50 ± 0.47
truck	69.66	80.23	84.09	68.98 ± 0.56	84.92 ± 2.35	78.70 ± 1.97
bus	76.90	88.59	85.35	71.57 ± 0.60	87.25 ± 1.33	79.95 ± 1.15
train	70.35	83.33	81.87	63.11 ± 3.17	89.63 ± 1.61	68.13 ± 3.93
motorcycle	24.45	28.57	62.92	32.92 ± 1.13	53.91 ± 2.07	45.89 ± 2.21
bicycle	54.57	59.30	87.24	59.01 ± 0.61	71.62 ± 2.43	77.20 ± 3.38
mean over $\mathcal{Y}$	68.63	79.79	80.94	68.53 ± 0.27	83.32 ± 0.28	77.17 ± 0.60
mean over $\mathcal{Y}^+$	64.82	75.36	76.44	66.94 ± 0.27	82.05 ± 0.25	75.86 ± 0.55

Table 4.3: IoU, precision and recall values (on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively) evaluated on the Cityscapes validation data for both, the initial and the extended DNN from the first experiment.

Besides the novel class, which achieves an IoU value of nearly 40% with approximately 50-60% precision and recall, the incremental training has only little impact on previously-known classes. For many classes, however, we observe an improvement in precision at the expense of the corresponding recall values, *e.g.* for the

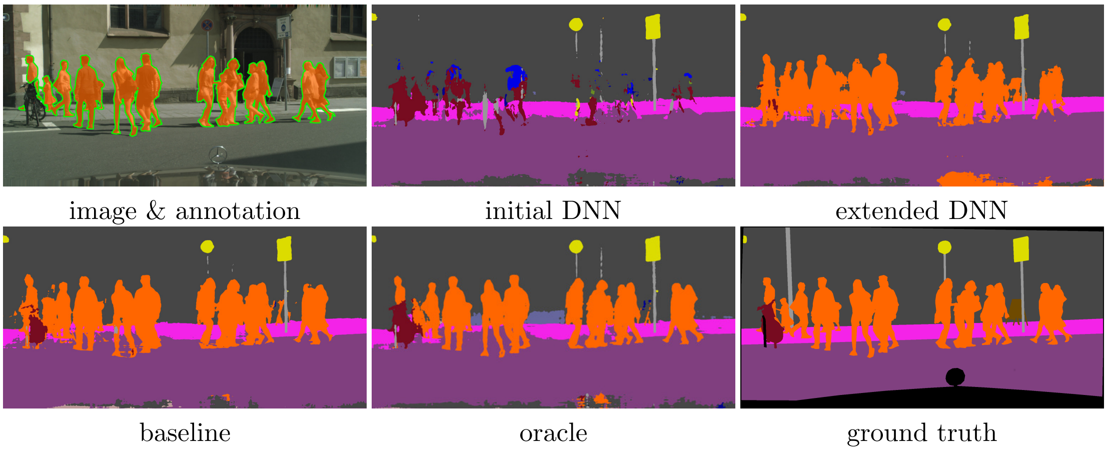


Figure 4.7: Comparison of the semantic segmentation predictions of all DNNs evaluated in the first experiment for an exemplary scene from the Cityscapes validation data.

classes *fence*, *truck* and *train*. This is also reflected in the mean precision and recall values over $\mathcal{Y}$, *i.e.*, while precision increases by 3.53%, recall decreases by 3.77%. Especially the classes *motorcycle* and *bicycle* gain performance regarding the IoU and precision, which is mainly due to human pixels initially assigned to those classes, while the proportion of bikes (motor- or bicycles) that are predicted correctly drops significantly. A comparison of all evaluated models in the first experiment is illustrated for an example image in Figure 4.7. We observe a reduction of noise in the model's predictions, starting from the initial DNN, to the extended DNN, the baseline and the oracle. Nonetheless, the predicted segmentation of our extended DNN comes close to those predicted by the comparative models that both require ground truth for the novel class.

Experiment 2 The setup of the second experiment is the same as in the first one (DeepLabV3+, Cityscapes dataset), but excluding busses from the set of known classes instead of humans. This novelty belongs to the vehicle category, thus being akin to other vehicle classes as *train* or *truck*. These are also the classes the objects declared as novel were predicted for the most part, as we illustrated in Figure 4.5. On that account, at least 50% of the 55 images in $\mathcal{S}^{\text{novel}}$ contain trucks, 30% trains. As a consequence of the visual relatedness, trucks and trains that exhibit a low prediction quality, *i.e.*, that are treated as anomalies, contaminate the cluster of busses in the two-dimensional embedding space. We observed, that the segmentation network predicts most of these "detected" trucks and trains correctly, while it assigns multiple classes, *i.e.*, multiple segments in the semantic segmentation prediction, to a bus. Thus, we delete anomalies from the embedding space, whose predicted segmentation consists of only one segment (ignoring segments with less than 500 pixels).

2. experiment	DeepLabV3+					
Cityscapes, bus	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	97.63	98.81	98.80	97.57 ± 0.03	98.76 ± 0.09	98.79 ± 0.08
sidewalk	81.60	89.65	90.09	81.57 ± 0.10	90.07 ± 0.46	89.63 ± 0.45
building	90.19	94.50	95.19	89.90 ± 0.10	94.22 ± 0.26	95.15 ± 0.25
wall	48.77	78.07	56.51	44.89 ± 3.11	79.23 ± 1.36	50.94 ± 4.20
fence	53.86	70.97	69.08	51.74 ± 0.81	71.82 ± 0.62	64.92 ± 1.27
pole	55.03	75.71	66.83	54.05 ± 0.61	77.62 ± 1.11	64.06 ± 1.54
traffic light	55.87	77.29	66.84	54.70 ± 0.92	80.15 ± 2.02	63.35 ± 2.46
traffic sign	68.21	87.02	75.94	67.88 ± 0.32	87.87 ± 0.98	74.91 ± 1.08
vegetation	90.35	93.98	95.91	90.21 ± 0.09	93.70 ± 0.33	96.04 ± 0.26
terrain	54.03	79.90	62.53	52.77 ± 0.46	75.06 ± 1.14	64.00 ± 1.01
sky	93.64	96.14	97.30	93.26 ± 0.29	95.55 ± 0.63	97.49 ± 0.36
person	71.65	83.27	83.70	71.02 ± 0.21	82.22 ± 0.87	83.92 ± 0.65
rider	48.77	68.86	62.58	47.15 ± 0.73	70.85 ± 1.32	58.55 ± 1.99
car	91.90	94.65	96.94	91.76 ± 0.11	95.35 ± 0.61	96.07 ± 0.62
truck	47.51	51.19	86.87	54.14 ± 1.85	69.81 ± 4.17	71.09 ± 5.25
bus	00.00	00.00	00.00	44.73 ± 1.46	58.33 ± 3.13	66.15 ± 5.16
train	43.57	48.58	80.88	55.46 ± 1.64	74.35 ± 5.75	69.19 ± 5.46
motorcycle	44.35	61.76	61.13	41.66 ± 1.17	71.22 ± 1.70	50.16 ± 2.38
bicycle	68.00	77.42	84.82	67.52 ± 0.28	76.38 ± 0.64	85.35 ± 0.44
mean over $\mathcal{Y}$	66.94	79.32	79.55	67.07 ± 0.12	82.46 ± 0.56	76.31 ± 0.46
mean over $\mathcal{Y}^+$	63.42	75.15	75.36	65.89 ± 0.10	81.19 ± 0.54	75.78 ± 0.34

Table 4.4: IoU, precision and recall values (on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively) evaluated on the Cityscapes validation data for both, the initial and the extended DNN from the second experiment.

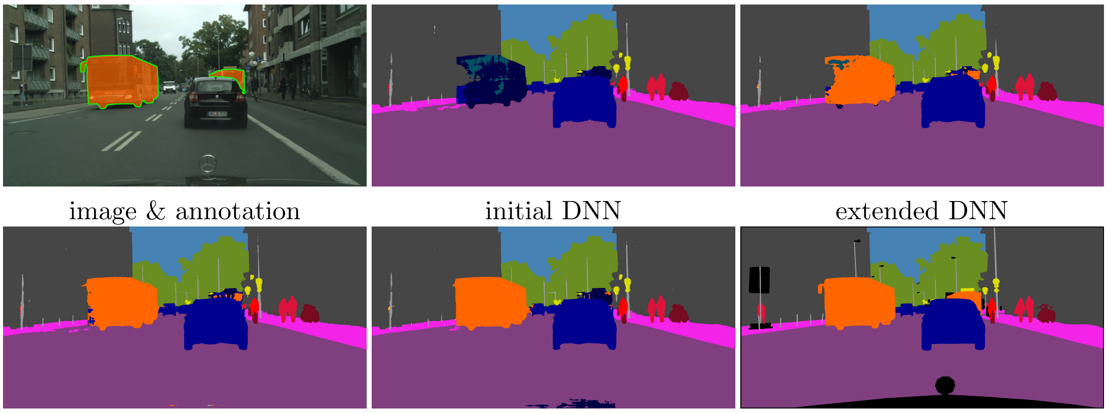


image & annotation

initial DNN

extended DNN

baseline

oracle

ground truth

Figure 4.8: Comparison of the semantic segmentation predictions of all DNNs evaluated in the second experiment for an example image from the Cityscapes validation data.

Again, we provide a class-wise evaluation on the Cityscapes validation split in Table 4.4 and present a comparison of different models for one exemplary street scene in Figure 4.8. Here, large parts of the bus in the foreground are predicted correctly by our extended DNN. The bus in the background is even better recognized by our network than by the baseline and oracle. Analogous to the first experiment, the most similar classes *truck* and *train* show increasing IoU and precision, but decreasing recall values. Averaged over the known classes $q \in \mathcal{Y}$, we again observe improvement in IoU and precision with a concurrent drop in recall. Averaged over the extended class set $\mathcal{Y}^+$, all three performance measures increase after class-incremental learning.

Experiment 3 In the next experiment we extend the previous ones by enlarging the set of novel classes, withholding the classes *pedestrian* & *rider*, *bus* and *car*. Again, we trained a DeepLabV3+ network on the Cityscapes dataset to learn the remaining, non-novel classes. We reconsidered our approach to reject possibly

3. experiment	DeepLabV3+					
Cityscapes, multi	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	95.43	96.41	98.95	96.62 ± 0.07	98.29 ± 0.20	98.27 ± 0.22
sidewalk	77.23	83.84	90.74	76.42 ± 0.26	84.27 ± 0.98	89.16 ± 0.91
building	87.21	91.05	95.39	87.42 ± 0.12	92.66 ± 0.30	93.92 ± 0.40
wall	45.86	68.38	58.20	40.36 ± 0.59	76.67 ± 1.57	46.03 ± 1.07
fence	47.86	59.63	70.79	41.15 ± 1.47	69.23 ± 2.40	50.44 ± 2.54
pole	51.63	69.15	67.09	48.68 ± 0.48	73.74 ± 1.13	58.93 ± 1.42
traffic light	55.61	77.70	66.17	45.62 ± 0.47	72.64 ± 0.85	55.09 ± 1.07
traffic sign	64.84	80.37	77.04	58.34 ± 0.74	86.84 ± 0.70	64.01 ± 1.23
vegetation	88.26	91.27	96.40	88.61 ± 0.22	91.80 ± 0.43	96.22 ± 0.21
terrain	53.22	72.42	66.74	45.43 ± 0.77	79.11 ± 1.55	51.66 ± 1.67
sky	93.58	96.11	97.27	92.41 ± 0.16	95.56 ± 0.19	96.56 ± 0.10
human	00.00	00.00	00.00	40.22 ± 1.77	68.74 ± 4.84	49.65 ± 4.80
car	00.00	00.00	00.00	81.27 ± 1.16	86.56 ± 2.20	93.05 ± 1.12
truck	9.31	9.41	89.35	25.59 ± 7.41	61.27 ± 5.50	30.77 ± 9.90
train	41.70	45.05	84.87	49.87 ± 5.21	60.85 ± 8.56	73.99 ± 2.61
motorcycle	4.03	4.12	66.09	14.30 ± 2.72	63.79 ± 3.44	15.64 ± 3.31
bicycle	39.13	41.30	88.15	51.97 ± 1.58	71.26 ± 1.98	65.95 ± 4.30
mean over $\mathcal{Y}$	56.99	65.75	80.88	57.52 ± 0.80	78.53 ± 1.20	65.78 ± 1.00
mean over $\mathcal{Y}^+$	50.29	58.01	71.37	57.90 ± 0.68	78.43 ± 1.10	66.43 ± 0.94

Table 4.5: In-depth evaluation on the Cityscapes validation data for the third experiment, where we incrementally extend a DeepLabV3+ by the novel classes *human* and *car* on the Cityscapes dataset. We provide IoU, precision and recall values obtained for both, the initial and the extended DNN, on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively.

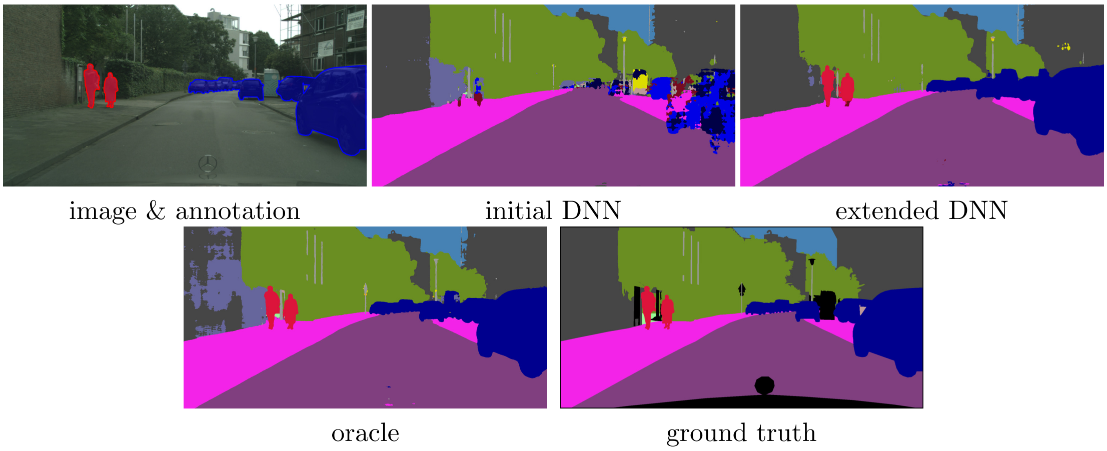


Figure 4.9: Comparison of the semantic segmentation predictions of all DNNs evaluated in the third experiment for an example image from the Cityscapes validation data.

known objects from the embedding space to improve the purity of novel object clusters. Instead of rejecting anomalous segments that consist of only one predicted segment in the semantic segmentation mask, we include a random choice of objects / segments from each known class into the embedding space. If an anomalous object can be assigned to an existing class, it is no longer taken into account in the further procedure. To decide whether an object is novel or known, we consider its 2.75-neighborhood. If this contains at least 10 known objects from which at least 80% belong to the most frequent class, we assume the anomaly belongs to even this class, *i.e.*, we reject it. Consequently, we discard the detected bus segments since these are closely related to the classes *truck* and *train*. However, we obtain two clusters, one for the class *car* (1375 segments) and one for the class *human* (135 segments). We incrementally expand the model by these classes, achieving a similar IoU value (around 40%) for the *human* class as in experiment 1, where we only learned a single class. For the *bus* class, we even get an IoU value of more than 80%. Detailed results are provided in Table 4.5.

Experiment 4a The fourth experiment involves two different network architectures. Results for the first one are shown in experiment 4(a), results for the other one in 4(b). We start with a DeepLabV3+ network trained on the Cityscapes dataset and aim to detect and learn the *guardrail* class using images taken from the A2D2 dataset. To mitigate a performance drop caused by the domain shift from Cityscapes to A2D2, we first fine-tune the decoder for 70 epochs on our A2D2 training split, applying the same hyperparameters we used for the incremental training. By that, we improve the mean IoU of the initial network from 59.38% to 75.77%. The classes which suffer the most are *person*, *motorcycle* and *bicycle*, which is presumably due to their rare occurrence on country roads

4. experiment (a)	DeepLabV3+					
A2D2, guardrail	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	95.59	97.21	98.29	95.93 ± 0.06	97.94 ± 0.18	97.91 ± 0.15
sidewalk	72.01	86.73	80.92	72.08 ± 0.41	85.29 ± 0.84	82.33 ± 1.28
building	87.82	93.58	93.44	85.75 ± 0.67	93.13 ± 0.53	91.54 ± 1.01
fence	59.35	81.59	68.53	56.76 ± 0.37	79.89 ± 2.40	66.29 ± 1.63
pole	56.13	76.39	67.91	54.31 ± 0.24	77.86 ± 0.52	64.23 ± 0.66
traffic light	68.41	85.10	77.72	65.48 ± 0.19	84.21 ± 0.77	74.65 ± 0.83
traffic sign	76.34	86.78	86.38	74.53 ± 0.38	89.98 ± 1.11	81.30 ± 1.19
vegetation	91.61	94.01	97.29	92.00 ± 0.23	94.81 ± 0.38	96.89 ± 0.17
sky	97.96	98.72	99.22	97.81 ± 0.03	98.57 ± 0.07	99.22 ± 0.04
person	67.60	79.28	82.11	64.27 ± 0.58	87.70 ± 0.87	70.65 ± 1.21
car	93.19	96.73	96.22	92.42 ± 0.11	96.04 ± 0.35	96.08 ± 0.35
truck	84.99	88.51	95.53	80.98 ± 2.66	84.75 ± 3.29	94.82 ± 0.69
motorcycle	48.68	84.71	53.37	26.05 ± 2.72	90.18 ± 2.09	26.85 ± 3.04
bicycle	61.08	80.65	71.57	50.65 ± 3.27	85.78 ± 2.10	55.43 ± 4.78
guardrail	00.00	00.00	00.00	46.10 ± 4.79	80.41 ± 2.12	52.09 ± 6.42
mean over $\mathcal{Y}$	75.77	87.86	83.47	72.07 ± 0.39	89.01 ± 0.48	78.44 ± 0.52
mean over $\mathcal{Y}^+$	70.72	82.00	77.90	70.34 ± 0.50	88.44 ± 0.40	76.69 ± 0.47

Table 4.6: In-depth evaluation on the A2D2 validation data for the fourth experiment, where we first fine-tune and then incrementally extend a DeepLabV3+ by the novel class *guardrail* on the A2D2 dataset. We provide IoU, precision and recall values obtained for both, the initial and the extended DNN, on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively.

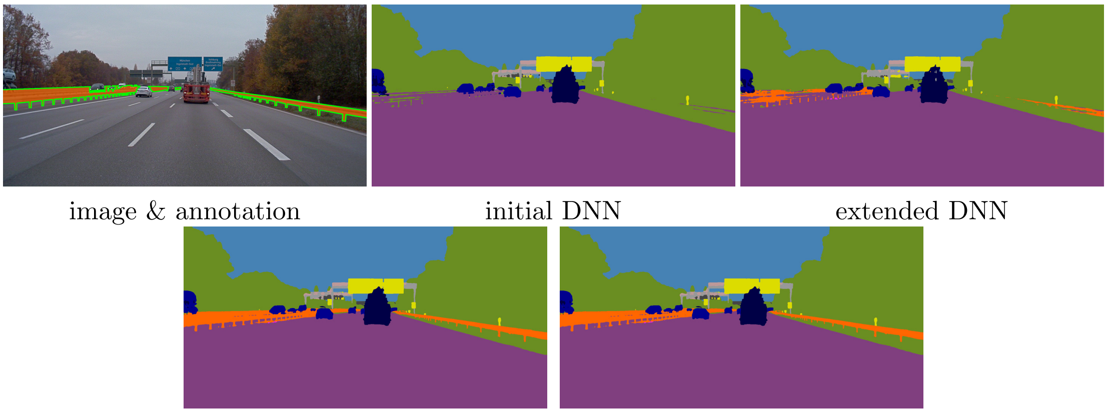


image & annotation

initial DNN

extended DNN

oracle

ground truth

Figure 4.10: Comparison of the semantic segmentation predictions of all DNNs evaluated in the fourth experiment (a) for an example image from the A2D2 dataset.

and highways, and therefore, low frequency in the re-training data, which involves only 30 pseudo-labeled and 30 replayed images. Details are provided in Table 4.6.

Experiment 4b In experiment 4(b), we employ the PSPNet instead of the DeepLabV3+, for the rest we proceed as in the previous subsection. Again, the training data consists of 30 images with pseudo ground truth and 30 labeled, replayed images (containing only old classes) from the A2D2 training split. Note that these 30 images are not the same as in experiment 4(a) due to the different network providing predictions of estimated low quality on different images. In total, the initial and the extended PSPNet are outperformed by DeepLabV3+, however, both architectures show similar patterns:

- extended DNN exhibits a high $\text{precision}_{\text{guardrail}}$ and a low $\text{recall}_{\text{guardrail}}$
- classes that are mostly affected by re-training: *person*, *motorcycle*, *bicycle*
- averaged over $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively, IoU and recall values decrease, precision values increase

For more detailed information we refer to Table 4.7.

4. experiment (b)	PSPNet					
A2D2, guardrail	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	95.18	97.10	97.96	94.93 ± 0.21	96.94 ± 0.55	97.86 ± 0.34
sidewalk	66.15	83.68	75.94	62.19 ± 2.28	82.28 ± 2.09	71.99 ± 4.75
building	84.32	92.46	90.54	82.38 ± 0.46	90.78 ± 0.86	89.91 ± 1.04
fence	54.48	76.84	65.18	50.67 ± 1.24	80.91 ± 1.85	57.62 ± 2.33
pole	44.60	63.94	59.59	42.15 ± 0.91	65.52 ± 2.19	54.31 ± 2.89
traffic light	58.94	81.14	68.30	56.07 ± 0.17	80.65 ± 1.85	64.83 ± 1.37
traffic sign	71.30	87.71	79.22	67.63 ± 0.47	87.61 ± 0.71	74.79 ± 0.56
vegetation	90.68	93.12	97.18	90.65 ± 0.11	93.71 ± 0.41	96.53 ± 0.32
sky	97.57	98.44	99.10	97.21 ± 0.12	98.06 ± 0.19	99.12 ± 0.10
person	59.17	82.53	67.64	46.20 ± 1.13	82.99 ± 0.99	51.04 ± 1.60
car	89.39	94.36	94.44	86.82 ± 0.34	93.90 ± 0.57	92.01 ± 0.60
truck	77.83	84.05	91.31	73.53 ± 1.91	82.11 ± 2.40	87.58 ± 1.25
motorcycle	19.73	76.72	20.99	7.00 ± 2.02	94.92 ± 3.73	7.04 ± 2.07
bicycle	53.49	71.82	67.70	46.05 ± 1.37	79.31 ± 2.49	52.44 ± 2.71
guardrail	00.00	00.00	00.00	32.79 ± 3.47	70.75 ± 2.04	38.04 ± 4.90
mean over $\mathcal{Y}$	68.77	84.57	76.79	64.54 ± 0.28	86.41 ± 0.77	71.22 ± 0.69
mean over $\mathcal{Y}^+$	64.19	78.93	71.67	62.42 ± 0.42	85.36 ± 0.78	69.01 ± 0.94

Table 4.7: In-depth evaluation on the A2D2 validation data for the fourth experiment, where we first fine-tune and then incrementally extend a PSPNet by the novel class *guardrail* on the A2D2 dataset. We provide IoU, precision and recall values obtained for both, the initial and the extended DNN, on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively.

Experiment 5 Finally, we perform the same experiment as in 4(a) without prior fine-tuning the initial DNN on A2D2. Consequently, the domain shift causes many

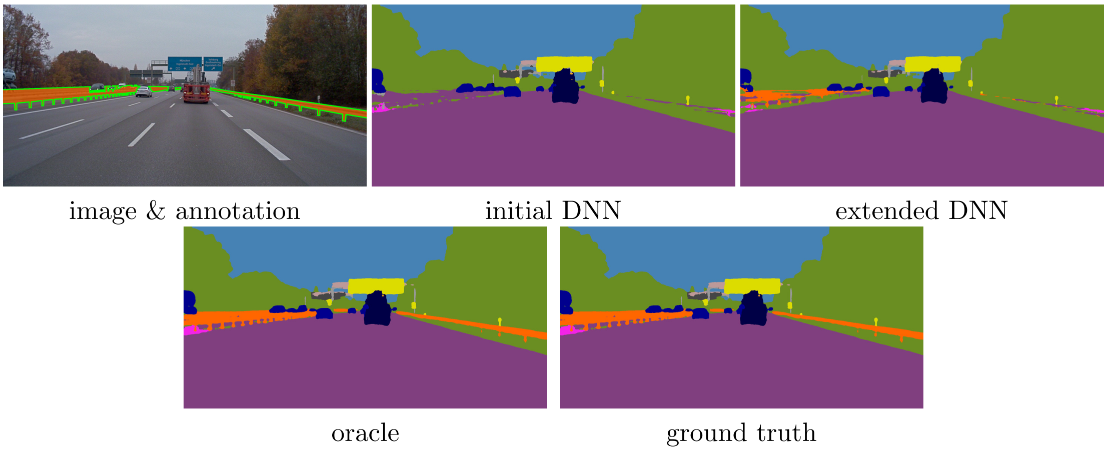


Figure 4.11: Comparison of the semantic segmentation predictions of all DNNs evaluated in the fourth experiment (*b*) for an example image from the A2D2 dataset.

noisy predictions, exhibiting low prediction quality estimates. We exclude such images from the further process based on two criteria:

1. mean quality score (averaged over pixels) less than 0.7
2. more than 1/3 of all pixels with quality estimate less than 0.9.

If at least one criterion holds, we reject the image, *cf.* Figure 4.12.

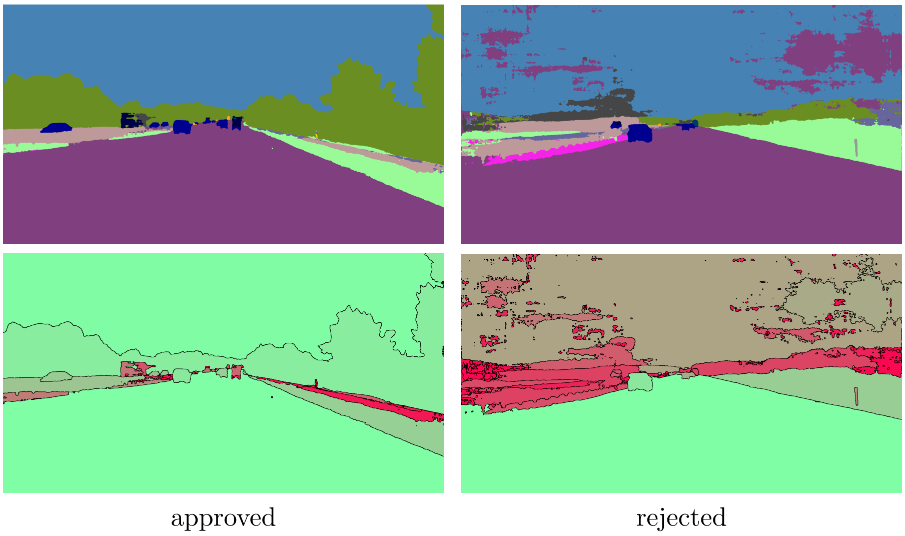


Figure 4.12: Illustration of prediction quality differences (green color indicates high, red color low prediction quality), caused by the domain shift from Cityscapes to A2D2, mainly due to weather conditions.

5. experiment A2D2, guardrail	DeepLabV3+					
	initial			extended		
class	**IoU**	**precision**	**recall**	**IoU**	**precision**	**recall**
road	89.88	92.18	97.30	93.15 ± 0.19	94.89 ± 0.23	98.07 ± 0.12
sidewalk	47.91	76.22	56.33	35.28 ± 2.43	86.95 ± 0.98	37.26 ± 2.67
building	70.94	86.88	79.45	71.25 ± 1.46	90.51 ± 0.89	77.03 ± 2.21
fence	26.08	35.30	49.94	26.20 ± 0.49	37.25 ± 1.46	46.99 ± 1.26
pole	42.59	59.24	60.25	42.77 ± 0.37	62.91 ± 0.73	57.21 ± 0.85
traffic light	47.59	85.85	51.64	52.52 ± 0.70	89.21 ± 1.15	56.10 ± 1.19
traffic sign	54.89	82.49	62.13	57.23 ± 0.25	87.34 ± 1.03	62.42 ± 0.43
vegetation	69.15	96.68	70.83	73.42 ± 0.41	95.05 ± 0.62	76.35 ± 0.34
sky	94.96	98.25	96.59	96.92 ± 0.09	97.81 ± 0.13	99.08 ± 0.05
person	59.77	71.00	79.08	59.58 ± 1.23	84.68 ± 2.45	66.88 ± 2.89
car	90.47	95.72	94.28	90.72 ± 0.16	96.14 ± 0.39	94.16 ± 0.53
truck	62.64	83.61	71.40	71.10 ± 0.24	89.44 ± 0.51	77.62 ± 0.36
motorcycle	28.39	70.82	32.15	32.77 ± 3.05	79.50 ± 3.43	35.96 ± 4.24
bicycle	46.04	78.74	52.57	43.84 ± 1.01	85.43 ± 1.50	47.41 ± 1.56
guardrail	00.00	00.00	00.00	20.90 ± 1.73	77.12 ± 3.95	22.32 ± 2.07
mean over $\mathcal{Y}$	59.38	79.50	68.14	60.48 ± 0.47	84.08 ± 0.49	66.61 ± 0.64
mean over $\mathcal{Y}^+$	55.42	74.20	63.60	57.84 ± 0.48	83.61 ± 0.68	63.66 ± 0.63

Table 4.8: In-depth evaluation on the A2D2 validation data for the fifth experiment, where we incrementally extend a DeepLabV3+ (trained on Cityscapes) by the novel class *guardrail* on the A2D2 dataset. We provide IoU, precision and recall values obtained for both, the initial and the extended DNN, on a class-level as well as averaged over the classes in $\mathcal{Y}$ and $\mathcal{Y}^+$, respectively.

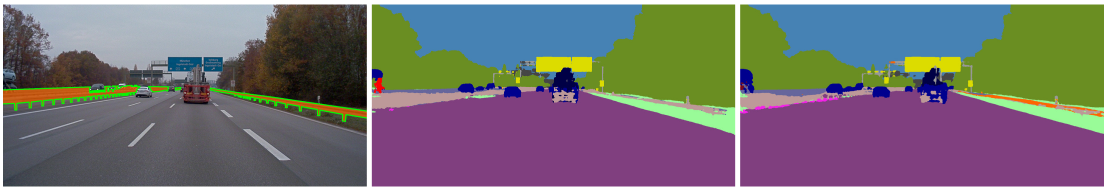


image & annotation

initial DNN

extended DNN

Figure 4.13: Comparison of the semantic segmentation predictions of all DNNs evaluated in the fifth experiment for an example image from the A2D2 dataset.

Applying our method, we obtain 70 pseudo-labeled images. The incorporation of data seen during training of the initial DNN, *i.e.*, the Cityscapes training data, restrains the network from adapting to the new domain. We therefore decided to extend the model only on $\mathcal{S}^{\text{novel}}$.

Class-wise evaluation results are reported in Table 4.8. Even with a domain shift, we achieve an IoU of 20.90 ± 1.73% for the novel class. This is less than the value obtained with prior fine-tuning. However, this DNN still outperforms the PSPNet from the previous experiment considering only the precision. The low recall values are tolerable since many guardrails are still assigned to the "supercategory" *fence*. For most other classes, the IoU values increase or remain roughly the same. In contrast to the other experiments, the *motorcycle* class improves in IoU, precision and recall values. Only classes that are rare in rural street scenes, *e.g. sidewalk* or *bicycle*, suffer from the incremental training.

Segmentations of the same input image are provided in Figure 4.10, Figure 4.11 and Figure 4.13 for the experiments 4(a), 4(b) and 5. All three extended DNNs have learned to predict the novel class to some extent. The prior fine-tuned networks show similar predictions, though DeepLabV3+ is much more precise than the PSPNet and better recognizes the guardrail on the right. The model from the fifth experiment predicts the left guardrail as *fence* (which is not totally mistaken), though it performs better on the right-hand guardrail than the others. Both oracles illustrate, that the *guardrail* class is learnable with high accuracy, still leaving room for improvement of unsupervised methods.

4.6 Conclusion

In this work, we have introduced a new and modular procedure for the class-incremental extension of a semantic segmentation network, where novel classes are detected, annotated and learned in an unsupervised fashion. While there already exists an unsupervised open world approach for semantic segmentation [104], we are the first in this field to extend a neural network's semantic space by robust novel classes. We performed five hierarchically structured experiments with an increasing level of difficulty. We demonstrated that our approach can deal with novelties that are either "well" separated or related to known categories, and that it is even applicable when the test data is sampled from a slightly different distribution than the DNN was trained on. Moreover, we applied two different models in the fourth experiment, where the initial DeepLabV3+ already outperformed the initial PSPNet. This performance gap is also reflected in the model's ability to learn the novel class, thus we conclude that our method benefits significantly from high performance networks.

Chapter 5

Detecting Novelties with Empty Classes

For open world applications, deep neural networks (DNNs) need to be aware of previously unseen data and adaptable to evolving environments. Furthermore, it is desirable to detect and learn novel classes which are not included in the DNNs underlying set of semantic classes in an unsupervised fashion. The method proposed in this article builds upon anomaly detection to retrieve OoD data as candidates for new classes. We thereafter extend the DNN by k empty classes and fine-tune it on the OoD data samples. To this end, we introduce two loss functions, which 1) entice the DNN to assign OoD samples to the empty classes and 2) to minimize the inner-class feature distances between them. Thus, instead of ground truth which contains labels for the different novel classes, the DNN obtains a single OoD label together with a distance matrix, which is computed in advance. We perform several experiments for image classification and semantic segmentation, which demonstrate that a DNN can extend its own semantic space by multiple classes without having access to ground truth.

5.1 Introduction

For computer vision tasks such as image classification or semantic segmentation, deep neural networks (DNNs) learn to classify instances, either on a per image- or per pixel-level, into a limited number of predefined classes. State-of-the-art DNNs achieve high accuracy when trained in a supervised fashion and deployed in a closed world setting, in which the learner is not confronted with OoD data. In practice, however, concepts not seen at training time might occur, which is why so-called open world recognition [6] has emerged as a practically more relevant problem formulation. It combines OoD detection with class-incremental learning, *i.e.*, retraining the model with newly observed classes. Nevertheless, methods

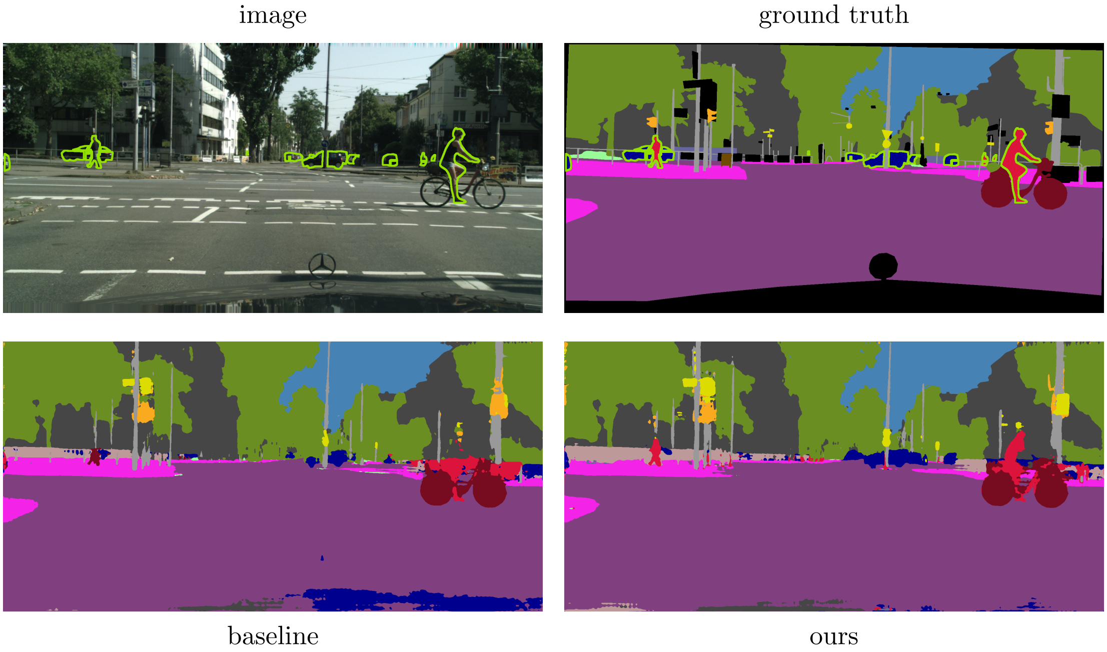


Figure 5.1: Comparison of two segmentation DNNs which were extended by the classes *human* and *car*. While the segmentation masks are similar for the initial classes, the humans and cars are much better segmented by the DNN which was extended by our empty classes approach. The novel classes are marked with green contours in the image and ground truth.

of this kind are typically updated in a supervised fashion, commonly employing humans for annotation.

First attempts to learn in an unsupervised manner have been made to achieve cheaper labeling. In open world image classification, clustering methods like *k-means* [90] or *DBSCAN* [36] allow for an unsupervised labeling of instances in feature regions that appear to be novel. Approaches in this direction leverage such methods to obtain pseudo-labels for detected OoD images [50, 128]. However, the quality of these pseudo-labels strongly depends on the clustering performance. Furthermore, the OoD candidates are assigned to fixed labels, which are likely to be noisy and thus unreliable, whereas in our method, they are put in relation to each other. In open world semantic segmentation [104, 134], pseudo-labeling on a per pixel-level is required, rendering the problem more complex. More recently, few-shot learning [19], where a model is trained to generalize well on novel classes with only few labeled examples, has also been proposed to deal with the lack of labeled data as another (semi-)supervised strategy.

In our work, we introduce a new unsupervised approach for incrementally extending DNNs by capturing novel concepts in additional classes of hypothetical nature. To this end, we proceed from an initial model aware of hitherto known classes, which is augmented by an OoD detection mechanism to distinguish these classes from unknown categories. When treating additional data with potentially additional but unknown classes, we suggest extending the model by additional auxiliary neurons in the DNN's output layer constituting the suspected novel classes to be recognized, which we dub *empty classes*. To predict outcomes of these classes, our model is fine-tuned by a clustering loss that aims to recognize similar concepts for OoD data, allowing to flexibly adapt the learned feature representations to distinguish the already known classes from the new learning outcomes.

We conduct experiments on several datasets with increasing level of difficulty, starting with image classification of MNIST [74] digits as well as the slightly more sophisticated data from FashionMNIST [144]. Next, we apply our approach to low- and medium-resolution images from the CIFAR10 [69] and Animals10 dataset, respectively. Finally, we also adapt our method to the complex task of semantic segmentation of street scenes from the Cityscapes [29] dataset. In three out of four image classification experiments, our method outperforms the baseline, where a DNN is fine-tuned on k-means labeled OoD data. Furthermore, our extended segmentation DNN achieves better results than the baseline [134] for the novel class *car*, and significantly reduces the number of overlooked humans. See Figure 5.1 for an example.

5.2 Related Works

Open world recognition [6] refers to the problem of adapting a learning system to a non-delimitable and potentially constantly evolving target domain. As such, it combines the disciplines of open set learning [125], where incomplete knowledge over the target domain is assumed at training time, with incremental learning [18], in which the model is updated by exploring additional target space regions at test time, thereby adapting to novel target information. Typically, open set recognition is formalized by specifying a novelty detector, a labeling process and an incremental learning function, allowing for a generalized characterization of such systems [6].

Most of previous approaches consider the open world recognition problem in the context of classification, where novel concepts are in form of previously unseen classes. While a plethora of methods has been proposed to tackle the individual sub-problems for classification problems, for which we refer to [110] for a more

https://www.kaggle.com/datasets/alessiocorrado99/animals10

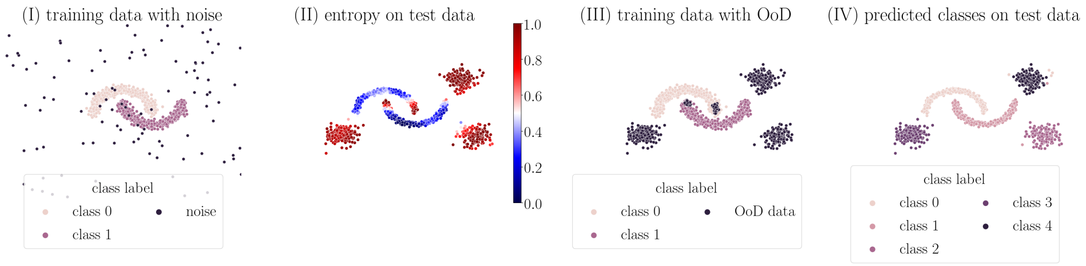


Figure 5.2: (I) A binary classification model is trained on two classes and additional noise data for entropy maximization. (II) OoD samples in the test data are obtained by entropy thresholding. (III) The training data is enriched with the OoD samples and a distance matrix, containing their pair-wise Euclidean distances. (IV) The model is class-incrementally extended by three novel classes.

comprehensive overview, literature on holistic approaches for open world classification is rather scarce. In [128], a metric learning approach is used to distinguish between pairs of instances belonging to the same classes, allowing to detect instances that can not be mapped to known classes and being used to learn novel class concepts. Moreover, [106] suggests a semi-supervised learning approach that applies clustering on learned feature representations to reason about unknown classes. Related to this, [141] describes a kernel method using an alternative loss formulation to learn embeddings to be clustered for class discovery. More recently, similar concepts have also been tailored to specific data modalities, such as tabular data [132].

In the domain of semantic segmentation, open world recognition is also covered under the term *zero-shot semantic segmentation* [14]. To predict unseen categories for classified pixels, a wide range of methods leverage additional language-based context information [14, 86, 143]. Besides enriching visual information by text, unsupervised methods, *e.g.* , employing clustering based on visual similarity [134] or contrastive losses [19, 41], have also been considered. More recently, [20] adopts semantic segmentation based on LiDAR point clouds by augmenting conventional classifiers with predictors recognizing unknown classes, thereby enabling incremental learning.

In a more general context, unsupervised representation learning [116] constitutes a major challenge to generalize learning methods to unseen concepts. Methods of this kind are typically tailored to data modalities, *e.g.* , by specifying auxiliary tasks to be solved [43, 151]. In the domain of images, self-supervised learning approaches have emerged recently [16, 79], which commonly apply knowledge distillation between different networks, allowing for learning in a self-supervised fashion. Other methods include ideas stemming from metric [47] or contrastive learning [27].

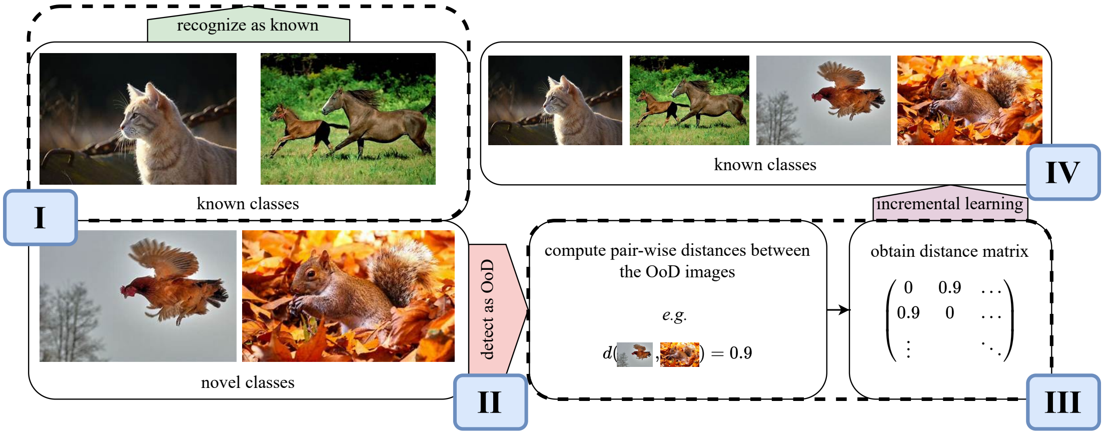


Figure 5.3: Open world recognition models must be able to recognize known classes while detecting OoD data from novel classes and furthermore to incrementally learn these novel classes. Instead of labeling the OoD samples, our method computes pair-wise distances between them, which serve as input for a clustering loss function.

5.3 Method Description

This section introduces a training framework for unsupervised class-incremental learning with empty classes. For the sake of brevity, all equations are introduced for image classification and adapted to semantic segmentation in Section 5.4. First, we give a motivating example in Figure 5.2, where we enrich data stemming from the Two Moons dataset with OoD samples and extend the model by three novel classes. As in-distribution data 1,000 samples are drawn from the Two Moons dataset with noise = 0.1. Additionally, 100 OoD data samples are drawn from a uniform distribution over $[-4,4]^2$. Then, a shallow neural network, consisting of 4 fully connected layers, is trained on these samples to minimize the cross entropy with respect to the Two Moons data while maximizing the entropy on the OoD data. As test data, another 750 samples are drawn from the Two Moons dataset, together with 500 OoD samples belonging to three blobs, centered at $\tilde{x}_1 = (-1.5, -0.95), \tilde{x}_2 = (2.5, 1.5)$ and $\tilde{x}_3 = (3, -1)$, respectively, with 0.25 standard deviation. These blobs represent the novel classes. The test data is then fed into the trained model, and is considered to be OoD if the softmax entropy exceeds a threshold of 0.8. Finally, the initial model is extended by three empty classes in the last layer and then fine-tuned on the Two Moons training samples plus the OoD samples detected in the test data. The OoD data is clustered into the empty classes through our proposed loss function, without requiring any previous (pseudo-)labeling.

The following method description is also illustrated in Figure 5.3.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

I) Learning Model For an input image $x \in \mathcal{X}$, let $f_\theta(x) \in (0,1)^Q$ denote the softmax probabilities of some image classification model $f_\theta : \mathcal{X} \to (0,1)^Q$ with underlying classes $\mathcal{Y} = \{1, \dots, Q\}$. Consider a test dataset which includes images from classes $q \in \{1, \dots, Q, Q+1, \dots\}$. Note that our framework does not necessarily assume labels for the test data as these will be only used for evaluation and not during the training. Furthermore, let $u(x) \in [0,1]$ denote some arbitrary uncertainty score which derives from the predicted class-probabilities $f_\theta(x)$. Thus, a test image x is considered to be OoD, if $u(x) > \tau$ for some threshold $\tau \in [0,1]$.

Next, we extend the initial model f_θ by $k \in \mathbb{N}$ empty classes in the final classification layer, which is then denoted as $f_{\theta^{+k}} : \mathcal{X} \to (0,1)^{Q+k}$, and fine-tune it on the OoD data $\mathcal{U}^{\text{OoD}} \subset \mathcal{X}$. Therefore, we compute pairwise distances $d_{ij} = d(x_i, x_j)$ for all $(x_i, x_j) \in \mathcal{U}^{\text{OoD}} \times \mathcal{U}^{\text{OoD}}$ as a pre-processing step, *e.g.* using the pixel-wise Euclidean distance or any distance metric in the feature space of some embedding network. The model $f_{\theta^{+k}}$ is then fine-tuned on (a subset of) the initial training data $\mathcal{S}^{\text{train}}$, enriched with the unlabeled OoD samples from the test data. For the in-distribution samples (x, y), we compute the cross-entropy loss

$$\ell_{\text{ce}}(x, y) = -\sum_{q=1}^{Q} \mathbb{1}_{\{q=y\}} \log(f_{\theta^{+k}}(x)_q) \ . \tag{5.1}$$

Further, we entice the model to predict one of the empty classes $Q+1, \dots, Q+k$ for OoD data by minimizing the class-probabilities $f_{\theta^{+k}}(x)_1, \dots, f_{\theta^{+k}}(x)_Q$, $x \in \mathcal{U}^{\text{OoD}}$, *i.e.*, by computing

$$\ell_{\text{ext}}(x) = \frac{1}{Q} \sum_{q=1}^{Q} f_{\theta^{+k}}(x)_q \ . \tag{5.2}$$

Finally, we aim to divide the data among the empty classes based on their similarity. Thus, our clustering loss is computed pair-wise as

$$\ell_{\text{cluster}}(x_i, x_j) = \frac{\alpha}{Q+k} \cdot d_{ij} \cdot \sum_{q=1}^{Q+k} f_{\theta^{+k}}(x_i)_q f_{\theta^{+k}}(x_j)_q \ , \tag{5.3}$$

where $\alpha \in \mathbb{R}_{>0}$ can be adjusted to control the impact of the clustering loss function. Together, these three loss functions give the overall objective

$$\begin{aligned} \mathcal{L} = \ & \lambda_1 \mathbb{E}_{(x,y)\sim\mathcal{S}^{\text{train}}}[\ell_{\text{ce}}(x, y)] \\ & + \lambda_2 \mathbb{E}_{x\sim\mathcal{U}^{\text{OoD}}}[\ell_{\text{ext}}(x)] \\ & + \lambda_3 \mathbb{E}_{x_i, x_j\sim\mathcal{U}^{\text{OoD}}}[\ell_{\text{cluster}}(x_i, x_j)] \ , \end{aligned} \tag{5.4}$$

where the hyperparameters λ_1, λ_2 and λ_3 can be adjusted to balance the impact of the objectives.

II) OoD Detection OoD detection is a preprocessing part of our framework, which can be exchanged in a plug and play manner. In our experiments, we implemented entropy maximization [56] for image classification and thus perform OoD detection by thresholding on the softmax entropy.

The idea of entropy maximization is the inclusion of *known unknowns* into the training data of the initial model in order to entice it to exhibit a high softmax entropy

$$u(x) = -\frac{1}{\log(Q)} \sum_{q=1}^{Q} f_\theta(x)_q \log(f_\theta(x)_q) \tag{5.5}$$

on OoD data $x \in \mathcal{U}^{\text{OoD}}$. Therefore, during training the initial model, we compute the entropy maximization loss

$$\ell_{\text{em}}(x) = -\sum_{q=1}^{Q} \frac{1}{Q} \log(f_\theta(x)_q) \tag{5.6}$$

for known unknowns $x \in \mathcal{U}^{\text{OoD}}$, giving the overall objective

$$\begin{aligned} \mathcal{L} = {} & \lambda \ \mathbb{E}_{(x,y)\sim\mathcal{S}^{\text{train}}}[\ell_{\text{ce}}(x,y)] \\ & + (1-\lambda) \ \mathbb{E}_{x\sim\mathcal{U}^{\text{OoD}}}[\ell_{\text{em}}(x)] \ . \end{aligned} \tag{5.7}$$

In the Two Moons example, these OoD data was uniformly distributed noise. For image classification, we employ the domain-agnostic data augmentation technique mixup [153]. This is, an OoD image is obtained by computing the average of two in-distribution samples. Entropy maximization was also introduced for semantic segmentation of street scenes [22, 63], where the OoD samples originate from the COCO dataset [82]. Furthermore, the OoD loss and data was only included in the final training epochs, which means that existing networks can be fine-tuned for entropy maximization.

III) Distance Matrix Next, we compute pair-wise distances for the detected OoD samples, which constitute the OoD dataset for the incremental learning. For simple datasets such as Two Moons or MNIST, the distance can be measured directly between the data samples. For MNIST, this is done by flattening the images and computing the Euclidean distance between the resulting vectors. For more complex datasets, we employ embedding networks to extract useful features of the images. These embedding networks are arbitrary image classification models, trained on large datasets such as ImageNet [32] or CIFAR100 [69], which need to be chosen carefully and individually for each experiment as the clustering loss strongly depends on their ability to extract separable features for the known and especially the novel classes.

The feature distances are either computed in the high-dimensional feature space directly, or, for the sake of transparency and better visual control, in a low-dimensional re-arrangement. Applying the manifold learning medthod UMAP to the entire test data, we reduce the dimension of the feature space to two. The distance matrix is then computed as the Euclidean distances in the low-dimensional space for all pairs of OoD samples.

IV) Incremental Learning For class-incremental learning, we minimize three different loss functions defined in Equation 5.1, Equation 5.2 and Equation 5.3. The cross-entropy loss (Equation 5.1) is computed for in-distribution to mitigate catastrophic forgetting [94]. The OoD samples are pushed towards the novel classes by the extension loss (Equation 5.2), which is minimized whenever the probability mass is concentrated in the empty classes, *i.e.*,

$$\ell_{\text{ext}}(x) \to 0 \text{ for } \sum_{q=Q+1}^{Q+k} f_{\theta^{+k}}(x)_q \to 1,\ x \in \mathcal{U}^{\text{OoD}} . \tag{5.8}$$

The cluster loss (Equation 5.3) is computed for all pairs of OoD candidates contained in a batch. Thus, it has a runtime complexity of $\mathcal{O}(n^2)$, as for n OoD candidates, we need to compute $\frac{n^2-n}{2}$ terms. Furthermore, the minimum of the cluster loss is probably greater than zero, as samples which belong to the same class rarely share exactly the same features. To reach this minimum for two OoD samples x_i, x_j with a large distance, they should be assigned to different classes, *i.e.*, whenever $f_{\theta^{+k}}(x_i)_q$ is significantly different from zero, we desire that $f_{\theta^{+k}}(x_j)_q$ becomes small.

5.4 Adjustments for Semantic Segmentation

The softmax output of a semantic segmentation DNN $f_\theta : \mathcal{X} \to (0,1)^{|\mathcal{Z}|\times Q}$ provides class-probabilities for image pixels, denoted as $z = (h,w) \in \mathcal{Z}$. Thus, the OoD detector must not only identify OoD images, but also give information about their pixel positions. To store these information, we generate OoD instance masks as illustrated in Figure 5.4 by thresholding on the obtained OoD score and by distinguishing between connected components in the resulting OoD mask.

For semantic segmentation, the loss functions are computed for pixels of OoD objects instead of images. Let $\mathcal{Z}_s$ denote the set of pixel positions which belong to an OoD candidate $s \subseteq x$. The extension loss is computed equivalently to Equation 5.2 as

$$\ell_{\text{ext}}(s) = -\frac{1}{|\mathcal{Z}_s|} \sum_{z\in\mathcal{Z}_s} \frac{1}{Q} \sum_{q=1}^{Q} f_{\theta^{+k}}(x)_{z,q} . \tag{5.9}$$

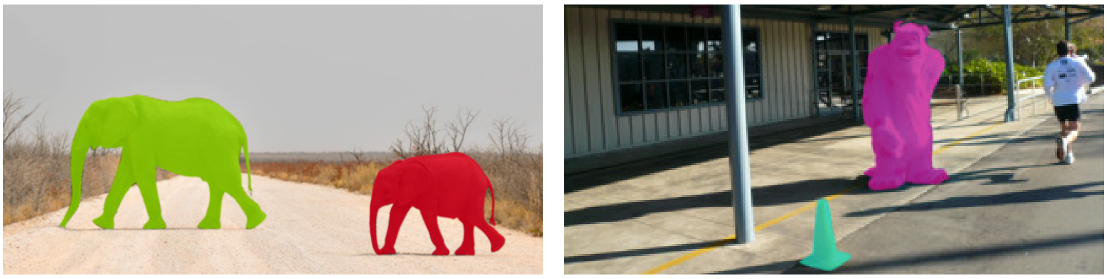


Figure 5.4: For semantic segmentation, each of the OoD objects is assigned a unique ID, no matter if they belong to the same novel class as the elephants, or to different classes as the cone and the monster costume [21].

For two OoD candidates $s_i \subseteq x_i, s_j \subseteq x_j$ with distance d_{ij}, the cluster loss is computed as

$$\ell_{\text{cluster}}(s_i, s_j) = \frac{\alpha}{Q+k} d_{ij} \sum_{q=1}^{Q+k} \overline{f_{\theta^{+k}}(x_i)_q}\ \overline{f_{\theta^{+k}}(x_j)_q}\ , \tag{5.10}$$

where

$$\overline{f_{\theta^{+k}}(x)_q} = \frac{1}{|\mathcal{Z}_s|} \sum_{z \in \mathcal{Z}_s} f_{\theta^{+k}}(x)_{z,q} \tag{5.11}$$

denotes the mean softmax probability over all pixels $z \in \mathcal{Z}_s$ for some class $q \in \{1, \dots, Q+k\}$.

For OoD detection in semantic segmentation, we adapt a meta regression approach [122, 123], using uncertainty measures such as the softmax entropy and further information which derives from the initial model's output, to estimate the prediction quality on a segment-level. Here, a segment denotes a connected component in the semantic segmentation mask, which is predicted by the initial model. That is, meta regression is a post-processing approach to quantify uncertainty aggregated over segments, and considering that the model likely is highly uncertain if confronted with OoD objects, it can be applied for OoD detection. In contrast to image classification, where images are either OoD or not, semantic segmentation is performed on images which can contain in-distribution and OoD pixels at the same time. Aggregating uncertainty scores across segments simplifies the detection of OoD objects as contiguous OoD pixels, since it removes the high uncertainty for class boundaries.

For an initial DNN, we use the training data to fit a gradient boosting model as meta regressor, which then estimates segment-wise uncertainty scores $u(s)$ for all segments $s \subseteq x \in \mathcal{U}^{\text{test}}$.

5.5 Numerical Experiments

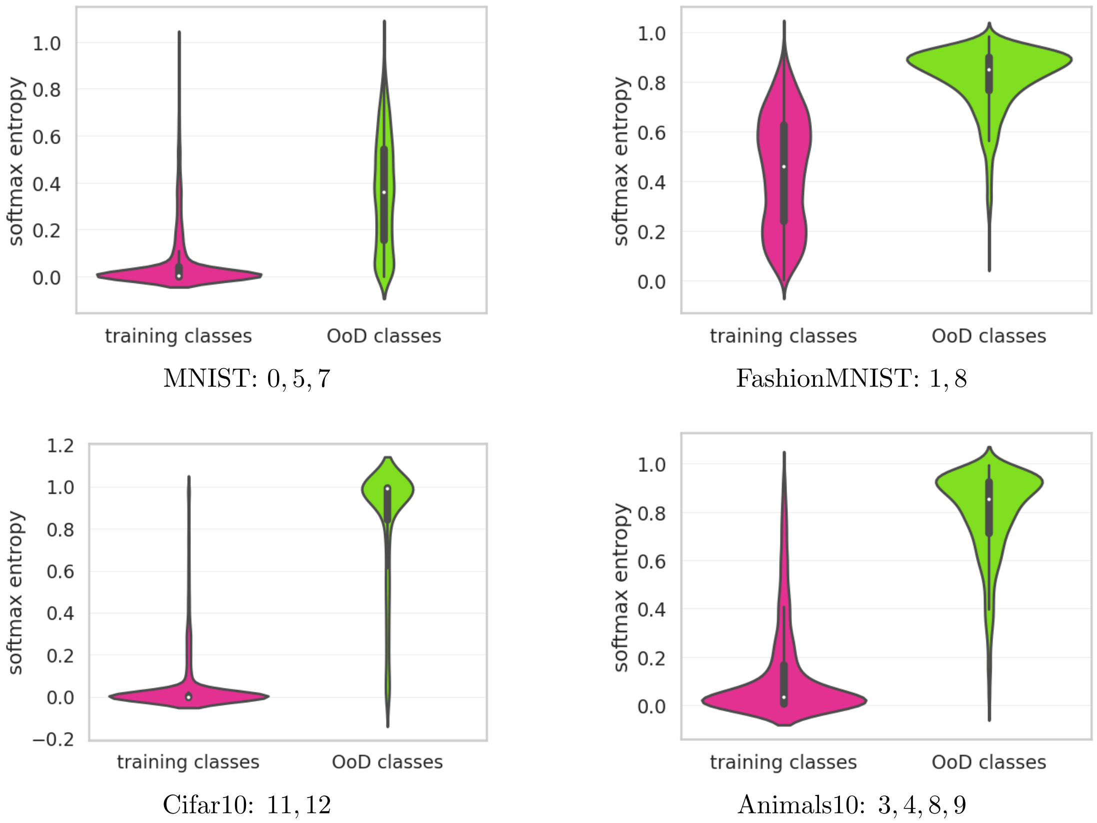
We perform several experiments for image classification on MNIST [74], FashionMNIST [144], CIFAR10 [69] and Animals10, as well as on Cityscapes [29] to evaluate our method for semantic segmentation. To this end, we extend the initial models by empty classes, *i.e.*, , neurons in the final classification layer with randomly initialized weights, and fine-tune them on OoD data, retraining with fixed encoder. For evaluation, we provide accuracy scores - separately for known and novel classes - for image classification, (mean) Intersection over Union (IoU), precision and recall values for semantic segmentation.

The OoD classes in the following experiments were all chosen in a way that they are semantically far away from each other. For example, the Animals10 classes *horse* (1), *cow* (6) and *sheep* (7) are semantically related, as they are all big animals which are mostly on the pasture, whereas *elephant* (2) and *spider* (8) are well separable classes, which is also visible in the two-dimensional feature space.

5.5.1 Experimental Setup

For each experiment, we consider the following dataset splits: the *training data* denotes images with ground truth for the initially known classes. We train the initial model on these images and replay them during the training of the extended model to avoid catastrophic forgetting. The *test data* consist of unlabeled images which include both, known and unknown classes. This dataset is fed into the OoD detector to identify *OoD data*, on which the model gets extended. The *evaluation dataset* includes images with ground truth for known and novel classes and is used to evaluate the models. If there are such labels available for the test data, evaluation images may be the same as the test images.

Our approach requires prior OoD detection. For image classification, we employed entropy maximization during training the initial model. The softmax entropy for the test data is visualized as a summary statistic in Figure 5.5 and sample-wise in Figure 5.6. We observe that the DNN exhibits high entropy scores on OoD data for all datasets except MNIST. However, the entropy for MNIST in-distribution data is sufficiently small, so that we detect most OoD samples using a threshold of $\tau = 0.1$. Further, the initial FashionMNIST DNN is uncertain regarding the in-distribution classes *t-shirt/top* (0), *pullover* (2) and *shirt* (6). However, this may be aleatoric uncertainty. To avoid too many false positive OoD predictions, we choose a high threshold $\tau = 0.75$. For the remaining datasets, in-distribution and OoD samples are well separable by the softmax entropy, thus, there is a large interval of proper thresholds.



MNIST: $0, 5, 7$

FashionMNIST: $1, 8$

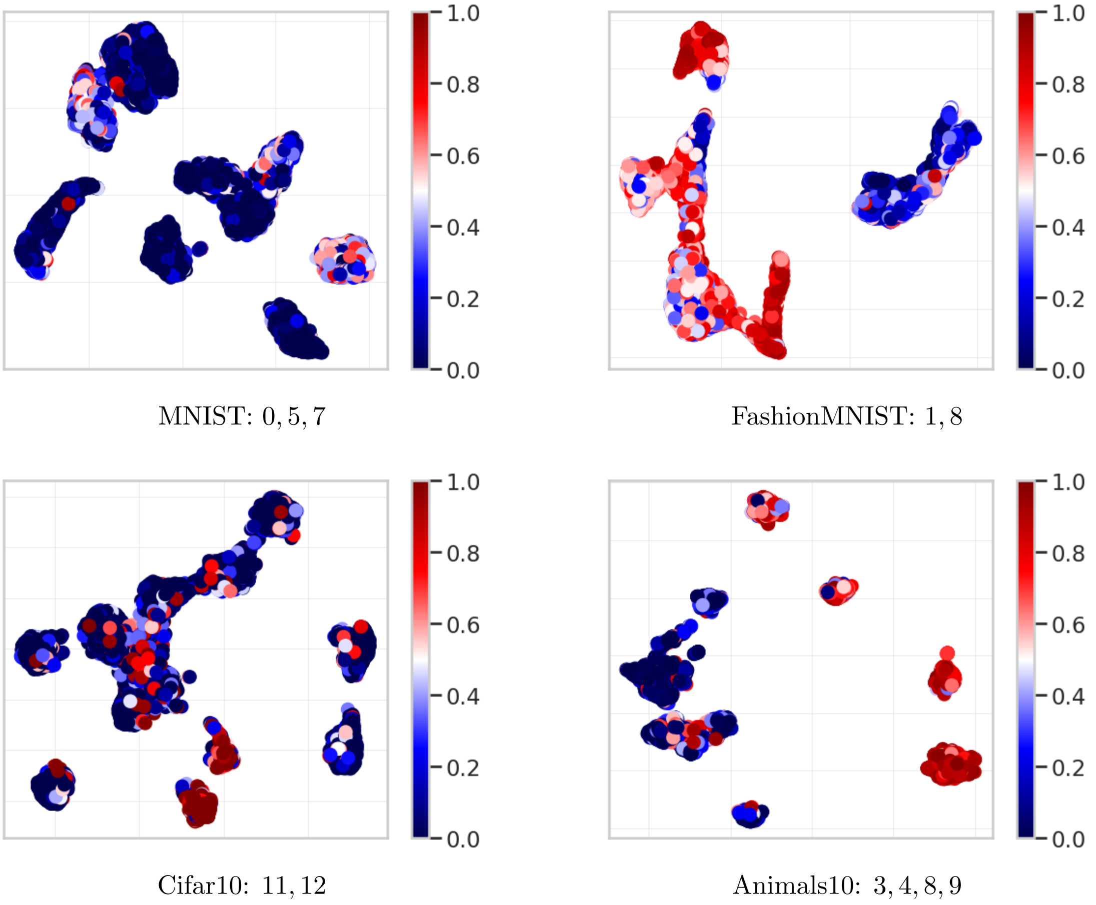
Cifar10: $11, 12$

Animals10: $3, 4, 8, 9$

Figure 5.5: Visualization of the softmax entropy, that the initial models exhibit on samples of known and OoD classes, respectively.

Here, we only provide the experimental setup for fine-tuning the extended model. For all experiments, we tuned the weighting parameters $\lambda_1, \lambda_2, \lambda_3$ in Equation 5.4 by observing all loss functions separately over several epochs using different parameter configurations to ensure that each loss term decreases. The following descriptions of the experiments, sorted by the datasets, include the network architecture, the known and novel classes, information about the dataset splits and the generation of the distance matrix. Further information about the experiments are provided in Table 5.1. We performed experiments using the Adam and the SGD optimizer as well as different batch sizes. In particular the batch size for semantic segmentation was bounded by memory limitations. The hyperparameters which are related to the loss functions, namely $\alpha, \lambda_1, \lambda_2, \lambda_3$, were selected by trying out and monitoring the course of the loss functions.

MNIST We employ a shallow neural network consisting of two convolutional layers, each followed by a ReLU activation function and max pooling, and a fully connected layer. From the digits $0, \ldots, 9$, we select $0, 5$ and 7 as novel classes. All



MNIST: $0, 5, 7$

FashionMNIST: $1, 8$

Cifar10: $11, 12$

Animals10: $3, 4, 8, 9$

Figure 5.6: Visualization of the softmax entropy per data sample, that the initial models exhibits on test samples.

images in the MNIST training set which belong to these classes are excluded from our training data. The MNIST test images compose our test set, and together with the original labels, our evaluation set. The distance matrix is computed as pixel-wise Euclidean distance between the OoD images.

FashionMNIST Using the same network architecture as for MNIST, our initial model is trained on eight out of ten classes, excluding the classes *trouser* (1) and *bag* (8). Our dataset splits are created analogously to those from MNIST. Also, the distance matrix is obtained analogously by computing the pixel-wise Euclidean distances between the OoD images.

CIFAR10 The setting for CIFAR10 differs slightly from the other experiments to ensure comparability with existing approaches. Thus, as initial model, we employ

dataset	MNIST	FashionMNIST	Cifar10	Animals10	Cityscapes
# empty classes	3	2	2	4	2
# epochs	30	30	30	30	200
optimizer	adam	sgd	sgd	adam	adam
learning rate	1e-2	1e-2	1e-2	5e-3	5e-3
momentum	-	0	0.9	-	-
weight decay	-	0	1e-4	-	-
batch size	2,500	500	1,000	1,000	10
α	5	2.5	5	2.5	2.5
λ_1	0.45	0.45	0.45	0.45	0.375
λ_2	0.45	0.45	0.45	0.45	0.375
λ_3	0.1	0.1	0.1	0.1	0.25

Table 5.1: Overview of training parameters for each dataset.

a ResNet18 which is trained on the whole CIFAR10 training split, including all ten classes. For testing, we enrich the CIFAR10 test split with images from CIFAR100. Therefore, we split CIFAR100 into an unlabeled and a labeled subset: the classes $\{0, \ldots, 49\}$ are possible OoD candidates, thus, all samples belonging to these classes are considered to be unlabeled. We extend the CIFAR10 test data by the classes *apple* (0) and *clock* (22), mapping them onto the labels (10) and (11), respectively. As before, we evaluate our models on the labeled test data. The labeled CIFAR100 subset includes the classes $\{50, \ldots, 99\}$ and is used together with the CIFAR10 training data to train a ResNet18 as an embedding network. To compute the distances, we feed the whole test data into this embedding network and extract the features of the penultimate layer. These are further projected into a 2D space with UMAP. Then, the distance matrix is computed as the pixel-wise Euclidean distance between the 2D representations of the OoD images.

Animals10 As initial model, we employ a ResNet18 which is trained on six out of ten classes. As novel classes we selected *butterfly* (3), *chicken* (4), *spider* (8) and *squirrel* (9). The dataset splits are obtained analogously to those from MNIST. The distances are computed as for CIFAR10, but employing a DenseNet201, which is trained on ImageNet with 1,000 classes, as embedding network.

Cityscapes For comparison reasons with the baseline, we adapt the experimental setup from [134], where the class labels *human (person, rider), car* and *bus* are excluded from the 19 Cityscapes evaluation classes. Like the baseline, we extend the DNN by two empty classes and exclude the class *bus* from the evaluation. Thus, we train a semantic segmentation DeepLabV3+ with WideResNet38 backbone on 2,500 training samples with 15 trainable classes. We apply meta regression to the Cityscapes test data and crop out image patches tailored to the

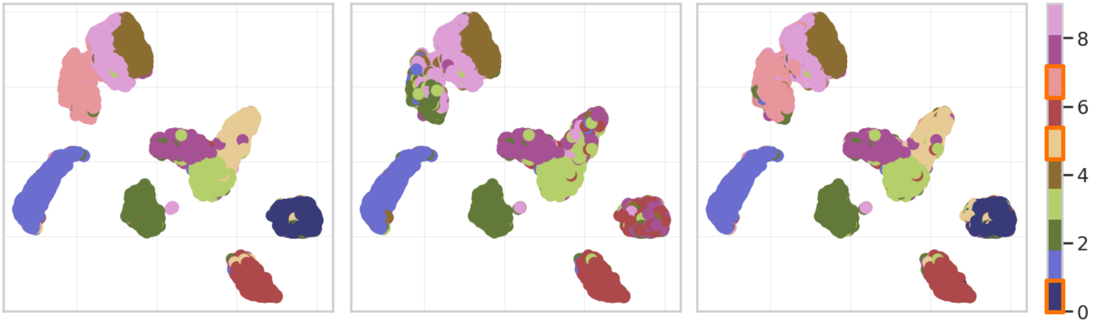


Figure 5.7: Visualized ground truth *(left)* and prediction of the MNIST dataset by the initial *(middle)* and extended *(right)* model. The three novel classes $0, 5$ and 7 are outlined in orange. The extended model's accuracy is $\sim 94\%$.

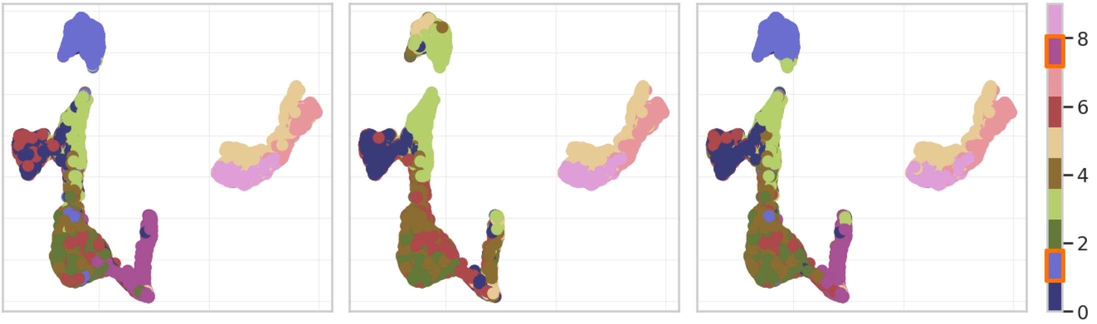


Figure 5.8: Visualized ground truth *(left)* and prediction of the FashionMNIST dataset by the initial *(middle)* and extended *(right)* model. The two novel classes 1 and 8 are outlined in orange. The extended model's accuracy is $\sim 85\%$.

predicted OoD segments, *i.e.*, , connected component of OoD pixels. Afterwards, we compute distances between these image patches analogously to Animals10 as the Euclidean distances between 2D representations of features which we obtain by feeding the patches into a DenseNet201 trained on 1,000 ImageNet classes.

5.5.2 Evaluation & Ablation Studies

We compare our evaluation results to the following baselines. For image classification, we employ the k-means clustering algorithm to pseudo-label the OoD data samples and fine-tune the model on the pseudo-labeled data using the cross-entropy loss. For semantic segmentation, we compare with the method presented in [134], which also employs clustering algorithms in the embedding space to obtain pseudo-labels. Furthermore, to get an idea of the maximum achievable performance, we train oracle models which have learned all available classes in a fully supervised manner.

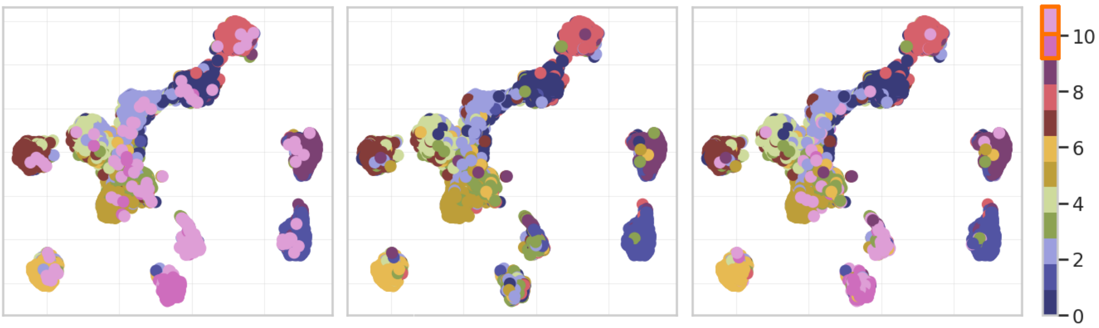


Figure 5.9: Visualized ground truth *(left)* and prediction of the CIFAR10 dataset by the initial *(middle)* and extended *(right)* model. The two novel classes 10 and 11 are outlined in orange. The extended model's accuracy is $\sim 89\%$.

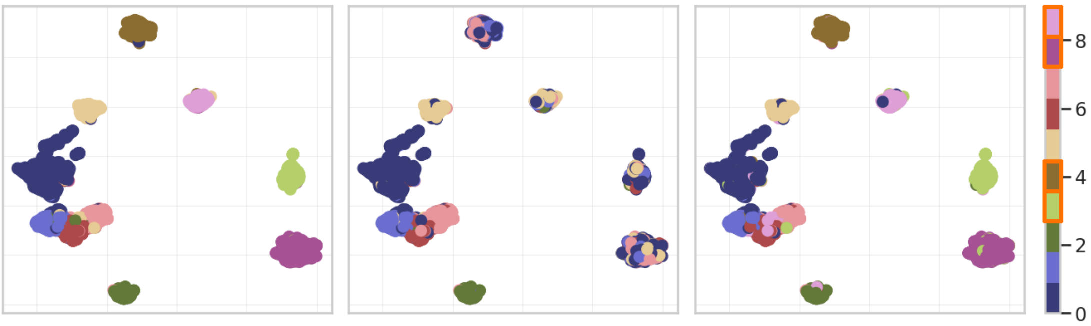


Figure 5.10: Visualized ground truth *(left)* and prediction of the Animals10 dataset by the initial *(middle)* and extended *(right)* model. The four novel classes $3, 4, 8$ and 9 are outlined in orange. The extended model's accuracy is $\sim 95\%$.

For the ablation studies, we evaluate our image classification approach on "clean" OoD data (−detection). Therefore, we do not detect the OoD samples in the test data by thresholding on some anomaly score, but by considering the ground truth. In this way, we simulate a perfect OoD detector. Since the results of our method are also affected by the quality of the distance matrix, we further analyze our method for a synthetic distance matrix (−−distance), where two OoD samples $x_i, x_j \in \mathcal{U}^{\text{OoD}}$ have a distance $d(x_i, x_j) = 0$ if they stem from the same class, $d(x_i, x_j) = 1$ otherwise. Thus, the OoD samples are labeled by the distance matrix and the fine-tuning is supervised, allowing a pure comparison of our loss functions with the cross-entropy loss. We do not provide ablation studies for semantic segmentation, since the Cityscapes test data does not include publicly available annotations.

Image Classification As shown in Table 5.2 and visualized in Figure 5.7, Figure 5.8, Figure 5.9 and Figure 5.10, our approach exceeds the baseline's accuracy for novel classes by 36.60 and 24.09 percentage points (pp) for CIFAR10 and Animals10, respectively. This is mainly caused by in-distribution samples which are false positive OoD predictions, or by OoD samples which are embedded far away from their class centroids. Consequently, different OoD classes are assigned to the same cluster by the k-means algorithm. As our approach uses soft labels, the DNN is more likely to reconsider the choice of the OoD detector during fine-tuning.

In the ablation studies, we omit the OoD detector (−detection) and select the OoD samples based on their ground truth instead. Thereby, we observe an improvement of the accuracy of novel classes for the CIFAR10 and Animals10 datasets, while the performance remains constant for FashionMNIST and significantly decreases for MNIST. We further compute a ground truth distance matrix (−−distance) with distances 0 and 1 for samples belonging to the same or to different classes, respectively. Since this is supervised fine-tuning, these DNNs are comparable to oracles. We observe, that the oracles tend to perform better on the initial and worse on the novel classes. However, this might be a consequence of the class-incremental learning.

Image Classification								
			supervised		unsupervised		ablation studies	
dataset	**OoD**	**accuracy**	**initial**	**oracle**	**ours**	**baseline**	**−detection**	**−−distance**
MNIST	0 5 7	known	96.68%	98.54%	**96.20%**	95.94%	97.45%	96.54%
		novel	-	95.85%	**97.94%**	84.62%	74.52%	97.00%
FashionMNIST	1 8	known	81.54%	83.75%	81.41%	**85.08%**	81.89%	81.39%
		novel	-	90.83%	90.05%	**92.85%**	89.90%	95.00%
CIFAR10	10 11	known	91.45%	91.86%	**90.51%**	90.29%	88.90%	86.94%
		novel	-	89.53%	**70.00%**	33.40%	78.80%	87.00%
Animals10	3 4 8 9	known	96.29%	95.80%	**93.76%**	92.78%	94.46%	95.20%
		novel	-	97.65%	**96.68%**	72.59%	97.02%	97.90%

Table 5.2: Quantitative evaluation of the image classification experiments. For all evaluated models, the accuracy is stated separately for the previously-known and the unlabeled novel classes. The highest scores for the unsupervised approaches are bolded.

Semantic Segmentation The quantitative results of our semantic segmentation method, reported in Table 5.3, demonstrate, that the empty classes are "filled" with the novel concepts *human* and *car*. Thereby, the performance on the previously-known classes is similar to the baseline even without including a distillation loss [98]. For the *car* class, our method outperforms the baseline with respect to IoU (+2.87 pp), precision (+0.55 pp) and recall (+3.06 pp). We lose performance in terms of IoU for the *human* class due to a higher tendency for false positives, *cf.* Figure 5.12. However, the false negative rate is significantly reduced, which is indicated by an increase in the recall value of 26.89 pp. The improved recall score is also visible in Figure 5.11, showing two examples from the Cityscapes validation dataset. In the top row, several pedestrians are crossing the street, which are mostly segmented by our DNN, whereas the baseline DNN

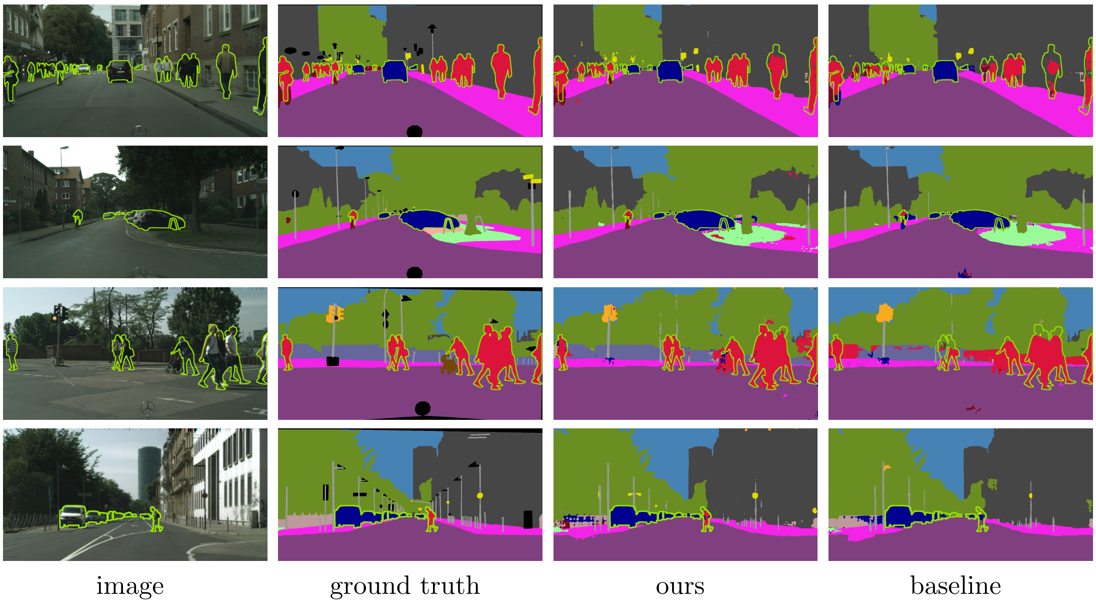


Figure 5.11: Visual comparison of the segmentation masks produced by our method and by the baseline for two image cutouts from the Cityscapes validation dataset. The ground truth contours of the novel classes are highlighted with green.

mostly misses the persons in the center as well as all heads. In the bottom row, the person in front of the car is completely overlooked by the baseline, and also some cars in the background are missed.

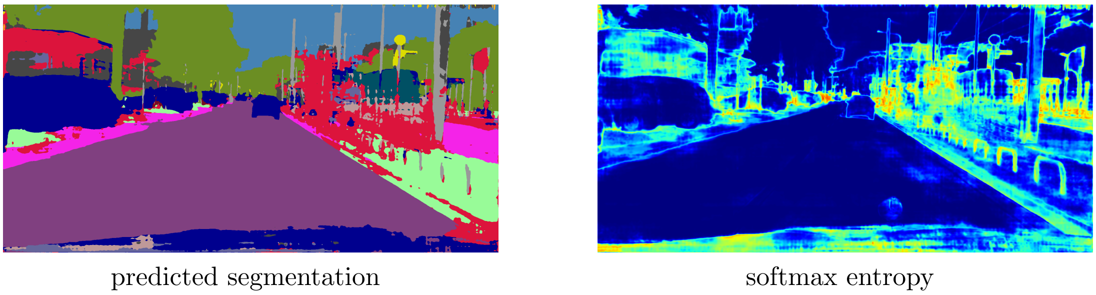


Figure 5.12: For highly uncertain regions, the extended DNN tends to predict the novel *human* class, which causes the low precision score.

When examining the OoD masks, we observed that the connected components are often very extensive, which is caused by neighboring OoD objects. Thus, the embedding space contains many large image patches which are not tailored to a single OoD object, but rather to a number of parked cars, a crowd of people or even a bicyclist riding next to a car, which appreciably impairs our results.

Semantic Segmentation						
			supervised		unsupervised	
dataset	**class**	**metric**	**initial**	**oracle**	**ours**	**baseline**
Cityscapes	$0,\dots,14$	mean IoU	56.99%	77.28%	**59.72%**	57.52%
	15 (human)	IoU	-	81.90%	33.87%	**40.22%**
	16 (car)	IoU	-	94.94%	**84.14%**	81.27%
	$0,\dots,14$	mean precision	65.75%	88.03%	**84.63%**	78.53%
	15 (human)	precision	-	89.22%	37.80%	**68.74%**
	16 (car)	precision	-	96.83%	**87.11%**	86.56%
	$0,\dots,14$	mean recall	80.88%	85.38%	65.38%	**65.78%**
	15 (human)	recall	-	90.90%	**76.54%**	49.65%
	16 (car)	recall	-	97.99%	**96.11%**	93.05%

Table 5.3: Quantitative evaluation of the semantic segmentation experiment on the Cityscapes dataset. IoU, precision and recall values are provided for both novel classes, as well as averaged over the previously-known classes. The highest scores for the unsupervised approaches are bolded.

5.6 Conclusion & Outlook

In our work, we proposed a solution to open world classification for image classification and semantic segmentation by learning novel classes in an unsupervised manner. We suggested postulating empty classes, which allow one to capture newly observed classes in an incremental learning approach. This way, we allow our model to detect new classes in a flexible manner, potentially whitewashing mistakes of previous OoD detectors.

As our method employs several hyperparameters, *e.g.* , to specify the number of novel empty classes, we envision an automatic derivation of the optimal number of new classes as future work. In this regard, replacing the *Elbow method* in the eventual clustering by more suitable criteria appears desirable [126]. Moreover, we shall investigate approaches to improve the generalizability of our approach to embedding models of arbitrary kind to derive distance matrices, not being tailored to specific datasets. Furthermore, the semantic segmentation performance could be improved by incorporating depth information into the OoD segmentation method to obtain OoD candidates on instance- instead of segment-level.

Chapter 6

Conclusion & Outlook

This thesis addressed the discovery and incremental learning of novel classes in the semantic segmentation of street scenes. While state-of-the-art convolutional neural networks already achieve remarkable performance when their training and testing domains coincide, they are not capable of continuously adapting to evolving environments. Neural networks used as perception systems in automated vehicles require a huge amount of labeled training data coupled with reliable anomaly segmentation approaches.

Conclusion To enhance the comparability of anomaly segmentation approaches, and motivated by the lack of appropriate data, we invested time in the acquisition and labeling of several datasets, along with a benchmark suite to evaluate anomaly segmentation methods using pixel- and segment-level performance metrics. While using anomaly segmentation as an alerting system is sufficient for infrequently occurring anomalous objects, it seemed reasonable to include frequently occurring unknown classes in the DNN's semantic space. We realized that large amounts of unlabeled data can be easily collected and examined for anomalies. To discover novel classes among the anomalies, we proposed a feature clustering approach that extracts features of image patches tailored to the predicted anomalies, using an image classification network trained on $1,000$ classes. The intention of this was that an almost *omniscient* classification DNN is capable of producing separable features for different types of anomalies, allowing the detection of novel classes by clustering algorithms.

We further investigated incremental learning of novel classes using pseudo-labels obtained by combining the binary anomaly segmentation masks with the division of the anomalies into different clusters. Limited by the lack of suitable datasets, we performed several experiments on the Cityscapes dataset with artificial novel classes obtained by class exclusion. We found that our method is prone to errors in several aspects. First, anomaly segmentation approaches typically produce false positive and false negative predictions. Thus, instances from known classes contaminate the feature space, complicating the feature clustering. In the context of video sequences, we proposed to use tracking information as a filter, assuming that true positives are consistently tracked over multiple frames. For single frames, we purposely inserted known classes into the feature space to identify anomaly predictions that are likely to be false positives. The main problems with false negative anomaly predictions are, that they reduce the amount of training data, and that pseudo-labels are very fuzzy when only fragments of the objects are detected. Furthermore, binary anomaly masks are not able to detect instance-level anomalies. Another hindrance is the quality of the extracted features. We observed, that features of related classes such as *bus* and *train* are not separable in the proposed feature space. Furthermore, the anomaly classes have different densities in the feature space depending on their visual diversity, which impedes the clustering.

We then developed a method that replaces hard pseudo-labels with pairwise distances between the anomalies. Therefore, the clustering was incorporated into the training of the extended DNN in the form of an additional loss function. The intention was to preferably, but not strictly, cluster the anomalies into novel classes. Instead of fine-tuning the DNN only on the data related to the formed clusters, we used the total amount of OoD data. We observed, that the increased amount of data significantly improved the recall scores for the anomaly classes at the cost of an increased number of false positives. We noticed, that these false positives are usually accompanied by high softmax entropy.

Outlook Both of the open world semantic segmentation methods we presented are based on anomaly segmentation and feature embedding methods that can easily be replaced in a *plug-and-play* manner by more advanced approaches. Anomaly scores obtained by any method can be averaged over segments and then used as (additional) input to the meta regression model. Instance-level anomalies could be detected by incorporating depth-information, or by using class-agnostic object detectors as proposed in [7]. Furthermore, zero-shot learning and self-supervised contrastive learning are promising research directions to improve the performance of the feature extractor, *e.g.* by learning visual features from natural language supervision [115]. Self-supervised feature extractors can even be leveraged for unsupervised training of a semantic segmentation DNN [152]. Semi-supervised learning could be applied as a refinement of our methods by using and iteratively updating the confident predictions of the extended DNN as pseudo-labels for unlabeled data. In summary, unsupervised open world semantic segmentation is a growing area of research with much potential for future work, but there is still a lack of appropriate datasets to provide a fair evaluation.

List of Figures

List of Tables

Bibliography

[1] S. Albelwi, *Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging*, Entropy, 24 (2022).

[2] M. Ankerst, M. M. Breunig, H. peter Kriegel, and J. Sander, *Optics: Ordering points to identify the clustering structure*, ACM Press, 1999, pp. 49–60.

[3] R. Arandjelović and A. Zisserman, *Multiple queries for large scale specific object retrieval*, in BMVC, 2012.

[4] H. Bay, T. Tuytelaars, and L. V. Gool, *Surf: Speeded up robust features*, in ECCV, 2006.

[5] R. Bellman, *Dynamic programming*, Science, 153 (1966), pp. 34–37.

[6] A. Bendale and T. E. Boult, *Towards open world recognition*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2015), pp. 1893–1902.

[7] V. Besnier, A. Bursuc, A. Briot, and D. Picard, *Instance-aware observer network for out-of-distribution object segmentation*, in NeurIPS ML Safety Workshop, 2022.

[8] P. Bevandic, I. Kreso, M. Orsic, and S. Segvic, *Simultaneous semantic segmentation and outlier detection in presence of domain shift*, in GCPR, 2019.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[10] H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena, *Fishyscapes: A benchmark for safe semantic segmentation in autonomous*

driving, 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), (2019), pp. 2403–2412.

[11] H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena, *The Fishyscapes Benchmark: Measuring Blind Spots in Semantic Segmentation*, International Journal of Computer Vision (IJCV), 129 (2021).

[12] D. Bogdoll, S. Uhlemeyer, K. Kowol, and J. M. Zollner, *Perception datasets for anomaly detection in autonomous driving: A survey*, ArXiv, abs/2302.02790 (2023).

[13] K. Boyd, K. H. Eng, and C. D. Page, *Area under the precision-recall curve: Point estimates and confidence intervals*, in Machine Learning and Knowledge Discovery in Databases, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 451–466.

[14] M. Bucher, T.-H. VU, M. Cord, and P. Perez, *Zero-shot semantic segmentation*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, eds., vol. 32, Curran Associates, Inc., 2019.

[15] O. Calin, *Deep Learning Architectures: A Mathematical Approach*, Springer Publishing Company, Incorporated, 1st ed., 2020.

[16] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, *Emerging properties in self-supervised vision transformers*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 9630–9640.

[17] F. M. Castro, M. J. Marín-Jiménez, N. G. Mata, C. Schmid, and A. Karteek, *End-to-end incremental learning*, ArXiv, abs/1807.09536 (2018).

[18] G. Cauwenberghs and T. A. Poggio, *Incremental and decremental support vector machine learning*, in NIPS, 2000.

[19] J. Cen, P. Yun, J. Cai, M. Y. Wang, and M. Liu, *Deep metric learning for open world semantic segmentation*, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), (2021), pp. 15313–15322.

[20] J. Cen, P. Yun, S. Zhang, J. Cai, D. Luan, M. Y. Wang, M. Liu, and M. Tang, *Open-world semantic segmentation for lidar point clouds*, ArXiv, abs/2207.01452 (2022).

[21] R. Chan, K. Lis, S. Uhlemeyer, H. Blum, S. Honari, R. Siegwart, P. Fua, M. Salzmann, and M. Rottmann, *SegmentMeIfYouCan: A Benchmark for Anomaly Segmentation*, in Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2021.

[22] R. Chan, M. Rottmann, and H. Gottschalk, *Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation*, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 10 2021, pp. 5128–5137.

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. P. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 40 (2018), pp. 834–848.

[24] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, *Rethinking atrous convolution for semantic image segmentation*, ArXiv, abs/1706.05587 (2017).

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

[26] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, *A simple framework for contrastive learning of visual representations*, in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 1597–1607.

[27] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, *Big self-supervised models are strong semi-supervised learners*, in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[28] X. Chen, Y. Yuan, G. Zeng, and J. Wang, *Semi-supervised semantic segmentation with cross pseudo supervision*, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2021), pp. 2613–2622.

[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*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 3213–3223.

[30] C. Creusot and A. Munawar, *Real-time small obstacle detection on highways using compressive rbm road reconstruction*, in 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 162–167.

[31] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*, Cambridge University Press, 2020.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in CVPR, 2009.

[33] T. Devries and G. W. Taylor, *Learning confidence for out-of-distribution detection in neural networks*, ArXiv, abs/1802.04865 (2018).

[34] G. Di Biase, H. Blum, R. Siegwart, and C. Cadena, *Pixel-wise anomaly detection in complex driving scenes*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021, pp. 16918–16927.

[35] C. Doersch, A. K. Gupta, and A. A. Efros, *Unsupervised visual representation learning by context prediction*, 2015 IEEE International Conference on Computer Vision (ICCV), (2015), pp. 1422–1430.

[36] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, in KDD, 1996.

[37] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, *The pascal visual object classes (voc) challenge*, International Journal of Computer Vision, 88 (2009), pp. 303–338.

[38] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, *Query by image and video content: the qbic system*, Computer, 28 (1995), pp. 23–32.

[39] D. A. Forsyth, *Applied machine learning*, in Cambridge International Law Journal, 2019.

[40] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, p. 1050–1059.

[41] W. V. Gansbeke, S. Vandenhende, S. Georgoulis, and L. V. Gool, *Unsupervised semantic segmentation by contrasting object mask proposals*, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), (2021), pp. 10032–10042.

[42] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. G. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, *A2d2: Audi autonomous driving dataset*, ArXiv, abs/2004.06320 (2020).

[43] S. Gidaris, P. Singh, and N. Komodakis, *Unsupervised representation learning by predicting image rotations*, in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[44] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[45] Y. Grandvalet and Y. Bengio, *Semi-supervised learning by entropy minimization*, in Conférence francophone sur l'apprentissage automatique, 2004.

[46] M. Grcić., P. Bevandić., and S. Segvić., *Dense Open-set Recognition with Synthetic Outliers Generated by Real NVP*, in International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), 2021.

[47] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, *Bootstrap your own latent - A new approach to self-supervised learning*, in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[48] S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell, *Open-vocabulary object retrieval*, in Robotics: Science and Systems, 2014.

[49] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, Springer, 2 ed., 2009.

[50] J. He and F. Zhu, *Unsupervised continual learning via pseudo labels*, ArXiv, abs/2104.07164 (2021).

[51] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2015), pp. 770–778.

[52] A. M. Hein, *Identification and bridging of semantic gaps in the context of multi-domain engineering*, in Proceedings 2010 Forum on Philosophy, Engineering & Technology, 01 2010.

[53] M. Hein, M. Andriushchenko, and J. Bitterwolf, *Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6 2019.

[54] D. Hendrycks, S. Basart, M. Mazeika, A. Zou, J. Kwon, M. Mostajabi, J. Steinhardt, and D. Song, *Scaling Out-of-Distribution Detection for Real-World Settings*, in International Conference on Machine Learning (ICML), 2022.

[55] D. Hendrycks and K. Gimpel, *A baseline for detecting misclassified and out-of-distribution examples in neural networks*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[56] D. Hendrycks, M. Mazeika, and T. Dietterich, *Deep anomaly detection with outlier exposure*, Proceedings of the International Conference on Learning Representations, (2019).

[57] G. E. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, ArXiv, abs/1503.02531 (2015).

[58] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, *Natural language object retrieval*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 4555–4564.

[59] G. Huang, Z. Liu, and K. Q. Weinberger, *Densely connected convolutional networks*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 2261–2269.

[60] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods*, Mach. Learn., 110 (2021), pp. 457–506.

[61] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in Proceedings of the 32nd International Conference on Machine Learning, F. Bach and D. Blei, eds., vol. 37 of Proceedings of Machine Learning Research, Lille, France, 07–09 Jul 2015, PMLR, pp. 448–456.

[62] K. J. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, *Towards open world object detection*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021, pp. 5830–5840.

[63] N. Jourdan, E. Rehder, and U. Franke, *Identification of uncertainty in artificial neural networks*, in Proceedings of the 13th Uni-DAS e.V. Workshop Fahrerassistenz und automatisiertes Fahren, 7 2020.

[64] H. Jung, J. Ju, M. Jung, and J. Kim, *Less-forgetful learning for domain expansion in deep neural networks*, in AAAI, 2018.

[65] D. Kim, J. Bae, Y. Jo, and J. Choi, *Incremental learning with maximum entropy regularization: Rethinking forgetting and intransigence*, ArXiv, abs/1902.00829 (2019).

[66] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015.

[67] A. Kirillov, K. He, R. B. Girshick, C. Rother, and P. Dollár, *Panoptic Segmentation*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, Computer Vision Foundation / IEEE, 2019, pp. 9404–9413.

[68] M. Klingner, A. Bär, P. Donn, and T. Fingscheidt, *Class-incremental learning for semantic segmentation re-using neither old data nor old labels*, 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), (2020), pp. 1–8.

[69] A. Krizhevsky, *Learning multiple layers of features from tiny images*, 2009.

[70] J. B. Kruskal, *Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis*, Psychometrika, 29 (1964), pp. 1–27.

[71] ——, *Nonmetric multidimensional scaling: A numerical method*, Psychometrika, 29 (1964), pp. 115–129.

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

[73] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, in NIPS, 2016.

[74] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.

[75] Y. LeCun and C. Cortes, *The mnist database of handwritten digits*, 2005.

[76] D.-H. Lee, *Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks*, ICML 2013 Workshop : Challenges in Representation Learning (WREPL), (2013).

[77] K. Lee, K. Lee, H. Lee, and J. Shin, *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018, pp. 7167–7177.

[78] K. Lee, K. Lee, J. Shin, and H. Lee, *Overcoming catastrophic forgetting with unlabeled data in the wild*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 312–321.

[79] C. Li, J. Yang, P. Zhang, M. Gao, B. Xiao, X. Dai, L. Yuan, and J. Gao, *Efficient self-supervised vision transformers for representation learning*, in The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[80] Z. Li and D. Hoiem, *Learning without forgetting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 40 (2018), pp. 2935–2947.

[81] S. Liang, Y. Li, and R. Srikant, *Enhancing the reliability of out-of-distribution image detection in neural networks*, in International Conference on Learning Representations, 2018.

[82] 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*, in European Conference on Computer Vision, 2014.

[83] K. Lis, S. Honari, P. Fua, and M. Salzmann, *Detecting road obstacles by erasing them*, 2020.

[84] K. Lis, K. Nakka, P. Fua, and M. Salzmann, *Detecting the unexpected via image resynthesis*, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 10 2019.

[85] J. Liu, D. Wang, S. Yu, X. Li, Z. Han, and Y. Tang, *A survey of image clustering: Taxonomy and recent methods*, in 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2021, pp. 375–380.

[86] Q. Liu, Y. Wen, J. Han, C. Xu, H. Xu, and X. Liang, *Open-world semantic segmentation via contrasting and clustering vision-language embedding*, in European Conference on Computer Vision, 2022.

[87] G. LoweDavid, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, (2004).

[88] K. Maag, R. Chan, S. Uhlemeyer, K. Kowol, and H. Gottschalk, *Two Video Data Sets for Tracking and Retrieval of Out of Distribution Objects*, in Asian Conference on Computer Vision (ACCV), 2023.

[89] K. Maag, M. Rottmann, and H. Gottschalk, *Time-dynamic estimates of the reliability of deep semantic segmentation networks*, 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), (2020), pp. 502–509.

[90] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, 1967.

[91] S. Maji and S. Bose, *Cbir using features derived by deep learning*, ACM/IMS Transactions on Data Science (TDS), 2 (2021), pp. 1 – 24.

[92] S. Mallat, *A wavelet tour of signal processing (2. ed.).*, Academic Press, 1999.

[93] J. Mao, J. Huang, A. Toshev, O.-M. Camburu, A. L. Yuille, and K. P. Murphy, *Generation and comprehension of unambiguous object descriptions*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 11–20.

[94] M. McCloskey and N. Cohen, *Catastrophic interference in connectionist networks: The sequential learning problem*, Psychology of Learning and Motivation, 24 (1989), pp. 109–165.

[95] L. McInnes and J. Healy, *Umap: Uniform manifold approximation and projection for dimension reduction*, ArXiv, abs/1802.03426 (2018).

[96] A. Meinke and M. Hein, *Towards neural networks that provably know when they don't know*, in International Conference on Learning Representations, 2020.

[97] U. Michieli and P. Zanuttigh, *Incremental learning techniques for semantic segmentation*, 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), (2019), pp. 3205–3212.

[98] ——, *Knowledge distillation for incremental learning in semantic segmentation*, Computer Vision and Image Understanding, 205 (2021), p. 103167.

[99] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in Advances in Neural Information Processing Systems, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds., vol. 26, Curran Associates, Inc., 2013.

[100] J. Mukhoti and Y. Gal, *Evaluating bayesian deep learning methods for semantic segmentation*, 2019.

[101] A. Munawar, P. Vinayavekhin, and G. De Magistris, *Limiting the reconstruction capability of generative neural network using negative learning*, in 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), 2017, pp. 1–6.

[102] K. P. Murphy, *Probabilistic Machine Learning: An introduction*, MIT Press, 2022.

[103] E. Naaz and T. Kumar, *Enhanced content based image retrieval using machine learning techniques*, 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), (2017), pp. 1–12.

[104] Y. Nakajima, B. Kang, H. Saito, and K. Kitani, *Incremental class discovery for semantic segmentation with rgbd sensing*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 972–981.

[105] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, *The mapillary vistas dataset for semantic understanding of street scenes*, 2017 IEEE International Conference on Computer Vision (ICCV), (2017), pp. 5000–5009.

[106] J. Nixon, J. Z. Liu, and D. Berthelot, *Semi-supervised class discovery*, ArXiv, abs/2002.03480 (2020).

[107] H. Noh, S. Hong, and B. Han, *Learning deconvolution network for semantic segmentation*, 2015 IEEE International Conference on Computer Vision (ICCV), (2015), pp. 1520–1528.

[108] P. Oberdiek, M. Rottmann, and G. A. Fink, *Detection and retrieval of out-of-distribution objects in semantic segmentation*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), (2020), pp. 1331–1340.

[109] Y. Ouali, C. Hudelot, and M. Tami, *An overview of deep semi-supervised learning*, ArXiv, abs/2006.05278 (2020).

[110] J. Parmar, S. S. Chouhan, V. Raychoudhury, and S. S. Rathore, *Open-world machine learning: Applications, challenges, and opportunities*, ACM Computing Surveys, 55 (2021), pp. 1 – 37.

[111] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, *Context encoders: Feature learning by inpainting*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016), pp. 2536–2544.

[112] K. Pearson F.R.S., *Liii. on lines and planes of closest fit to systems of points in space*, Philosophical Magazine Series 1, 2, pp. 559–572.

[113] P. Pinggera, S. Ramos, S. K. Gehrig, U. Franke, C. Rother, and R. Mester, *Lost and found: detecting small road hazards for self-driving vehicles*, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2016), pp. 1099–1106.

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

[115] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, *Learning transferable visual models from natural language supervision*, in International Conference on Machine Learning, 2021.

[116] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, in 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2016.

[117] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, *icarl: Incremental classifier and representation learning*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 5533–5542.

[118] H. E. Robbins, *A stochastic approximation method*, Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[119] A. Robins, *Catastrophic forgetting, rehearsal and pseudorehearsal*, Connection Science, 7 (1995), pp. 123–146.

[120] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[121] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, Psychological Review, 65 (1958), pp. 386–408.

[122] M. Rottmann, P. Colling, T.-P. Hack, F. Hüger, P. Schlicht, and H. Gottschalk, *Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities*, 2020 International Joint Conference on Neural Networks (IJCNN), (2020), pp. 1–9.

[123] M. Rottmann and M. Schubert, *Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), (2019), pp. 1361–1369.

[124] J. Sammon, *A nonlinear mapping for data structure analysis*, IEEE Transactions on Computers, C-18 (1969), pp. 401–409.

[125] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult, *Toward open set recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 1757–1772.

[126] E. Schubert, *Stop using the elbow criterion for k-means and how to choose the number of clusters instead*, ArXiv, abs/2212.12189 (2022).

[127] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning - From Theory to Algorithms.*, Cambridge University Press, 2014.

[128] L. Shu, H. Xu, and B. Liu, *Unseen class discovery in open-world classification*, ArXiv, abs/1801.05609 (2018).

[129] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. C. Jain, *Content-based image retrieval at the end of the early years*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 1349–1380.

[130] O. Tasar, Y. Tarabalka, and P. Alliez, *Incremental learning for semantic segmentation of large-scale remote sensing data*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12 (2019), pp. 3524–3537.

[131] Z. Tian, H. Zhao, M. Shu, Z. Yang, R. Li, and J. Jia, *Prior guided feature enrichment network for few-shot segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2020), pp. 1050–1065.

[132] C. Troisemaine, J. Flocon-Cholet, S. Gosselin, S. Vaton, A. Reiffers-Masson, and V. Lemaire, *A method for discovering novel classes in tabular data*, 2022 IEEE International Conference on Knowledge Graph (ICKG), (2022), pp. 265–274.

[133] S. Uhlemeyer, J. Lienen, E. Hüllermeier, and H. Gottschalk, *Detecting novelties with empty classes*, to appear on ArXiv, (2023).

[134] S. Uhlemeyer, M. Rottmann, and H. Gottschalk, *Towards unsupervised open world semantic segmentation*, in The 38th Conference on Uncertainty in Artificial Intelligence, 2022.

[135] L. van der Maaten and G. E. Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research, 9 (2008), pp. 2579–2605.

[136] A. Vasilev, V. Golkov, I. Lipp, E. Sgarlata, V. Tomassini, D. K. Jones, and D. Cremers, *q-space novelty detection with variational autoencoders*, ArXiv, abs/1806.02997 (2018).

[137] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, *Deep high-resolution representation learning for visual recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (2021), pp. 3349–3364.

[138] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, *Panet: Few-shot image semantic segmentation with prototype alignment*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 9196–9205.

[139] Y. Wang, H. Wang, Y. Shen, J. Fei, W. Li, G. Jin, L. Wu, R. Zhao, and X. Le, *Semi-supervised semantic segmentation using unreliable pseudo-labels*, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2022), pp. 4238–4247.

[140] Y. Wang, Q. Yao, J. T.-Y. Kwok, and L. M. Ni, *Generalizing from a few examples: A survey on few-shot learning*, arXiv: Learning, (2019).

[141] Z. Wang, B. Salehi, A. Gritsenko, K. R. Chowdhury, S. Ioannidis, and J. G. Dy, *Open-world class discovery with kernel networks*, 2020 IEEE International Conference on Data Mining (ICDM), (2020), pp. 631–640.

[142] Y. Xia, Y. Zhang, F. Liu, W. Shen, and A. Yuille, *Synthesize then compare: Detecting failures and anomalies for semantic segmentation*, in Proceedings of the European Conference on Computer Vision (ECCV), 2020.

[143] Y. Xian, S. Choudhury, Y. He, B. Schiele, and Z. Akata, *Semantic projection network for zero- and few-label semantic segmentation*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2019), pp. 8248–8257.

[144] H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, ArXiv, abs/1708.07747 (2017).

[145] D. Xu and Y. jie Tian, *A comprehensive survey of clustering algorithms*, Annals of Data Science, 2 (2015), pp. 165–193.

[146] X. Xu, T. Wang, Y. Shi, H. Yuan, Q. Jia, M. Huang, and J. Zhuang, *Whole heart and great vessel segmentation in congenital heart disease using deep neural networks and graph matching*, in Medical Image Computing and Computer Assisted Intervention – MICCAI 2019, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, eds., Cham, 2019, Springer International Publishing, pp. 477–485.

[147] J. Yang, K. Zhou, Y. Li, and Z. Liu, *Generalized out-of-distribution detection: A survey*, ArXiv, abs/2110.11334 (2021).

[148] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, *Adversarial feature alignment: Avoid catastrophic forgetting in incremental task lifelong learning*, Neural Computation, 31 (2019), pp. 2266–2291.

[149] M. Yi-de, L. Qing, and Q. Zhi-bai, *Automated image segmentation using improved pcnn model based on cross-entropy*, in Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004., 2004, pp. 743–746.

[150] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, *Bdd100k: A diverse driving dataset for heterogeneous multitask learning*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2020), pp. 2633–2642.

[151] S. Yun, H. Lee, J. Kim, and J. Shin, *Patch-level representation learning for self-supervised vision transformers*, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2022), pp. 8344–8353.

[152] A. Zadaianchuk, M. Kleindessner, Y. Zhu, F. Locatello, and T. Brox, *Unsupervised semantic segmentation with self-supervised object-centric representations*, in The Eleventh International Conference on Learning Representations, 2023.

[153] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, *mixup: Beyond empirical risk minimization*, ArXiv, abs/1710.09412 (2017).

[154] X. Zhang and Y. LeCun, *Universum prescription: Regularization using unlabeled data*, in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[155] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, *Pyramid scene parsing network*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 6230–6239.