

# Domain Knowledge Guided Representation Learning for Traffic Scenarios

**Jonas Wurst**

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology  
der Technischen Universität München zur Erlangung eines

**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**

genehmigten Dissertation.

Vorsitz: Prof. Dr. Reinhard Heckel

Prüfende der Dissertation: 1. Prof. Dr.-Ing. Wolfgang Utschick  
2. Prof. Dr.-Ing. Michael Botsch

Die Dissertation wurde am 15.09.2023 bei der Technische Universität München eingereicht  
und durch die TUM School of Computation, Information and Technology am 10.06.2024  
angenommen.



# Acknowledgement

I would like to express my deepest gratitude to my supervisor, Prof. Dr.-Ing. Botsch, for his unwavering support, trust, and the freedom he granted me. Without his guidance, I would not be the researcher I am today.

I am equally thankful to my supervisor, Prof. Dr.-Ing. Utschick, for his continuous support throughout this journey. Our discussions and his supervision were incredibly fruitful, teaching me to take a step back and reflect on my work from various perspectives.

I extend my thanks to Prof. Dr. Heckel for organizing and chairing my doctoral examination. Furthermore, I am grateful to my doctoral mentor, Dr. Keck, for his consistent input. Our frequent meetings kept me focused and inspired, providing me with new ideas.

I also want to thank the working groups of both Prof. Utschick and Prof. Botsch. I am especially grateful to all the PhD candidates and engineers I collaborated with during my research. A special thank you goes to Lakshman Balasubramanian, with whom I had countless deep technical discussions that often resulted in joint publications. Another special thanks to Friedrich Kruber, who not only supervised me during my early research days but also co-authored several publications with me. I also want to express my sincere appreciation to all my colleagues who made my doctoral journey enjoyable, filled with discussions, learnings, and friendships. Thank you all!

Lastly, and most importantly, I want to thank my friends and family. Foremost, I am grateful to my parents, Rüdiger and Brigitte Wurst for their lifelong support. Without you, I would not have achieved this milestone. Thank you. I am equally thankful to my wife and our children for their patience during long days, for uplifting my spirits after stressful times, and for sharing joyous moments with me.





# Dedication

I dedicate this work to my mother.  
Without her effort and faith in me, this work would not exist.

**I wish you could have seen me conclude this chapter.**



# Abstract

The success of Autonomous Vehicles (AVs) depends on the proof of their safety through validation, besides the development of the autonomous driving functionalities. For the validation of AVs, scenario-based testing is considered as one of the feasible approaches, where an AV is tested only in relevant traffic scenarios. One of the challenges that remain, is to identify such relevant traffic scenarios. In this work, relevant traffic scenarios are categorized into representative, unknown and critical traffic scenarios. Identifying representative traffic scenarios is realized by clustering, detecting unknown traffic scenarios is realized by novelty detection, and critical scenarios are identified with criticality detection. Each of those tasks is addressed in this work, to identify relevant traffic scenarios.

This work presents three novel representation learning methods to improve the clustering and novelty detection of traffic scenarios. Each of the three representation learning methods is guided with domain knowledge leading to a superior performance in terms of clustering and novelty detection. Two of the methods utilize a graph representation of the traffic scenarios. In order to find similar traffic scenarios, the database is analyzed with respect to similar graph representations. This way it is possible to embed domain knowledge specific objectives into the representation space formed by the representation learning methods. The third introduced approach proposes a domain knowledge guided transformation to generate two similar traffic scenarios given an origin traffic scenario. All three domain knowledge guided representation learning methods are evaluated thoroughly and are compared to alternative approaches. The three domain knowledge guided representation learning methods enable high clustering and novelty detection performance for traffic scenarios, and hence to find representative and unknown traffic scenarios.

To motivate the necessity of the scenario-based testing approach, a statistical analysis for random field tests of AVs with German highway accident statistics is provided. A novel novelty detection method is presented and it is applied to detect unknown infrastructures. Also, alternative novelty detection methods are applied to the problem of detecting novel infrastructures. As a fallback strategy, an approach is introduced, relying only on domain knowledge to detect representative and novel traffic scenarios based on their graph representation. In order to detect critical scenarios, the implementation of well-established methods in a computationally efficient framework is presented. Furthermore, the background for all used machine learning methods is explained in detail.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Outline . . . . .	3
1.2	Contribution . . . . .	4
1.3	Publications . . . . .	5
1.4	Repositories and Websites . . . . .	6
<b>2</b>	<b>Data-Driven Methods and Representation Learning</b>	<b>9</b>
2.1	Neural Networks . . . . .	10
2.2	Dimensionality Reduction . . . . .	26
2.3	Outlier Detection . . . . .	30
2.4	Clustering . . . . .	39
2.5	Representation Learning . . . . .	41
2.5.1	Reconstruction . . . . .	42
2.5.2	Metric Learning . . . . .	43
2.5.3	Self-Supervised Learning . . . . .	50
<b>3</b>	<b>Identifying Relevant Traffic Scenarios</b>	<b>59</b>
3.1	Statistical Motivation . . . . .	61
3.2	State of the Art . . . . .	69
3.3	Entropy-Based Novelty Detection of Road Infrastructure Images . . . . .	74
3.4	Alternative Outlier Detection Methods . . . . .	87
3.5	Domain Knowledge-Based Methods . . . . .	90
<b>4</b>	<b>Representation Learning for Identifying Relevant Traffic Scenarios</b>	<b>103</b>
4.1	Domain Knowledge guided Triplet Autoencoder . . . . .	104
4.1.1	Method . . . . .	105
4.1.2	Experiments . . . . .	113
4.1.3	Conclusion . . . . .	118
4.2	Domain Knowledge guided Quadruplet Autoencoder . . . . .	119
4.2.1	Method . . . . .	120
4.2.2	Experiments . . . . .	130

4.2.3	Conclusion	142
4.3	Domain Knowledge guided Augmentations for Self-Supervised Learning	143
4.3.1	Method	145
4.3.2	Experiments	149
4.3.3	Conclusion	156
<b>5</b>	<b>Conclusion and Outlook</b>	<b>159</b>
	<b>List of Acronyms</b>	<b>163</b>
	<b>References</b>	<b>165</b>
<b>A</b>	<b>Appendix</b>	<b>181</b>
A.1	Cross Entropy-based Contrastive Loss as Mutual Information Maximization	181
A.2	Statistical Theory	183
A.2.1	Statistical Moments	183
A.2.2	Probability Distributions	184

# Chapter 1

## Introduction

Autonomous driving is considered to be the next revolution of transportation. The potential improvements are many, such as increased driving comfort, optimized traffic flow and increased vehicle safety. Besides the required development of the autonomous driving functions, it is necessary to ensure the safety of *Autonomous Vehicles* (AVs). The confidence of the consumers and the public in AVs will be crucial for the further progress of autonomous driving [FBBA18].

Proofing that an AV performs significantly safer than the average human driver through a simple statistical testing approach, would require more than 8 billion kilometers to be driven<sup>1</sup> (see Section 3.1). As an alternative testing method, the scenario-based approach is commonly applied to demonstrate the safety of AVs [PEG]. In scenario-based testing the safety of an AV is demonstrated only on relevant scenarios. However, for this approach, it is crucial that the relevant scenarios are known. Accordingly, identifying relevant scenarios is required [JWKW18] for the scenario-based validation approach. Besides handcrafted

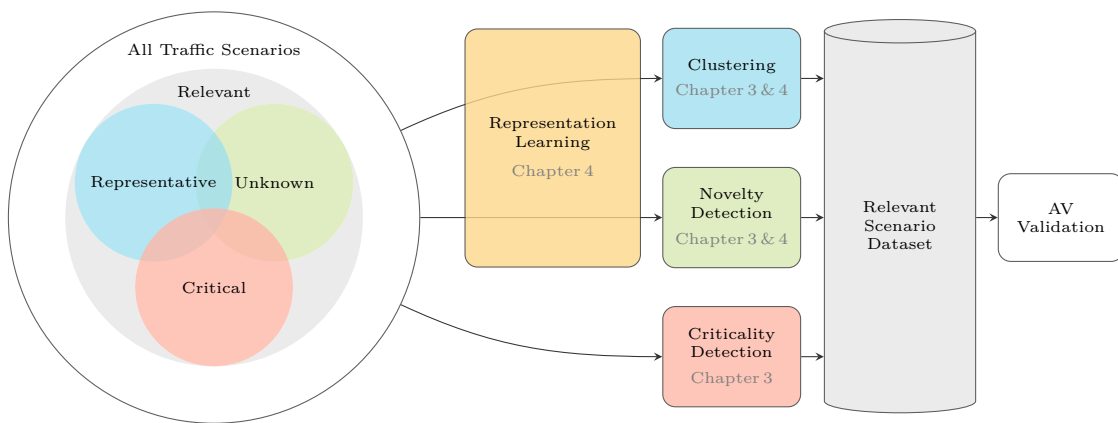



Figure 1.1: Methodological overview of this work and its embedding in the validation process of AVs. : Representation Learning component is only used in Chapter 4.

<sup>1</sup>Given a confidence of 95% and an improvement of 20% over the average human driver in terms of fatal accidents.

scenarios, real-world data should be taken into consideration, since the variety of situations and sensor inputs are hard to model in advance. In this work, relevant traffic scenarios are either

- representative traffic scenarios, which represent a group of traffic scenarios well,
- unknown traffic scenarios, which are potentially untested, or
- critical traffic scenarios.

This is also illustrated on the left side of Figure 1.1.

Finding representative traffic scenarios in real-world data can be achieved with clustering. Clusters of similar traffic scenarios are represented by one or some representative traffic scenarios from that group. Therefore, many recent works focus on clustering methods for traffic scenarios. Another important component for the scenario-based validation is to identify if an observed scenario is covered by the validation data. Detecting such unknown traffic scenarios is typically addressed through novelty or outlier detection. Besides unknown, also critical and potentially hazardous scenarios should be detected and tested thoroughly, since such scenarios should definitely be avoided by an AV. In summary, detecting representative, unknown, and critical scenarios from real-world data are key enablers for the scenario-based validation of AVs. Furthermore, the ISO 21448 [ISO21] requires a deployed AV to be monitored to enable continuous validation, which can be partially realized by detecting representative, unknown, and critical scenarios. This work addresses all three aspects by introducing novel methods and concepts to detect and process such traffic scenarios. In Figure 1.1, the three detection methods, one for each scenario type, is shown. They lead to the dataset consisting of the detected relevant traffic scenarios, which can be used for the validation of an AV.

The before mentioned tasks can be addressed with numerous of methods, specialized for the specific problem. Those methods can either be data-driven or mainly defined through domain knowledge. However, as analyzed in Chapter 3, most of the methods either fail to handle such complex data like a traffic scenario, or are not designed for the specific needs in this field of research. This work focuses on transforming a traffic scenario into a meaningful representation which can be used for clustering, novelty detection, and further analysis. As illustrated in Figure 1.1, transforming the traffic scenarios into such representations is realized with representation learning methods in this dissertation. The aim of representation learning is to train models which realize the transformation from the input space to the representation space. In this work the focus is on methods whose resulting representation spaces are suited for the tasks of clustering and novelty detection of traffic scenarios. In order for the representation learning methods to satisfy the representation space requirements, domain knowledge is used to guide the representation learning methods. This domain knowledge guidance is conceptually realized by defining



the similarity of traffic scenario. Three approaches for domain knowledge guided representation learning are presented. Two of the methods define a similarity for existing traffic scenarios. The third method enhances similarity by defining how to artificially create similar traffic scenarios.

As demonstrated by the results of this work, representation learning which relies on carefully designed domain knowledge outperforms approaches without domain knowledge. Moreover, the results show that the performance for tasks like novelty detection and clustering improves significantly when applied on the representations of such domain knowledge guided representation learning methods. The introduced representation learning methods contribute to the research on the validation of AVs, since the representations of traffic scenarios formed by these methods are better suited for clustering and novelty detection, the tasks required for the validation of AVs.

## 1.1 Outline

This work is split into three main parts. First, Chapter 2 summarizes the background from the field of machine learning with focus on representation learning. Second, existing approaches are discussed and novel concepts are introduced to identify relevant traffic scenarios in Chapter 3. Third, novel methods for transforming the traffic scenarios by means of representation learning guided by domain knowledge are presented in Chapter 4. In the following, the content of each chapter is briefly summarized.

Chapter 2 provides the necessary background from the field of machine learning with a focus on representation learning. Various neural network architectures, which are used in this work, are summarized in Section 2.1. This is followed by dimensionality reduction techniques which are used throughout the work (Section 2.2). The different applied outlier detection methods are explained in Section 2.3. Next, clustering methods are briefly discussed (Section 2.4). The methods from the field of reconstruction-based representation learning, metric learning, and self-supervised learning are covered in Section 2.5.

In Chapter 3, various aspects and methods for identifying relevant traffic scenarios are discussed. It is motivated in detail why scenario-based testing and hence the identification of relevant scenarios is necessary. Novel approaches for the identification of unknown traffic scenarios are introduced and analyzed. Besides, some recent alternative approaches are summarized. In detail Chapter 3 is structured as follows. A statistical motivation for the scenario-based validation approach is provided by showing the infeasibility of random field test to proof an AVs safety (Section 3.1). State-of-the-art approaches that target either clustering or novelty detection for traffic scenarios are summarized in Section 3.2. Since the infrastructure is one of the core elements of a traffic scenario, in Section 3.3 a novel method to detect unknown road infrastructures is introduced, which was presented in [WFBU20] as part of this work. The problem of detecting unknown road infrastruc-

tures is further explored by applying other outlier detection techniques in Section 3.4. In Section 3.5, model-based methods, which do not utilize machine learning, are presented. First, methods to identify critical scenario with various amount of available information are shown. Also, an automatic domain knowledge driven method to categorize traffic scenarios is introduced. In conclusion, the discussion in Chapter 3 highlights the need for the representation learning methods as introduced in Chapter 4.

The novel domain knowledge guided representation learning methods, the core contribution of this work, are introduced in Chapter 4. In total three approaches are presented and thoroughly analyzed with respect to traffic scenario clustering, novelty detection, and feature stability. The results show that applying the representation learning methods improves the performance for those tasks. In Chapter 4, each method is explained and analyzed separately, leading to the following overall structure. In Section 4.1, an autoencoder is combined with triplet learning [WBBU21]. This way, a latent space is formed which enables basic outlier detection methods to detect unknown road infrastructures. The triplet loss is used to embedded domain knowledge into the latent space. The method is extended to a quadruplet autoencoder (Section 4.2), such that besides the infrastructure also dynamic information is used [WBBU22]. Here, the domain knowledge is applied through the quadruplet loss. The representation space formed in this way is analyzed thoroughly with respect to various characteristics. An alternative approach for domain knowledge guided representation learning is presented in Section 4.3 [BWE<sup>+</sup>22]. There, domain knowledge guided augmentations for existing self-supervised learning approaches are presented.

In Chapter 5 the work is concluded and possible further research directions emerging from the presented methods are discussed.

## 1.2 Contribution

This work contributes to the area of AV validation, with methods that enable the identification of relevant traffic scenarios. The contributions of this work can be summarized as follows.

**Section 3.3** A novel outlier detection method is presented which is applied to road infrastructure images [WFBU20].

**Section 3.4** Alternative methods for the detection of unknown road infrastructures are explored.

**Section 3.5** Methods based only on domain knowledge to identify critical traffic scenarios and to automatically categorize traffic scenarios are introduced.

**Section 4.1** A domain knowledge guided triplet autoencoder is introduced which projects

road infrastructures into a representation that is better suited for novelty detection [WBBU21].

**Section 4.2** The quadruplet autoencoder guided by domain knowledge is presented [WBBU22]. It extends the triplet autoencoder method so that dynamic information is used in addition. Its traffic scenarios representations can be used for novelty detection and clustering.

**Section 4.3** Domain knowledge guided augmentations to be used in self-supervised learning for traffic scenarios are introduced [BWE+22].

## 1.3 Publications

**Core Publications** As part of this dissertation, some methods have been published. The core publications for this work are:

[WFBU20] **Jonas Wurst**, Alberto Flores Fernández, Michael Botsch and Wolfgang Utschick. *An Entropy Based Outlier Score and its Application to Novelty Detection for Road Infrastructure Images*. 2020 IEEE Intelligent Vehicles Symposium (IV), 2020.

[WBBU21] **Jonas Wurst**, Lakshman Balasubramanian, Michael Botsch and Wolfgang Utschick. *Novelty Detection and Analysis of Traffic Scenario Infrastructures in the Latent Space of a Vision Transformer-Based Triplet Autoencoder*. 2021 IEEE Intelligent Vehicles Symposium (IV), 2021.

[WBBU22] **Jonas Wurst**, Lakshman Balasubramanian, Michael Botsch and Wolfgang Utschick. *Expert-LaSTS: Expert-Knowledge Guided Latent Space for Traffic Scenarios*. 2022 IEEE Intelligent Vehicles Symposium (IV), 2022.

[BWE+22] Lakshman Balasubramanian\*, **Jonas Wurst\***, Robin Egolf, Michael Botsch, Wolfgang Utschick and Ke Deng. *ExAgt: Expert-guided Augmentation for Representation Learning of Traffic Scenarios*. 2022 25th International Conference on Intelligent Transportation Systems (ITSC), 2022. \*: equal contribution

**Other Publications** Other publications of the author in the field of traffic scenario clustering and analysis are:

[KWB18] Friedrich Kruber, **Jonas Wurst** and Michael Botsch. *An Unsupervised Random Forest Clustering Technique for Automatic Traffic Scenario Categorization*. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018.

- [**KWM<sup>+</sup>19**] Friedrich Kruber, **Jonas Wurst**, Eduardo Sánchez Morales, Samarjit Chakraborty and Michael Botsch. *Unsupervised and Supervised Learning with the Random Forest Algorithm for Traffic Scenario Clustering and Classification*. 2019 IEEE Intelligent Vehicles Symposium (IV), 2019.
- [**BKBD21**] Lakshman Balasubramanian, **Jonas Wurst**, Michael Botsch and Ke Deng. *Traffic Scenario Clustering by Iterative Optimisation of Self-Supervised Networks Using a Random Forest Activation Pattern Similarity*. 2021 IEEE Intelligent Vehicles Symposium (IV), 2021.
- [**FFWSM<sup>+</sup>22**] Alberto Flores Fernández, **Jonas Wurst**, Eduardo Sánchez Morales, Michael Botsch, Christian Facchi and Andrés García Higuera. *Probabilistic Traffic Motion Labeling for Multi-Modal Vehicle Route Prediction*. MDPI Sensors Journal Volume 22 Issue 12, 2022.
- [**KME<sup>+</sup>22**] Friedrich Kruber, Eduardo Sánchez Morales, Robin Egolf, **Jonas Wurst**, Samarjit Chakraborty and Michael Botsch. *Micro- and Macroscopic Road Traffic Analysis using Drone Image Data*. Leibniz Transactions on Embedded Systems, Volume 8, Issue 1, 2022.
- [**KWBC23**] Friedrich Kruber, **Jonas Wurst**, Michael Botsch and Samarjit Chakraborty. *Unsupervised Random Forest Learning for Traffic Scenario Categorization*. Chapter in: Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems, 2023
- [**BWE<sup>+</sup>23**] Lakshman Balasubramanian\*, **Jonas Wurst\***, Robin Egolf, Michael Botsch, Wolfgang Utschick and Ke Deng. *SceneDiffusion: Conditioned Latent Diffusion Models for Traffic Scene Prediction*. 2023 26th International Conference on Intelligent Transportation Systems (ITSC), 2023.

### 1.4 Repositories and Websites

For some methods presented in this dissertation, implementations have been made publicly available. Moreover, some results are additionally presented in publicly accessible websites. The published repositories are:

**openDRIVE-Matlab** Tool to parse and plot OpenDRIVE format in MATLAB. <https://github.com/JWTHI/openDRIVE-Matlab>

**ULEF** Implementation for the ULEF outlier detection score as presented in Section 3.3. <https://github.com/JWTHI/ULEF>

**ViTAL-SCENE** Implementation of the triplet autoencoder as presented in Section 4.1. <https://github.com/JWTHI/ViTAL-SCENE>

**Expert-LaSTS** Implementation of the quadruplet autoencoder as presented in the method Expert-LaSTS in Section 4.2. <https://github.com/JWTHI/Expert-LaSTS>

and the published websites are:

**SCENATLAS** Interactive visualization of the representation space when applying the triplet autoencoder as presented in Section 4.1 and additional dimensionality reduction. <https://jwthi.github.io/SCENATLAS/>

**Expert-LaSTS** Interactive visualization of the representation space when applying the method Expert-LaSTS (c. f. Section 4.2) and additional dimensionality reduction. <https://jwthi.github.io/Expert-LaSTS/>



## Chapter 2

# Data-Driven Methods and Representation Learning

The methods proposed in this work require background knowledge about various machine learning concepts and methods. In this chapter, those concepts and methods as well as the necessary background is summarized. Figure 2.1 recaps the overall framework of this work and highlights some components and the related sections in this chapter. As one important element of this work, applied in outlier detection and representation learning, various neural network architectures and basic components are explained. Specifically, the concepts of the *MultiLayer Perceptron* (MLP), *Convolutional Neural Network* (CNN), *deep Residual Network* (ResNet), *Long Short-Term Memory* (LSTM), transformers, and *Vision Transformer* (ViT) are summarized. The dimensionality reduction methods *Uniform Manifold Approximation and Projection* (UMAP), *t-Stochastic Neighbor Embedding* (t-SNE), and *Principal Component Analysis* (PCA) are discussed. Neighborhood-based

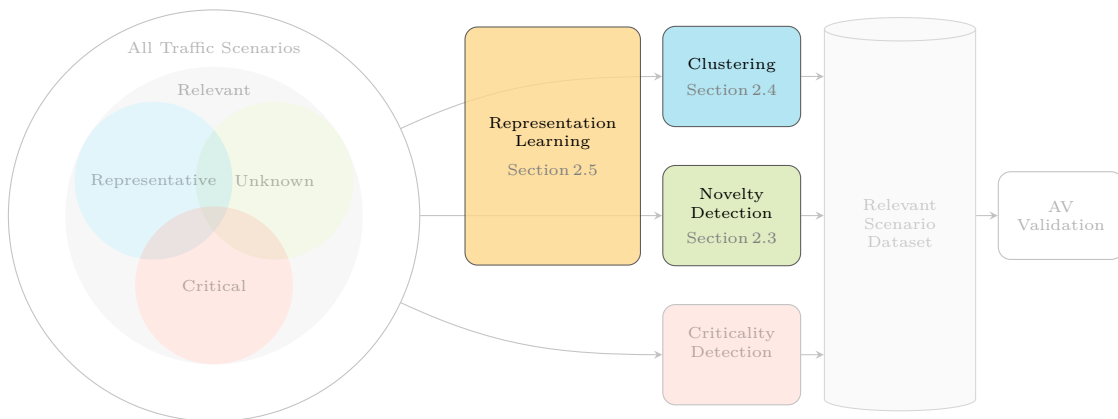


Figure 2.1: Methodological overview of this work and its embedding in the validation process of AVs. The background for the highlighted components is provided in the corresponding sections.

and reconstruction-based outlier detection methods, as well as the isolation forest and the one-class support vector machine are described. Clustering with *Hierarchical Clustering* (HC) and  $K$ -means is presented. Methods and architectures to realize representation learning are summarized. Three branches of representation learning are considered. First, reconstruction-based methods. Second, methods related to metric learning, such as contrastive, triplet, and cross entropy-based contrastive learning. Third, from the field of self-supervised learning the methods Barlow Twins, *Swapping Assignments between Views* (SwAV), and *Variance-Invariance-Covariance Regularization* (VICReg) are explained.

Machine learning methods, aim to identify statistical relations and patterns in data. Therefore, machine learning is often termed as pattern recognition or statistical learning as well. To find those relations and patterns in a data set  $\mathcal{D}$ , optimization techniques are applied. A set  $\mathcal{D}$  consists of  $M$  data points  $\mathbf{x}_m \in \mathbb{R}^N$ . Typically, machine learning methods are grouped with respect to the task they aim to solve, e.g., classification, regression, clustering, representation learning. Furthermore, it is distinguished between supervised and unsupervised tasks, meaning either a ground truth for the respective task is available or not. Various models can be used to fulfill the tasks, while the objective of a task is usually defined through the loss  $\mathcal{L}$ , which is the learning objective.

### 2.1 Neural Networks

Artificial neural networks play a key role in the success of nowadays machine learning methods. Typically, artificial neural networks consist of multiple layers, thereby realizing deep architectures – coined the term deep learning. In this section, various network architectures are discussed. The different architectures are used to realize trainable mapping  $f : \mathbf{x} \mapsto \mathbf{y}$ , such that learning objective can be fulfilled. Each architecture is specifically designed for a certain type of input data, as vectors, images, and sequences, corresponding architectures are shown in the following. Therefore, architectures for inputs as: vectors, images, and sequences are shown in the following.

The MLP is explained first, which is basic building block in more complex architectures. Networks suited for image-like inputs, such as vanilla CNNs, ResNets, and their layers are discussed next. In order to process sequence data, architectures like LSTMs and transformers are commonly used, which are covered as well. The ViT, a special variant of the transformer suited for images is described last.

Artificial neural networks are trainable mathematical models, which can be used to solve various training objectives. Those architectures typically consist of multiple artificial neurons, which can be connected in multiple ways. An artificial neuron generates an output



$y$  based on the weighted sum of the  $N$  inputs  $\mathbf{x} = [x_i]_1^N$ , as

$$y = \phi\left(\sum_{i=1}^N w_i x_i + b\right), \quad (2.1)$$

$$= \phi(\mathbf{w}^T \mathbf{x} + b), \quad (2.2)$$

with  $w_i$  the weight for the  $i$ -th input, the bias  $b$ , and the activation function  $\phi$  of the neuron. During training of neural networks, the weights  $\mathbf{w}$  and bias values  $b$  are optimized to realize the learning objective. Activation functions typically used are non-linear e. g., the sigmoid function, the hyperbolic tangent function, and the *Rectified Linear Unit* (ReLU) function.

### 2.1.1 Multi-Layer Perceptron

The MLP is a widely used artificial neural network topology, in which multiple layers of artificial neurons are connected in a feed-forward way. This architecture is used for various tasks and is utilized within other architectures as well. The MLP is suited for inputs represented as vectors.

Let one layer in the MLP be a group of  $N_l$  artificial neurons, here the neurons within a layer are not connected. A MLP consists of  $L + 2$  layers, with one input layer and one output layer. Between the input and the output layer,  $L$  so-called hidden layers are used. All neurons of one layer are connected with all neurons of the consecutive layer – fully-connected. The output  $\mathbf{y}^{(l)} \in \mathbb{R}^{N_l}$  of the  $N_l$  neurons in the  $l$ -th layer is defined as

$$\mathbf{y}^{(l)} = \phi(\mathbf{W}^{(l)} \mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}), \quad (2.3)$$

where the input is the output of the previous layer  $\mathbf{y}^{(l-1)} \in \mathbb{R}^{N_{l-1}}$ . The matrix  $\mathbf{W}^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$  contains the weights for each connection from layer  $l - 1$  to layer  $l$ . The bias values for the  $N_l$  neurons are represented by  $\mathbf{b}^{(l)} \in \mathbb{R}^{N_l}$ . The outputs of the neurons in the last layer can be determined by applying Equation (2.3) for each layer successively. In Figure 2.2, the architecture of a MLP is depicted. The circles symbolize the neurons and the edges the weighted connections between the neurons. The bias values are not separately shown, they are assumed to be included in the neurons. During the training process, the weight matrices  $\mathbf{W}$  and the bias values  $\mathbf{b}$  for each layer are learned. The number of neurons per layer, the number of layers as well as the used activation functions are hyperparameters.

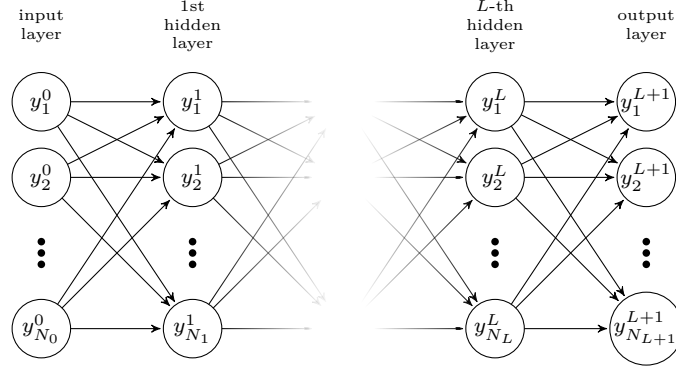


Figure 2.2: Multilayer Perceptron with  $L$  hidden layers and  $N_{\dots}$  neurons per layer.

### 2.1.2 Convolutional Neural Networks

The training of neural networks on tasks that include image data are typically solved with other architectures than the MLP<sup>1</sup>. For this purpose, the CNNs are introduced [LBD<sup>+</sup>90], which are well suited for processing image data. Due to the convolution operation, CNNs are able to encode visual features and achieve remarkable results with fewer parameters when compared to a MLP. Typically, a CNN consists of multiple layers, where this stacking enables the learning of complex image features. The building blocks commonly used in a CNN are the convolution, activation function, pooling operation, and residual connection. In this section, first the building blocks are discussed, then some widely used architectures are summarized.

#### 2.1.2.1 Layer Types

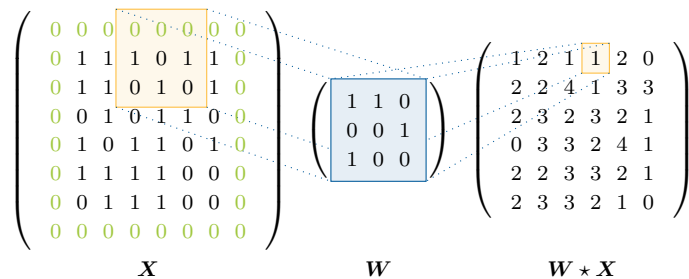
**Convolution** The core element of a CNN is the convolution operation. Within a convolution layer, the  $l$ -th layer input  $\mathbf{X}^{(l)} \in \mathbb{R}^{N_H^{(l)} \times N_W^{(l)} \times N_C^{(l)}}$ , also called input feature maps, is convolved with the  $N_C^{(l+1)}$  weight matrices  $\mathbf{W}^{(l)} \in \mathbb{R}^{N_H^{(l)} \times N_W^{(l)} \times N_C^{(l)} \times N_C^{(l+1)}}$ , adding the bias  $\mathbf{b}^{(l)} \in \mathbb{R}^{N_C^{(l+1)}}$  leads to the output feature maps  $\mathbf{Y}^{(l)} \in \mathbb{R}^{N_H^{(l+1)} \times N_W^{(l+1)} \times N_C^{(l+1)}}$  as

$$\mathbf{Y}_{::,m}^{(l)} = b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \mathbf{W}_{::,c,m}^{(l)} * \mathbf{X}_{::,c}^{(l)}, \quad (2.4)$$

$$\mathbf{Y}_{i,j,m}^{(l)} = b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \sum_h \sum_w \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{i-h,j-w,m}^{(l)}, \quad (2.5)$$

---

<sup>1</sup>Using a MLP for image input would require a huge amount of parameters what is omitted by the CNNs due to the shared parameter concept. Moreover, CNNs in contrast to MLPs are specifically designed to consider spatial relations of the input. Nevertheless, CNNs can be interpreted as a special case of MLPs using weight sharing and the spatial concept.

Figure 2.3: Convolution with zero-padding.<sup>3</sup>

with  $m = 1, \dots, N_C^{(l+1)}$  creating  $N_C^{(l+1)}$  channels in the output. However, in deep learning the convolution operation is typically realized through a cross correlation<sup>2</sup>. Therefore, the operation is defined as,

$$\mathbf{Y}_{::,m}^{(l)} = b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \mathbf{W}_{::,c,m}^{(l)} \star \mathbf{X}_{::,c}^{(l)}, \quad (2.6)$$

$$\mathbf{Y}_{i,j,m}^{(l)} = b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \sum_{h=1}^{N_{\text{HW}}^{(l)}} \sum_{w=1}^{N_{\text{WW}}^{(l)}} \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{i-1+h,j-1+w,m}^{(l)}. \quad (2.7)$$

The only difference between the cross correlation and the convolution is that the weight matrices are not flipped along both axes. In the following, the term convolution is used with respect to CNNs, even though it is actually a cross correlation. The Equation (2.7), describes the operation performed by one convolution layer. The number of weight matrices  $N_C^{(l+1)}$ , their width  $N_{\text{WW}}^{(l)}$ , and height  $N_{\text{HW}}^{(l)}$  are the hyperparameters of this layer. The weight matrices  $\mathbf{W}^{(l)}$  and the bias vector  $\mathbf{b}^{(l)}$  for each layer are learned during the training process. The weight matrices are also known as so-called shared weights, since in contrast to MLPs, the same weights are used for multiple parts of the input.

The convolution operation reduces the height  $N_{\text{H}}$  and width  $N_{\text{W}}$  of the input feature maps. This can be prevented by applying padding at the edges of the feature maps. The height and width of the feature maps are increased by attaching columns and rows at the outer border of the input,  $N_{\text{H}} + N_{\text{H}_0}$  and  $N_{\text{W}} + N_{\text{W}_0}$ , called as padding. Typically, those columns and rows are filled with zeros – zero-padding. The  $N_{\text{H}_0}$ ,  $N_{\text{W}_0}$ , and location of padded rows and columns are hyperparameters. A common strategy is to select the values such that the height and width is maintained between the input and the output feature maps.

In Figure 2.3, an example of a convolution between a weight matrix  $\mathbf{W}$  and a feature map  $\mathbf{X}$  is shown, where  $N_C = 1$ . In this example, zero-padding is applied such that the

<sup>2</sup>For example in the framework PyTorch, the convolution is realized through a cross correlation <https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html>

<sup>3</sup>Inspired from <https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz>

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 \\ 2 & 2 & 2 \\ 2 & 3 & 2 \end{pmatrix}$$

$\mathbf{X}$                        $\mathbf{W}$                        $\mathbf{W} \star \mathbf{X}$

Figure 2.4: Convolution with zero-padding and stride of 2.

resulting feature map has the same height and width as the input.

On the contrary, reducing the size of the output can be achieved by changing the hyperparameter stride  $S$  of the convolution operation,

$$\mathbf{Y}_{i,j,m}^{(l)} = b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \sum_{h=1}^{N_{H_W}^{(l)}} \sum_{w=1}^{N_{W_W}^{(l)}} \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{S(i-1)+h, S(j-1)+w, m}^{(l)}. \quad (2.8)$$

When combined with the zero-padding, the size of the output feature maps can be adjusted. In Figure 2.4, a convolution with  $N_C = 1$ , zero-padding and a stride of 2 is shown, leading to an output with half width and half height of the input.

When setting the weight matrix height  $N_{H_W} = 1$  and width  $N_{W_W} = 1$  and having an input  $\mathbf{X}^{(l)} \in \mathbb{R}^{1 \times 1 \times N_C^{(l)}}$ , the convolution layer becomes similar to a layer in a MLP, except the activation function.

The convolution operation can be formulated as a matrix multiplication. For this, the input feature maps  $\mathbf{X}^{(l)}$  is vectorized to  $\mathbf{x}^{(l)} \in \mathbb{R}^{N_W^{(l)} N_H^{(l)} N_C^{(l)}}$  in row-major order<sup>4</sup>. The weight matrices are converted into the convolution operation realizing matrices  $\mathfrak{W} \in \mathbb{R}^{(N_W^{(l+1)} N_H^{(l+1)}) \times (N_W^{(l)} N_H^{(l)} N_C^{(l)}) \times N_C^{(l+1)}}$ , such that

$$\text{reshape}(\mathfrak{W}_{:, :, m}^{(l)} \mathbf{x}^{(l)}) = \sum_{c=1}^{N_C^{(l)}} \sum_{h=1}^{N_{H_W}^{(l)}} \sum_{w=1}^{N_{W_W}^{(l)}} \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{S(i-1)+h, S(j-1)+w, m}^{(l)} \quad (2.9)$$

holds. For example, the  $\mathfrak{W}$  for a convolution without padding and  $N_{H_W} = 2$ ,  $N_{W_W} = 2$ ,  $N_H^{(l)} = 3$ ,  $N_W^{(l)} = 3$ ,  $N_C^{(l)} = 1$ ,  $N_C^{(l+1)} = 1$  and  $S = 1$  is defined as,

$$\mathfrak{W} = \begin{bmatrix} W_{1,1} & W_{1,2} & 0 & W_{2,1} & W_{2,2} & 0 & 0 & 0 & 0 \\ 0 & W_{1,1} & W_{1,2} & 0 & W_{2,1} & W_{2,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_{1,1} & W_{1,2} & 0 & W_{2,1} & W_{2,2} & 0 \\ 0 & 0 & 0 & 0 & W_{1,1} & W_{1,2} & 0 & W_{2,1} & W_{2,2} \end{bmatrix}. \quad (2.10)$$

<sup>4</sup>Row-major order: attaching each row of the first feature map. Then attaching each row of the first feature map and so forth.

Reshaping the resulting vector of  $\mathfrak{W}\mathbf{x}$  into a matrix with shape  $(N_{\text{H}}^{(l+1)} = 4 \times N_{\text{W}}^{(l)} = 3)$ , yields the same result as the normal convolution operation.

**Transpose Convolution** In some architectures, e. g., convolutional autoencoders, an up-sampling layer is required, which increases the size of the feature maps. This is usually realized through the so-called transpose convolution operation. As shown before, the convolution operation can be expressed through  $\mathfrak{W}\mathbf{x}$ . Considering the case where an up-sampling should be performed on  $\mathbf{X}$  such that it increases the size of the feature map to the size of some input feature map  $\bar{\mathbf{X}}$ . This can be achieved by the transposed convolution operation,

$$\mathbf{Y}_{::,m}^{(l)} = b_m^{(l)} + \text{reshape}\left\{\mathfrak{W}_{::,m}^{(l)\text{T}}\mathbf{x}^{(l)}\right\} \quad (2.11)$$

where  $\mathbf{Y}$  has the same shape as  $\bar{\mathbf{X}}$ . Therefore, the transposed convolution can be seen as a learnable up-sampling process, reverting the size of a hypothetical convolution. Typically, the  $\mathbf{W}$  underlying the  $\mathfrak{W}^{\text{T}}$  per transposed convolution layer is learned independently of the convolution layers. In deep learning frameworks (e. g., `PyTorch`), the settings of the stride and padding of the transposed convolution have to be given as if parameterizing the convolution operation to be transposed. For example, if the up-sampling corresponds to a hypothetical convolution of  $3 \times 3$  weights, stride of one, and no padding, this is the setting to be used in the transposed layer definition. Intuitive illustrations of the transposed convolution can be found in [DV16].

The transposed convolution can also be formulated as [GBC16]

$$\mathbf{Y}_{i,j,m}^{(l)} = b_m^{(l)} + \sum_c \sum_{\substack{k,h \\ \text{s.t.} \\ S(k-1)+h=i}} \sum_{\substack{n,w \\ \text{s.t.} \\ S(n-1)+w=j}} \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{S(i-1)+h,S(j-1)+w,m}^{(l)} \quad (2.12)$$

The transposed convolution is also performed when back propagating the error through a normal convolutional layer while training.

**Activation Function** Another frequently applied operation in CNNs are activation functions. This is realized by applying the activation function to each element of the corresponding feature maps. Let  $\mathbf{X}$  be the input to the activation function, then the output can be formulated as

$$\mathbf{Y} = \phi(\mathbf{X}). \quad (2.13)$$

As in the MLP, an activation function is typically applied after each convolution layer. The definition of the convolution layer Equation (2.8) combined with the activation func-

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} \frac{4}{9} & \frac{4}{9} & \frac{4}{9} \\ \frac{4}{9} & \frac{6}{9} & \frac{6}{9} \\ \frac{4}{9} & \frac{8}{9} & \frac{4}{9} \end{pmatrix}$$

$\mathbf{X}$   $\mathbf{Y}$

Figure 2.5: Average-pooling with zero-padding and stride of 2.

tion is given by

$$\mathbf{Y}_{i,j,m}^{(l)} = \phi \left( b_m^{(l)} + \sum_{c=1}^{N_C^{(l)}} \sum_{h=1}^{N_{HW}^{(l)}} \sum_{w=1}^{N_{WW}^{(l)}} \mathbf{W}_{h,w,c,m}^{(l)} \cdot \mathbf{X}_{S(i-1)+h, S(j-1)+w, m}^{(l)} \right). \quad (2.14)$$

**Pooling** In contrast to changing the stride, an alternative approach to reduce the size of the feature maps is to use pooling. As in the convolution, a region of interest with height  $N_{HP}$  and width  $N_{WP}$  is slid with stride  $S_P$  over the feature map as,

$$\mathbf{Y}_{i,j,m}^{(l)} = \underset{\substack{h=1, \dots, N_{HP} \\ w=1, \dots, N_{WP}}}{\text{pooling}} \left( \mathbf{X}_{S_P(i-1)+h, S_P(j-1)+w, m}^{(l)} \right). \quad (2.15)$$

The function  $\text{pooling}(\dots)$ , takes the values of the region specified by  $S_P(i-1)+h, S_P(j-1)+w$  with  $h = 1, \dots, N_{HP}$ ,  $w = 1, \dots, N_{WP}$  and returns a single value. The pooling can be realized by various functions. Typically, max-pooling or average-pooling is used, returning the maximum and the average value, respectively. Figure 2.5 depicts an example of an average-pooling with a stride of 2 on a padded feature map.

**Fully-Connected** The *Fully-Connected* FC layer as used in MLPs is also utilized in CNNs. If the input is a feature map, it is vectorized, such that it suits the definition. Hence, the output of the FC layer is also a vector. As in the MLP, the weights and biases are the learnable parameters for this layer type.

**Softmax** For classification tasks, the output of CNNs are typically converted into one-hot-encoded vectors. For a 10 class problem, this would lead to a vector of dimensionality 10, where the vector is zero except at the position of the actual class. This encoding is achieved by the softmax layer, which is attached at the end of classification architectures.

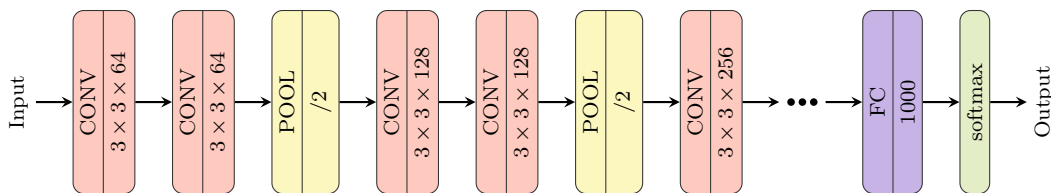


Figure 2.6: Vanilla CNN: parts of VGG-19. CONV: feature map size maintaining convolution followed by activation (dimensions specify the kernel size); POOL: pooling with /2 to reduce size by the half; FC: fully connected layer with output dimension as specified.

The softmax is realized as,

$$\mathbf{y} = \text{softmax}(\mathbf{x}), \quad (2.16)$$

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N_y} e^{x_j}} \quad (2.17)$$

where  $N_y$  is the number of classes and the size of the input vector  $\mathbf{x}$ .

### 2.1.2.2 Vanilla CNN

The vanilla version of a CNN can have arbitrary architectures, which consists only of the before described layers, connected in a successive feed-forward manner. The VGG-19 [SZ14] is such an architecture and is designed for image classification. In Figure 2.6, the parts of the VGG-19 architecture are used to show the typical structure in vanilla CNNs. Various layers are successively stacked, yielding a deep architecture. The convolution blocks shown in the figure already include the activation functions and are set up to maintain the size of the feature maps. The pooling block is parameterized to reduce the size of the feature map by half, as indicated by the /2. The end of the network is formed by FC layers and a final softmax layer.

### 2.1.2.3 Residual CNN

In contrast to the vanilla CNN, in the *residual CNN* (ResNet) the layers are not only connected in a successive manner. This is realized by "shortcut connections", which are connections between two layers, but skipping one or more successive layer. The intuition behind the shortcut connections is the so-called residual learning. It is motivated from the degradation problem which occurs for deep architecture without residual learning. One architecture using such a residual setup is the ResNet-18 [HZRS16]. Parts of a ResNet-18 are depicted in Figure 2.7 in order to illustrate the shortcut connections. As before, in the CONV blocks, the convolution is always followed by an activation and setup such that the size of feature maps stays the same. In the CONV+ block, the second input is added to the convolution result before fed to the activation. The CONV /2 block is a convolution with activation but setup to halve the feature map size. The solid shortcut connection is

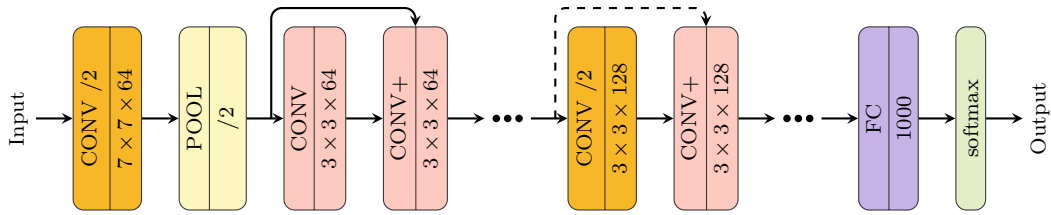


Figure 2.7: Residual CNN: parts of ResNet-18. CONV: feature map size maintaining convolution followed by activation function (dimensions specify the kernel size); CONV /2: feature map size halving convolution followed by activation function; CONV+: feature map size maintaining convolution followed by summation of the shortcut connection followed by activation function; POOL: pooling with /2 to reduce size by the half; FC: fully connected layer with output dimension as specified.

realized as an identity mapping. The dashed shortcut connection increases the depth of the feature map through a linear projection and reduces the width and height by stride of 2.

### 2.1.3 LSTM

The LSTM [HS97] belongs to the group of *Recurrent Neural Networks* (RNNs), which are specifically suited for sequential data. One characteristic of RNNs are the feedback connections, which feed back outputs of the model to the model itself [GBC16]. In contrast to CNNs, where parameter sharing is realized spatially, in RNNs, the parameter sharing is realized across the sequence, and hence temporal. The LSTM is able to learn long-term dependencies.

Let  $(\mathbf{x})_{N_S} = (\mathbf{x}_1, \dots, \mathbf{x}_{N_S})$  with  $\mathbf{x}_t \in \mathbb{R}^N$  be an input sequence of length  $N_S$ . The LSTM iterates over the sequence, producing the hidden states  $(\mathbf{h})_{N_S} = (\mathbf{h}_1, \dots, \mathbf{h}_{N_S})$  for each element in the sequence. The hidden state  $\mathbf{h}_t \in \mathbb{R}^{N_h}$  for the  $t$ -th element in  $(\mathbf{x})_{N_S}$  is



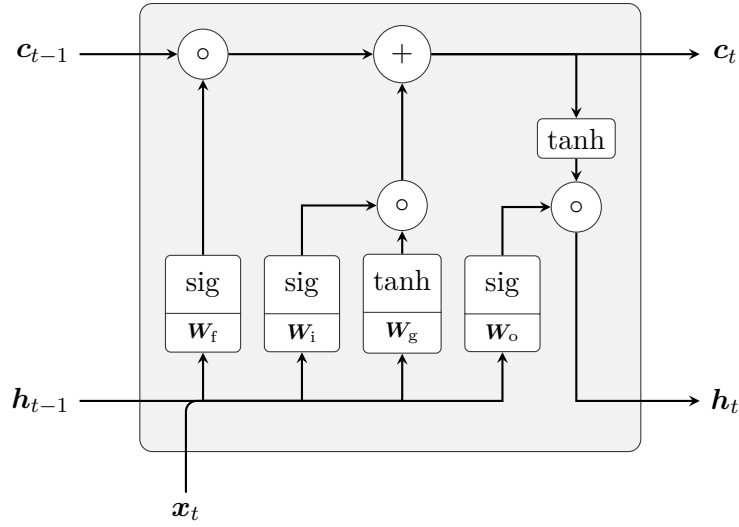


Figure 2.8: LSTM cell.  $\mathbf{W}_{\dots}$ : learnable weight matrix;  $\mathbf{c}_{\dots}$ : cell state;  $\mathbf{h}_{\dots}$ : hidden state;  $\mathbf{x}_{\dots}$ : input;  $\text{---}$ : concatenation;  $\circ$ : Hadamard product.<sup>5</sup>

calculated as,

$$\mathbf{h}_t = o_t \circ \tanh(\mathbf{c}_t), \text{ with} \quad (2.18)$$

$$\mathbf{c}_t = f_t \circ \mathbf{c}_{t-1} + i_t \circ g_t, \quad (2.19)$$

$$o_t = \text{sig} \left( \mathbf{W}_o \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_o \right), \quad (2.20)$$

$$f_t = \text{sig} \left( \mathbf{W}_f \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_f \right), \quad (2.21)$$

$$i_t = \text{sig} \left( \mathbf{W}_i \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_i \right) \text{ and} \quad (2.22)$$

$$g_t = \tanh \left( \mathbf{W}_g \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_g \right). \quad (2.23)$$

$$(2.24)$$

With sig the sigmoid function, the bias vectors  $\mathbf{b}_{\dots} \in \mathbb{R}^{N_h}$ , and the weight matrices  $\mathbf{W}_{\dots} \in \mathbb{R}^{N_h \times (N_h + N)}$ , such that the classical LSTM formulation  $\text{sig} \left( \mathbf{W}_{\dots} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{\dots} \right) = \text{sig}(\mathbf{W}_{h\dots} \mathbf{h}_{t-1} + \mathbf{W}_{x\dots} \mathbf{x}_t + \mathbf{b}_{\dots})$  is achieved, respectively for  $f_t$ ,  $o_t$ ,  $i_t$ , and  $g_t$ . The overall calculation flow in the LSTM is also shown in Figure 2.8.

The LSTM consists of the so-called cell state  $\mathbf{c}_t \in \mathbb{R}^{N_h}$ , which is updated during the

<sup>5</sup>Inspired from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

above described process. Hence, the cell state is recursively fed to the network, where  $\mathbf{c}_t$  is the current cell state and  $\mathbf{c}_{t-1}$  the cell state for the previous sequence element. To update the cell state, first the forget gate  $f_t$  controls what and how much to forget from the previous cell state  $\mathbf{c}_{t-1}$  by  $f_t \circ \mathbf{c}_{t-1}$  (multiply with low values for forgetting). Then, the cell state is updated, given the current input  $\mathbf{x}_t$  and the previous hidden state  $\mathbf{h}_{t-1}$  with  $i_t \circ g_t$ , where  $i_t$  is the input gate, selecting what and how much of the values to use for the update. The update value is determined by  $g_t$ . Therefore, updating the cell state is realized with  $f_t \circ \mathbf{c}_{t-1} + i_t \circ g_t$ . The current hidden state  $\mathbf{h}_t$  is based on the current cell state through  $\tanh(\mathbf{c}_t)$ , controlled by the output gate  $o_t$  which determines what to output from the projected cell state.

The weight matrices  $\mathbf{W}_{\dots}$  and the bias vectors  $\mathbf{b}_{\dots}$  are learned during the training process. The initial hidden state  $\mathbf{h}_0$  and initial cell state  $\mathbf{c}_0$  are typically set to  $\mathbf{0}$ .

### 2.1.4 Transformer

A more recent architecture suited for sequences is the transformer network [VSP<sup>+</sup>17]. In the base variant, it is designed for sequence to sequence modeling, such as translation tasks. There are a lot of successful applications of transformer variants for example: autoregressive language models (GPT-3 [BMR<sup>+</sup>20]), natural language processing pre-training (BERT [DCLT19]), object detection in images (DETR [CMS<sup>+</sup>20]), image classification (ViT [DBK<sup>+</sup>21]) and image generation from text (DALL-E [RPG<sup>+</sup>21])

The base transformer was introduced as an encoder-decoder structure. Here, the encoder processes the input data first, then the decoder is generating the output given another input and the encoder output. Transformers can handle sequences of varying length, like LSTMs. Different to LSTMs, the sequence elements are processed in one step, where the processing of each element can use information of any other element. In some cases it might be necessary to allow the transformer to only access information from the previous elements, this is realized through causal masking. One of the key components of transformers is the so-called attention mechanism, which provides the possibility to combine the information of any element. In the following, the attention mechanism, the layer-norm, and the complete architecture of the transformer are explained.

#### 2.1.4.1 Attention

The attention mechanism provides the possibility to access all the information of the different sequence elements, depended on their actual states. For this, the three inputs: value  $\mathbf{V}$ , key  $\mathbf{K}$ , and query  $\mathbf{Q}$  are generated. The result of the attention mechanism is the weighted sum of the values  $\mathbf{V}$ . The weights – how much to attend to what – are

determined based on the queries and keys. The attention mechanism is defined as,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{N_k}}\right)\mathbf{V}, \quad (2.25)$$

with

$$\begin{aligned} \mathbf{Q} &= \mathbf{X}_Q \mathbf{W}_Q, \text{ with } \mathbf{Q} \in \mathbb{R}^{N_{S_Q} \times N_K}, \mathbf{X}_Q \in \mathbb{R}^{N_{S_Q} \times N}, \mathbf{W}_Q \in \mathbb{R}^{N \times N_K}, \\ \mathbf{K} &= \mathbf{X}_K \mathbf{W}_K, \text{ with } \mathbf{K} \in \mathbb{R}^{N_{S_K} \times N_K}, \mathbf{X}_K \in \mathbb{R}^{N_{S_K} \times N}, \mathbf{W}_K \in \mathbb{R}^{N \times N_K}, \\ \mathbf{V} &= \mathbf{X}_V \mathbf{W}_V, \text{ with } \mathbf{V} \in \mathbb{R}^{N_{S_K} \times N_V}, \mathbf{X}_V \in \mathbb{R}^{N_{S_K} \times N}, \mathbf{W}_V \in \mathbb{R}^{N \times N_V} \text{ and} \\ \mathbf{Z} &= \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \text{ with } \mathbf{Z} \in \mathbb{R}^{N_{S_Q} \times N_V}. \end{aligned}$$

The inputs  $\mathbf{X}_{\dots}$  are sequences of length  $N_{S_Q}$  and  $N_{S_K}$  in matrix form  $[\mathbf{x}_1, \dots, \mathbf{x}_{N_{S_{\dots}}}]^T$ . The attention mechanism is discussed in more detail in the following. Two special cases of the attention mechanisms are used typically: self-attention and cross-attention. In self-attention the inputs are chosen as  $\mathbf{X}_Q = \mathbf{X}_K = \mathbf{X}_V = \mathbf{X}$ , therefore, the output is determined by attending to the input  $\mathbf{X}$  itself. Whereas in cross-attention, the inputs are chosen as  $\mathbf{X}_K = \mathbf{X}_V = \mathbf{X}$  and  $\mathbf{X}_Q = \tilde{\mathbf{Y}}$ . The output for the input  $\tilde{\mathbf{Y}}$  is determined by attending to the other input  $\mathbf{X}$ . In the remainder of this explanation, sequence are expressed through their matrix representation.

**Self-Attention** Given the input sequence  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{N_{S_Q}}]^T$ , the output sequence with the same length  $\mathbf{Z} = [z_1, \dots, z_{N_{S_Q}}]$  is determined through self-attention  $f: \mathbf{X} \mapsto \mathbf{Z}$ . For self-attention, the inputs are selected as  $\mathbf{X}_Q = \mathbf{X}_K = \mathbf{X}_V = \mathbf{X}$ . Each  $\mathbf{x}$  is linearly projected by  $\mathbf{W}_V$  leading to the values  $\mathbf{v}_1, \dots, \mathbf{v}_{N_{S_Q}} \in \mathbb{R}^{N_V}$ . The output elements  $z_1, \dots, z_{N_{S_Q}}$  are the weighted sums of all values  $\mathbf{v}_1, \dots, \mathbf{v}_{N_{S_Q}}$ , where the weights depend on the input sequence  $\mathbf{X}$  itself. Considering the first output element  $z_1$ , the weights or attention values  $\mathbf{w}(\mathbf{x}_1, \mathbf{X})$  depend on the complete input sequence  $\mathbf{X}$  and the first input element  $\mathbf{x}_1$ . The first input element  $\mathbf{x}_1$  is used to determine the query  $\mathbf{q}_1$  for the first output element  $z_1$  and the sequence  $\mathbf{X}$  to generate the keys  $\mathbf{K}$ . The attention  $\mathbf{att}(\mathbf{x}_1, \mathbf{X})$  is determined by the softmax of  $[\mathbf{q}_1 \mathbf{K}^T]$ . Hence, for the ranking of the attention values  $\mathbf{att}(\mathbf{x}_1, \mathbf{X})$  it holds: the larger the projection of  $\mathbf{k}_i$  in the direction of  $\mathbf{q}_1$  ( $\|\mathbf{k}_i\| \cos \theta$ ) the higher the rank of the corresponding attention value. Since the ranking is preserved by the softmax, the closer a key  $\mathbf{k}_i$  to the query  $\mathbf{q}_1$ , the more the generation of  $z_1$  attends to the corresponding value  $\mathbf{v}_i$ . In Figure 2.9, the previous explanation is illustrated.

**Cross-Attention** The output sequence  $\mathbf{Z} = [z_1, \dots, z_{N_{S_Q}}]$  is determined using  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{N_{S_K}}]$  in dependency of  $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_{N_{S_Q}}]$  as  $f: \mathbf{X} | \tilde{\mathbf{Y}} \mapsto \mathbf{Z}$ . In contrast to the self-attention, the outputs for each element in  $\tilde{\mathbf{Y}}$  are generated by cross-attending to  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{N_{S_K}}]$ . The output  $\mathbf{Z}$  is the weighted sum of the values  $\mathbf{V}$  of  $\mathbf{X}$ , dependent

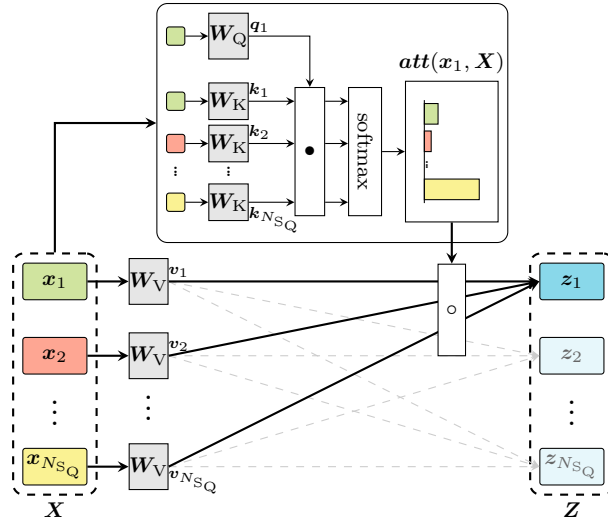


Figure 2.9: Illustration of self-attention mechanism. Upper part depicts the calculation of the attention values  $att(x_1, \mathbf{X})$  for the output  $z_1$ .  $\bullet$ : dot product between all keys  $\mathbf{k}$  and the query  $\mathbf{q}_1$ ;  $\mathbf{W}\dots$ : learnable matrix.

on the second input sequence  $\tilde{\mathbf{Y}}$ , therefore  $\tilde{\mathbf{Y}}$  and  $\mathbf{Z}$  are of the same length  $N_{S_Q}$ . For example, the weights  $att(\tilde{\mathbf{y}}_1, \mathbf{X})$  for the first output element dependent on the sequence  $\mathbf{X}$  and  $\tilde{\mathbf{y}}_1$ . Here,  $\mathbf{X}$  is used to generate the keys  $\mathbf{K}$  and  $\tilde{\mathbf{y}}_1$  to generate the first query  $\mathbf{q}_1$ . The cross-attention is also illustrated in Figure 2.10.

The matrices  $\mathbf{W}_V$ ,  $\mathbf{W}_K$ , and  $\mathbf{W}_Q$  are shared across the sequence, hence varying length sequences can be fed to the attention mechanism. The learnables of an attention layer are  $\mathbf{W}_V$ ,  $\mathbf{W}_K$ , and  $\mathbf{W}_Q$ .

The attention mechanism flexibility is limited as the size of  $\mathbf{QK}^T$  grows quadratic with the sequence length for self-attention ( $N_S \times N_S$ ). Various approaches attempt to solve this bottleneck [TDBM20] for example through: limiting the attendance area [HKWS19, BPC20], low-Rank approximations [WLK+20, XZC+21] or kernelization [KVPF20, CLD+21].

Typically, in attention layers  $h$  instances of the attention mechanism are applied in parallel, called multi-head attention, leading to  $h$  different versions of the matrices  $\mathbf{W}_V$ ,  $\mathbf{W}_K$ , and  $\mathbf{W}_Q$ . The  $h$  results are concatenated and linearly projected by the learnable matrix  $\mathbf{W}_O \in \mathbb{R}^{hN_V \times N}$ . Since none of the learnable matrices depends on the input sequence length, the attention mechanism can be applied to arbitrary sequence lengths.

#### 2.1.4.2 Layer-Norm

Another component used in the transformer is the layer normalization [BKH16]. In this layer, the inputs are normalized and transformed as,

$$\mathbf{y} = \frac{\mathbf{x} - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}} \circ \mathbf{g} + \mathbf{b}, \quad (2.26)$$

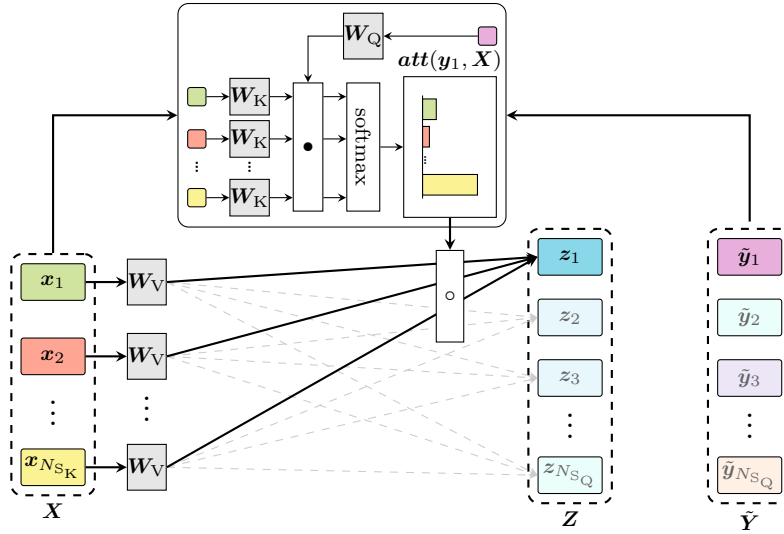


Figure 2.10: Illustration of self-attention mechanism. Upper part depicts the calculation of the attention values  $att(\mathbf{y}_1, \mathbf{X})$  for the output  $z_1$ .  $\bullet$ : dot product between all keys  $\mathbf{k}$  and the query  $\mathbf{q}_1$ ;  $W_{\dots}$ : learnable matrix.

with the mean  $\mu_x = \frac{1}{N_l} \sum_{n_l} x_{n_l}$ , the standard deviation  $\sigma_x = \sqrt{\frac{1}{N_l} \sum_{n_l} (x_{n_l} - \mu_x)^2}$ , and the learnable parameters  $\mathbf{g} \in \mathbb{R}^{N_l}$  and  $\mathbf{b} \in \mathbb{R}^{N_l}$ .

#### 2.1.4.3 Positional Encoding

Compared to classical RNNs, the transformer does not process the data sequentially, moreover, the order of the input is not relevant in transformers. Due to this, the so-called positional encoding is introduced, such that information about the position of an element within the sequence is provided. This can be realized in multiple ways, here only two variants are discussed. For each input element  $\mathbf{x}$ , a positional encoding vector  $\mathbf{pos}$  is determined and added to the actual input vector  $\mathbf{x} + \mathbf{pos}$ . In [VSP<sup>+</sup>17], the  $\mathbf{pos}$  is determined as

$$\mathbf{pos} = \begin{cases} \begin{pmatrix} \sin\left(\frac{\mathbf{pos}}{1000^{\frac{2\lfloor i/2 \rfloor}{N}}}\right) \\ \cos\left(\frac{\mathbf{pos}}{1000^{\frac{2\lfloor i/2 \rfloor}{N}}}\right) \end{pmatrix} & \text{if } 2|i \\ \text{else} \end{cases} \begin{matrix} N-1 \\ i=0 \end{matrix}, \quad (2.27)$$

with  $2|i$  indicates  $i$  divisible by 2. This provides a unique code for each position in a sequence.

Another approach is presented in [DBK<sup>+</sup>21], where a fix sequence length is used. There, for each position, the corresponding positional encoding vector  $\mathbf{pos}$  is learned.

#### 2.1.4.4 Vanilla Transformer

The base variant of the transformer as introduced in [VSP<sup>+</sup>17] consists of an encoder and a decoder. The encoder is used to process the input, such as a sentence which shall be

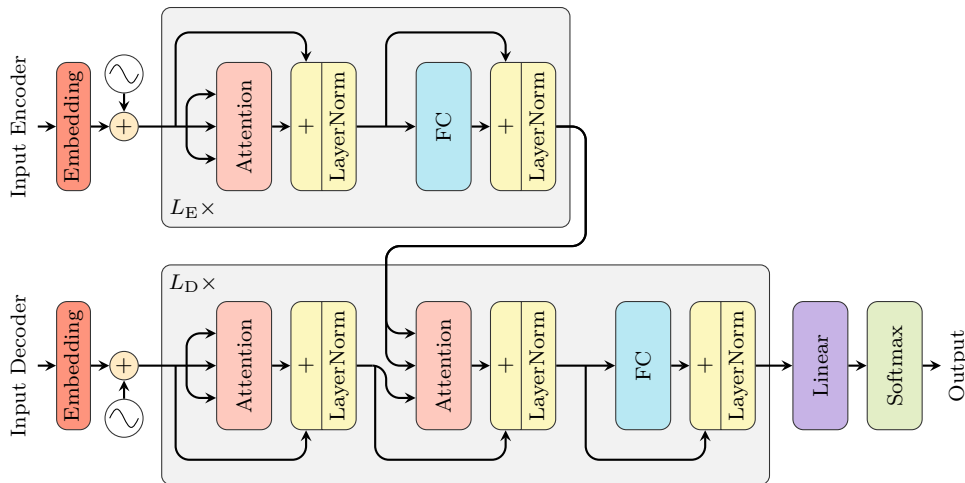


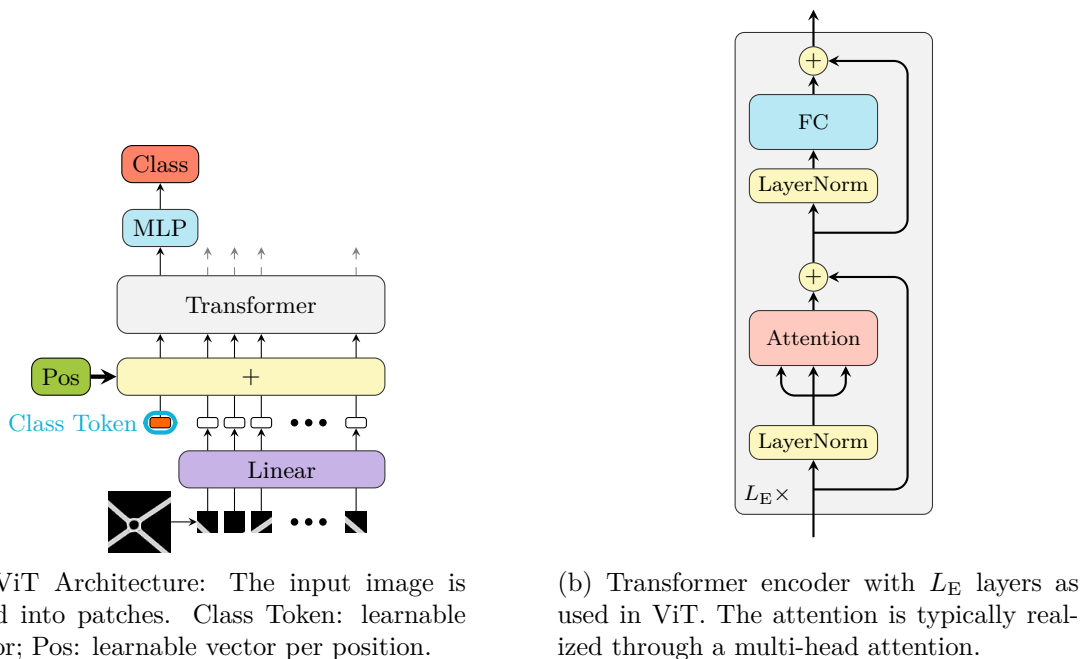
Figure 2.11: Transformer [VSP<sup>+</sup>17]. Upper part: encoder consisting of  $L_E$  layers; Lower Part: decoder consisting of  $L_D$  layers;  $\hat{\sim}$ : positional encoding. The attention is typically realized through a multi-head attention.

translated. The decoder generates the output given the information of the encoded input. The overall architecture of the transformer can be seen in Figure 2.11. The encoder processes the input by linearly embedding each input element and adding the positional encoding before sequentially feeding it through the  $L_E$  encoder layers. Each encoder layer consists of a multi head self-attention, followed by an element-wise layer normalization. A skip connection from the multi head self-attention input to the layer normalization input is added. Then each element is fed through a FC layer with ReLU activations, what is followed by element-wise layer normalization. A skip connection is used from the input of the FC layer to the input of the layer normalization.

The decoder takes its input as query in the attention, and the encoder output as keys and values. Like in the encoder, the elements in the decoder input are first linearly projected and the positional encodings are added. Then the decoder input together with the encoder output is processed through  $L_D$  decoder layers. The output elements are linearly projected and fed through a softmax to produce the overall output. Like in the encoder, first the combination of self-attention, layer normalization, and skip connection is used. Followed by the same combination but with a cross-attention. The cross attention takes the decoder input as query and the encoder output as key and value. The last stage of a layer is formed by the combination of a FC layer, layer normalization, and skip connection as in the encoder.

#### 2.1.4.5 ViT

In [DBK<sup>+</sup>21] the ViT, a variant of the transformer suited for image classification is presented. The naive way of applying the transformer on image data by considering each pixel as one input element leads to large attention matrices ( $N_{\text{pixel}} \times N_{\text{pixel}}$ ) and hence is com-

Figure 2.12: ViT [DBK<sup>+</sup>21]

putationally difficult to realize. The approach in [DBK<sup>+</sup>21] is to create non-overlapping patches, where each patch contains multiple pixels. The ViT reaches state-of-the-art performance on classification task, without using convolutions.

The ViT utilizes an encoder-only transformer architecture, hence the decoder part of the classical transformer is not used. First, the  $N_S$  image patches are linearly projected. Then, the so-called class token is added as an input element. In the next step, the positional encodings are added to the input elements, here, the positional encodings are learned per position. The input is fed through a transformer encoder with  $L_E$  layers. The layers are slight variations from the classical encoder layers. The layer normalization is moved from the end of the attention and FC blocks to the beginning. Hence, the first block is layer normalization followed by self-attention with a skip connection from the layer normalization input to the output of the self-attention. The second block with layer normalization followed by a FC layer are connected respectively. The class prediction is based on the output of the encoder. However, only the output corresponding to the class token input is used for the prediction, by processing the class token output with a MLP to generate the class estimation. The architecture of the ViT and the adjusted encoder layer are shown in Figure 2.12.

## 2.2 Dimensionality Reduction

In this work, dimensionality reduction techniques are applied for multiple purposes. The most straight-forward utilization for dimensionality reduction is visualization, there a high-dimensional space  $N > 3$  is projected into a displayable number of dimensions  $N_{\text{red}} \leq 3$ . However, dimensionality reduction is also commonly used as a pre-processing step for other methods.

For each dimensionality reduction technique, the projection is achieved by optimizing a certain objective. Different objectives as well as the methods are explained in the following. In this work, the dimensionality reduction techniques *Principal Component Analysis* (PCA), *t-distributed Stochastic Neighbor Embedding* (t-SNE), and *Uniform Manifold Approximation and Projection* (UMAP) are applied and explained.

### 2.2.1 PCA

The PCA is one of the most used dimensionality reduction techniques. The main advantage of the PCA is that it performs a linear mapping, and hence the results are interpretable. The objective of the PCA is to find a projection of the data, such that the variance along the new dimensions is maximized. This is realized by determining the orthogonal principal components (directions) of the data with the highest variance.

Given the dataset  $\mathcal{D}$  with  $M$  samples, arranged in the dataset array of shape  $N \times M$  as  $\mathbf{X}_{\mathcal{D}} = [\mathbf{x}_1 - \boldsymbol{\mu}_{\mathbf{x}}, \dots, \mathbf{x}_M - \boldsymbol{\mu}_{\mathbf{x}}]$  with zero mean per dimension, the covariance matrix is defined as

$$\mathbf{C}_{\mathbf{xx}} = \mathbb{E}_{\mathbf{x}} \left\{ (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\text{T}} \right\} \quad (2.28)$$

$$= \frac{1}{M} \mathbf{X}_{\mathcal{D}} \mathbf{X}_{\mathcal{D}}^{\text{T}}, \quad (2.29)$$

with  $\boldsymbol{\mu}_{\mathbf{x}} \in \mathbb{R}^N$ , the mean values for each dimension  $N$ . The PCA aims to find orthogonal unit vectors, such that if the data is projected on them the variance along each vector is maximized. Moreover, the dimension shall be sorted in descending order of variance, hence the first dimension should have the highest variance. This can be realized by iteratively solving

$$\begin{aligned} \mathbf{w}_i &= \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ \mathbf{w} \mathbf{C}_{\mathbf{xx}} \mathbf{w}^{\text{T}} \right\} & (2.30) \\ \text{s.t. } & \mathbf{w}^{\text{T}} \mathbf{w} = 1, \\ & \mathbf{w}^{\text{T}} \mathbf{w}_j = 0 \quad \forall 1 \geq j < i. \end{aligned}$$

Where  $\mathbf{w} \mathbf{C}_{\mathbf{xx}} \mathbf{w}^{\text{T}}$  is the variance of the data projected on  $\mathbf{w}$ . Through the constraints it is ensured that the  $i$ -th vector is of length one and orthogonal to all previously  $\forall 1 \geq j < i$  found vectors.



The problem in Equation (2.30) for all  $N$  vectors  $\mathbf{w}_1, \dots, \mathbf{w}_N$  can be solved through the eigendecomposition of the covariance matrix as  $\mathbf{C}_{\mathbf{xx}} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T$ . Hence, when sorting the eigenvectors and eigenvalues descending to their eigenvalues, the weight matrix  $\mathbf{W}$  contains the vectors  $\mathbf{w} \in \mathbb{R}^N$  as defined before.

By using only the first  $N_{\text{red}}$  components, the covariance matrix is approximated as  $\hat{\mathbf{C}}_{\mathbf{xx}} = \mathbf{W}_{\text{red}}\mathbf{\Lambda}_{\text{red}}\mathbf{W}_{\text{red}}^T$ . Projecting an input vector  $\mathbf{x}$  to the  $N_{\text{red}}$  components, hence, generating the dimensionality reduced output vector  $\mathbf{y}$  with  $N_{\text{red}}$  dimensions is determined as

$$\mathbf{y} = \mathbf{W}_{\text{red}}^T(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}). \quad (2.31)$$

When performing the dimensionality reduction on all input vectors, the dimensionality reduced dataset  $\mathcal{D}_{\text{red}} = \{\mathbf{y}_m\}_{m=1}^M$ , with  $\mathbf{y} \in \mathbb{R}^{N_{\text{red}}}$  is generated.

### 2.2.2 t-SNE

Non-linear dimensionality reductions, as performed by t-SNE [MH08], are successfully used for visualizing complex and high-dimensional data. The embeddings produced by t-SNE can reveal lots of structure of the high-dimensional data. The objective in t-SNE is to make high-dimensional pairwise similarities as similar as possible to their low-dimensional pairwise similarities. The Barnes-Hut-SNE [vdM14] is a variant of t-SNE, which is typically applied in practice, due to its performance advantage over the original t-SNE. In the following, the t-SNE and its accelerated version Barnes-Hut-SNE are briefly summarized. The overall objective in t-SNE is to determine low-dimensional embeddings  $\mathcal{D}_{\text{red}} = \{\mathbf{y}_m\}_{m=1}^M$  with  $\mathbf{y} \in \mathbb{R}^{N_{\text{red}}}$  such that the embeddings best reflect the structure of the data points in  $\mathcal{D} = \{\mathbf{x}_m\}_{m=1}^M$  with  $\mathbf{x} \in \mathbb{R}^N$ .

The first step in t-SNE is to determine the pairwise similarities in the high-dimensional space. For this purpose, a pointwise neighborhood adaptive similarity measure is introduced. The pointwise similarity is interpreted as the conditional probability of the  $j$ -th data point being selected as neighbor of the  $i$ -th data point. It is assumed that the selection of a neighbor is modeled through a Gaussian centered at the  $i$ -th data point, parameterized by  $\sigma_i$ . This is formulated as

$$v_{j|i} = \frac{\exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_j)^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_k)^2}{2\sigma_i^2}\right)}, \quad (2.32)$$

where  $\mathbf{x}$  is a sample from  $\mathcal{D}$  and the  $d$  is a distance measure.

The accelerated version, the Barnes-Hut-SNE, approximates the pointwise similarities by only using the set of  $K$  nearest neighbor indices  $\mathcal{K}_i$  of each data point. Hence, the

computational effort is reduced significantly. The pointwise similarities are determined as

$$v_{j|i} = \begin{cases} \frac{\exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_j)^2}{2\sigma_i^2}\right)}{\sum_{k \in \mathcal{K}_i} \exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_k)^2}{2\sigma_i^2}\right)} & \text{if } j \in \mathcal{K}_i \\ 0 & \text{else} \end{cases}. \quad (2.33)$$

For both versions, the  $\sigma_i$  is selected such that the perplexity over  $v_{\cdot|i}$  matches  $u$ , hence

$$u = \prod_{j=1}^M v_{j|i}^{-v_{j|i}} \text{ and} \quad (2.34)$$

$$u = \prod_{j \in \mathcal{K}_i} v_{j|i}^{-v_{j|i}} \quad (2.35)$$

hold respectively. In Barnes-Hut-SNE, the number of neighbors is selected as  $K = \lfloor 3u \rfloor$ .

Next, the pointwise similarities are symmetrized, leading to pairwise similarities as

$$v_{ij} = \frac{v_{j|i} + v_{i|j}}{2M}. \quad (2.36)$$

The pairwise similarities describe the data points in the high-dimensional space. By selecting an appropriate perplexity, the focus on local or global structure in the data can be adjusted.

To create a low-dimensional representations of the data points, pairwise similarities are defined in the low-dimensional space as well. In contrast to the previous definition, a normalized Student t-kernel with one degree of freedom is used as similarity function

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_l - \mathbf{y}_k\|_2^2)^{-1}}. \quad (2.37)$$

In t-SNE, the objective is to make the low-dimensional similarities  $q_{ij}$  similar to the high-dimensional similarities  $v_{ij}$  of the data points. This is realized by minimizing the Kullback-Leibler divergence

$$\mathcal{L}_{\text{t-SNE}} = \sum_{i \neq j} v_{ij} \log\left(\frac{v_{ij}}{q_{ij}}\right) \quad (2.38)$$

through gradient descent. Therefore, the low-dimensional embeddings  $\mathbf{y}$  are generated through this optimization procedure. The low-dimensional representations are typically randomly initialized.

In the accelerated version, the gradient is approximated through the Barnes-Hut algorithm [vdM14]. Within this algorithm, points are grouped when their contributions can be represented by a single point [MHM18]. Therefore, it is not required to determine the

similarities between all the embeddings.

### 2.2.3 UMAP

UMAP [MHM18] is a non-linear dimensionality reduction technique. In contrast to t-SNE, UMAP is derived from Riemannian geometry and algebraic topology. In UMAP, data points are assumed to be uniformly distributed on a manifold. For this, the manifold is approximated through many local manifolds, each covering only the neighborhood of a data point. For details on the derivation see [MHM18]. As before, the overall objective is to determine the low-dimensional embeddings  $\mathcal{D}_{\text{red}} = \{\mathbf{y}_m\}_{m=1}^M$  with  $\mathbf{y} \in \mathbb{R}^{N_{\text{red}}}$  such that the embeddings reflect the structure of the data points in  $\mathcal{D} = \{\mathbf{x}_m\}_{m=1}^M$  with  $\mathbf{x} \in \mathbb{R}^N$ .

The steps to generate the low-dimensional embeddings are very similar to the steps in t-SNE, therefore, the same notation is used here to highlight the similarities. However, in UMAP the relations between data points are not expressed through conditional probabilities but by the edge weights of a directed nearest neighbor graph.

The first step in UMAP is to generate the so-called local fuzzy simplicial sets, obtained by a weighted directed nearest neighbor graph as

$$v_{j|i} = \begin{cases} \exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_j) - \rho_i)}{\sigma_i}\right) & \text{if } j \in \mathcal{K}_i \\ 0 & \text{else} \end{cases}. \quad (2.39)$$

Here,  $\rho_i$  is the dissimilarity between  $\mathbf{x}_i$  and its most similar neighbor. The parameter  $\sigma_i$  is selected such that

$$\sum_{j \in \mathcal{K}_i} \exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_j) - \rho_i)}{\sigma_i}\right) = \log_2(K) \quad (2.40)$$

holds. In difference to t-SNE, no normalization is required to calculate the pointwise similarities.

With the probabilistic t-conorm, the pointwise similarities is symmetrized to achieve the pairwise similarities

$$v_{ij} = (v_{j|i} + v_{i|j}) - v_{j|i}v_{i|j}. \quad (2.41)$$

Hence, the directed graph is converted into an undirected graph.

The pairwise similarity of the low-dimensional representations  $\mathbf{y}$  is defined by

$$q_{ij} = \frac{1}{1 + a\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2b}}, \quad (2.42)$$

where  $a$  and  $b$  are hyperparameters. In UMAP,  $a$  and  $b$  are automatically selected by a non-linear fitting, such that a minimum distance between the representation points is realized.

As in t-SNE, the objective is to find projections that maximize the similarity between

the low-dimensional and the high-dimensional pairwise similarities  $q_{ij}$  and  $v_{ij}$ . In UMAP, this is realized by minimizing the cross entropy

$$\mathcal{L}_{\text{UMAP}} = \sum_{i \neq j} v_{ij} \log \left( \frac{v_{ij}}{q_{ij}} \right) + (1 - v_{ij}) \log \left( \frac{1 - v_{ij}}{1 - q_{ij}} \right) \quad (2.43)$$

of the two fuzzy sets (high- and low-dimensional), using stochastic gradient descent. The embedding is typically initialized with a spectral embedding [vL07].

## 2.3 Outlier Detection

The objective in the area of outlier detection is to find data points, which do not match the patterns and characteristics of the dataset, these points are referred to as outlier or abnormal data points. Aside from finding outliers in an existing database, those methods sometimes are applied for novelty or out-of-distribution detection. An observation is declared as novel or not, depending on if it fits in the existing database. In this section, various outlier detection methods are summarized. First, a group of neighborhood-based methods is explained. Next, *Isolation Forest* (IF) and *One-Class Support Vector Machine* (OCSVM) are discussed. Last, approaches from the group of reconstruction-based methods are summarized.

In the following, various outlier detection methods are introduced. The outlier score  $\mathcal{A}_{\dots}$  is the core part of an outlier detection method. An outlier score returns a value for each tested data point. Identifying outliers is typically realized by considering the data points above / below some thresholds as outliers. The notation  $\mathcal{A}_{\dots}^{\uparrow}$  represents outlier scores where high values indicate that the investigated data point is more likely to be an outlier. On the contrary, a data point is more likely to be an outlier, the lower the value of scores with the notation  $\mathcal{A}_{\dots}^{\downarrow}$ . If not stated differently, outliers are detected if they are more likely to be an outlier than all other data points or at least a big share of the other data points. Hence, the outlier score is usually considered with respect to all other data points.

### 2.3.1 Neighborhood-Based Methods

Neighborhood-based methods aim to detect local outliers. For this purpose, the data point under investigation  $\mathbf{x}_i$  is compared to its  $K$ -neighborhood  $\mathcal{K}_i$ . As defined in Section 2.2.2,  $\mathcal{K}_i$  contains the indices of the  $K$  nearest neighbors of  $\mathbf{x}_i$ , given some distance function. Let  $\mathcal{R}_i$  be the inverse neighborhood set, it contains all the indices for which  $\mathbf{x}_i$  is inside their  $K$ -neighborhood, hence  $\mathcal{R}_i = \{j \in \mathcal{I} | i \in \mathcal{K}_j\}$ , where  $\mathcal{I}$  denotes the set of all available indices in the dataset. Next, the various neighborhood-based outlier scores are summarized briefly.

**$K$ -dist** One of the simplest approaches is to measure the outlier strength is to use the distance to the  $K$ -th neighbor [RRS00]. Therefore, the farther the  $K$ -th neighbor, the more outlying the data point  $\mathbf{x}_i$  is. With  $\mathcal{K}_i(K)$  being the  $K$ -th nearest neighbor of the  $i$ -th data point, the outlier score is defined as

$$\mathcal{A}_K^\uparrow(\mathbf{x}_i) = d(\mathbf{x}_i, \mathbf{x}_{\mathcal{K}_i(K)}). \quad (2.44)$$

This score is simple to be determined, however, it does not consider the complete  $K$ -neighborhood. Moreover, the densities within the neighborhood are not considered.

**$\Sigma K$ -dist** In order to consider the complete  $K$ -neighborhood of the  $i$ -th data point, taking the sum over all  $K$  distances is another outlier score [AP02]

$$\mathcal{A}_{\Sigma K}^\uparrow(\mathbf{x}_i) = \sum_{j \in \mathcal{K}_i} d(\mathbf{x}_i, \mathbf{x}_j). \quad (2.45)$$

Like the score  $\mathcal{A}_K^\uparrow$ ,  $\mathcal{A}_{\Sigma K}^\uparrow$  follows the intuition that outlying points should be a far way from their neighborhood.

**ODIN** In *Outlier Detection using Indegree Number* (ODIN) [HKF04], instead of the neighborhood of the  $i$ -th point, the neighborhoods of the other points are investigated. The concept is, that an outlying point less often occurs in others'  $K$ -neighborhoods. Hence, the smaller the inverse neighborhood set

$$\mathcal{A}_{\text{ODIN}}^\downarrow(\mathbf{x}_i) = |\mathcal{R}_i|, \quad (2.46)$$

the more likely it is for a point to be an outlier. When considering the  $K$ -neighborhood relationships in a directed graph that would be equivalent to the incoming edges, therefore the indegree of the  $i$ -th vertex.

**LOF** One of the most popular outlier scores is the *Local Outlier Factor* (LOF) [BKNS00]. This score is based on local density comparison. The local density for the  $i$ -th point is compared to the average density in the  $K$ -neighborhood. First, the so-called reachability distance is defined as

$$d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j) = \max\left\{d(\mathbf{x}_j, \mathbf{x}_{\mathcal{K}_j(K)}), d(\mathbf{x}_i, \mathbf{x}_j)\right\}, \quad (2.47)$$

which can be understood as the reachability of the  $i$ -th data point from the  $j$ -th data point. This can be the true distance between two data points or at least it is the  $K$ -distance of the  $j$ -th data point. Therefore, all data points in the  $K$ -neighborhood of  $j$  have a reachability of  $K$ -distance. Given the reachability distance definition, the local

reachability density  $lrd$  is determined as the inverse average reachability distances from all points in the neighborhood of  $i$  to the  $i$ -th point itself

$$lrd(\mathbf{x}_i) = \frac{1}{\frac{1}{|\mathcal{K}_i|} \sum_{j \in \mathcal{K}_i} d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j)}. \quad (2.48)$$

The  $lrd$  inversely reflects how reachable the  $i$ -th point is from all its neighbors  $\mathcal{K}_i$ . Provided with the density estimates per point, the LOF is determined as the average density in the neighborhood relative to  $i$ -th density

$$\mathcal{A}_{\text{LOF}}^\uparrow(\mathbf{x}_i) = \frac{\frac{1}{|\mathcal{K}_i|} \sum_{j \in \mathcal{K}_i} lrd(\mathbf{x}_j)}{lrd(\mathbf{x}_i)}. \quad (2.49)$$

If the  $i$ -th data point is outlying, that will lead to a small density  $lrd(\mathbf{x}_i)$  compared to the densities of the neighborhood. Hence, the  $\mathcal{A}_{\text{LOF}}^\uparrow(\mathbf{x}_i)$  will become high.

**LDOF** In [ZHJ09], the *Local Distance-based Outlier Factor* (LDOF) is presented. It compares the average distance between the  $i$ -th point to its neighbors to the average distance between all  $K(K-1)$  pairs of neighbors as

$$\mathcal{A}_{\text{LDOF}}^\uparrow(\mathbf{x}_i) = \frac{\frac{1}{|\mathcal{K}_i|} \sum_{j \in \mathcal{K}_i} d(\mathbf{x}_i, \mathbf{x}_j)}{\frac{1}{|\mathcal{K}_i|(|\mathcal{K}_i|-1)} \sum_{j, j' \in \mathcal{K}_i | j \neq j'} d(\mathbf{x}_j, \mathbf{x}_{j'})}. \quad (2.50)$$

**ABOD** Where the former methods are based on distances, the *Angle-Based Outlier Detection* (ABOD) [KhZ08] utilizes angles to determine outliers. Here, only the special case fast ABOD is discussed, since it is considering only the  $K$ -neighborhood. First,  $K$  vectors are constructed spanning from the  $i$ -th data point to each neighbor. Then, for all possible  $K(K-1)/2$  vector pairs, the angle between the vectors is determined. The variance over all angles for the  $i$ -th data point yields the outlier score as

$$\mathcal{A}_{\text{ABOD}}^\downarrow(\mathbf{x}_i) = \text{Var}_{j, j' \in \mathcal{K}_i | j \neq j'} \left( \frac{(\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_{j'} - \mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|^2 \cdot \|\mathbf{x}_{j'} - \mathbf{x}_i\|^2} \right). \quad (2.51)$$

The more inlying a data point is, the more the angles to the neighborhood vary. Therefore, the smaller the value of the ABOD, the more outlying the data point is.

**Other** The scores listed above are a few representatives from the group of neighborhood-based outlier detection methods. A few more will be briefly summarized in the remainder of this subsection, based on [WFBU20].

In the comprehensive survey [CZS+16a], various nearest neighborhood based methods are applied to several datasets, providing an empirical analysis. The datasets are made publicly available to act as a benchmark for outlier detection.

Several extensions of the LOF have been presented. The *Connectivity-based Outlier Factor* (COF) [TCcFC02] using on connectivity chains to estimate the local reachability density. In the *Local Outlier Probabilities* (LoOP) [KKSZ09], the density estimates are replaced by the reciprocal of the mean quadratic distance. A more complex replacement of LOF’s density estimates can be found in the *Local Density Factor* (LDF) [LLP07], where a Gaussian kernel density estimate is used. Another variant is the *Simplified LOF* (SLOF) [SZK12], where LOF’s reachability distance is replaced with the distance to the  $K$ -th neighbor.

The following methods are based on definitions from dimensionality reduction techniques. Using the weighted directed graph constructed in the first stage of t-SNE [MH08], the outlier score *Stochastic Outlier Selection* (SOS) was introduced in [JHPvdH12]. Based on t-SNE’s faster approximation Barnes-Hut-SNE, the outlier scores *K Nearest Neighbor SOS* (KNNsOS) and *Intrinsic SOS* (ISOS) are presented in [SG17]. The former can be considered as an approximation of the SOS, where in ISOS, the distance measures are transformed based on an intrinsic dimensionality estimate.

### 2.3.2 Isolation Forest

The *Isolation Forest* (IF) [LTZ08] is another method for outlier detection. The basic assumption is, that if a data point is outlying, it is easy to be isolated.

An IF  $\{T_b(\mathbf{x}, \Theta_b)\}$  is constructed, leading to  $B$  isolation trees, with  $\Theta_b$  being the random parameter vector for the  $b$ -th tree  $T_b$ . A sub-sampled dataset consisting of  $M_\psi$  samples is generated per tree. Provided with the sub-sampled dataset, a tree is grown by randomly choosing a dimension and randomly choosing a split threshold (c.f. extremely randomized trees [GEW06]). The growing procedure stops whenever all data points have the same values or the tree reaches the depth limit  $\lceil \log_2 M_\psi \rceil$ . After growing the IF, all data points are passed through all trees. Let  $\mathcal{T}_{i,b} = \{t_{j_{i_2},b}, \dots, t_{j_{i_J},b}\}$  be the path of the  $i$ -th data point through the  $b$ -th tree. Here  $t_{j_{i_2},b}$  denotes the second node of the  $b$ -th tree the  $i$ -th element has passed and  $t_{j_{i_J},b}$  the terminal node for the  $i$ -th data point in the  $b$ -th tree. The average length of the paths for the  $i$ -th data point over all trees is

$$\bar{l}(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^B |\mathcal{T}_{i,b}|. \quad (2.52)$$

The average length is normalized by the average length of an unsuccessful search path in a binary search tree, which is given through

$$\bar{l}_{\text{norm}}(M_\psi) = 2H(M_\psi - 1) - (2(M_\psi - 1)/M_\psi). \quad (2.53)$$

The harmonic number  $H(\cdot)$  can be estimated via  $\ln(\cdot) + 0.5772156649$  [LTZ08]. Finally,

the outlier score of the IF is determined through

$$\mathcal{A}_{\text{IF}}^\uparrow(\mathbf{x}_i) = 2^{\frac{\bar{l}(\mathbf{x}_i)}{l_{\text{norm}}(M_\psi)}}. \quad (2.54)$$

This score gives values closer to one, the lesser splits are required to separate a data point, and hence the more outlying it is. On the contrary, this score will be smaller for inside points. More specifically, in [LTZ08] it is stated that for scores much smaller than 0.5 it is quite safe to regard them as normal instances.

### 2.3.3 One-Class Support Vector Machine

Another outlier detection method was presented in [SWS<sup>+</sup>00] based on the concept of *Support Vector Machines* (SVMs). The idea is to consider a one class case by the *One-Class SVM* (OCSVM), where the objective is to separate all data points from the origin in the feature space with a hyperplane. Hence, all data points are treated as one class. With  $\Phi(\cdot)$ , the projection from the input to the feature space, the problem definition is given as

$$\min_{\mathbf{w}, \xi, \rho} \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{1}{\nu M} \sum_{i=1}^M \xi_i - \rho, \quad (2.55)$$

$$\text{s.t. } \mathbf{w}^\top \Phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad (2.56)$$

$$\xi_i \geq 0, \quad (2.57)$$

where minimizing  $\frac{1}{2} \|\mathbf{w}\|_2^2$  contributes to maximizing the margin like in the normal SVM. Minimizing  $-\rho$  contributes to shift the decision hyperplane  $\mathbf{w}^\top \Phi(\mathbf{x}) - \rho = 0$  as far as possible from the origin, but also in maximizing the margin  $\rho / \|\mathbf{w}\|_2$ . As in the normal SVM,  $\xi$  are the slack variables enabling a soft margin classifier. Using the kernel trick  $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$  and the method of Lagrange multipliers, the dual problem can be stated as

$$\min_{\lambda} \frac{1}{2} \sum_{ij} \lambda_i \lambda_j k(\mathbf{x}_i, \mathbf{x}_j), \quad (2.58)$$

$$\text{s.t. } 0 \leq \lambda_i \leq \frac{1}{\nu M} \quad (2.59)$$

$$\sum_i \lambda_i = 1. \quad (2.60)$$

As stated in [SWS<sup>+</sup>00], if  $\rho \neq 0$ ,  $\nu$  is an upper bound on the fraction of outliers in the training data and  $\nu$  is a lower bound on the fraction of support vectors. Under more constraints (see [SWS<sup>+</sup>00] for details),  $\nu$  asymptotically equals both fractions. After determining the coefficients  $\lambda$  of the dual problem, the outlier ship of the  $i$ -the data point



is determined as

$$\mathcal{A}_{\text{OCSVM}}^{\downarrow}(\mathbf{x}_i) = \text{sgn} \left( \sum_{j=1}^M \lambda_j k(\mathbf{x}_j, \mathbf{x}_i) - \rho \right). \quad (2.61)$$

Ideally, outliers should have an outlier score of  $-1$ , while normal data points (inliers) should have  $+1$ . By assuming a certain fraction of outliers  $\nu$  in the training data and by the choice of the kernel and its parameters, the closeness of the decision boundary to the data can be adjusted.

### 2.3.4 Reconstruction-Based Methods

Various approaches try to utilize generative neural networks for outlier detection. The narrative is that if a generative model is able to reconstruct the input properly that input can be interpreted as known data. For unknown data, which was not part of the training, the network might fail to reconstruct that. Those approaches are well suited for images, though not being limited to them. A General issue with the reconstruction based method is that the model would need to be retrained with the unknown data if it should be added to the known set. Moreover, the network should not generalize well in order to actually fail to reconstruct unknown data. One example for that might be when the generative model was trained on images of fours and sevens, it should still not reconstruct a nine properly. However, that is likely to happen for well generalizing networks. In the following, only a few approaches are discussed, but there exist various reconstruction-based approaches and applications.

#### 2.3.4.1 Autoencoder

With the family of autoencoders, a straight forward method for solving the reconstruction tasks exists. Autoencoders find application in many approaches e. g., [SY14, AC15, LBR<sup>+</sup>18]. Some use basic autoencoders, others use variants like denoising or variational autoencoders (see Section 2.5.1.1 for details on vanilla autoencoders). Nevertheless, the concept remains the same: The autoencoder is trained with normal/known data. In inference, new data is passed through the trained autoencoder. The outlier score is estimated by comparing the reconstructed output with the input, using the reconstruction error. Let  $g$  be the decoder and  $f$  be the encoder, the outlier score can then be defined as

$$\mathcal{A}_{\text{ROD}}^{\uparrow}(\mathbf{x}_i) = \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|_2^2, \quad (2.62)$$

where ROD stand for *Reconstruction-based Outlier Detection*. The worse the reconstruction the higher the outlier score and hence the more likely the input is an outlier.

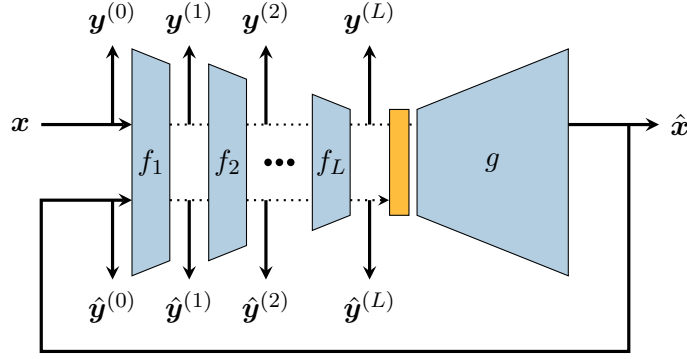


Figure 2.13: Reconstruction along Projection Pathway [KSL+20].  $\mathbf{y}^{(\dots)}$ : hidden states for the input;  $\hat{\mathbf{y}}^{(\dots)}$ : hidden states for the input;  $\mathbf{x}$ : input;  $\hat{\mathbf{x}}$ : reconstruction;  $f_{\dots}$ : layers of the encoder;  $g$ : decoder.

### 2.3.4.2 RaPP

In *Reconstruction along Projection Pathway* (RaPP) [KSL+20], another reconstruction-based approach based on autoencoder is presented. Instead of using only the difference of the input and the generated output, also the hidden states of the encoder are used. Specifically, two new outlier scores are presented, *Simple Aggregation along Pathway* (SAP) and normalized aggregation along pathway. In this work, only the SAP outlier score is applied, since the latter requires singular value decomposition of all hidden states flattened and concatenated. Due to large amount of hidden states in deep networks, this would lead to high computational effort.

The assumption behind RaPP is that also the hidden states of the input and the hidden states of the reconstructed output should be very similar if it is a known input. For this purpose, in [KSL+20] it is proposed to first process the input by the autoencoder as  $\hat{\mathbf{x}} = g(f(\mathbf{x}))$ . In order to access the hidden states, the outputs of the various layers ( $f_{\dots}$ ) of the encoder are used. Here, they are denoted by  $\mathbf{y}^{(l)}$  where  $l$  represents the layer number, with  $\mathbf{y}^{(0)}$  the input to the encoder and  $\mathbf{y}^{(L)}$  the latent representation. The hidden states of the input  $\mathbf{x}$  are concatenated to  $\mathbf{y}^{(0\dots L)}$  and the hidden states for the reconstructed output  $\hat{\mathbf{x}}$  are concatenated to  $\hat{\mathbf{y}}^{(0\dots L)}$ . Hence, the reconstructed output  $\hat{\mathbf{x}}$  is fed through the encoder. In Figure 2.13 the generation of  $\mathbf{y}^{(0\dots L)}$  and  $\hat{\mathbf{y}}^{(0\dots L)}$  is depicted. Given the above definitions, the outlier score for SAP follows as

$$\mathcal{A}_{\text{SAP}}^{\uparrow}(\mathbf{x}_i) = \|\mathbf{y}_i^{(0\dots L)} - \hat{\mathbf{y}}_i^{(0\dots L)}\|_2^2. \quad (2.63)$$

Since  $\mathbf{y}^{(0)}$  represents  $\mathbf{x}$  and  $\hat{\mathbf{y}}^{(0)}$  represents  $\hat{\mathbf{x}}$ , the outlier score includes the normal reconstruction-based outlier score. The less similar the input and the reconstruction as well as their hidden states are, the higher the outlier score will be and the more likely the data point is an outlier.

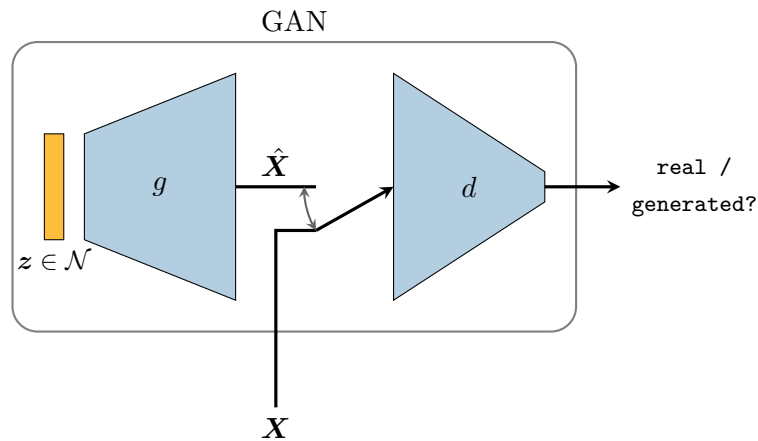


Figure 2.14: f-AnoGAN: GAN training;  $z \in \mathcal{N}$ : latent representation is sampled from a normal distribution;  $g$ : generator;  $d$ : discriminator; **real / generated**: estimation of the discriminator.

### 2.3.4.3 f-AnoGAN

*Generative Adversarial Networks* (GANs) are also used for outlier detection based on the reconstruction concept, e. g., [ZRF<sup>+</sup>18, SSW<sup>+</sup>17, SSW<sup>+</sup>19, DVR<sup>+</sup>19]. In the context of this work only the *fast Anomaly detection with GANs* (f-AnoGAN) method is summarized and applied. Like before, the assumption is that if the network is able to generate an output which is very similar to the input, that input can be assumed to be known. Moreover, also the discriminator’s output for the original and the generated input is taken into consideration. In the following, the method is briefly summarized.

**GAN Training** In the first step, the GAN is trained to produce samples that are very similar to the ones in the data set. For this purpose, a generator network  $g$  is given a random latent realization, commonly drawn from Gaussian distribution,  $z \in \mathcal{N}$ . This way, the output  $\hat{X} = g(z)$  is generated. Ideally, a well-trained generator will produce very realistic outputs. The discriminator network  $d$  aims to judge whether the input is real data  $X$  or a generated sample  $\hat{X}$ . See Figure 2.14 for a graphical illustration. More details about the GANs and their training can be found in e. g., [GBC16, SSW<sup>+</sup>19].

**Encoder Training** After the successful training of a GAN, it is required to find a latent representation from which the generator  $g$  can create a realistic output. For this purpose, in [SSW<sup>+</sup>19] an encoder  $f$  is introduced to learn the mapping of an input  $X$  to the latent representation  $z$ . During the training of the encoder, the generator network  $g$  and the discriminator network  $d$  are frozen, hence only the encoder network is optimized to realize the mapping. Figure 2.15 illustrates the learning procedure. The loss to train the encoder

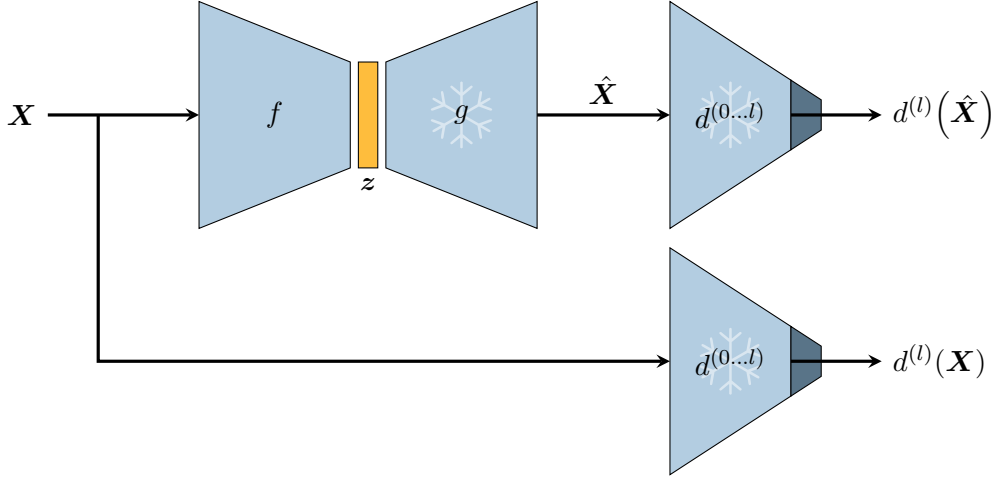


Figure 2.15: f-AnoGAN: Encoder training.  $\ast$ : frozen network.  $f$ : encoder;  $g$ : generator;  $d^{(0...l)}$ : first  $l$  layers of discriminator;  $d^{(l)}(\mathbf{X})$ : output of discriminator's  $l$ -th layer;  $\mathbf{X}$ : input.

is formulated as

$$\mathcal{L}_{f\text{-AnoGAN}} = \frac{1}{N} \|\mathbf{X} - g(f(\mathbf{X}))\|_2^2 - \frac{\kappa}{N_{d^{(l)}}} \|d^{(l)}(\mathbf{X}) - d^{(l)}(g(f(\mathbf{X})))\|_2^2. \quad (2.64)$$

The first term is the normal reconstruction loss between the input and the generated sample. The second term compares the  $l$ -th layer output  $\in \mathbb{R}^{N_{d^{(l)}}}$  of the discriminator between the input and the generated sample. Hence, the last layers of the discriminator  $d$  are not accessed. The hyperparameter  $\kappa$  controls the influence of the second term. In [SSW<sup>+</sup>19], it is stated that the second term improves the anomaly detection performance.

**Outlier Detection** After the training of the GAN and the encoder, the networks can be used to estimate the outlierhsip of samples. As already stated, the outlier detection is based on the reconstruction assumption like the methods before. A sample is projected into the latent space by using the encoder  $f$  and then reconstructed by the generator  $g$ . Additionally, the input  $\mathbf{X}$  and the reconstructed sample  $\hat{\mathbf{X}}$  are fed through the discriminator  $d$  in order to compare the  $l$ -th hidden representations. Hence, the anomaly score is the same as the loss for training the encoder:

$$\mathcal{A}_{f\text{-AnoGAN}}^\uparrow(\mathbf{X}) = \frac{1}{N} \|\mathbf{X} - g(f(\mathbf{X}))\|_2^2 - \frac{\kappa}{N_{d^{(l)}}} \|d^{(l)}(\mathbf{X}) - d^{(l)}(g(f(\mathbf{X})))\|_2^2. \quad (2.65)$$

Like the other reconstruction based scores, the higher the value, the more outlying the sample is.

## 2.4 Clustering

With clustering methods groups can be identified in unlabeled data. In this work, two clustering techniques  $K$ -means and *hierarchical clustering* HC are explained and applied. This section is based on [Wur18] and [KWBC23]. Clustering aims to find  $K$  clusters  $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ , such that the data inside each cluster is similar [Bis06]. This relation is described in terms of the proximity which can be defined through the similarity or dissimilarity, e.g., the Euclidean distance or Pearson correlation [XW05]. The proximity measure influences how the data is going to be clustered. Some methods are explicitly or implicitly connected to a specific proximity [XW05].

### 2.4.1 $K$ -Means

One of the most prominent methods for clustering is the  $K$ -means algorithm. The explanation is inspired from [MB20]. The objective is to find  $K$  cluster representatives  $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ , hence, one per cluster  $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ . The clusters and the representatives are selected such that the all data points in a cluster are close to the cluster representative. This can be formulated as the optimization problem

$$\{\mathbf{c}_1, \dots, \mathbf{c}_K\} = \underset{\{\mathbf{c}'_1, \dots, \mathbf{c}'_K\}}{\operatorname{argmin}} \left\{ \sum_{m=1}^M \min_{k=1, \dots, K} d(\mathbf{x}_m, \mathbf{c}'_k) \right\}. \quad (2.66)$$

The  $K$ -means algorithm solves the optimization problem by using the block coordinate descent method. The overall procedure is to iteratively optimize the cluster assignments and the representatives until the representatives are stable. Given a pre-selected  $K$ , and  $K$  randomly initialized cluster representatives, the  $K$ -means algorithm can be summarized as follows:

1. Each data point is assigned to the cluster with the closest representative. Here the cluster representatives are fixed.
2. The cluster representatives are updated according to Equation (2.66), given the assignment from step 1. This is equal to calculating the centroids of the clusters (average of all data points in a cluster) when using the squared Euclidean distance.
3. The steps 1 and 2 are repeated until the representatives are stable.

This way,  $K$ -means efficiently detects clusters in unlabeled data.

### 2.4.2 Hierarchical Clustering

Hierarchical clustering methods aim to find a hierarchy of clusters. They can be classified as agglomerative or divisive, where only agglomerative HC is applied in this work. This

section is mainly based on [Wur18]. In this subsection, the term dissimilarity instead of distance is used, since it is not necessarily an actual distance.

At the beginning, each data point is considered as a single cluster. Then they are successively merged until one single cluster remains. Typically, the most similar clusters are merged. The linkage function determines the new dissimilarity between a merged cluster and all remaining clusters. For this purpose, the dissimilarity  $D_{kl}$  between the clusters  $\mathcal{C}_k$  and  $\mathcal{C}_l$  is calculated according to the equations shown below. The dissimilarity between two points  $i$  and  $j$  is defined as  $d_{ij}$ . In the following multiple types of linkage functions are briefly introduced.

**Single Linkage** The minimum dissimilarity between all the elements of both clusters is the dissimilarity of the clusters:

$$D_{kl} = \min_{\substack{i \in \mathcal{C}_k \\ j \in \mathcal{C}_l}} \{d_{ij}\}. \quad (2.67)$$

**Complete Linkage** The maximum dissimilarity between all the elements of both clusters is the dissimilarity of the clusters:

$$D_{kl} = \max_{\substack{i \in \mathcal{C}_k \\ j \in \mathcal{C}_l}} \{d_{ij}\}. \quad (2.68)$$

**Average Linkage** The dissimilarity is determined through the average of all dissimilarities between the points of the two clusters:

$$D_{kl} = \frac{1}{|\mathcal{C}_k||\mathcal{C}_l|} \sum_{i \in \mathcal{C}_k} \sum_{j \in \mathcal{C}_l} d_{ij}. \quad (2.69)$$

**Weighted Average Linkage** The average of the dissimilarities from both previous clusters (before merging) to the arbitrary one is used:

$$D_{kl} = \frac{D_{kp} + D_{kq}}{2}. \quad (2.70)$$

**Centroid Linkage** The dissimilarity between the clusters is calculated by the Euclidean distance between the centroids of the clusters:

$$D_{kl} = \|\bar{\mathbf{x}}_k - \bar{\mathbf{x}}_l\|_2, \quad (2.71)$$

with the centroids

$$\bar{\mathbf{x}}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i, \quad (2.72)$$

$$\bar{\mathbf{x}}_l = \frac{1}{|\mathcal{C}_l|} \sum_{j \in \mathcal{C}_l} \mathbf{x}_j. \quad (2.73)$$

By applying these methods, one achieves a binary tree  $H$  representing the hierarchy, stating which clusters are merged with which dissimilarity. This hierarchy is typically visualized using a dendrogram. One possibility to perform cluster identification based on a hierarchical tree is to cut the tree into subtrees. All leaves of a subtree are considered as members of one cluster. The trees are cut by choosing a dissimilarity threshold.

The choice of the linkage method mainly depends on the given data set  $\mathcal{D}$  and the structure of the data inside it. Moreover, the choice also depends on the used dissimilarity measure. One main drawback of HC is its sensitivity to noise and outliers.

## 2.5 Representation Learning

Representation learning methods are a key part of the techniques proposed and developed in this work. The learned representations of traffic scenarios are used for tasks like clustering and outlier detection. Therefore, relevant learning strategies from the field of representation learning are summarized in this section.

Representation learning aims to find appropriate representations for a given input, where the appropriateness is defined by the training objective. Models trained with the summarized methods, project the input data to the so-called representation space. In a properly formed representation space, the data can be further processed with methods like clustering, classification or outlier detection.

Most of the presented methods are trained using multiple samples/instances simultaneously. There, a definition of similarity between the instance is required. During the training phase of a model, the objective is to pull similar instances close together in the representation space and to push dissimilar instance far from each other. Typically, there are two ways to determine similar and dissimilar instances. First, by using a measure to determine similarity between two given instances, e. g., are the two instances from the same class. Second, generating similar instances by augmentation. By the use of augmentation, the pretext knowledge is taken advantage of that the augmented instances origin from the same source and thereby should be similar.

Methods summarized in this section are contrastive learning, triplet learning, cross entropy based learning, *Swapping Assignments between Views* (SwAV) [CMM<sup>+</sup>20], Barlow Twins [ZJM<sup>+</sup>21], and *Variance-Invariance-Covariance Regularization* (VICReg) [BPL21]. Moreover, typical augmentations are summarized. The autoencoder is explained as well,

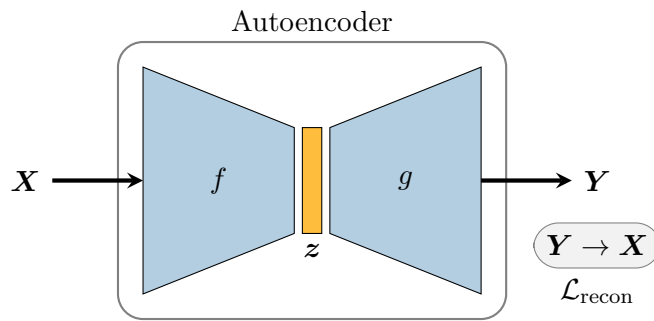


Figure 2.16: Autoencoder consisting of the encoder  $f$  and the decoder  $g$ . Training objective is depicted in the gray box.  $\mathbf{X}$ : input;  $\mathbf{Y}$ : reconstruction.

which relies on the reconstruction regime.

## 2.5.1 Reconstruction

### 2.5.1.1 Autoencoder

Autoencoders find various applications in generative models, outlier detection (Section 2.3.4.1) or in the field of representation learning, as presented in the following. The procedure of an autoencoder is divided into two parts. First, an input is converted into the latent representation. Second, it is aimed to reconstruct the original input value from the latent representation. The generated reconstruction is the output of the autoencoder. As the latent representation is usually of lower dimension a bottleneck is created. Therefore the task is to learn the characteristics of the input and compress them. Autoencoders find also application in the domain of image processing, as in [MMCS11] an autoencoder using CNNs is presented.

An autoencoder consists of two main components, the encoder  $f$  and the decoder  $g$ . The encoder generates the latent representation  $z \in \mathbb{R}^{N_z}$ , given the input as  $z = f(\mathbf{X})$ . The decoder takes the latent representation to generate the output as  $\mathbf{Y} = g(z)$ . As already mentioned, the latent representation is typically of lower dimension than the input, i. e.,  $N_z < N$ . The architecture of such an autoencoder is depicted in Figure 2.16.

In the standard setting, the objective is to optimize  $f$  and  $g$ , such that the output  $\mathbf{Y}$  best possible matches the input  $\mathbf{X}$ . For this, the reconstruction loss

$$\mathcal{L}_{\text{recon}} = \|\mathbf{X} - \mathbf{Y}\|_2^2 = \|\mathbf{X} - g(f(\mathbf{X}))\|_2^2 \quad (2.74)$$

is used.

The trained encoder can be used for the representation generation as required in this work. However, during the training of the autoencoder, the latent representations are only indirectly considered since the optimization is realized based on the reconstruction.



## 2.5.2 Metric Learning

Metric learning refers to a special field of representation learning. A network is trained such that the resulting representations mimic the objective metric as well as possible. Typically, the metric is represented by providing examples which are meant to be similar and examples which are dissimilar. Therefore, no exact definition of the metric/measure is required, simple extreme cases are sufficient. Typically, metric learning is applied in a supervised setting, for example knowing the class of instances and aiming to maximize the similarity of similar ones, while minimize similarities of dissimilar ones.

### 2.5.2.1 Contrastive Loss

One of the most basic representation learning strategies is contrastive learning [HCL06]. In contrast to the autoencoder, the loss is directly applied on the representations and hence, the network is optimized such that the representations follow certain definitions.

The objective of contrastive learning is to pull representations of similar instances close to each other, while pushing dissimilar ones away. During training, pairs of data points  $(\mathbf{X}_1, \mathbf{X}_2)$  are provided where they are either similar  $y = 0$  or dissimilar  $y = 1$ . The data points  $(\mathbf{X}_1, \mathbf{X}_2)$  are passed through the neural network  $f$  individually, hence the same network is used for both inputs. The network is therefore sometimes called Siamese or twin network. In order to achieve the former objective, the network  $f$  is trained such that the latent representations  $\mathbf{z}_1 = f(\mathbf{X}_1)$  and  $\mathbf{z}_2 = f(\mathbf{X}_2)$  fulfill the similarity constraint. With the squared distance  $d(\mathbf{z}_1, \mathbf{z}_2)^2$  between the pair of latent representations, the contrastive loss is defined as

$$\mathcal{L}_{\text{cont}} = (1 - y)d(\mathbf{z}_1, \mathbf{z}_2)^2 + y \max\{\alpha - d(\mathbf{z}_1, \mathbf{z}_2)^2, 0\}, \quad (2.75)$$

where  $\alpha$  is the margin. Hence, the dissimilar examples shall be at least  $\sqrt{\alpha}$  away from each other, while the positive examples shall be as close as possible to each other. In Figure 2.17, the learning objective and the network are illustrated.

### 2.5.2.2 Triplet Loss

The triplet loss [SKP15] is an extension of the contrastive loss. Instead of considering the positive and negative cases independently, they are coupled where both refer to the same *anchor* data point. The objective is to pull the similar example closer to the anchor than the negative example. Vice versa, the negative example shall be pushed away, at least farther than the positive example plus some margin. During training, for each step a data point (anchor), a similar data point (positive) and a dissimilar data point (negative) needs to be provided. Hence, the training is based on triplets of data points  $(\mathbf{X}_a, \mathbf{X}_p, \mathbf{X}_n)$ , an anchor  $\mathbf{X}_a$ , a positive example  $\mathbf{X}_p$  and a negative example  $\mathbf{X}_n$ .

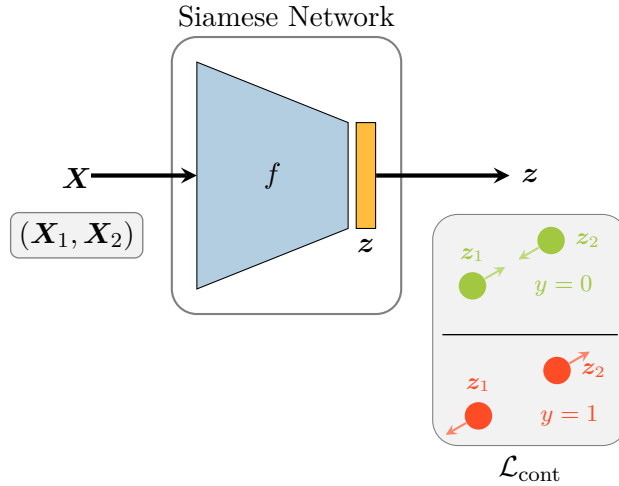


Figure 2.17: Contrastive learning with a Siamese network. The training is depicted in the gray boxes.  $(\mathbf{X}_1, \mathbf{X}_2)$ : input pair for training;  $(z_1, z_2)$ : latent representations for training pair (each input is passed separately);  $y = 0/1$ : pair is similar/dissimilar.

A neural network  $f$  is trained such that the resulting representations fulfill the triplet objective. For this, the data triplet is fed through the network sequentially, leading to  $z_a$ ,  $z_p$ , and  $z_n$ . Let the dissimilarity between the anchor representation and the positive representation be  $d_p = d(z_a, z_p)$  and  $d_n$  be the dissimilarity between the anchor and the negative representation, given a dissimilarity measure  $d$ . The requirement of a positive data point being more similar to the anchor than a negative can be expressed as

$$d_n \geq d_p + \alpha, \quad (2.76)$$

where  $\alpha$  is the margin, controlling how dissimilar the negative should be with respect to  $d_p$ . From this constraint, the triplet loss emerges as

$$\mathcal{L}_{\text{tri}} = \max\{d_p - d_n + \alpha, 0\}. \quad (2.77)$$

When using a similarity measure  $s$  instead of a dissimilarity measure, the constraint follows  $s_p \geq s_n + \alpha$  and hence, the loss is formulated as

$$\mathcal{L}_{\text{tri}} = \max\{s_n - s_p + \alpha, 0\}. \quad (2.78)$$

In practice, the  $\max\{\dots, 0\}$  is realized by a ReLU function. The triplet network as well as the training is depicted in Figure 2.18.

The training stability using triplet learning is strongly affected by the selected negative example. For this reason, various negative sample mining strategies are proposed. One typical choice is the so-called semi-hard sampling [SKP15], it provides high stability and fast learning. Given an anchor and a positive example, three different types

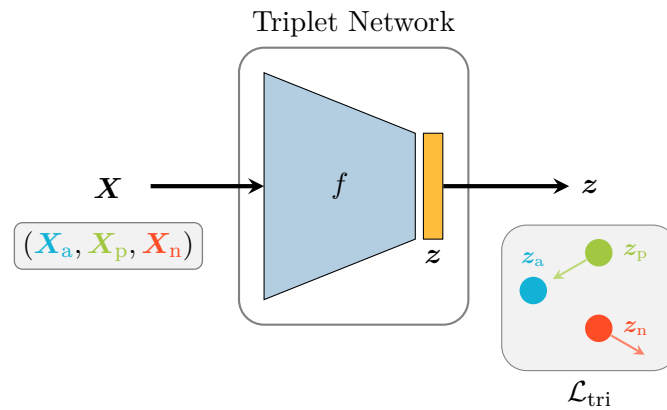


Figure 2.18: Triplet learning with a triplet network.  $(X_a, X_p, X_n)$ : training triplet consisting of an anchor, a positive example, and a negative example;  $z_{\dots}$ : the latent representations corresponding to the training triplet (each input is passed separately through  $f$ ).

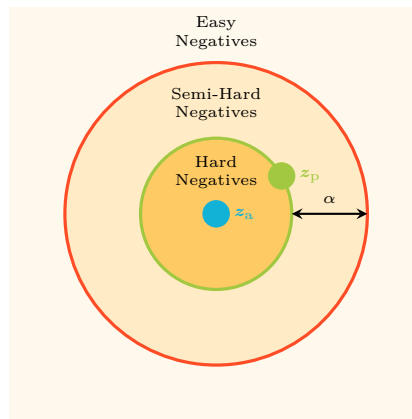


Figure 2.19: Negative types based on the anchor latent representation  $z_a$ , the positive latent representation  $z_p$  and the margin  $\alpha$ .<sup>6</sup>

of negatives can be distinguished: 1. hard negatives  $d_n < d_p$ , 2. semi-hard negatives  $d_n > d_p \wedge d_n < d_p + \alpha$  and 3. easy negatives  $d_n > d_p + \alpha$ . The three different types are illustrated in Figure 2.19. Considering the triplet loss Equation (2.77), easy negatives do not contribute to the learning. Hence, sampling easy negatives should be avoided for fast learning. Contrary, hard negatives can lead to an early stopping in a local minimum [SKP15] or even in so-called *mode collapse*, which will produce the same latent representation for any input. Therefore, hard negatives should be avoided to ensure a stable training. In conclusion, semi-hard negatives provide fast and stable training and are a typical choice when sampling negatives.

In summary, triplet learning is used to embed a given similarity knowledge into the representation space. This way, complex inputs can be projected into representations which follow the required similarity. The representation space constructed with triplet

<sup>6</sup>Inspired from [https://omindrot.github.io/assets/triplet\\_loss/triplets.png](https://omindrot.github.io/assets/triplet_loss/triplets.png).

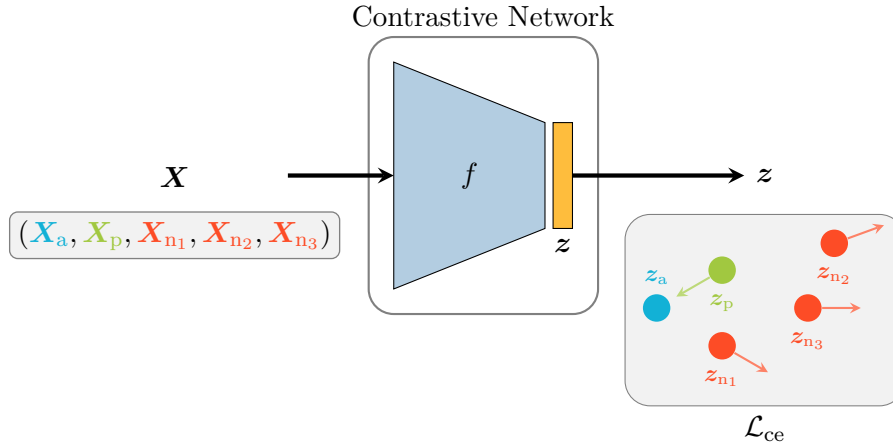


Figure 2.20: Cross entropy-based contrastive learning for  $K = 3$  negative instances.  $(\mathbf{X}_a, \dots, \mathbf{X}_{n_3})$ : training input tuple;  $z_{\dots}$ : latent representation corresponding to input training tuple (each input is passed separately).

learning is successfully used for tasks like face re-identification [SKP15].

### 2.5.2.3 Cross Entropy-based Contrastive Loss

Using multiple negative instances is realized through cross entropy-based contrastive loss, which is widely used in representation learning [OLV18, WXYL18, Soh16, HFW<sup>+</sup>20, CKNH20]. As in triplet learning, a network is trained to optimize the contrastive loss.

The cross entropy-based contrastive loss follows the same intuition as the triplet loss, hence, similar instances shall be pulled close to each other in the representation space while dissimilar ones shall be pushed far from each other. Given  $K$  negative instances, one positive instance, and an anchor instance, the cross entropy-based contrastive loss is defined as

$$\mathcal{L}_{ce} = -\log \left( \frac{\exp(\frac{1}{\tau} s_p)}{\exp(\frac{1}{\tau} s_p) + \sum_{k=1}^K \exp(\frac{1}{\tau} s_{n_k})} \right), \quad (2.79)$$

with  $\tau$  the temperature hyperparameter. The training of a network  $f$  using  $\mathcal{L}_{ce}$  is illustrated in Figure 2.20. The similarity  $s_{\dots}$  follows the same definition as for the triplet learning.

Cross entropy-based contrastive learning has a strong relation to triplet learning. In the following, it is shown that the special case of the cross entropy-based contrastive loss with only a single negative instance approximates the triplet loss. Moreover, the general case of the cross entropy-based contrastive loss approximates triplet loss combined with hardest sampling. The relation to the triplet loss can aid the interpretation of the cross entropy-based contrastive learning. Moreover, under some constraints and assumptions, the contrastive loss  $\mathcal{L}_{ce}$  maximizes a lower bound on the mutual information [OLV18] (see

Appendix A.1).

**Single Negative Cross Entropy-based Contrastive Loss as Approximated Triplet Loss** It can be shown, that the special case of  $\mathcal{L}_{ce}$  with only one negative instance is a smooth approximation of the triplet loss  $\mathcal{L}_{tri}$  [LBCP<sup>+</sup>20]. Using the triplet loss from Equation (2.78) and the softplus function  $\text{softplus}(\dots) = \log(1 + \exp(\dots))$ , which is a smooth approximation of  $\max\{\dots, 0\}$ , the following holds:

$$\mathcal{L}_{tri} = \max\{s_n - s_p + \alpha, 0\} \quad (2.80)$$

$$\mathcal{L}_{tri} \approx \text{softplus}(s_n - s_p + \alpha) \quad (2.81)$$

$$\mathcal{L}_{tri} \approx \log(1 + \exp(s_n - s_p + \alpha)) \quad (2.82)$$

$$\mathcal{L}_{tri} \approx \log(1 + \exp(s_n - s_p + \alpha)) - \log(1) \quad (2.83)$$

$$\mathcal{L}_{tri} \approx -\log\left(\frac{1}{1 + \exp(s_n - s_p + \alpha)}\right) \quad (2.84)$$

$$\mathcal{L}_{tri} \approx -\log\left(\frac{\exp(s_p)}{\exp(s_p) + \exp(s_n + \alpha)}\right). \quad (2.85)$$

When using a margin of zero  $\alpha = 0$ , and by introducing the parameter  $\beta$ , which controls how well the function is approximated, the approximated triplet loss can be written as

$$\mathcal{L}_{tri} \approx -\frac{1}{\beta} \log\left(\frac{\exp(\beta s_p)}{\exp(\beta s_p) + \exp(\beta s_n)}\right). \quad (2.86)$$

Hence, by selecting  $\beta = \frac{1}{\tau}$  the approximated triplet loss is a linear-scaled version of  $\mathcal{L}_{ce}$  for one negative instance. The influence of the parameter  $\beta$  for the approximation is depicted in Figure 2.21. A typical choice is  $\tau = 0.1$ , which leads to  $\beta = 10$ , hence for this setting, the single negative  $\mathcal{L}_{ce}$  approximates the  $\mathcal{L}_{tri}$  closely. Moreover, the smaller  $\tau$  is selected the stricter the single negative  $\mathcal{L}_{ce}$  follows  $\mathcal{L}_{tri}$ . With  $s_n - s_p$  substituted by  $x$ , the softplus approximation of  $\max\{x, 0\}$  can be seen from the limit value analysis, since for  $x > 0$

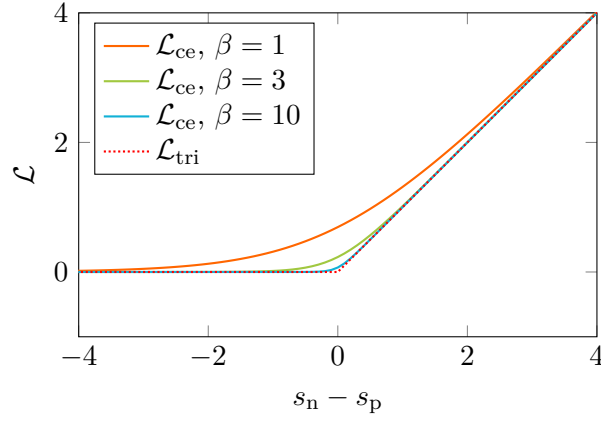
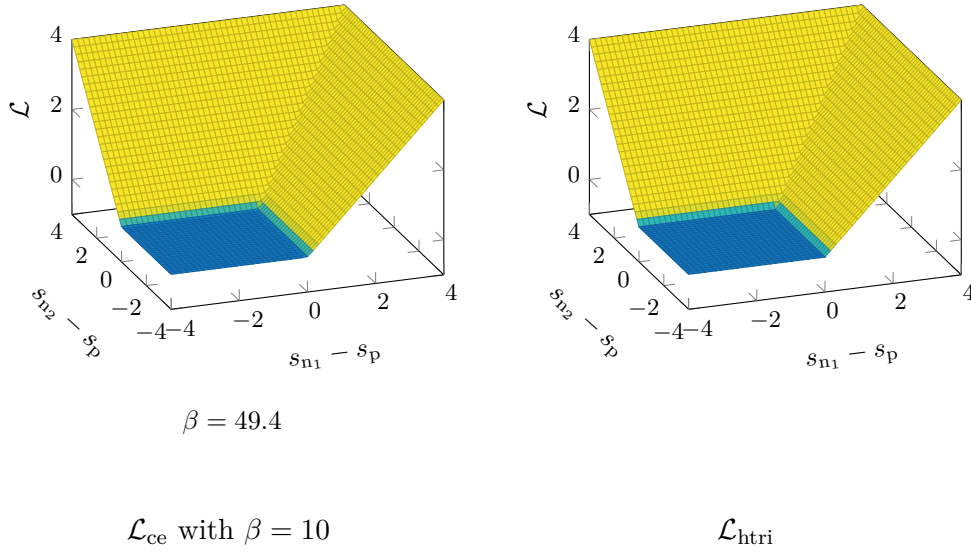
$$\lim_{\beta \rightarrow \infty} \frac{1}{\beta} \log(1 + \exp(\beta x)) \quad (2.87)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} [\log(1 + \exp(-\beta x)) + \log(\exp(\beta x))] \quad (2.88)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} [\log(1 + 0) + \beta x] \quad (2.89)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} \beta x \quad (2.90)$$

$$= x \quad (2.91)$$


 Figure 2.21: Approximation of the triplet loss  $\mathcal{L}_{\text{tri}}$  by  $\mathcal{L}_{\text{ce}}$  with different values for  $\beta$ .

 Figure 2.22: Approximation of the hardest sampling triplet loss  $\mathcal{L}_{\text{htri}}$  by  $\mathcal{L}_{\text{ce}}$ .

holds. For negative values  $x < 0$

$$\lim_{\beta \rightarrow \infty} \frac{1}{\beta} \log(1 + \exp(\beta x)) \quad (2.92)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} [\log(\exp(\beta x)(\exp(\beta x) + 1)) + \log(\exp(-\beta x))] \quad (2.93)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} [\log(0(0 + 1)) + \log(\exp(-\beta x))] \quad (2.94)$$

$$= 0 \quad (2.95)$$

holds, since  $\log(0(0 + 1)) \rightarrow -\infty$  and  $\log(\exp(-\beta x)) \rightarrow \infty$  because  $x > 0$ . For  $x = 0$   $\lim_{\beta \rightarrow \infty} \frac{1}{\beta} \log(1 + \exp(\beta x)) = 0$  holds.

**Cross Entropy-based Contrastive Loss as Approximated Triplet Loss with Hardest Sampling** In the standard case, multiple negative instances are used in the cross entropy-based contrastive loss. In the following it is shown, that this case can be seen as a smooth approximation of the triplet loss with hardest negative sampling. Therefore, using multiple negatives in the triplet loss can be realized by sampling the hardest negative from multiple negatives. Hardest sampling from  $K$  negatives can be formulated as

$$\mathcal{L}_{\text{htri}} = \max\left\{\max\left\{\{s_{n_k} - s_p\}_{k=1}^K\right\}, 0\right\}. \quad (2.96)$$

The  $\mathcal{L}_{\text{htri}}$  can be approximated by the temperature-scaled cross entropy-based contrastive loss  $\mathcal{L}_{\text{ce}}$ , which leads to

$$\mathcal{L}_{\text{htri}} \approx -\frac{1}{\beta} \log\left(\frac{\exp(\beta s_p)}{\exp(\beta s_p) + \sum_{k=1}^K \exp(\beta s_{n_k})}\right), \quad (2.97)$$

where the approximation becomes stricter the smaller  $\beta$ . In Figure 2.22, the special case of only two negative examples is depicted. As for the single instance case, the approximation for  $\tau = 0.1$  or  $\beta = 10$  is already close. The approximation of the hardest sampling can be shown as follows. First, the temperature scaled contrastive loss can be written as

$$\text{softplus}_{\beta}(\text{LSE}_{\beta}(s_{n_1}, \dots, s_{n_K}) - s_p) \quad (2.98)$$

$$= \frac{1}{\beta} \log\left[1 + \exp\left(\beta \frac{1}{\beta} \log\left[\sum_{k=1}^K \exp(\beta s_{n_k})\right] - \beta s_p\right)\right] \quad (2.99)$$

$$= \frac{1}{\beta} \log\left[1 + \sum_{k=1}^K \exp(\beta s_{n_k}) \exp(-\beta s_p)\right] \quad (2.100)$$

$$= -\frac{1}{\beta} \log\left[\frac{\exp(\beta s_p)}{\exp(\beta s_p) + \sum_{k=1}^K \exp(\beta s_{n_k})}\right], \quad (2.101)$$

where  $\text{LSE}_{\beta}$  is the  $\beta$  parameterized log-sum-exp function  $(\frac{1}{\beta} \log[\sum_{k=1}^K \exp(\beta x)])$ , which is a smooth approximation of the maximum function. Hence,  $\text{LSE}_{\beta}$  will return approximately the highest  $s_{n_k}$  which is equal to hardest sampling. Like for the softplus, the higher  $\beta$  the stricter the approximation gets. Assume the  $K$   $s_{n_k}$  to be sorted such that  $s_{n_1} \geq s_{n_2} \geq \dots \geq s_{n_K}$  then the approximation of the maximum function for  $\beta \rightarrow \infty$  with  $\text{LSE}_{\beta}$  can

be seen from

$$\lim_{\beta \rightarrow \infty} \frac{1}{\beta} \log \left( \sum_{k=1}^K \exp(\beta s_{n_k}) \right) \quad (2.102)$$

$$= \lim_{\beta \rightarrow \infty} \frac{1}{\beta} \left[ \log(\exp(\beta s_{n_1})) + \log \left( 1 + \sum_{k=2}^K \exp(\beta (s_{n_k} - s_{n_1})) \right) \right] \quad (2.103)$$

$$= \lim_{\beta \rightarrow \infty} s_{n_1} + \frac{1}{\beta} \log \left( 1 + \sum_{k=2}^K \exp(\beta (s_{n_k} - s_{n_1})) \right) \quad (2.104)$$

$$= \lim_{\beta \rightarrow \infty} s_{n_1} + \frac{1}{\beta} \log(1 + 0) \quad (2.105)$$

$$= s_{n_1}, \quad (2.106)$$

where  $\beta(s_{n_k} - s_{n_1}) = -\infty$ , since  $s_{n_1} \geq s_{n_k}$ . Hence, the  $\text{LSE}_\beta$  approaches the max function for  $\beta \rightarrow \infty$ .

### 2.5.3 Self-Supervised Learning

Self-supervised learning represents another branch of representation learning methods. While being closely related to metric learning, the key difference lies in the fact that no similarity definition is used for supervision directly. Instead of taking two positive/negative examples which are sampled based on the similarity definition, in self-supervised learning the positive and negative pairs are generated in an automated fashion directly from the input data. From an input sample typically two random deviations are created by a set of selected transformations. These two new views of the input sample are treated as the positive pair, since they origin from the same input sample. Therefore, in self-supervised learning the selected transformations define the similarity. The transformations are typically called augmentations.

In the following subsections, some recent self-supervised learning methods for image data are summarized in more detail [CMM<sup>+</sup>20, ZJM<sup>+</sup>21, BPL21]. Other important self-supervised methods, not being cover in the following subsections are briefly discussed in the remainder of this section.

In *Simple framework for Contrastive Learning of visual Representations* (SimCLR) [CKNH20], the positive examples are generated with a set of strong augmentations from an input. The negative examples are the remaining augmented images in the batch. Both augmented views are processed with the same network. In *Bootstrap Your Own Latent* (BYOL) [GSA<sup>+</sup>20], only positive pairs are used, hence BYOL is a non-contrastive method. Unlike SimCLR, the two networks are different, the *online* network and the *target* network. The *online* network is optimized by the loss while the *target* network is updated using a moving exponential average of the weights of the *online* network. Hence, no gradients flow through the target network, this is typically called as stop-gradient. Alternatively,



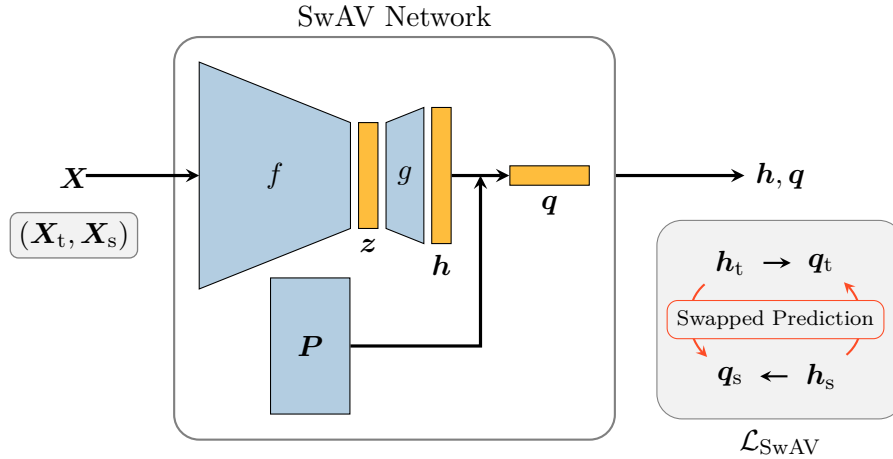


Figure 2.23: SwAV learning, using the encoder  $f$ , projector  $g$ , and the prototype matrix  $P$ .  $h$ : projected representation;  $q$ : assignment code of projected representation to the prototypes;  $(\mathbf{X}_t, \mathbf{X}_s)$ : training input pair;  $h_{\dots}$ : projected representation corresponding to the training input;  $q_{\dots}$ : assignment code corresponding to the training input.

exploring Simple Siamese representation learning (SimSiam) [CH20] relies on the same setting as BYOL, but the *target* network is selected to be similar to the *online* network.

### 2.5.3.1 SwAV

In [CMM<sup>+</sup>20], the self-supervised learning method *Swapping Assignments between Views* (SwAV) is presented. As many other self-supervised methods, SwAV relies only on positive examples, which are generated by augmentation. SwAV realizes the representation learning through a swapped prototype assignment problem. This way, a network and a codebook (prototypes) are learned, where the network is realizing the projection to the representation space.

Let  $\mathbf{X}_t$  and  $\mathbf{X}_s$  be two positive input examples. Hence, those two examples can be considered as somehow similar (e. g., different crops of the same image). First, each input is fed through the network  $f$  leading to the latent representations  $z_t = f(\mathbf{X}_t)$  and  $z_s = f(\mathbf{X}_s)$ . For further steps, the latent representations are projected by the network  $g$  and normalized as  $h = \frac{g(z)}{\|g(z)\|_2}$ . Then, for each projected representation  $h \in \mathbb{R}^{N_h}$ , the so-called assignment code  $q \in \mathbb{R}^K$  is determined. The code indicates the assignment strengths of the projected representation to the  $K$  prototypes  $\{p_1, \dots, p_K\}$ , with  $\|q\|_1 = 1$ . The prototypes are collected in the prototype matrix  $P \in \mathbb{R}^{K \times N_h}$ . The training in SwAV is realized by predicting the code given the other projected representation, i. e., predicting  $q_s$  from  $h_t$  and vice versa predicting  $q_t$  from  $h_s$ . In the following, the code calculation as well as the swapped prediction learning are explained. In Figure 2.23, the overall method is depicted.

**Swapped Prediction Learning** Training the networks  $f$  and  $g$  as well as the prototypes  $\mathbf{P}$  is realized through the swapped assignment prediction. Given two positive instances  $\mathbf{X}_t$  and  $\mathbf{X}_s$ , the aim is to predict the assignment code for the other instance. Hence, predicting  $\hat{\mathbf{q}}_t$  from  $\mathbf{h}_s$  and predicting  $\hat{\mathbf{q}}_s$  from  $\mathbf{h}_t$ . The target assignment codes  $\mathbf{q}_t$  and  $\mathbf{q}_s$  are determined by the Sinkhorn-Knopp algorithm, i. e.,  $\mathbf{q}_t$  from  $\mathbf{h}_t$  and  $\mathbf{q}_s$  from  $\mathbf{h}_s$  and are fixed throughout a swapped prediction learning step. The swapped prediction loss is formulated as

$$\mathcal{L}_{\text{SwAV}} = l(\mathbf{h}_t, \mathbf{q}_s) + l(\mathbf{h}_s, \mathbf{q}_t) \quad (2.107)$$

$$l(\mathbf{h}_t, \mathbf{q}_s) = - \sum_k q_{s,k} \log \left( \frac{\exp\left(\frac{1}{\tau} \mathbf{h}_t^T \mathbf{p}_k\right)}{\sum_{k' \neq k} \exp\left(\frac{1}{\tau} \mathbf{h}_t^T \mathbf{p}_{k'}\right)} \right). \quad (2.108)$$

The loss in Equation (2.107) becomes minimal, if the predicted assignment of one instance matches the code of the other instance. The predicted assignment is realized through the softmax in Equation (2.108). Assuming a one-hot code, the loss will force the positive instances and the selected prototype to become similar while making the positive instances dissimilar from all other prototypes. When compared to contrastive learning, the matching prototype can be interpreted as the positive or contracting part, while the remaining prototypes can be seen as the negatives or contrastive part. Given that the prototypes form a compressed version of the training data, the contrastive part represents the compressed dataset except the most similar prototype.

**Code Calculation** Given a batch of projected latent representations  $\mathbf{H} \in \mathbb{R}^{B \times N_h}$ , it is determined to which prototype from  $\mathbf{P}$  each of the latent representations should be assigned. This leads to the corresponding assignment codes  $\mathbf{Q}$ . The online calculation of the codes, using the fixed  $\mathbf{P}$  and  $\mathbf{H}$ , is based on considering the assignment as entropic regularized optimal transport problem [CMM+20]

$$\begin{aligned} & \max_{\mathbf{Q}} \left\{ \text{Tr}(\mathbf{Q}^T \mathbf{P}^T \mathbf{H}) - \varepsilon \sum_{ij} Q_{ij} (\log(Q_{ij}) - 1) \right\} \\ & \text{s.t. } \mathbf{Q} \in \mathbb{R}_+^{K \times B} \\ & \quad \mathbf{Q} \mathbf{1}_B = \frac{1}{K} \mathbf{1}_K \\ & \quad \mathbf{Q}^T \mathbf{1}_K = \frac{1}{B} \mathbf{1}_B. \end{aligned} \quad (2.109)$$

The first term describes the optimal transport, while the second term of the problem is the entropic regularizer. Hence, the assignment codes  $\mathbf{Q}$  are optimized to maximize the similarity between the projected representations  $\mathbf{H}$  and the prototypes  $\mathbf{P}$ .  $\varepsilon$  controls the smoothness of the mapping. With the constraints, it is ensured that on average  $\frac{B}{K}$  repre-

representations are assigned to each prototype in a batch. Therefore, an optimal assignment matrix  $Q$  assigns the representations to the most similar prototypes with soft codes, due to the entropic regularization.

The problem (2.109) is solved by [Cut13]

$$Q = \text{diag}(\mathbf{u}) \exp\left(\frac{P^T H}{\varepsilon}\right) \text{diag}(\mathbf{v}). \quad (2.110)$$

The vectors  $\mathbf{u} \in \mathbb{R}^K$  and  $\mathbf{v} \in \mathbb{R}^B$  and hence, the assignment  $Q$  of a batch of projected representations is being determined by the iterative Sinkhorn-Knopp algorithm [Cut13]. However, for small batch sizes, additionally, previous batches are considered through a queue, in order to increase the code accuracy.

As presented in [CMM+20], the two positive instances' case can be extended to the multi-positive instance case. For this, multi-crop strategies are introduced. The loss for the multi-positive instances case is defined as

$$\mathcal{L}_{\text{mSwAV}} = \sum_{i=1}^{K_P} \sum_{j=1}^{K_P} \mathbf{1}_{i \neq j} l(\mathbf{h}_{t_j}, \mathbf{q}_{t_i}), \quad (2.111)$$

with  $K_P$  the number of positive instances. It has to be noted that this loss is a slightly modified version to [CMS+20], since there  $i$  is restricted to select from only two positive instances. However, here the more general case is stated.

### 2.5.3.2 Barlow Twins

The self-supervised learning method Barlow Twins is presented in [ZJM+21]. The method is named after the neuroscientist H. Barlow, who presented the redundancy-reduction principle. The authors of [ZJM+21] emphasize, that their method applies redundancy-reduction to realize the self-supervised learning. This way, the problem of mode collapse is prevented. Other recent methods avoid the mode collapse by stop-gradient function (SimSiam, BYOL), non-differentiable clustering (SwAV) or negative examples (SimCLR). Barlow Twins achieves comparable results to other methods, while being less sensitive to the batch size.

The setup used in Barlow Twins is very similar to other self-supervised learning architectures. A pair of similar examples is processed by the same network to produce the representations. Here, two similar examples are created by augmentation from the original input. More details about typical augmentations can be found in Section 2.5.3.4. Like other methods, Barlow Twins aims to make the representations of the two positive examples as similar as possible. As depicted in Figure 2.24, given  $\{\mathbf{X}_t, \mathbf{X}_s\}_B$  a batch of  $B$  positive pairs, the respective representations  $\{\mathbf{h}_t, \mathbf{h}_s\}_B$  are generated by  $\mathbf{h} = g(f(\mathbf{X}))$ ,

<sup>7</sup>Inspired and adopted from [ZJM+21].

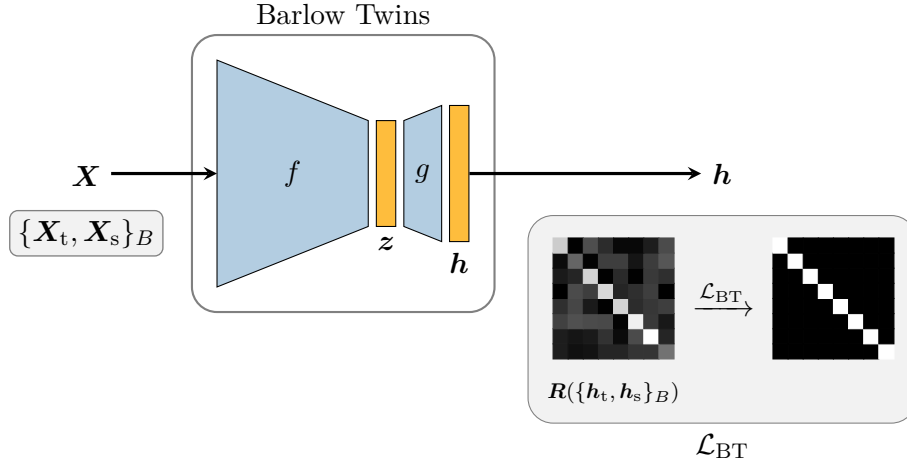


Figure 2.24: Barlow Twins learning<sup>7</sup>.  $\{\mathbf{X}_t, \mathbf{X}_s\}_B$ : training batch of  $B$  pairs;  $\mathbf{R}(\{\mathbf{h}_t, \mathbf{h}_s\}_B)$ : batch-wise cross-correlation matrix for the projected representations.

with  $\mathbf{h} \in \mathbb{R}^{N_h}$ . The redundancy reduction is realized by the loss function

$$\mathcal{L}_{\text{BT}} = \sum_i (1 - \mathbf{R}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} \mathbf{R}_{ij}^2. \quad (2.112)$$

$\mathbf{R}$  is the batch-wise cross-correlation matrix for  $\{\mathbf{h}_t, \mathbf{h}_s\}_B$ , hence  $\mathbf{R} \in \mathbb{R}^{N_h \times N_h}$ . Given that  $\mathbf{h}_t$  and  $\mathbf{h}_s$  are batch-normalized in Barlow Twins, the cross-correlation calculation simplifies to

$$\mathbf{R}_{ij} = \frac{\sum_{b=1}^B h_{t,i}^{(b)} h_{s,j}^{(b)}}{\sqrt{\sum_{b=1}^B (h_{t,i}^{(b)})^2} \sqrt{\sum_{b=1}^B (h_{s,j}^{(b)})^2}}. \quad (2.113)$$

The objective underlying the Barlow Twins loss is to make the cross correlation matrix as close possible to the identity matrix, this objective is also visualized in Figure 2.24. The first part of the loss represents the invariance term, it is responsible for making representatives of positive pairs as similar as possible. If that is realized for the complete batch, the diagonal of the correlation matrix will be close to one. The second term is realizing the redundancy reduction by enforcing the feature dimensions to decorrelate. Therefore, the second part is aiming to make all off-diagonal elements in  $\mathbf{R}$  close to zero. The authors argue that the redundancy reduction term is responsible for mode collapse prevention. The hyperparameter  $\lambda$  controls the trade-off between the first and second term.

### 2.5.3.3 VICReg

In [BPL21] *Variance-Invariance-Covariance Regularization* (VICReg), a self-supervised learning method is introduced. As already defined by the name, the method aims to solve

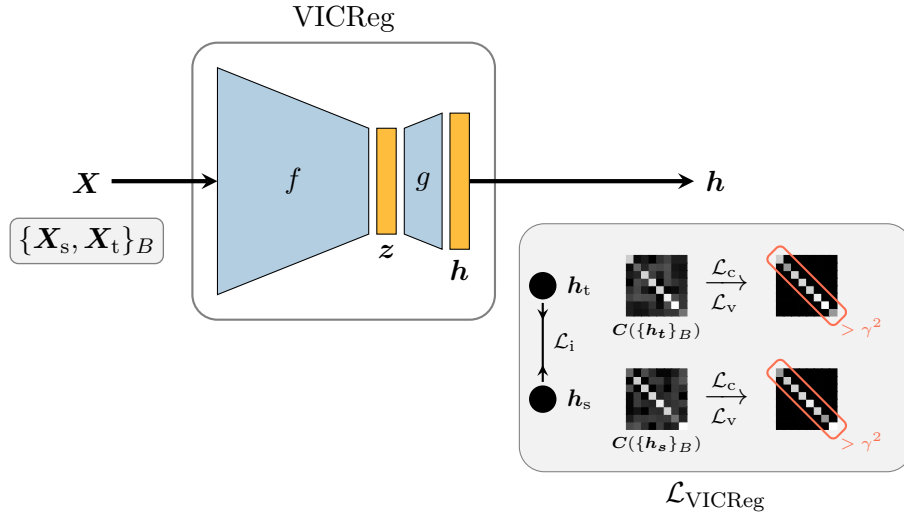


Figure 2.25: VICReg learning.  $\{\mathbf{X}_t, \mathbf{X}_s\}_B$ : training batch of  $B$  pairs;  $C(\{h_{\dots}\}_B)$ : batch-wise covariance matrix for projected representations;  $\mathcal{L}_{\dots}$ : individual loss terms of VICReg.

the representation task by regularizing the invariance, variance, and covariance. Like the methods discussed up to now, it is based on two positive examples generated through augmentations. Even though in the scope of this work VICReg is only used and presented in a typical twin architecture with identical branches, it is not restricted to that setting.

Like Barlow Twins, VICReg essentially introduces a new loss function for the self-supervised training regime. The setup is very similar to other common frameworks using pairs of positive examples. Again,  $\{\mathbf{X}_t, \mathbf{X}_s\}_B$  is a batch of  $B$  positive pairs and the representations  $\{h_t, h_s\}_B$  are generated through  $h = g(f(\mathbf{X}))$ , with  $h \in \mathbb{R}^{N_h}$ . The VICReg loss is split into three terms: the invariance term, the variance term, and the covariance term. Up next, each term is briefly discussed.

The invariance term is ensuring a high similarity between the representations of a pair. In VICReg, this is realized through the Euclidean distance. Hence, the loss aims to make the distance between two positive representations as small as possible. This is also depicted in Figure 2.25 by the contracting force between  $h_t$  and  $h_s$ . The invariance term is defined as

$$\mathcal{L}_i(\{h_s, h_t\}_B) = \frac{1}{B} \sum_{b=1}^B \|h_s^{(b)} - h_t^{(b)}\|_2^2. \quad (2.114)$$

The variance term aims to achieve high variances for each dimension of the representation across the batch. Hence, this term's objective is to prevent the mode collapse. The variance is determined for a batch of one instance independently, i.e.,  $\{h_s\}_B$  or  $\{h_t\}_B$ . The loss for the variance term is

$$\mathcal{L}_v(\{h\}_B) = \frac{1}{N_h} \sum_{i=1}^{N_h} \max(0, \gamma - \sqrt{\sigma_{h_i} + \epsilon}), \quad (2.115)$$

with  $\sigma_{h_i}$ , the empirical variance in dimension  $i$  for one batch and  $\epsilon$  a small scalar to prevent numerical instability. The loss forces the standard deviation to be at least  $\gamma$ . Therefore, in the covariance matrix, the diagonal elements should be at least  $\gamma^2$ , which is illustrated in Figure 2.25.

The covariance term is inspired from Barlow Twins, as VICReg aims reduce the covariance between the dimensions for one batch and hence make it less redundant. The covariance matrix is determined based only on a batch of one instance  $\{\mathbf{h}\}_B$  as

$$\mathbf{C}(\{\mathbf{h}\}_B) = \frac{1}{B-1} \sum_{b=1}^B (\mathbf{h}^{(b)} - \mu_{\mathbf{h}}) (\mathbf{h}^{(b)} - \mu_{\mathbf{h}})^{\text{T}}, \quad (2.116)$$

where  $\mu_{\mathbf{h}}$  is the mean representation vector within a batch. The covariance loss term is determined as

$$\mathcal{L}_c(\{\mathbf{h}\}_B) = \frac{1}{N_h} \sum_{i=1}^{N_h} \sum_{j \neq i} \mathbf{C}(\{\mathbf{h}\}_B)_{ij}^2. \quad (2.117)$$

The objective is to minimize the off-diagonal elements of the covariance matrix, what is visualized in Figure 2.25 as well.

The complete VICReg loss is defined as

$$\mathcal{L}_{\text{VICReg}} = \alpha_v [\mathcal{L}_v(\{\mathbf{h}_s\}_B) + \mathcal{L}_v(\{\mathbf{h}_t\}_B)] + \alpha_i \mathcal{L}_i(\{\mathbf{h}_s, \mathbf{h}_t\}_B) + \alpha_c [\mathcal{L}_c(\{\mathbf{h}_s\}_B) + \mathcal{L}_c(\{\mathbf{h}_t\}_B)], \quad (2.118)$$

with  $\alpha_i$ ,  $\alpha_v$ , and  $\alpha_c$  being the hyperparameters controlling the balance between the three terms. In summary, the objective of the VICReg loss is threefold:

1. make two positive representations as similar as possible (invariance),
2. maintain a high variance across each feature dimension and preventing mode collapse (variance), and
3. reduce redundancy between the feature dimensions (covariance).

#### 2.5.3.4 Augmentations

Methods like Barlow Twins, VICReg or SwAV make use of augmentations to generate two somehow similar examples from one input sample. The summarized self-supervised methods are all applied on image data. The augmentation functions, typically applied in these computer vision methods, are summarized here.

**Crop & Resize** The image is cropped at a random position with random size. The output is resized to the size of the input image. The size is typically sampled from 0.08 to 1.0 of the original size and the aspect ratio being sampled from 3/4 to 4/3 of the original aspect ratio [CKNH20].



Figure 2.26: Typical computer vision augmentations.

**Horizontal Flip** The image is flipped horizontally.

**Color Jitter** The brightness, contrast, saturation, and hue of an image are randomly jittered. For this, the following intervals are used for sampling the jitter strengths: brightness  $[0.2, 1.8]$ , contrast  $[0.2, 1.8]$ , saturation  $[0.2, 1.8]$ , and hue  $[-0.2, +0.2]$  [CKNH20].

**Grayscale** The image is converted to grayscale.

**Gaussian Blur** This augmentation method blurs the image by a Gaussian kernel. The kernel size is typically selected to be 10% of the image size and the standard deviation is sampled from  $[0.1, 2.0]$  [CKNH20].

**Solarization** Simulating the effect of extreme overexposure is realized by this augmentation. For this all pixels above a selected threshold are inverted. Typically, this threshold is selected to be 0.5 [GSA<sup>+</sup>20].

In many methods [ZJM<sup>+</sup>21, BPL21, GSA<sup>+</sup>20, CKNH20] random crop and resize is always applied. The other augmentations are only applied with some probability. In Figure 2.26 the effects of the various augmentations are shown. Expect for the Gaussian blur, the values as stated above are used. For the Gaussian blur, the values are changed to make the effect visible.





## Chapter 3

# Identifying Relevant Traffic Scenarios

This chapter presents novel approaches to identify relevant traffic scenarios. State-of-the-art approaches for the identification of relevant traffic scenarios are summarized as well. Furthermore, a statistical motivation for the necessity of relevant traffic scenario identification is provided. As already stated in Chapter 1, in this work representative, unknown, and critical scenarios are considered to be relevant traffic scenarios. The overall framework of this work is shown in Figure 3.1, where the representation learning component is not illustrated since it is not used by the methods introduced in this chapter.

Validating an Autonomous Vehicle (AV) by proofing that it performs statically safer than a human driver is infeasible through pure unconditioned test drives. It would require the AV to drive for millions to billions of kilometers. One approach to tackle this problem is the so-called scenario-based testing. The assumption is that testing only relevant traffic

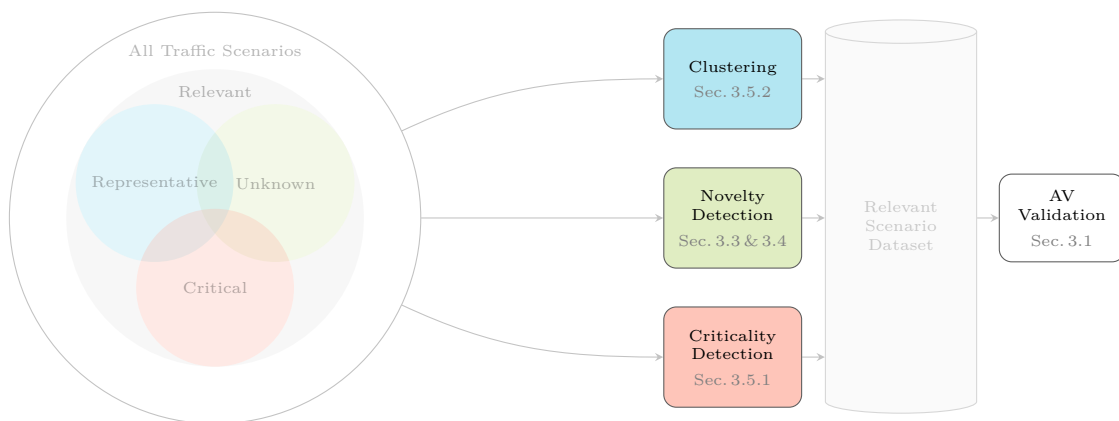


Figure 3.1: Methodological overview of this work and its embedding in the validation process of AVs. Novel methods for the highlighted components are introduced in the corresponding sections. Also, a statistical motivation for the validation of AV through scenario-based testing is provided in Section 3.1

scenarios is sufficient to prove an AV's safety. One prominent project relying on the scenario-based testing assumption is PEGASUS [PEG]. The key question in scenario-based testing is, how to identify such relevant traffic scenarios. A typical approach is to define traffic scenarios from domain knowledge. Another approach is to select relevant traffic scenarios from real-world data. Novel traffic scenarios, which are not covered by the validation scenarios, can occur after the deployment of AVs. In order to improve the safety of AVs, novel scenarios should be included into future validation processes, hence it should be possible to detect if a traffic scenario is unknown. This is also required by the ISO 21448 *Road vehicle – Safety of the intended functionality* (SOTIF) [ISO21]. The deployed AV shall be monitored to enable continuous validation. This can be partially realized by detecting representative, unknown, and critical scenarios. Hence, this chapter is focused on methods to detect representative, unknown, and critical scenarios, as they form a key role for the validation and monitoring of deployed AVs. The approaches discussed and presented in this chapter are used to find relevant traffic scenarios in real-world data. The embedding of the presented approaches into the framework of this work is shown in Figure 3.1.

The first section shows that it is infeasible to perform simple randomized driving for the validation of AVs. The second part discusses state-of-the-art approaches to find relevant traffic scenarios from real-world data. In the third part, a novel outlier method with the application of finding novel traffic infrastructures [WFBU20] is introduced and discussed. Furthermore, alternative data-driven approaches for detecting novel traffic scenarios are discussed and evaluated in the fourth section, using the example of road infrastructure images. In the fifth part of this chapter, methods relying only on domain knowledge to detect relevant traffic scenarios are introduced, where traffic scenarios are either detected due to their criticality or due to their category.

The experimental results show that, none of the presented approaches is capable of delivering desirable results. One option would be to change the applied clustering or novelty detection methods. The other option is to find representations of traffic scenarios that are better suited for clustering and novelty detection methods. The latter is what the methods presented in scope of this work are aiming to solve. As will be discussed, also the domain knowledge-based approaches are not sufficient to solve the issue of detecting relevant traffic scenarios. Combining data-driven approaches with domain knowledge based methods can be a promising solution to the mentioned shortcomings. Such a combination is demonstrated in Chapter 4, and it is shown that the use of domain knowledge to guide representation learning is beneficial for the tasks of clustering and novelty detection.

## 3.1 Statistical Motivation

The development of appropriate testing and validation strategies for AVs is necessary to examine their reliability in a feasible manner. In this section, naive forward validation approaches, as presented in [KP16], are analyzed and explained. The underlying question: *Is it possible to demonstrate the reliability of an AV, by just performing a certain amount of unconditioned tests drives?* For this purpose, different statistical methods can be used. In the following, the statistical methods used in [KP16] are explained, derived, and applied to German accident statistics. The required statistical theory is summarized in Appendix A.2.

### 3.1.1 Statistical Methods

This section summarizes the statistical methods, which can be used to model the testing of AVs through driving.

#### 3.1.1.1 Success Run Method

In terms of reliability theory, the success run method is used to determine the number of successful experiments which is required to prove a certain reliability with a certain confidence. In the following, two different derivations of the success run are shown.

**Binomial Distribution Derivation** The investigated experiment yields  $x \in \{0, 1\}$ , hence it can be modeled by a Bernoulli distribution. Here,  $0 \hat{=}$  success and  $1 \hat{=}$  no success holds. Given the reliability  $R$  and the failure rate  $F = 1 - R$ , the binomial distribution over the number of failures  $m$  can be written as [OK12]

$$p(m|n, R) = \binom{n}{m} F^m (1 - F)^{n-m}. \quad (3.1)$$

The *Cumulative Distribution Function* (CDF) of this *Probability Mass Function* (PMF) using  $F = 1 - R$  leads to

$$P(m \leq k) = 1 - \sum_{i=0}^k \binom{n}{i} R^{n-i} (1 - R)^i. \quad (3.2)$$

The probability of observing no failure ( $k = 0$ ) in  $n$  trials with a reliability of  $R$  is

$$P(m \leq 0) = C = 1 - R^n, \quad (3.3)$$

where  $C$  is the confidence level.

The number of required trials to prove a certain reliability  $R$  with a confidence of  $C$  is

$$n = \frac{\log(1 - C)}{\log(R)}. \quad (3.4)$$

Therefore,  $n$  successful trials are required to prove that the test object (AV in the scope of this work) has a reliability of  $R$  with a confidence of  $C$ . As can be seen, a reliability of 1, can not be proven, since it would require  $\infty$  successful trials, Similarly, also a confidence of 1 is impossible to prove.

**Bayes' Theorem Derivation** Contrary to the previous derivation, this approach is based on Bayes' theorem. First, similar to the former, the process is considered to be binomial distributed, but here indeed the success is modeled as such [KBG<sup>+</sup>97]. The prior probability is modeled by a beta distribution. Therefore, the probability of  $x$  given  $n$  trials is

$$p(x|n) = \frac{p(n|x)p(x)}{\int p(n|x)p(x)dx}, \quad (3.5)$$

$$= \frac{\binom{n}{m} x^m (1-x)^{n-m} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1}}{\int_0^1 \binom{n}{m} x^m (1-x)^{n-m} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1} dx}. \quad (3.6)$$

If the prior probability distribution is considered as a uniform distribution (no previous knowledge), i. e.,  $a = b = 1$  and  $m = n$  (all trials succeeding), the a-posteriori probability distribution Equation (3.6) simplifies to  $x_{|n} \sim \text{Beta}(n + 1, 1)$ , since

$$p(x|n) = \frac{x^n}{\frac{1}{n+1}} \quad (3.7)$$

$$= (n + 1)x^n. \quad (3.8)$$

The corresponding CDF using the reliability  $R$  as limit leads to

$$P(x_{|n} \leq R) = (n + 1) \int_0^R x^n dx \quad (3.9)$$

$$= R^{n+1}, \quad (3.10)$$

where 0 is the lower bound, because the beta distribution is defined for  $x \in [0, 1]$  in this case. The probability (confidence) of  $x_{|n} \geq R$  is

$$P(x_{|n} \geq R) = C = 1 - R^{n+1} \quad (3.11)$$

and hence the required number of trials  $n$  to prove that the reliability is at least  $R$  with

a confidence of  $C$  is

$$n = \frac{\log(1 - C)}{\log(R)} - 1. \quad (3.12)$$

According to the derivation based on Bayes' theorem one trial less is required, compared to the derivation based on the binomial distribution. However, the results are very similar.

### 3.1.1.2 Event Amount Precision Estimate Method

This section describes how the precision of the event rate can be estimated. The event rate is defined as  $y = \frac{\lambda}{\Delta I}$ , where  $\Delta I$  is the fixed Poisson interval underlying the Poisson distributed random variable  $m \sim \text{Poi}(\lambda)$  and  $\lambda$  the number of events occurring in the interval. As shown in Section A.2.2, the Poisson distribution can be approximated with a normal distribution if more than 30 events occur in a Poisson interval. Hence, with  $\lambda \geq 30$  the approximated confidence interval of  $\lambda$  follows

$$\left[ \lambda - z_{1-\alpha/2} \sqrt{\lambda}, \lambda + z_{1-\alpha/2} \sqrt{\lambda} \right], \quad (3.13)$$

where  $z_{1-\alpha/2}$  is the  $1 - \alpha/2$ -th quantile of the standard normal distribution (c.f. Section A.2.2) corresponding to the confidence interval. Then, the precision relative to the amount of events is given by

$$\delta = \frac{z_{1-\alpha/2} \sqrt{\lambda}}{\lambda} = \frac{z_{1-\alpha/2}}{\sqrt{\lambda}}. \quad (3.14)$$

Therefore, if one wants to prove, that the precision of the amount of successes is at least  $\delta$ ,

$$\lambda = \left( \frac{z_{1-\alpha/2}}{\delta} \right)^2 \quad (3.15)$$

events must occur within the Poisson interval. This way, it is possible to determine the minimum required size of the Poisson interval to prove this precision with

$$\Delta I = \frac{\left( \frac{z_{1-\alpha/2}}{\delta} \right)^2}{y}. \quad (3.16)$$

### 3.1.1.3 Hypothesis Test

Statistical testing is used to verify assumptions about the behavior or quantity of a random variable. The following explanations are based on [FHK<sup>+</sup>16].

A statistical test problem is modeled by two hypotheses, where one of the two has to be chosen. Therefore, the given assumption is formulated as the alternative hypothesis  $H_1$ , whereas the null hypothesis  $H_0$  represents the opposite of the assumption. Then, the decision is carried out by testing the null hypothesis  $H_0$ , which yields the possible result that

### 3. Identifying Relevant Traffic Scenarios

---

- $H_0$  is accepted or
- $H_0$  is rejected.

Two possible mistakes can occur: rejecting the null hypothesis when the null hypothesis is true (Type 1 error or  $\alpha$ -error) or accepting the null hypothesis when the alternative hypothesis is true (Type 2 error or  $\beta$ -error). This relation is illustrated in Table 3.1.

Table 3.1: Error types of a hypothesis test.

	$H_0$ true	$H_1$ true
$H_0$ accepted	-	Type 2 error / $\beta$ -error
$H_0$ rejected	Type 1 error / $\alpha$ -error	-

The probability of the Type 1 error is  $\alpha$ , it is also called the significance level of the test.  $\beta$  is the probability of the Type 2 error. Therefore, the probability of correctly deciding for the alternative hypothesis  $H_1$  is

$$\gamma = 1 - \beta, \quad (3.17)$$

referred to as power of the test.

For demonstrating a difference from a Poisson distributed variable from a certain value  $\lambda_0$  with a statistical significance, the minimum required size of the Poisson interval has to be determined. Let  $\lambda \geq 30$ , then the significance level

$$P(\lambda \geq \lambda_0) \leq \alpha \quad (3.18)$$

is fulfilled if

$$\lambda + z_{1-\alpha}\sqrt{\lambda} < \lambda_0. \quad (3.19)$$

Since the required Poisson interval is of interest, it is reformulated to

$$y\Delta I + z_{1-\alpha}\sqrt{y\Delta I} < y_0\Delta I, \quad (3.20)$$

which yields to

$$\Delta I = y \left( \frac{z_{1-\alpha}}{y_0 - y} \right)^2. \quad (3.21)$$

Furthermore, the power of the test

$$P(\lambda \leq \lambda_0) \geq 1 - \beta, \quad (3.22)$$

is fulfilled if

$$\lambda + z_{1-\alpha}\sqrt{\lambda} < \lambda_0 - z_{1-\beta}\sqrt{\lambda} \quad (3.23)$$

holds, what can be solved to

$$\Delta I = y \left( \frac{z_{1-\alpha} + z_{1-\beta}}{y_0 - y} \right)^2. \quad (3.24)$$

### 3.1.2 Application: How many Kilometers?

The previously explained methods can be used to determine the required amount of kilometers that a AV needs to be driven, in order to demonstrate certain statistical characteristics. Like in [KP16], also here the human driver failure rate is used as a reference, whereby the German numbers are considered.

According to the publication *Verkehr – Verkehrsunfälle 2017* presented by the *Federal Statistical Office of Germany* [Bun18], the following rates are predicted for the year 2017. Per 1 billion vehicle kilometers

- $\mathcal{F}_{\text{total}} = 3371$  accidents,
- $\mathcal{F}_{\text{pers}} = 386$  accidents with injuries, and
- $\mathcal{F}_{\text{fatal}} = 4.1$  fatal accidents happened in average.

The rates are determined based on the respective total amount of accidents and the total amount of kilometers driven by German vehicles, including the distances driven in foreign countries. The total amount of kilometers is estimated by the *Federal Highway Research Institute of Germany*. The failure rates, as used in the section before, are given by  $F_{\dots} = \mathcal{F}_{\dots}/10^9$ .

The questions from [KP16] are answered in this work, given the German traffic statistics. Therefore, the questions are formulated according to [KP16] as:

- How many kilometers would AVs have to be driven without failure to demonstrate that their failure rate is below some benchmark?
- How many kilometers would AVs have to be driven to demonstrate their failure rate to a particular degree of precision?
- How many kilometers would AVs have to be driven to demonstrate that their failure rate is statistical significantly lower than the failure rate of the average human driver?

In the next sections, these questions are answered based on the statistical methods shown in the former section. Furthermore, in the last section, a necessary correction term for the usage of the hypothesis test is provided.

### 3.1.2.1 Success Run Method

In order to answer the first question (How many kilometers would AVs have to be driven without failure to demonstrate that their failure rate is below some benchmark?), the success run method (Section 3.1.1.1) can be used. Therefore, each kilometer is considered as an independent Bernoulli trial. Moreover, for the whole distance, no accident is allowed to happen.

The results, given the failure rates  $\mathcal{F}_{...}$ , are shown in Figure 3.2. For example, if one wants to demonstrate a failure rate below  $\mathcal{F}_{\text{fatal}}$  with a confidence of 0.95, approximately 730 million kilometers need to be driven.

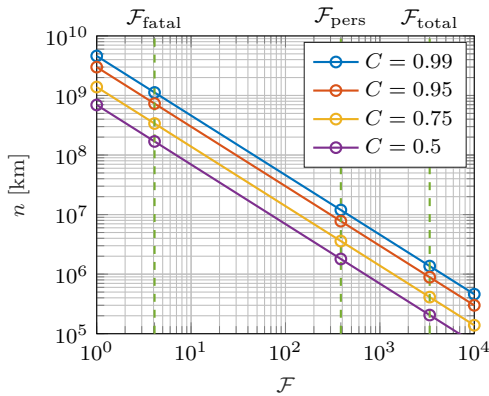


Figure 3.2: Kilometers need to be driven according to success run method.

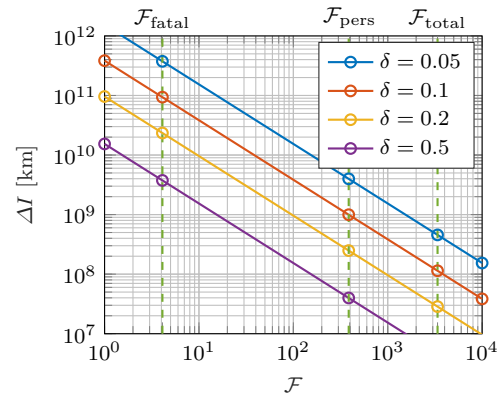


Figure 3.3: Kilometers need to be driven to demonstrate certain precision.

### 3.1.2.2 Event Amount Precision

How many kilometers would AVs have to be driven to demonstrate their failure rate to a particular degree of precision? The second question can be answered using the event amount precision method (see Section 3.1.1.2). Accordingly, the whole driving is modeled as Poisson distributed, where the minimum necessary interval size needs to be determined, given an objective degree of precision (e.g., the estimated failure rate should be within 5%, hence  $\delta = 0.05$ ). Moreover, it is assumed that more than 30 events occur within the interval (3.1.1.2).

The minimum size of the Poisson interval and hence the minimum number of kilometers need to be driven for different failure rates are depicted in Figure 3.3. In addition to the different failure rates, also various degrees of precision are shown. For all values, a 0.95 confidence interval is used.

### 3.1.2.3 Hypothesis Test

The last question is: How many kilometers would AVs have to be driven to demonstrate that their failure rate is statistically significantly lower than the human driver failure



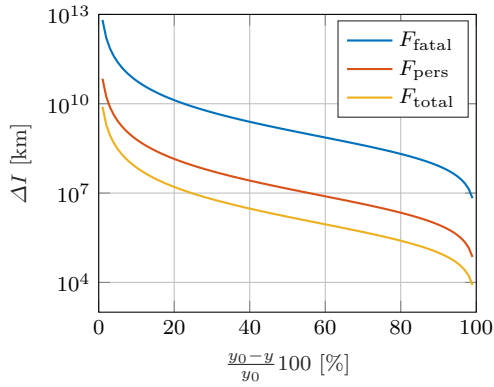


Figure 3.4: Kilometers need to be driven to demonstrate statistically significantly lower failure rates, with a significance level of  $\alpha = 0.05$ .

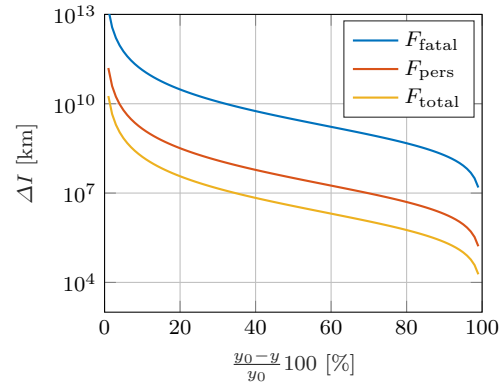


Figure 3.5: Kilometers need to be driven to demonstrate statistically significantly lower failure rates, with a significance level of  $\alpha = 0.05$  and a power of  $\gamma = 0.8$ .

rate? The answer is given by performing a statistical test. First, the hypothesis needs to be constructed. According to [KP16] it is defined as

- $H_0: \lambda \geq \lambda_0$  and
- $H_1: \lambda < \lambda_0$ .

Like before, the Poisson distribution is approximated by a normal distribution and hence  $\lambda \geq 30$  has to be fulfilled.

By evaluating the numbers at a significance level of  $\alpha = 0.05$ , the results can be seen in Figure 3.4. The x-axis shows the improvement of the AV over the human failure rates. Demonstrating that the AV is better than a human driver with statistical significance gets easier the better the AV performs.

If also a power of the test  $\gamma = 0.8$  is used, the resulting curves are shifted upwards compared to the previous (Figure 3.5). Accordingly, more kilometers need to be driven.

### 3.1.2.4 Hypothesis Test Corrected

The equation used to construct the plot of the previous section is based on the assumption that  $\lambda \geq 30$ . Therefore, one way to fulfill this condition is to modify the equation based on the significance as

$$\Delta I = y \left( \frac{z_{1-\alpha}}{y_0 - y} \right)^2 f(y_0, y) \quad (3.25)$$

with

$$f(y_0, y) = \begin{cases} 1 & \text{if } \frac{y_0 - y}{y_0} < \frac{z_{1-\alpha}}{\sqrt{30 + z_{1-\alpha}}} \\ \frac{30}{\left( y \frac{z_{1-\alpha}}{y_0 - y} \right)^2} & \text{if } \frac{y_0 - y}{y_0} \geq \frac{z_{1-\alpha}}{\sqrt{30 + z_{1-\alpha}}} \end{cases} \quad (3.26)$$

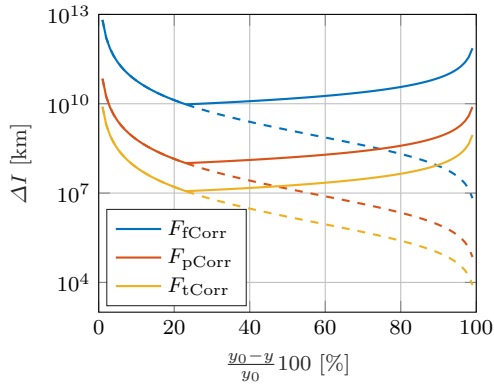


Figure 3.6: Kilometers need to be driven to demonstrate statistically significantly lower failure rates, with a significance level of  $\alpha = 0.05$  – Corrected.

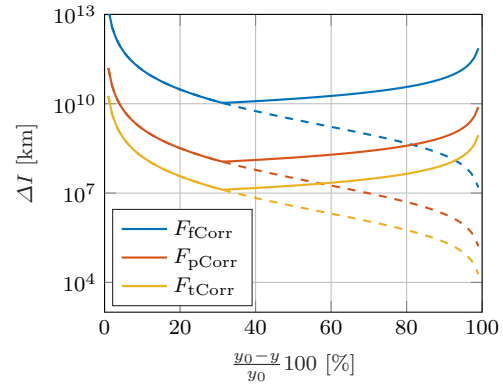


Figure 3.7: Kilometers need to be driven to demonstrate statistically significantly lower failure rates, with a significance level of  $\alpha = 0.05$  and a power of  $\gamma = 0.8$  – Corrected.

The equation based on the significance level and the power also needs to be modified, where

$$\Delta I = y \left( \frac{z_{1-\alpha} + z_{1-\beta}}{y_0 - y} \right)^2 f(y_0, y) \quad (3.27)$$

with

$$f(y_0, y) = \begin{cases} 1 & \text{if } \frac{y_0 - y}{y_0} < \frac{z_{1-\alpha} + z_{1-\beta}}{\sqrt{30 + z_{1-\alpha} + z_{1-\beta}}} \\ \frac{30}{\left( y \frac{z_{1-\alpha} + z_{1-\beta}}{y_0 - y} \right)^2} & \text{if } \frac{y_0 - y}{y_0} \geq \frac{z_{1-\alpha} + z_{1-\beta}}{\sqrt{30 + z_{1-\alpha} + z_{1-\beta}}} \end{cases} \quad (3.28)$$

holds.

The resulting curves are depicted in Figure 3.6 and 3.7 as solid lines. The amount of required kilometers drops until to the point where the event rate per Poisson interval reaches 30. From this point on, the amount of required kilometers increases. This is due to the fact, that if the failure rate decreases, the size of the Poisson interval needs to be increased in order to achieve at least 30 events.

### 3.1.3 Conclusion

In this section, statistical methods are explained which can be used to determine the required amount of kilometers an AV needs to be driven in order to demonstrate certain statistical reliability. The methods are applied to German accident statistics.

As a result of this section, and also of [KP16], it becomes clear, that proving reliability by just driving the necessary amount of kilometers is infeasible. As stated in [KP16], even under aggressive testing assumptions, existing fleets would take tens and sometimes hundreds of years to drive the required kilometers.

Therefore, new concepts of validation need to be developed. One possible approach is

the so-called scenario-based testing [PEG, JWKW18]. Within this approach, only relevant scenarios are used for the validation of AVs. Nevertheless, also other approaches are known such as formal verification, where *Responsibility Sensitive Safety* RSS [SSS17] is one of the most prominent methods building on that approach. In this work, the focus lies on scenario-based testing.

## 3.2 State of the Art

The validation of AVs through scenario-based testing is considered to be one of the feasible approaches to prove their safety [JWKW18]. In the survey [RPL<sup>+</sup>20], various works utilizing the scenario-based validation approach are summarized. The survey also lists works trying to identify and define relevant scenarios. In the remainder of this section, various related work dealing with the identification of relevant traffic scenarios are summarized.

In this work, the following definitions are used. The EGO vehicle is the vehicle under test, while the OBJ vehicle is a vehicle in the surrounding of the EGO vehicle.

**Traffic Scenario Clustering with an Unsupervised Random Forest** In [KWB18] and the successor work [KWM<sup>+</sup>19], a data-adaptive similarity measure is introduced. An unsupervised random forest is grown by separating noise from real data. Then the similarity between data points is measured by the normal proximity [KWB18] or by the path proximity as introduced in [KWM<sup>+</sup>19]. With HC the proximity matrix is reordered, then visualized and manually analyzed to identify categories of scenarios. This overall procedure is applied to traffic scenarios in both works. In [KWB18] the traffic scenario is described through various features like velocity at different timestamps, and brake indicator of the EGO vehicle and a OBJ vehicle. Also, the relative angle between the EGO and the OBJ is used, as well as some infrastructure information like the approximated radius of the road segment, the speed limit, and the number of lanes. The feature list in [KWM<sup>+</sup>19] is more comprehensive. For example, the distance information from the EGO vehicle to multiple surrounding vehicles, dynamic information of the EGO vehicle, and many other more high-level features are used.

**Unknown Traffic Scenario Detection with an Autoencoder** Detecting unknown scenarios through novelty detection is the aim of [LBR<sup>+</sup>18]. A reconstruction-based approach is pursued for outlier detection, hence an autoencoder is trained on known data. In the test phase, if the reconstruction error is higher than a specified threshold, the scenario is assumed to be unknown. To add unknown scenarios to the training data an iterative procedure is suggested and the autoencoder is retrained with the new training data. This way, the training data is automatically extended. The features used in the application are the presence or absence of a car in front of the EGO as well as the curvature, slope, speed

limit, and urbanity of the road. Each feature is collected over the span of a scenario.

**Traffic Scenario Clustering with  $K$ -medoids and Dynamic Length Scenario Extraction** Finding scenario clusters is the objective of [LGO<sup>+</sup>19]. An approach for extracting dynamic length scenarios is presented. Based on scenario dividing signals, the dynamic length scenario can be automatically generated from continuous recordings. The used dividing signals are the street type, the speed limit, the environment type, and the curviness. Whenever there is a strong change in one of the dividing signals, the scenario is split. For each dynamic length scenario a describing feature vector is generated including: the dividing signals plus the length of the segment, characteristic of the slope, average velocity, number of traffic lights, and number of braking maneuvers. The describing feature vectors are then used for clustering using  $K$ -medoids [KR87].

**Traffic Scenario Clustering with a Novel Scenario Similarity Measure** A novel similarity measure for traffic scenarios is introduced in [KWG<sup>+</sup>20]. For this purpose, the similarity between two scenes is determined first. A scene represents a slice of a scenario at a specific timestamp. The eight-car neighborhood around the EGO vehicle is used for this. Per possible neighbor, it is determined if there is a vehicle in both scenes and what is the difference in normalized distance between the two vehicles  $[0, 1]$ . If one scene has an object where the other does not, this cell is penalized with 1.5. Therefore, each cell contains a dissimilarity value, summing up the cells leads to the scene distance  $[0, 12]$ . The scenario distance is determined by averaging the scene distances. This distance is then used in HC. Moreover, maneuver detection, spatial and temporal filtering is introduced to be applied before the similarity calculation.

**Traffic Scenario Clustering with DTW and  $K$ -means** Clustering pairs of trajectories, also called encounter scenarios, is the objective of [WRZ<sup>+</sup>20]. The overall procedure can be summarized as follows: first, the driving encounters are transformed to be unified length. Second, the unified length trajectories are processed with representation learning/feature extraction in order to achieve the driving encounter representations, which are then used for clustering in the third and final step. The unification is realized through re-sampling using interpolation. For the representation learning/feature extraction various approaches have been examined: LSTM-autoencoder, convolutional autoencoder, *Dynamic Time Warping* (DTW) [SC78, M07] (using the cost matrix as feature per driving encounter), and normalized Euclidean distance (distance for each trajectory element). The clustering is realized with  $K$ -means on the extracted features. In [WRZ<sup>+</sup>20], the best feature extraction strategy is DTW.

**Traffic Scenario Clustering with DTW, PCA and  $K$ -means** The objective in [HGSP20] is to use as little “mental model” as possible to cluster traffic scenarios. For this

purpose, the time series data which describes the traffic scenario is first z-normalized and then further processed. The DTW distance between pairs of time series data is calculated. This leads to a feature vector for each traffic scenario which expresses the difference to all other scenarios. Those feature vectors are feature-wise normalized and then further processed using PCA. The  $K$ -means clustering is performed in the space reduced by PCA.

#### **Traffic Scenario Clustering with HC in the Latent Space of Deep Autoencoders**

In [HBG20], two different deep learning architectures are presented which are used to reconstruct the input trajectories through a latent space. In that latent space, HC analysis is performed. Both architectures follow the autoencoding regime. The first architecture is realized through a convolutional autoencoder. The input trajectories are rasterized into a sequence of grids. The second architecture uses a so-called deep set network, which transform all positions of all objects per timestamp into a hidden representation. That hidden representation is processed sequentially with a *Recurrent Neural Network* (RNN), yielding the latent representation after processing all timestamps. Hence, the encoder is composed of the deep set network and a RNN. Given this latent representation, the input is reconstructed through a RNN and a deep set prediction network.

#### **Traffic Scenario Clustering with DBSCAN using a PCA or t-SNE projected Latent Space of a LSTM-based Autoencoder**

Another approach utilizing deep learning and the autoencoding regime is presented in [DARC20], where LSTMs are used to encode a trajectory and reconstruct it from the latent representation. The latent representations are further projected with PCA or t-SNE before clustering with DBSCAN [EKS<sup>+</sup>96]. The reconstruction error is used to estimate the novelty of a trajectory. However, the focus of [DARC20] is the generation of scenarios, which is realized through recurrent conditional GANs.

#### **Traffic Scenario Classification and Clustering with $K$ -means in the Latent Space of an Iteratively Learned Deep Neural Network**

A different problem setting is presented in [BWBD21], where some known classes and also unknown classes are assumed. In a multi-step training process, a deep neural network is trained, such that it finds representations suited for classification and clustering of the unknown classes. The steps include self-supervised pre-training, classification, and mixed training (clustering and classification). For the clustering training, a novel representation based on the random forest is introduced, the so-called *Random Forest Activation Pattern* (RFAP). The Hamming similarity between two RFAPs of the latent representation of two data points is used as objective for the clustering loss. Hence, the deep network is forced to mimic the similarity constructed by the Hamming similarity and the RFAPs. After the training, the latent representations are clustered with  $K$ -means. The training input consists of a sequence of images, representing the infrastructure and the objects as well.

#### **Traffic Scenario Clustering with $K$ -means in the Latent Space of an Architecture consisting of an Autoencoder and a Recurrent Neural Network**

Also in [ZFYZ21], an image sequence which contains the infrastructure and the objects at each timestamp is used as input. Each frame is fed through an autoencoder which is trained using the reconstruction loss and the triplet loss. For the triplet loss, temporally closer frames shall be closer in the latent space. The latent representations of the image sequence are transformed into a sequence latent representation, which is then used for clustering. This transformation is realized by a RNN architecture, which aims to reconstruct frames and predict future frames. The sequence representations of the scenarios are clustered using  $K$ -means.

#### **Traffic Scenario Clustering with Gaussian Mixture Model based on DTW, t-SNE, min-max Distance and Multidimensional Scaling**

Instead of using deep learning, in [HRC21] a tool chain of dimensionality reduction techniques and similarity measures is applied for clustering trajectories. First the dissimilarity between trajectories is determined with the DTW distance. The dissimilarity matrix is embedded using t-SNE. In the embedding space, the distance between the embedded trajectories is determined using the min-max distance, which takes all possible paths from one point to the other and returns the smallest of all largest gaps among all different paths. After embedding the distance matrix from the min-max dissimilarity with multidimensional scaling, the resulting embeddings are clustered using Gaussian mixture model. Additionally, in [HRC21], the former described clustering procedure is used to validate trajectories generated with a GAN.

#### **Traffic Scenario Clustering with HC based on the Chi-squared Distance Between Trajectory Histogram Representations**

In [BSS21], trajectories are clustered by HC with Chi-squared distance. Each trajectory is transformed into a histogram representation. Each point of a trajectory is encoded as an area group label. The area group labels are defined by a Gaussian mixture model, which is fitted on the complete dataset. Given the trajectory points encoded as area labels, a trajectory can be represented as histogram over the assigned area labels. The trajectories are then clustered using HC with Chi-squared distance between the histograms. The cluster centers are generated through averaging the histograms within a cluster.

#### **Traffic Scenario Clustering based on DTW for Multiple Objects**

Clustering traffic scenarios by the EGO information and other OBJs' information is realized in [RRB<sup>+</sup>21]. To determine if two scenarios are similar, the following procedure is used. First: Do the scenarios contain the same object types? If yes, then second: DTW distance between EGO trajectories below a certain threshold? If yes, then third: DTW distance between any trajectories from the same object type are below a certain threshold? If yes, scenarios

are similar. If any of the previous statements is not true, the scenarios are considered to be not similar.

This way, an iterative clustering can be realized.

**Unknown Traffic Scenario Detection with Deep Learning Approaches** Detecting anomalous driving scenarios is the objective in [WSK<sup>+</sup>22]. A benchmark dataset with abnormal driving scenarios was generated manually. Also, some baseline anomaly detection methods are proposed. Either a single trajectory, multiple trajectories or multiple trajectories paired with the infrastructure information is used as input. They propose four different model architectures: (i) linear, (ii) deep, (iii) two-stage, and (iv) end-to-end. The linear and the deep models follow the typical autoencoder regime for outlier detection. The two-stage models are trained as autoencoders, where the outlier detection is realized with OCSVM in the latent space. The end-to-end model combines the autoencoder training with some OCSVM like loss for the latent space, which forces all data points to be close to some center. Here the outlier detection is realized through the distance in the latent space.

In summary, many methods have been presented recently to address the problem of clustering traffic scenarios and detecting unknown scenarios. Typical strategies to enable the clustering or novelty detection is to either transform the scenarios into valuable representations or to use specific similarity measures. Most of the approaches explicitly or implicitly make use of certain domain knowledge:

1. Design of features with domain knowledge [KWB18, KWM<sup>+</sup>19].
2. Design of features and scenario dividing process based on domain knowledge [LGO<sup>+</sup>19].
3. Domain knowledge-based similarity measure of traffic scenarios [KWG<sup>+</sup>20].
4. Measuring distance of time series signals with DTW [WRZ<sup>+</sup>20, HGSP20, HRC21, RRB<sup>+</sup>21]. Where using the DTW implicitly defines what is interpreted as similar.
5. Using known classes to guide representation learning [BWBD21].
6. Domain knowledge: temporally closer frames are more similar [ZFYZ21].
7. Domain knowledge: trajectory points can be abstracted through spatial groups to form histograms [ZFYZ21].

Using domain knowledge can aid the clustering and novelty detection to perform as expected and hence to follow certain similarity assumptions. On the other hand, one might argue that domain knowledge might bias the method [HGSP20]. However, as can be seen from the previous list, relying on domain knowledge is a trend in recent approaches. In

order to prevent a strong bias, the design of the used features and the application of the domain knowledge itself has to be done carefully.

Even though lots of recent work utilize domain knowledge, the representation learning methods only use domain knowledge a little or not at all. The representation learning methods present in [WRZ<sup>+</sup>20] rely only on the autoencoder regime, which is not based on domain knowledge for the training except for the architecture and input selection. The autoencoder concept is also applied in [HBG20, DARC20], only for the input representation selection and network selection, domain knowledge is used. As already stated, known classes in training procedure can be seen as utilizing domain knowledge in [BWBD21]. Another autoencoder like setting can be found in [HRC21]. It is extended with the domain knowledge that frames which are close to each other in time should also have more similar representations. The autoencoding regime is a common approach in recent works, since it enables the network to extract meaningful and expressive features. However, more explicitly guiding representation learning methods with domain knowledge has not been investigated in the area of traffic scenario clustering and novelty detection.

Moreover, recent work has focused on trajectories and dynamic information. Static information has been used only rarely. Only the works [BWBD21, ZFYZ21] used comprehensive static information, since a rasterized version of the infrastructure is used. Hence, investigating the benefit of infrastructure information for clustering and novelty detection of traffic scenarios is still not widely considered. Since the infrastructure is a crucial part of a traffic scenario, this should be analyzed further.

### 3.3 Entropy-Based Novelty Detection of Road Infrastructure Images

Detecting novel traffic scenarios is a key factor for scenario-based testing, which is crucial for the validation of AVs. Identifying novel scenarios can be interpreted as an outlier detection task, where an observed scenario is tested with respect to its novelty. In this section a nearest neighbor graph-based outlier detection method is introduced, therefore it is an alternative approach to the methods summarized in the previous section. A traffic scenario consists of several components. Besides the dynamic objects, the infrastructure forms an important part. In this section, the outlier detection method is applied to infrastructure images to detect novel infrastructures. The methods presented in this section have been published in [WFBU20], which is part of this dissertation.

The novel outlier score is based on a directed nearest neighbor graph. Graphs allow one to use the data points itself to determine outliers. Such type of graphs are constructed also in dimensionality reduction techniques such as t-SNE (Section 2.2.2), Barnes-Hut-SNE (Section 2.2.2), LargeVis [TLZM16], and UMAP (Section 2.2.3). In this section, an entropy-based outlier score is presented, which can be embedded into the before men-



tioned directed graph-based dimensionality reduction techniques. The novel outlier score is called *Local Entropy Factor* (LEF). The score LEF is based on the definitions of closeness or distance as used in the dimensionality reduction techniques. Hence, the score can be computed directly from the graph constructed during dimensionality reduction. Therefore, no additional graph or model has to be constructed, if a dimensionality reduction is performed. The method presented in this section can also be used without dimensionality reduction, the graph is constructed independently. Further, it is shown how an outlier score presented in another work based on t-SNE [JHPvdH12, SG17] is embedded into UMAP.

The contributions of this research on nearest neighbor-based outlier detection for traffic scenarios are as follows:

1. Introduction of the novel unsupervised outlier score LEF based on weighted normalized entropy, which can be applied within graph-based dimensionality reduction techniques.
2. Embedding of a previously defined outlier score into UMAP.
3. Comparing the proposed scores with various state-of-the-art neighborhood-based outlier scores on different benchmark datasets.
4. Applying the scores to a road infrastructure images dataset to validate its capabilities in identifying new road infrastructures.
5. Providing a publicly available implementation of the LEF in combination with UMAP and the developed OpenDRIVE [D<sup>+</sup>15] parsing and plotting tool for MATLAB<sup>1</sup>.

Moreover, from the experiments it can be seen that the proposed method is more robust with respect to the chosen number of neighbors, compared to the state-of-the-art methods. This is an important characteristic, since in unsupervised problems, the appropriate number of neighbors is not known.

The unsupervised outlier detection is based on an unlabeled dataset  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$  containing  $M$  data points  $\mathbf{x}_m \in \mathbb{R}^N$ . Here, a dataset is represented as a weighted directed graph  $G$ , where each vertex  $v_m$  represents a data point  $\mathbf{x}_m$ . An edge  $e_{j|i}$  from data point  $\mathbf{x}_i$  to  $\mathbf{x}_j$  is weighted with  $P_{j|i}$ .

### 3.3.1 Local Entropy Factor

The novel entropy-based outlier score LEF is presented in this section. It can be used in any graph-based dimensionality reduction technique, or it can be used standalone too. This score is designed to contain information about inlierness in addition to identifying

<sup>1</sup>Implementations can be found in <https://github.com/JWTHI>.

local outliers. Hence, the definition of inliers is required. Here the membership strength of an inlier is defined as maximal if it is the nearest neighbor to all its  $k$  neighbors and minimal if it is not in the neighborhood of any of its neighbors. Additionally, the score shall be higher when the incoming similarities are equally distributed. Moreover, the value has to be weighted by the sum of incoming similarities, such that a point being nearest neighbor to all its neighbors gets a higher score than a point being an equally weak neighbor to all its neighbors. This definition favors data points which have the same similarities to all their neighbors over varying similarities. The entropy-based score is meant to detect outlying points within their neighborhood. Hence, big outlying groups will not be detected as outliers, since they form a neighborhood.

The Shannon entropy fulfills the equally distributed requirement of the above definition. Therefore, a weighted normalized entropy is utilized in this work as the measure of inlier strength. Identifying local outliers is achieved by selecting the data points with a low inlier value. Let the relative incoming similarities be defined as

$$\hat{P}_{i|j} = \frac{P'_{i|j}}{\sum_{j \in \mathcal{K}_i} P'_{i|j}}, \quad (3.29)$$

where  $P'_{i|j}$  is the incoming similarity, which is defined in the following paragraph. The LEF is determined as a weighted normalized entropy:

$$\mathcal{A}_{\text{LEF}}^\downarrow(\mathbf{x}_i) = -\frac{\sum_{j \in \mathcal{K}_i} P'_{i|j}}{K} \sum_{j \in \mathcal{K}_i} \hat{P}_{i|j} \log_2(\hat{P}_{i|j}). \quad (3.30)$$

By using the relative incoming similarities, it is ensured that they sum up to 1 and hence the normalized entropy is bound to 1. The normalized entropy of the relative incoming similarities would lead to 1 for data points with equal incoming similarities, independent of the actual value of the similarities. Therefore, the normalized entropy is weighted with the sum of incoming similarities divided by the maximum possible value  $K$ .

The LEF can be used in multiple graph-based dimensionality reduction methods. Here, the focus is on UMAP and t-SNE. In Table 3.2 the four different variants used in this work are listed, specifically: *UMAP graph-based non-sparse LEF* (ULEF), *UMAP graph-based Sparse LEF* (USLEF), *t-SNE graph-based non-sparse LEF* (tLEF), and *t-SNE graph-based Sparse LEF* (tSLEF). Extended neighborhood means, that the incoming similarity of a missing incoming edge is estimated by using the similarity as defined for the respective neighbor  $\mathbf{x}_j$ , but ignoring the fact that the point under investigation  $\mathbf{x}_i$  is not in the neighborhood of  $\mathbf{x}_j$ . In  $\mathcal{A}_{\text{USLEF}}^\downarrow$  (UMAP) and  $\mathcal{A}_{\text{tSLEF}}^\downarrow$  (t-SNE) only existing edges are considered, hence they are called *Sparse LEF*. The non-sparse versions are  $\mathcal{A}_{\text{ULEF}}^\downarrow$  (UMAP) and  $\mathcal{A}_{\text{tLEF}}^\downarrow$  (t-SNE), which use extended neighborhoods as described before. When using the similarities from UMAP, they have to be normalized by  $\log_2(K)$ , such that the outgoing similarities sum to 1, see Equation (2.40). The similarities of t-SNE are already normalized.

Table 3.2: LEF variants and the corresponding substitutions.

Type	$P'_{i j}$ in Eq. (2.29) & Eq. (3.30)	extended neighborhoods
$\mathcal{A}_{\text{ULEF}}^\downarrow$	Eq. (2.39) / $\log_2(K)$	yes
$\mathcal{A}_{\text{USLEF}}^\downarrow$	Eq. (2.39) / $\log_2(K)$	no
$\mathcal{A}_{\text{tLEF}}^\downarrow$	Eq. (2.33)	yes
$\mathcal{A}_{\text{tSLEF}}^\downarrow$	Eq. (2.33)	no

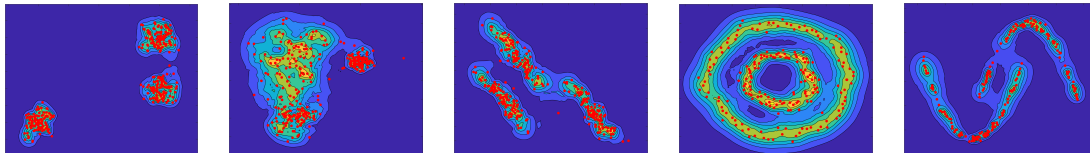


Figure 3.8: ULEF application to toy datasets. Background color indicating the value of  $\mathcal{A}_{\text{ULEF}}^\downarrow$  at the specific positions.

To visualize the ULEF scoring, it is applied to some two-dimensional toy datasets. In Figure 3.8 the red dots are considered to be the inlier dataset, and one single point is interpreted as test sample. The test points are sampled from a dense grid spanning the complete plotted area. Therefore, the background color indicates the  $\mathcal{A}_{\text{ULEF}}^\downarrow$  value for the test sample at that specific point. As can be seen from the plots, the scoring is density adaptive. This can best be seen from the second plot where three groups of inlier data points (red dots) with three different densities are illustrated. A new data point at the border of the group with the low density (middle), would likely be recognized as inlier. Whereas, when introducing a new point at the border of the right most group, it would most likely be interpreted as an outlier.

### 3.3.2 KNNSOS with UMAP

In this section, the *UMAP graph-based KNNSOS* (USOS) is introduced, which is the outlier score KNNSOS [SG17] implemented into UMAP. The KNNSOS score [SG17] was defined within Barnes-Hut-SNE. The initial score SOS [JHPvdH12] is defined using t-SNE conditional probabilities. An outlier is defined as a data point which is frequently not likely to be linked. In SOS, all the data points are used to determine the score, where in KNNSOS only the reverse nearest neighbor set of a point is used. Hence, only the incoming edges are considered. Therefore, the outlier score of  $\mathbf{x}_i$  is defined as

$$\mathcal{A}_{\text{KNNSOS}}^\uparrow(\mathbf{x}_i) = \prod_{j \neq i} 1 - P_{i|j}, \quad (3.31)$$

with  $P_{j|i}$  calculated as in Equation (2.33). The score leads to high values for data points being weakly linked to other data points. If for the data point  $\mathbf{x}_i$  no link exists at all, the score will be 1. In other words, the data point  $\mathbf{x}_i$  is not a member of any neighborhood.

The application of the KNNSOS for UMAP as such is straight forward. But, the similarities generated by UMAP need to be adjusted. This is due to the local connectivity requirement of UMAP, which results in the nearest neighbor having a similarity of 1. Each data point which is the nearest neighbor of any other point would have an outlier score of 0, no matter what the other similarities are. Therefore, the weights of the graph are adjusted by using  $P'_{j|i} = P_{j|i} / \log_2(K)$  with  $P_{j|i}$  the UMAP similarities, calculated as defined in Equation (2.39). Provided with the adjusted weights, the application of KNNSOS in UMAP is given by the outlier score USOS, defined as

$$\mathcal{A}_{\text{USOS}}^\uparrow(\mathbf{x}_i) = \prod_{j \neq i} 1 - P'_{i|j}. \quad (3.32)$$

Since the weights are adjusted, the interpretation of the outlier score changes. The smallest possible value which can be achieved is  $\mathcal{A}_{\text{USOS}}(\mathbf{x}_i) = (1 - 1/\log_2(K))^{|R_i|}$ , with  $R_i$  being the reverse nearest neighbor set. Hence, the smallest possible number varies for each data point. This may need to be considered in data analysis steps. In this work, the score  $\mathcal{A}_{\text{USOS}}$  is not adjusted, since if more incoming edges exist, the data point can be considered less as an outlier than others, which consist of less incoming edges. This characteristic is used in ODIN (Section 2.3) as well. If UMAP is applied as dimensionality reduction technique, the USOS score can be determined with little effort, since the graph is already constructed.

### 3.3.3 Experiments

In this subsection, the presented scores ULEF, USLEF, USOS, tLEF, and tSLEF are investigated with respect to their outlier detection capabilities. To simplify the understandability, in the remainder of this section, the scores introduced in this work are written italic, e.g., *ULEF*. Therefore, they are applied to several real world datasets and compared to state-of-the-art outlier methods. This section provides a brief summary of the used datasets and the evaluation measure. The results are shown and discussed as well. The implementation of the UMAP-based scores uses parts of the publicly available implementation [MHSG18], such as performing the binary search. The Barnes-Hut-SNE-based scores are a modified version of [Jan12].

#### 3.3.3.1 Datasets

The outlier datasets for the experiments in this section were taken from [CZS<sup>+</sup>16b], where each dataset contains labels for outliers. All datasets were normalized and do not contain

Table 3.3: Used datasets specifications [CZS<sup>+</sup>16b].

Dataset	$M$	$M_O$	$N$	Variants
Anthyroid	7129	534	21	–
Arrhythmia	450	206	259	–
Cardiotocography	2114	466	21	–
Glass	214	9	7	–
HeartDisease	270	120	13	–
Hepatitis	80	13	19	–
InternetAds	1966	368	1555	–
Ionosphere	351	126	32	–
Lymphography	148	6	18	–
PageBlocks	5393	510	10	–
Parkinson	195	147	22	–
PenDigits	9868	20	16	10
Pima	768	268	8	–
Shuttle	1013	13	9	10
SpamBase	4207	1679	57	–
Stamps	340	31	9	–
WBC	223	10	9	10
WDBC	367	10	30	10
WPBC	198	47	33	–
Waveform	3443	100	21	10
Wilt	4819	257	5	–

duplicates. Furthermore, in case of categorical attributes, the *Inverse Document Frequency* (IDF) coded versions are used. No further processing on the datasets is applied. In this work only the standard versions of the datasets are used. Let the number of samples per dataset be  $M$ , the number of outliers  $M_O$ .

### 3.3.3.2 Evaluation Method

The evaluation is performed in such a way, that each outlier data point is analyzed individually with respect to all the inliers. For example, the dataset Glass (see Table 3.3) yields 9 individual outlier datasets  $\mathcal{D}_{O,1} \dots \mathcal{D}_{O,9}$  each containing 205 instances and one outlier. This differs from the evaluation performed in [CZS<sup>+</sup>16a], as this work focuses on the local structure.

Each of the outlier datasets is then processed with different outlier detection methods, including the ones introduced in this work. For this purpose, the number of neighbors  $k$  is varied from 3 to 100. The  $k$  is selected larger or equal to 3 such that the perplexity in

### 3. Identifying Relevant Traffic Scenarios

Table 3.4:  $\overline{AUC}_{\max}$  per dataset and method.

Dataset	<i>ULEF</i>	<i>USLEF</i>	<i>USOS</i>	<i>ILEF</i>	<i>ISLEF</i>	KNNSOS	ISOS	<i>K</i>	$\Sigma K$	LOF	LDOF	LoOP	COF
Anthyroid	.78 ± .24	.78 ± .24	<b>80 ± .23</b>	.79 ± .24	.79 ± .24	<b>80 ± .23</b>	.79 ± .24	.67 ± .26	.68 ± .26	.74 ± .26	.78 ± .23	.77 ± .24	.73 ± .27
Arrhythmia	.76 ± .25	<b>.76 ± .24</b>	.76 ± .25	.76 ± .25	<b>.76 ± .24</b>	.76 ± .25	.76 ± .25	.76 ± .27	.76 ± .26	.75 ± .27	.76 ± .25	.76 ± .25	.75 ± .25
Cardiotocography	.86 ± .19	.85 ± .19	<b>.87 ± .18</b>	.85 ± .19	.86 ± .19	.87 ± .19	.86 ± .19	.80 ± .23	.82 ± .22	.83 ± .21	.83 ± .22	.86 ± .21	.84 ± .23
Glass	.85 ± .11	.83 ± .12	.83 ± .10	.80 ± .13	.80 ± .12	.79 ± .16	.81 ± .12	.88 ± .06	.89 ± .06	<b>.92 ± .06</b>	.84 ± .07	.87 ± .08	.89 ± .11
HeartDisease	.78 ± .20	.78 ± .20	.77 ± .21	.79 ± .20	.79 ± .20	.78 ± .20	.77 ± .21	<b>.85 ± .18</b>	.84 ± .18	.84 ± .18	.82 ± .20	.82 ± .20	<b>.85 ± .18</b>
Hepatitis	.72 ± .19	.73 ± .21	.68 ± .19	.71 ± .20	.73 ± .19	.65 ± .21	.66 ± .20	.82 ± .11	.79 ± .15	<b>.82 ± .19</b>	.79 ± .14	.79 ± .15	.82 ± .14
InternetAds	.92 ± .20	.92 ± .16	.92 ± .18	.92 ± .18	.92 ± .16	.91 ± .20	<b>.94 ± .15</b>	.86 ± .24	.87 ± .23	.92 ± .18	.92 ± .19	.92 ± .19	.91 ± .19
Ionosphere	.95 ± .10	.95 ± .10	.95 ± .11	.95 ± .10	.95 ± .10	.95 ± .11	.95 ± .11	<b>.97 ± .07</b>	.97 ± .08	.95 ± .09	.93 ± .12	.95 ± .09	<b>.97 ± .07</b>
Lymphography	<b>1.00 ± .00</b>	1.00 ± .01	1.00 ± .01	1.00 ± .01	1.00 ± .01	1.00 ± .01	1.00 ± .01	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>
PageBlocks	.93 ± .13	.93 ± .13	.92 ± .15	.92 ± .13	.93 ± .13	.92 ± .14	.93 ± .13	.92 ± .14	.93 ± .14	.95 ± .10	<b>.96 ± .06</b>	.94 ± .09	.91 ± .18
Parkinson	.95 ± .10	.94 ± .11	.95 ± .09	.95 ± .09	.94 ± .09	<b>.97 ± .08</b>	.95 ± .08	.89 ± .13	.92 ± .11	.93 ± .16	.85 ± .21	.94 ± .12	.93 ± .11
PenDigits	<b>1.00 ± .00</b>	1.00 ± .01	1.00 ± .01	1.00 ± .01	1.00 ± .01	.99 ± .01	.99 ± .01	<b>1.00 ± .00</b>	<b>1.00 ± .00</b>	1.00 ± .01	.95 ± .08	<b>1.00 ± .00</b>	1.00 ± .01
Pima	.68 ± .27	.68 ± .27	.68 ± .26	.67 ± .27	.67 ± .27	.67 ± .27	.67 ± .27	.76 ± .22	.75 ± .22	.74 ± .20	.67 ± .27	.69 ± .26	<b>.78 ± .21</b>
Shuttle	.98 ± .01	.98 ± .02	.98 ± .01	.98 ± .01	.98 ± .01	.98 ± .02	<b>.99 ± .01</b>	.96 ± .03	.97 ± .02	.98 ± .02	.98 ± .02	<b>.99 ± .01</b>	.98 ± .02
SpamBase	<b>.82 ± .21</b>	.81 ± .23	.80 ± .24	.82 ± .22	.82 ± .22	.80 ± .24	.80 ± .24	.73 ± .22	.75 ± .22	.77 ± .21	.78 ± .20	.78 ± .21	.75 ± .20
Stamps	.95 ± .04	.95 ± .05	.90 ± .07	.94 ± .06	.95 ± .06	.90 ± .08	.92 ± .06	.95 ± .03	<b>.96 ± .02</b>	.94 ± .04	.93 ± .03	.94 ± .04	.95 ± .04
WBC	.93 ± .08	.93 ± .08	.94 ± .09	.93 ± .09	.93 ± .08	.93 ± .11	.94 ± .09	<b>.99 ± .02</b>	<b>.99 ± .02</b>	<b>.99 ± .02</b>	.98 ± .03	.98 ± .03	.98 ± .02
WDBC	.97 ± .06	<b>.97 ± .05</b>	<b>.97 ± .05</b>	<b>.97 ± .05</b>	.97 ± .06	<b>.97 ± .05</b>	<b>.97 ± .05</b>	.95 ± .08	.95 ± .08	<b>.97 ± .05</b>	<b>.97 ± .05</b>	<b>.97 ± .05</b>	.95 ± .07
WPBC	.53 ± .24	.53 ± .27	.52 ± .25	.52 ± .24	.52 ± .24	.53 ± .28	.51 ± .24	.57 ± .23	.56 ± .23	.55 ± .23	.55 ± .23	.53 ± .24	<b>.59 ± .25</b>
Waveform	.76 ± .27	.76 ± .27	.76 ± .27	.76 ± .28	.76 ± .27	.75 ± .27	.76 ± .27	<b>.78 ± .24</b>	.78 ± .25	.77 ± .26	.78 ± .24	.77 ± .27	.73 ± .27
Wilt	.80 ± .19	.81 ± .20	.83 ± .20	.83 ± .18	.82 ± .19	<b>.84 ± .17</b>	.83 ± .18	.58 ± .19	.61 ± .19	.72 ± .26	.83 ± .15	.80 ± .20	.74 ± .23

Barnes-Hut-SNE is at least 1. For each  $k$ , an evaluation measure is calculated. Here the AUC ROC<sup>2</sup> as used in [CZS<sup>+</sup>16a]

$$AUC_k = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \begin{cases} 1 & \text{if } \mathcal{A}^\uparrow(o) > \mathcal{A}^\uparrow(i) \\ 0.5 & \text{if } \mathcal{A}^\uparrow(o) = \mathcal{A}^\uparrow(i) \\ 0 & \text{if } \mathcal{A}^\uparrow(o) < \mathcal{A}^\uparrow(i) \end{cases} \quad (3.33)$$

is utilized, where here  $o$  is the outlier and  $\mathcal{I}$  the set of inliers of the given outlier dataset, with its cardinality  $|\mathcal{I}|$ . The scores are represented by  $\mathcal{A}^\uparrow$ . An  $AUC_k$  of 1 indicates that the outlier has the highest outlier score of all data points. Whereas, a value of 0 indicates that the outlier has the lowest value and hence is not detected as such.

The maximum and the average values over all  $k$ -s of  $AUC_k$  are determined per outlier dataset. Then, the averages of both scores per dataset are determined as

$$\overline{AUC}_{\max}(\mathcal{D}) = \frac{1}{M_O} \sum_{i=1}^{M_O} \max_{k=3, \dots, 100} (AUC_k(\mathcal{D}_{\mathcal{O}, i})) \quad (3.34)$$

$$\overline{AUC}_{\text{avg}}(\mathcal{D}) = \frac{1}{M_O} \sum_{i=1}^{M_O} \frac{1}{98} \sum_{k=3}^{100} AUC_k(\mathcal{D}_{\mathcal{O}, i}). \quad (3.35)$$

Those values can be considered as indicators for maximum possible and average performance given a certain dataset. In a final step, both values are averaged for all outlier datasets.

#### 3.3.3.3 Results

The application and evaluation of the outlier scores to the various datasets as described in the former sections are discussed here. The average of all datasets for the max and the

<sup>2</sup>Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve.

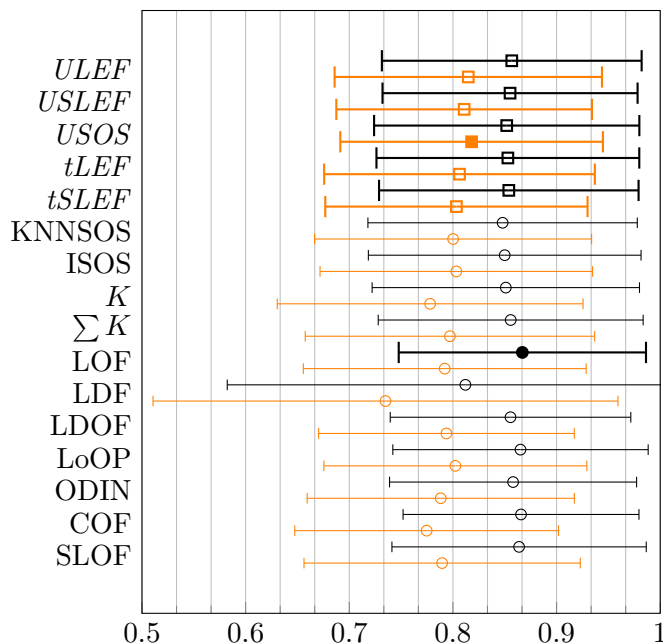


Figure 3.9: Performance per method: ( $\square/\circ/\bullet$ ) average  $\overline{AUC}_{\max}$  and ( $\square/\circ/\blacksquare$ ) average  $\overline{AUC}_{\text{avg}}$  over all datasets. The squares ( $\square/\square/\blacksquare$ ) indicate scores introduced in this work and the circles ( $\circ/\bullet/\circ$ ) other scores. The filled shapes ( $\blacksquare/\bullet$ ) indicate the best performing scores.

mean characteristics are shown in Figure 3.9. The results of the scores introduced in this work are drawn thick and the square is an indicator of the average value. The remaining methods are depicted normal and with a circle. The whiskers in all cases indicate the standard deviation. The best performing score is indicated thick and with a filled marker.

From the plots shown in Figure 3.9, it can be concluded that over all, the introduced scores are at a comparable level in terms of maximum performance ( $\square/\circ/\bullet$ ). However, LOF performed better than the rest. In terms of the average values, the introduced scores, especially the ones in combination with UMAP (*ULEF*, *USLEF*, *USOS*) show the best performance over all scores. They offer a higher robustness against variations of  $k$ , as it becomes clear from Figure 3.9, where the average values  $\overline{AUC}_{\text{avg}}$  over all  $k$ -s are shown ( $\square/\circ/\blacksquare$ ). Hence, for a randomly chosen  $k$ , the methods with the lowest risk are *ULEF* and *USOS*. This is an essential attribute, since selecting an appropriate  $k$  is difficult because the ground truth is not available for real world unsupervised tasks.

The scores *ULEF* and *tLEF* as well as their sparse versions *USLEF* and *tSLEF* are at a comparable level for the maximum values. But, the average values  $\overline{AUC}_{\text{avg}}$  ( $\square/\circ/\blacksquare$ ) of the non-sparse versions are slightly better. The non-sparse versions are causing additional computational cost, since the missing incoming edges are calculated as well. Therefore, deciding between the sparse and the non-sparse versions can be considered as trade-off between computational cost and accuracy.

A more detailed list of the different maximum values is depicted in Table 3.4. To ease



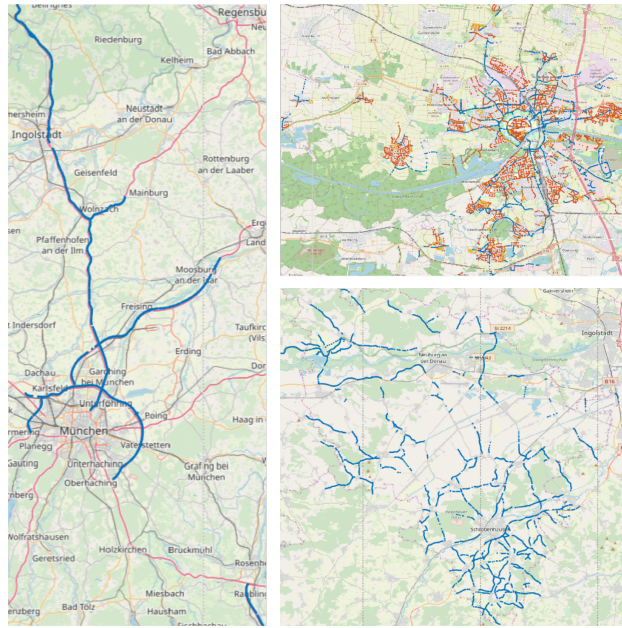


Figure 3.10: Position of the nodes used to create the images. Left image: (i) highway. Right bottom image: (ii) rural. Right top blue, red, and orange: (iii) city 50, (iv) city 30, and (v) city 5. [Ope19]

readability, only the best performing state-of-the-art methods are listed. As with the other scores, the introduced scores may suit for certain datasets better than for others. There is no score that is generally superior for all datasets. For example, LOF is outperforming the *ULEF* on 8 datasets, whereas *ULEF* is outperforming LOF on 7. Accordingly, in terms of maximum accuracy, the LOF is superior to *ULEF*, as already depicted in Figure 3.9. Contrary, the *ULEF* is outperforming the LOF in terms of average accuracy, thereby depicting its robustness against the number of neighbors  $k$  (see Figure 3.9).

#### 3.3.3.4 Application to Road Infrastructure Images

The application of the introduced scores to road infrastructure images is presented in the following section. Furthermore, the construction of the image dataset is explained. Experiments are performed with various scores, such that an application based analysis can be provided.

As already mentioned above, identifying newly observed traffic scenarios is a key aspect for the validation process of AVs. The road infrastructure builds a crucial part of a scene. The results of this work can serve as a case study. Here, a dataset of highway infrastructure images is considered as pre-recorded dataset. Various other infrastructure classes are compared in terms of outlieriness to the highway dataset. It is important to note, that the used classes are just defined to validate the method, since it can be assumed that a highway should be differentiable from the most other infrastructure images. Hence, the task is not



Table 3.5: Road infrastructure dataset description.

Class	Search Area	Search Criteria	Nodes
(i) Highway	Upper Bavaria	A9, A92, A93, A99	5 447
(ii) Rural	Neub. Schrobenh.	70, 80, 90, 100 km/h	5 636
(iii) City 50	Ingolstadt	50 km/h	4 604
(iv) City 30	Ingolstadt	30 km/h	5 152
(v) City 5	Ingolstadt	5 km/h	1 146

to differentiate between highway and not highway, but the task is to differentiate between unknown and known infrastructure images.

### 3.3.3.5 Data Generation

The dataset generation used for this application is part of this work. Five different infrastructure classes with a total number of 21 985 images are generated. The classes are (i) highway, (ii) rural roads; and inner-city roads with a speed limit of (iii) 50 km/h, (iv) 30 km/h, and (v) 5 km/h. The detailed steps are described below.

The images are generated from map data which is provided by *OpenStreetMap* (OSM) [Ope19]<sup>3</sup>. In the first step, for a given geographic area, all nodes<sup>4</sup> which fit the search criteria of the class are extracted using the Overpass API<sup>5</sup>. All the data is extracted from the region of Ingolstadt, Germany.

The highway nodes (i) are extracted by using Upper Bavaria as overall search area and the highways A9, A92, A93, and A99. The district of Neuburg Schrobenhausen, which borders Ingolstadt, is used to extract the rural road nodes (ii). There, all nodes which have a speed limit of either 100 km/h, 90 km/h, 80 km/h or 70 km/h are considered. For the inner city nodes (iii) - (v), Ingolstadt is selected as search area. For the classes (iii) and (iv), the corresponding speed limits are used as filters. In the case of class (v), the road type is filtered for living streets. The various settings as well as the number of resulting nodes is summarized in Table 3.5. The positions of the nodes are visualized in Figure 3.10.

In the second step, each node is considered as the center of a bounding box with the size of 100 m  $\times$  100 m. The corresponding maps are downloaded and then converted into the OpenDRIVE format [D<sup>+</sup>15] using the `netconvert` tool of SUMO [LBBW<sup>+</sup>18]. As part of this work, a tool is developed to generate adjustable images given OpenDRIVE maps as input. The following elements are not rendered for the image generation: bike lanes,

<sup>3</sup>Map data copyrighted OpenStreetMap contributors and available from <https://www.openstreetmap.org>.

<sup>4</sup>Nodes in OSM define a single position on the map. A way consists of an ordered list of nodes, defining the shape of the road.

<sup>5</sup>Overpass API: <https://overpass-turbo.eu/> online data filtering tool for OSM

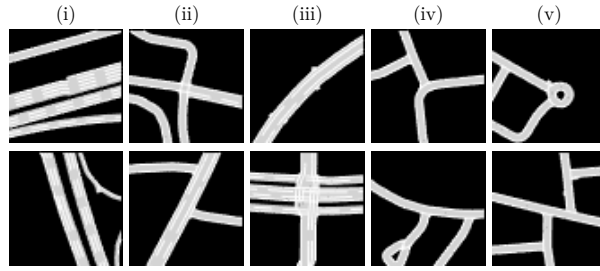


Figure 3.11: Example images of the different road infrastructure classes (Table 3.5).

sidewalk, restricted roads, rails, traffic signs, traffic signals, and zebra crossings. Since this application is considered as proof of concept, omitting such infrastructure information is assumed to be acceptable. The generated images are of size  $64 \times 64$  pixels. The lane surface itself is colored in gray, the lane markings are given in white, and the remaining area is set to black. In Figure 3.11 some example images for the different classes are depicted.

### 3.3.3.6 Results

The evaluation of the outlier scores follows the same intuition as in the general experiments. Hence, the outlier detection remains unsupervised. The classes are only used for the evaluation. Here, the highway class is considered to be the respective inlier dataset. All the remaining classes are considered to be outliers. In terms of the application, this can be thought of having a base dataset  $D_{\text{base}}$  collected from highways infrastructure and compare freshly gathered data from other infrastructures to the base dataset. Then the novelty of the freshly observed data in regard to the base dataset is evaluated by the outlier score. As described in Section 3.3.3, each image of the freshly recorded data (classes (ii)-(v)) is used separately to evaluate the outlier score compared to the base dataset (highway).

For this application, the number of neighbors is varied as  $k \in \{5, 15, 40, 60, 80, 100\}$ . As before, for each  $k$ , the  $AUC$  values of all outliers are averaged as  $\overline{AUC}$ . The results, when using the classes (ii) to (v) as outliers are depicted in Figure 3.12. The results based on the methods which use the UMAP definitions are colored in red, blue for the Barnes-Hut-SNE, and black for the state-of-the-art methods. As it becomes clear from the plots, the scores based on UMAP and Barnes-Hut-SNE perform more stable with respect to variations of  $k$  than the state-of-the-art methods (Note: Here only LDOF is shown, since it was the overall best performing score in terms of stability). This property is crucial, since in the unsupervised task to detect novel road infrastructures, the appropriate number of neighbors  $k$  is not known and can not be determined as in this setting. Therefore, the following discussion considers the overall performance for all given  $k$ .

Comparing the plots of the various classes, a clear tendency is that the performance increases from Rural to City 5. This characteristic is intuitive since rural roads are more similar to highway roads than city roads. Overall, the scores based on UMAP

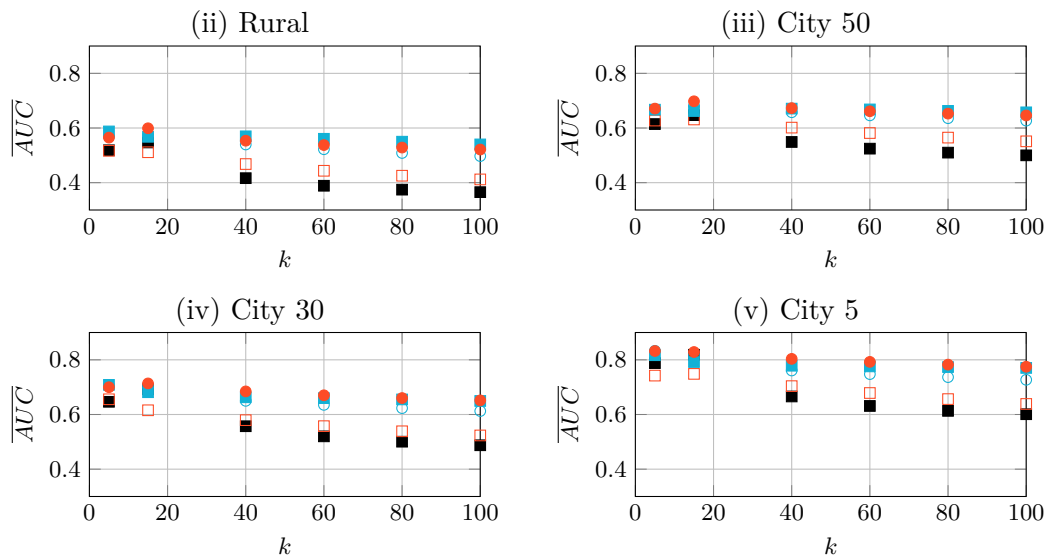


Figure 3.12: Outlier detection performance  $\overline{AUC}$  over various number of neighbors  $k$ . The highway dataset (class (i)) is considered as base dataset. For each experiment, the samples from the classes (ii)-(v) are checked with respect to their outlier score. The more likely they are all an outlier the better, since they can be considered as novel with respect to the base dataset.  $\bullet$   $ULEF$ ,  $\square$   $USOS$ ,  $\circ$   $tLEF$ ,  $\blacksquare$   $KNNSOS$ ,  $\blacksquare$   $LDOF$ .

and Barnes-Hut-SNE are superior to the state-of-the-art methods.  $USOS$ , which shows good performance for the benchmark datasets, performs the worst of the UMAP/Barnes-Hut-SNE-based outlier scores in this application. In comparison, the introduced  $ULEF$  score performs better. This shows that the  $ULEF$  score is preferable if UMAP is used. On a comparable level to  $ULEF$  performs the  $KNNSOS$  score based on Barnes-Hut-SNE. To conclude this, based on the application of road infrastructure images one should opt for the novel introduced score  $ULEF$  when using UMAP and for  $KNNSOS$  when using Barnes-Hut-SNE. The performance difference between the scores is negligible.

To highlight the capabilities further, a dataset for an exemplary drive is constructed. For this purpose, images for all nodes of the highway A9 from Munich to Ingolstadt-South are collected, but here, the orientation of the images is changed, such that the driving direction is pointing upwards. The driving direction is determined by the position of the current node and the position of the next node on the road. The images collected this way are used as base dataset. Then, using the same orientation correction, the images for the route from Ingolstadt-South to the Technische Hochschule Ingolstadt are extracted and tested if they fit in the base dataset. For the first part, this route stays on the highway and in the second part the route enters inner city infrastructure. In Figure 3.13, the route is shown, it starts from the bottom. The points represent the used nodes (best viewed with magnification). The color represents the outlier score relative to the range of the outlier scores of the base dataset, where red indicates a high outlier score and green a

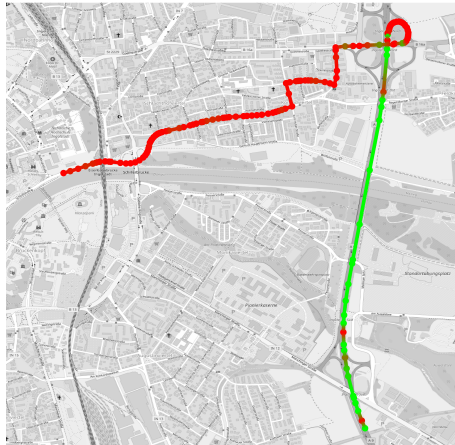


Figure 3.13: Outlier score for the test route, given a base dataset A9 (highway). Each road infrastructure image is represented by a point, while the points are connected to visualize the route. Red: Outlier, Green: Inlier

low. The first part of the route along the highway results in a low outlier score visualized green, and hence already known infrastructure images, which is reasonable, since the base dataset is also a highway (A9). However, some images seem to differentiate even for the highway, since some are marked red. The first red one for example, represents an infrastructure, where an additional lane on the north heading direction is added. Hence, it is most probable that such a constellation is not part of the base dataset. The other red highway points are mainly due to the shape of the merging/leaving lanes. As expected, for the inner city, the most nodes are considered as outliers. However, the first part after the highway tends a bit more towards green. This is plausible since this part of the road has two lanes on each side and a separation between both directions.

### 3.3.4 Conclusion

The novel outlier score  $LEF$  is presented in this section. The score is designed in such a fashion that it can be embedded into dimensionality reduction techniques which use a directed graph in the first phase. The  $LEF$  is applied in UMAP ( $ULEF$ ) and Barnes-Hut-SNE ( $tLEF$ ). Besides the  $LEF$ , the score  $USOS$  is introduced, which is the application of the KNNSOS [SG17] within UMAP. Both scores strongly focus on local out- and inliers. A key factor of both scores is that the same definitions of similarity as in the dimensionality reduction are used. Furthermore, if the dimensionality reduction is performed, the calculation of the scores is of low costs because the required graph has already been constructed. Nevertheless, the scores can be used as standalone, by just constructing the directed graphs without performing the actual embedding.

The scores are applied to outlier detection benchmark datasets and evaluated in comparison to other  $K$ -nearest neighbor outlier scores. Comparing the best achieved accuracy,

given the optimal number of neighbors, the scores  $ULEF$ ,  $tLEF$ , and  $USOS$  are at a comparable level with the state-of-the-art methods. In terms of robustness to the number of neighbors selected, the new scores are superior to the state-of-the-art methods, since the average accuracy over all  $k$ -s is higher. This fact is of special interest, since in unsupervised tasks, the optimal number of neighbors is not known.

Besides the application to the outlier detection benchmark dataset, a special focus is the application to novelty detection for road infrastructure images. This is required for identifying new and representative scenarios, as it is crucial for the validation process of AVs. The novel score  $ULEF$  alongside with the score  $KNNSOS$  are the overall best performing in identifying novel road infrastructures. UMAPs advantages [MHM18] in handling large-scale datasets and preserving global data structures, makes  $ULEF$  the recommendation of the analyzed methods for the purpose of outlier detection in combination with dimensionality reduction.

Even though the introduced scores showed equally and more robust performance to other state-of-the-art neighborhood-based outlier scores, the performance for image data is still low. Therefore, for image data, or also other more complex data inputs, alternative methods have to be considered for outlier detection. One approach might be to investigate methods that handle image data directly, such as reconstruction-based methods. Another approach could be to transform the complex data into representations which are better suited for simple outlier methods like the neighborhood-based methods introduced in this section.

### 3.4 Alternative Outlier Detection Methods

As highlighted in the section before, the analyzed outlier methods, as well as the introduced  $LEF$ , suffer from poor performance when applied on image data, specifically the application to infrastructure images. To identify representative and novel traffic sceneries, one of the requirements is to have a reliable outlier detection method. In this section, some alternative outlier detection approaches and their performance on infrastructure images are investigated. The approaches are either vector-based, where the input has to be in form of a vector, or image-based, where the input has to be an image.

#### 3.4.1 Vector-Based Methods

The methods used in the previous section, e. g.,  $LEF$ ,  $LOF$ ,  $KNNSOS$ , are all vector-based methods. Therefore, the infrastructure images are simply vectorized before processing them with the outlier detection methods. Here, some more vector-based methods are briefly discussed and their application to infrastructure images is shown.

**ABOD** The method *Angle-Based Outlier Detection* (ABOD) is based on the neighborhood of data points, like the methods introduced in the previous section. The method and its mathematical definition is summarized in Section 2.3.1. The method investigates the angle between all possible vector pairs inside a neighborhood. A high angle variance within a neighborhood indicates low outlier strength for the investigated data point. Here, the number of neighbors for ABOD is set to  $K = 15$  for all experiments.

**IF** The *Isolation Forest* (IF) is used for the analysis as well. The concept behind the IF is that outlying points are easily separable, hence only few splits in a tree are required. More details about the IF can be found in Section 2.3.2. Here, the number of trees is set to 500, while all other parameters are kept as in the original IF.

**OCSVM** With the OCSVM, another vector-based method is investigated. The OCSVM aims to separate all data points from the surrounding in the input space, by separating the data from the origin with a hyperplane in the projected space, see Section 2.3.3. The radial basis function is used as kernel, where  $\gamma = \frac{1}{N\text{Var}_x}$  and  $\nu = 0.5$ .

#### 3.4.2 Image-Based Methods

Instead of using methods specialized for vector data, using methods optimized for image data could be more beneficial, since the traffic scenarios are represented as infrastructure images. For this purpose, the following reconstruction-based approaches are used for the investigations.

**Autoencoder** A convolutional Autoencoder is trained and the reconstruction error is used as outlier strength indicator (ROD), as discussed in Section 2.3.4.1. The encoder used here consists of four convolution layers<sup>6</sup>, each followed by a ReLU layer, and a final FC layer. The latent representation has a dimensionality of  $N_{\text{AE}} = 32$ . The decoder uses six transposed convolution layers<sup>7</sup>, each except the last is followed by a ReLU.

**SAP** Using the convolutional autoencoder as defined above, the *Simple Aggregation along Pathway* (SAP) can be determined as shown in Section 2.3.4.2. It aims to utilize the hidden feature maps as well for the outlier score estimation.

**f-AnoGAN** The last reconstruction-based approach is f-AnoGAN. In contrast to the before mentioned methods, it is using a GAN and an encoder to determine the outlier strength of an image. Details about f-AnoGAN can be found in Section 2.3.4.3. The network used here is similar to the one used in the original work [SSW<sup>+</sup>19].

---

<sup>6</sup>Kernel sizes: 3-3-3-3; Kernels: 64-32-32-16; Stride:2-2-2-2; Padding: same-same-same-same

<sup>7</sup>Kernel sizes: 4-3-3-3-3-3; Kernels: 16-32-32-32-64-1; Stride:4-2-2-2-2-1; Cropping: same-same-same-same-same-same

Table 3.6:  $\overline{AUC}$  for the different methods across various outlier types.  $\mathcal{D}_{\text{base}}$ : highway; Bold: best overall; Underlined: best vector-based method.

	LOF	ULEF	ABOD	IF	OCSVM	ROD	SAP	f-AnoGAN
City 5	0,665	0,773	0,738	0,201	0,252	<b>0,981</b>	0,969	0,860
City 30	0,577	0,650	<u>0,710</u>	0,165	0,211	<b>0,889</b>	0,876	0,756
City 50	0,634	0,644	<u>0,729</u>	0,256	0,315	<b>0,899</b>	0,889	0,760
Rural	0,543	0,519	<u>0,655</u>	0,184	0,235	<b>0,763</b>	0,757	0,737
Mean	0,586	0,612	<u>0,700</u>	0,196	0,247	<b>0,855</b>	0,845	0,758

### 3.4.3 Experiments

The experimental setting is similar to the first one in Section 3.3.3.5. Therefore, all methods use the highway images as the base dataset  $\mathcal{D}_{\text{base}}$ . Then, all images from the classes (ii)-(v) are considered as outlier, and it is investigated how well the different outlier scores reflect that. In Table 3.6, the resulting  $\overline{AUC}$  per outlier group and method are listed, where Mean highlights the overall outlier performance with respect to  $\mathcal{D}_{\text{base}}$ . For reference, also the neighborhood-based method LOF and the introduced score ULEF are listed.

Among the vector-based methods, the neighborhood-based methods perform the best, especially ABOD, which outperforms the other vector-based methods clearly. The IF and the OCSVM seem to struggle on infrastructure images, since they reach only low performance values. The image-based methods, have a clear advantage over the other methods. The simple reconstruction error (ROD) achieved the highest performance, slightly above the SAP score. The f-AnoGAN performed the worst among the image-based methods.

### 3.4.4 Conclusion

The vector-based outlier detection methods are not optimized for the application to image data. Especially for this use case with infrastructure images, they are clearly outperformed by image-based outlier detection methods. Therefore, one possibility to still be able to use simple vector-based outlier detection methods like LOF, would be to find a different more suitable representation for the infrastructure images or even more for the whole traffic scenario. The other obvious path would be to use one of the image-based methods. However, if a more suitable representation for traffic scenarios is available, it would be most probably also helpful for tasks like clustering or visualization purposes to further analyze the traffic scenario dataset. This would not necessarily be available through image-based outlier detection methods. Hence, one question arises: How to represent traffic scenarios, such that simple outlier detection methods and clustering methods can be used for traffic scenario identification?



## 3.5 Domain Knowledge-Based Methods

Instead of using data-driven methods like in the previous sections, domain knowledge-based methods can be used to identify representative and new traffic scenarios. In this section two main approaches are covered, namely criticality-based approaches and category-based approaches. The presented methods have been implemented prototypically as part of this research work, since they provide a fallback strategy, in order to safely detect critical or obviously new scenarios.

### 3.5.1 Criticality-Based

One obvious way to identify a subgroup of relevant traffic scenarios is to identify the critical ones, since in the concept of scenario-based testing, an AV should perform well even or especially under critical conditions. The criticality of a scenario can be determined through different criticality measures. In this work, different criticality measures are used in parallel here, where also different levels of available information is assumed. This is due to the fact that more sophisticated criticality measures like the safety distance from [SSS17] require a lot of well processed information. However, due to possible errors in the processing chain, also low-level information sources should be used for the identification of critical scenarios. Three levels of information availability are used here: (i) Only the EGO vehicle dynamics. (ii) The EGO vehicle dynamics and the surrounding objects (OBJs). And (iii) like (ii) but additionally including map and position information of the EGO vehicle.

#### 3.5.1.1 Dynamic Impact

Relying only on dynamic information of the EGO vehicle, as defined by (i), a rather simple criticality detection mechanism is used here. Assuming that a normal and safe driving would not lead to high accelerations, allows monitoring only the accelerations of the EGO vehicle to detect critical scenarios. For example, in case of emergency braking, the deceleration would be much stronger than in normal driving. Furthermore, in scenarios that involve crashes the accelerations would exceed the ones of a normal driving pattern even more. In [DKN+06], the longitudinal and lateral accelerations are part of the trigger criteria to identify crashes, near-crashes, and other incidents. Whenever the accelerations exceed some threshold, the event is triggered. In the scope of this work, those thresholds are used as low-level criticality estimation. The thresholds are selected according to [DKN+06] as

$$a_{\text{lat,max}} \quad 7 \text{ m/s}^2 \text{ and} \\ a_{\text{lon,max}} \quad 6 \text{ m/s}^2.$$

For the identification of a threshold exceedance, the absolute values  $|a_{\text{lat}}|$  and  $|a_{\text{lon}}|$  have to be used.



In order to lower false positive triggers, the threshold has to be violated for several timestamps. In case of many false positives, the trigger thresholds could be increased. Of course the trigger values also heavily depend on various factors, such as the vehicle type. Even though the dynamic impact can serve as criticality indicator, such a low-level trigger should be used only as a fallback solution. For more sophisticated criticality estimation, also the environment has to be considered.

### 3.5.1.2 TTC, THW, and RF

In order to determine the criticality of a traffic scenario, various approaches are known and used nowadays. The criticality measures used here are *Time To Collision* (TTC), *Time HeadWay* (THW), and *Risk Feeling* (RF). Each criticality measure requires information of the EGO vehicle and surrounding OBJs, as defined above by (ii). The narrative of the used criticality measures is simple, the smaller the TTC and THW, the more critical the scenario, since the possibilities to prevent a crash decrease.

In the following, a framework for efficient TTC, THW, and RF calculation is presented. Whenever the criticality measures violate certain thresholds for several timestamps, the scenario is considered to be critical. The framework follows definition (ii) and hence relies on the following dynamic information:

EGO: velocity, acceleration, yaw rate;

OBJ: velocity, position relative to EGO.

The TTC returns the time until a crash between two vehicles will happen, given some assumptions. Here, the assumptions are limited to: (a) The EGO vehicle's future movement is modelled with a simple vehicle model, keeping the turn rate and acceleration constant. (b) An OBJ's movement is modelled with an even simpler model, assuming constant velocity only. This way, the modelling of the future movement does rely only on few dynamic information of the surrounding vehicle. The TTC calculation in this work will return the time to a possible crash between the EGO and an OBJ, using the two vehicle models.

THW can be understood as the time gap to another vehicle. Unlike the TTC, no crash is assumed here, nevertheless very low THW values are critical<sup>8</sup>. Like for the TTC calculation, the EGO vehicle's movement is modelled with constant turn rate and acceleration. The THW calculation returns the time until the EGO vehicle possibly reaches the current position of another vehicle.

---

<sup>8</sup>For example travelling on a highway behind another car with almost no distance but the very same speed. This would not lead to a crash, hence no TTC can be determined. Still the situation is critical since if the same speed condition changes, a crash can happen immediately. THW reflects this criticality in measuring the time gap between the vehicles. Typically, THW finds application in traffic laws to define safe distances to other vehicles.

Table 3.7: Velocity and acceleration assumptions for TTC and THW.

	TTC	THW
$\mathbf{v}_{\text{EGO}} =$	$\mathbf{v}_{\text{EGO}}$	$\mathbf{v}_{\text{EGO}}$
$\mathbf{a}_{\text{EGO}} =$	$\mathbf{a}_{\text{EGO}}$	$\mathbf{a}_{\text{EGO}}$
$\mathbf{v}_{\text{OBJ}} =$	$\mathbf{v}_{\text{OBJ}}$	0
$\mathbf{a}_{\text{OBJ}} =$	0	0

**TTC and THW Calculation** The calculation of the TTC and the THW is realized in the same framework. In order to reduce computation cost, some dynamic pre-filtering is applied, this way, only OBJs which are of interest for the calculation are determined. The actual TTC and THW calculation is determined by simulating the future movement.

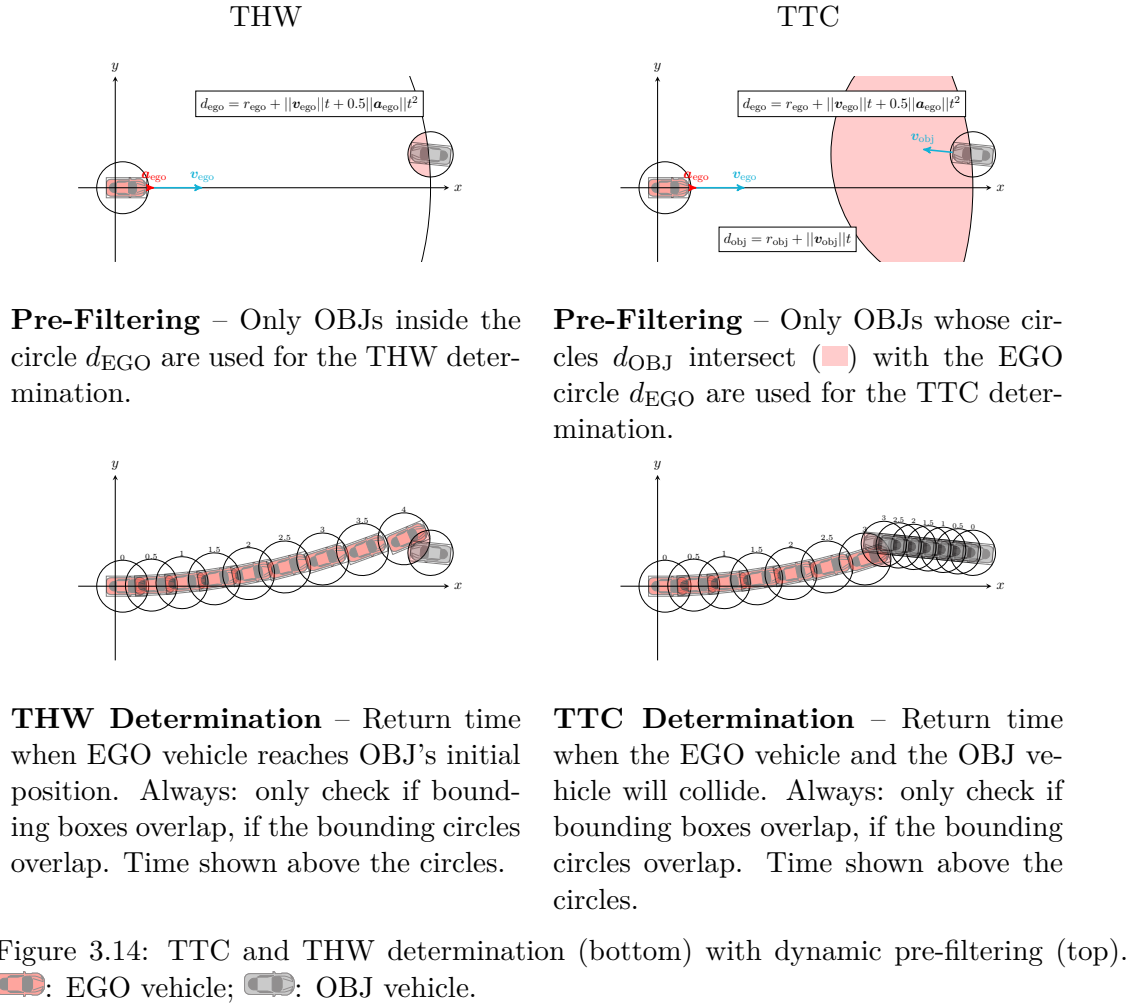
The dynamic pre-filtering is realized on some worst-case assumptions. For this purpose, the maximum travel distance  $d_{\text{max}}$  is determined for the EGO and the OBJ. Only if an OBJ is closer than

$$d_{\text{safe}} = d_{\text{max,EGO}} + r_{\text{EGO}} + d_{\text{max,OBJ}} + r_{\text{OBJ}} \quad (3.36)$$

to the EGO, it will be considered for the THW or TTC calculations, Here  $r$  is the radius, denoting the size of the vehicles and the maximum travel distance is calculated as

$$d_{\text{max}} = \|\mathbf{v}\|t_{\text{max}} + 0.5\|\mathbf{a}\|t_{\text{max}}^2, \quad (3.37)$$

where  $\mathbf{v}$  and  $\mathbf{a}$  are selected according to Table 3.7, dependent on whether TTC or THW needs to be determined. The maximum considered time span  $t_{\text{max}}$  should be selected greater than the threshold for TTC and THW. With the dynamic pre-filtering, the computational effort for the TTC and THW calculation is reduced, since only a subset of vehicles is considered. For each OBJ which passed the pre-filtering, a possible TTC or THW is determined. For this, the vehicle dynamics as summarized in Table 3.7 are used. The EGO's trajectory is then modelled with a constant turn rate and acceleration model for several timestamps  $t \in \{0, t_{\Delta}, 2t_{\Delta}, \dots, t_{\text{max}}\}$ ,  $t_{\Delta}$  is the simulation delta time. The same holds for the OBJ vehicle where instead, the constant velocity model is used. If the velocity and acceleration are 0, the future positions are not calculated since they remain constant over time. For each timestamp, it is checked whether the bounding boxes of the vehicles overlap. If they overlap, the procedure ends and the timestamp of the first overlap will be reported as the TTC or THW. For the overlap determination, another two-stage process is utilized in order to keep computational effort low. For each simulated timestamp, first the distance between the vehicles' centers is determined. Only if that distance is smaller than  $r_{\text{EGO}} + r_{\text{OBJ}}$ , it is checked if the bounding boxes overlap. Since the dynamics as defined in Table 3.7 are used, the THW is calculated with keeping the



OBJ vehicle fixed. Hence, the same framework as for TTC can be used.

The overall process is visualized in Figure 3.14. The top part shows the pre-filtering, indicated with the circles ( $d_{EGO}$  and  $d_{OBJ}$ ), and red highlighted intersection area. Only if the circles ( $d_{EGO}$  and  $d_{OBJ}$ ) intersect, the OBJ is considered for the TTC or THW calculation, which is shown at the bottom.

RF is a weighted combination of TTC and THW and aims to mimic the human risk perception. It is defined as

$$\text{RF} = \frac{w_{\text{THW}}}{\text{THW}} + \frac{w_{\text{TTC}}}{\text{TTC}}, \quad (3.38)$$

where in this work, the parameters are selected as  $w_{\text{THW}} = 1$  and  $w_{\text{TTC}} = 4$ , since it was shown in [KYK+08] that the risk perception increases/decreases parallel to the  $1/\text{THW} + 4/\text{TTC}$  line.

The thresholds for TTC and THW can be selected rather low, while the thresholds for the RF can be selected high. This would lead to detecting only very critical scenarios, which is desirable for this application.

#### 3.5.1.3 RSS Safety Distance

The criticality measures TTC, THW, and RF are implemented in this work independent of the infrastructure. Including the infrastructure information can aid the criticality assessment, since the future movement can be hypothesized through the infrastructure rather than simple vehicle models. In [SSS17], definitions for safe distances in various situation types is provided. It is part of the so-called *Responsibility Sensitive Safety* (RSS) framework presented in [SSS17]. In the following, it is focused on three of the situation types presented in [SSS17]. A situation represents a pair of vehicles in a specific infrastructural setting. The situation types are defined as:

- (i) the vehicles are driving on parallel routes in the same direction,
- (ii) the vehicles are driving on parallel routes in the opposite direction, and
- (iii) the vehicles are driving on overlapping routes.

In the following, only situations between the EGO vehicle and each of the OBJ vehicles are considered. For each situation type, safe distances are defined. If those distances are obeyed, the situation is safe.

**Parallel Routes (i) & (ii) – Projection** Before determining whether two vehicles are in safe distance to each other, the situation is projected into a lane-based coordinate system. In contrast to [SSS17], the project in this work is simplified and for situation types (i) & (ii), parallel routes in same and opposite direction, realized as follows. For each vehicle a reference line is assumed to be given. The reference lines of the two vehicles have to be parallel for at least some interval. For example, consider the case where two vehicles drive on a road with two lanes in opposite direction, the reference line (centerline) would be the same for both vehicles. Such a situation is illustrated in Figure 3.15.

In the following, the projection of a situation into the situation coordinate system  $(\tilde{x}^{(\text{sit})}, \tilde{y}^{(\text{sit})})$  is explained. The projection is also illustrated in Figure 3.15. The vehicles are described through their Frenet states  $(\tilde{x}, \tilde{y})$ , given the reference line<sup>9</sup>. This way, the maximum and minimum positions of a vehicle’s bounding box are determined, leading to  $\tilde{x}_{\max}$ ,  $\tilde{x}_{\min}$ ,  $\tilde{y}_{\max}$ , and  $\tilde{y}_{\min}$ . The situation in the situation coordinate system  $(\tilde{x}^{(\text{sit})}, \tilde{y}^{(\text{sit})})$  is constructed by considering the bounding box positions for both vehicles, and by projecting the velocity vectors according to  $(\tilde{x}, \tilde{y})$ <sup>10</sup>. As can be seen, the orientation of the vehicles are not considered in the situation coordinate system. If the reference lines are not the same, the projection is realized individually first. Then the coordinate systems are linked where the reference lines first become parallel. Each of the following safety distance calculations

---

<sup>9</sup>To determine the Frenet states, points are projected onto the reference line using the normal direction of the reference line.

<sup>10</sup>Projecting an input vector according to  $(\tilde{x}, \tilde{y})$  is realized by projecting it onto the normal and tangent vector of the reference line at the origin of the input vector.

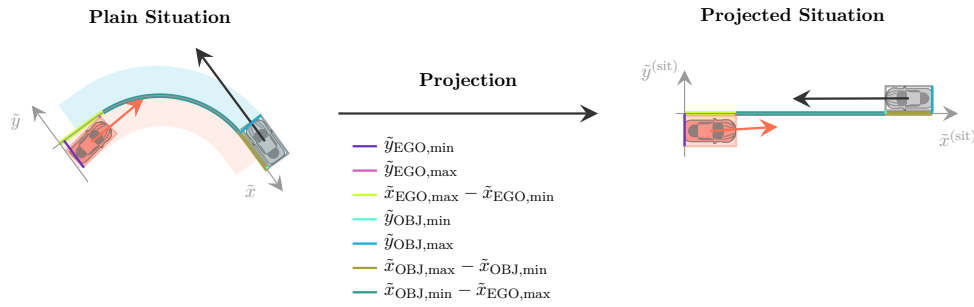




Figure 3.15: Visualization of the projection for parallel routes – RSS. Left: plain situation; Right: projected situation; : EGO vehicle; : OBJ vehicle. The black and red arrows illustrate the velocity vectors.

for type (i) & (ii) are based on a situation projected as described above. For a parallel routes situation of type (i) & (ii) to be critical, the lateral and longitudinal safe distance have to be violated.

**Parallel Routes (i) & (ii) – Lateral Safe Distance** The lateral safe distance indicates the lateral distance, two vehicles should have in a projected scenario. If the safe lateral distance is not violated, no further checks are required. The safe lateral distance, given a projected scenario is calculated as [SSS17]

$$d_{\text{lat},\text{min}} = \mu + \left[ \frac{v_l + v_{l,\rho}}{2} \rho + \frac{\text{sgn}(v_{l,\rho})v_{l,\rho}^2}{2a_{\text{min,brake}}} - \left( \frac{v_r + v_{r,\rho}}{2} \rho + \frac{\text{sgn}(v_{r,\rho})v_{r,\rho}^2}{2a_{\text{min,brake}}} \right) \right]_+, \quad (3.39)$$

where  $[\cdot]_+ = \max\{\cdot, 0\}$  and with:

$v_l$  := Lateral velocity left vehicle

$v_r$  := Lateral velocity right vehicle

$a_{\text{max,accel}}$  := Maximum lateral acceleration (abs)

$a_{\text{min,brake}}$  := Minimum lateral braking (abs)

$\mu$  := Lateral fluctuation distance

$\rho$  := Reaction time

$v_{l,\rho} := v_l + \rho a_{\text{max,accel}}$

$v_{r,\rho} := v_r - \rho a_{\text{max,accel}}$

The two lateral velocities are determined through the projection. The other values are hyperparameters to the safety distance. The safe lateral distance is calculated based on the worst case assumption, that during the reaction time  $\rho$  both vehicles will laterally accelerate towards each other with  $a_{\text{max,accel}}$ . After  $\rho$  both vehicles will stop moving

towards each other with a deceleration of  $a_{\min,\text{brake}}$ .

If the lateral distance of the two projected vehicles is less than  $d_{\text{lat},\min}$ , the situation is potentially critical and needs to be analyzed further.

**Parallel Routes Same Direction (i) – Longitudinal Safe Distance** If the situation is of type (i) (parallel routes with same directions), then the safe longitudinal distance is calculated as follows. Assume that one vehicle is driving in front (front vehicle), while one vehicle is driving behind the front vehicle (rear vehicle). Like before, a worst-case assumption is used. The safe longitudinal distance expresses the longitudinal distance, that would be required if the front vehicle suddenly brakes, while the rear vehicle will only react and brake after some reaction time  $\rho$ . Moreover, during  $\rho$ , the rear vehicle will accelerate. The safe longitudinal distance is defined for a projected situation of type (i) as

$$d_{\text{lon},\min,\text{same}} = \left[ \frac{v_{\text{r}} + v_{\text{r},\rho}}{2} \rho + \frac{v_{\text{r},\rho}^2}{2a_{\min,\text{brake}}} - \frac{v_{\text{f}}^2}{2a_{\max,\text{brake}}} \right]_+, \quad (3.40)$$

here the following holds:

$v_{\text{f}}$  := Longitudinal velocity front vehicle

$v_{\text{r}}$  := Longitudinal velocity rear vehicle

$a_{\max,\text{accel}}$  := Maximum longitudinal acceleration (abs)

$a_{\min,\text{brake}}$  := Minimum longitudinal braking (abs)

$a_{\max,\text{brake}}$  := Maximum longitudinal braking (abs)

$\rho$  := Reaction time

$v_{\text{r},\rho}$  :=  $v_{\text{r}} + \rho a_{\max,\text{accel}}$ .

**Parallel Routes Opposite Direction (ii) – Longitudinal Safe Distance** In contrast, for situations when driving on parallel routes but in opposite direction (ii), the worst-case assumption changes for the safe longitudinal distance calculation. One vehicle is assumed to travel in the correct direction, hence the correct lane for the travel direction. The other vehicle is assumed to travel in the wrong direction, overtaking etc. Here, the worst case would be that both vehicles accelerate during  $\rho$  before coming to a complete stop. The safe longitudinal distance is defined for a projected situation of type (ii) as

$$d_{\text{lon},\min,\text{opposite}} = \frac{v_1 + v_{1,\rho}}{2} \rho + \frac{v_{1,\rho}^2}{2a_{\min,\text{brake,correct}}} + \frac{|v_2| + v_{2,\rho}}{2} \rho + \frac{v_{2,\rho}^2}{2a_{\min,\text{brake}}} \quad (3.41)$$

with:

$$\begin{aligned}
v_1 &:= \text{Longitudinal velocity correct direction} \\
v_2 &:= \text{Longitudinal velocity wrong direction} \\
a_{\max, \text{accel}} &:= \text{Maximum long. acceleration (abs)} \\
a_{\min, \text{brake}} &:= \text{Minimum long. braking (abs)} \\
a_{\max, \text{brake, correct}} &:= \text{Minimum long. braking correct dir. (abs)} \\
\rho &:= \text{Reaction time} \\
v_{1, \rho} &:= v_1 + \rho a_{\max, \text{accel}} \\
v_{2, \rho} &:= |v_2| + \rho a_{\max, \text{accel}}.
\end{aligned}$$

**Parallel Routes (i) & (ii) – Critical Scenario Detection** As already stated above, for any situation of type (i) or (ii) to be critical, both, the safe longitudinal distance ( $d_{\text{lon, min, opposite}} / d_{\text{lon, min, same}}$ ) and the lateral distance ( $d_{\text{lat, min}}$ ) have to be violated in the projected situation. This logic can be applied to implement a more sophisticated detection method for critical scenarios, as required for scenario-based testing. In case too many scenarios are detected, one possible solution could be the adjustment of the hyperparameters ( $a_{\max, \text{accel}}$ ,  $a_{\min, \text{brake}}$  etc.).

**Overlapping Routes (iii) – Projection** For a situation with overlapping routes (iii), the projection is slightly different from the one above. In a first step, a driving tube for each vehicle is determined. The driving tube is constructed under the worst-case assumption from the lateral distance calculation ((i) & (ii)): the vehicle will accelerate laterally during  $\rho$  and will stop moving laterally after  $\rho$ . This way, the driving tube includes all possible lateral movements of the vehicle to the left and to the right. Therefore, to construct the driving tube, the lateral position to the reference line as well as the initial lateral velocity is determined using Frenet states. After the construction of the driving tubes, the intersection area of the tubes is determined. Then, the longitudinal distance of each vehicle along its driving tube to the start and end of the intersection is determined. In the case of a merging situation, only the start of the intersection area is considered for further calculations. Figure 3.16 roughly depicts the projection for the overlapping route case.

**Overlapping Routes (iii) – Safe Distance** For the safe distance calculation, the driving tubes are used as reference, where it is assumed that the vehicle can laterally be positioned everywhere inside the tube. Therefore, the safe distance only depends on the longitudinal actions, given the driving tubes.

The following definition is used to estimate if the vehicles are at a safe distance:

**If** The non prioritized vehicle can stop before entering the route of the prioritized vehicle.

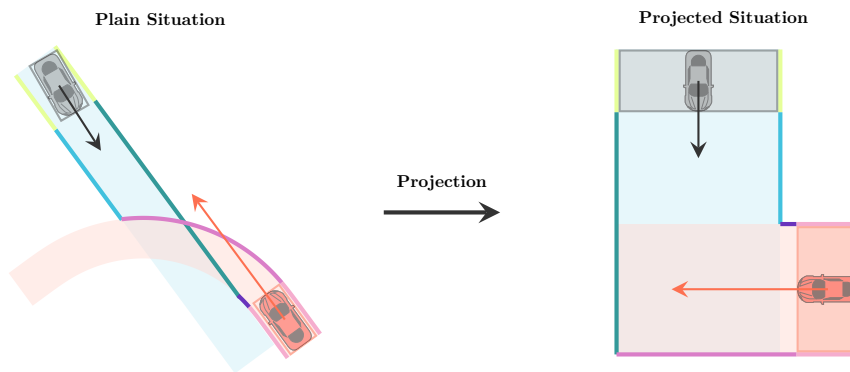




Figure 3.16: Visualization of the projection for overlapping routes – RSS. Left: actual situation with the driving tubes; Right: projected situation; : EGO vehicle; : OBJ vehicle.

**Else if** The prioritized vehicle can stop before entering the route of the non prioritized vehicle OR, for merging situations, the vehicle in front stays in front, after both apply a braking pattern.

**Else if** For all points of the overlapping area, the time intervals  $[t_{\min}, t_{\max}]$  to reach this point of both vehicles does not intersect.

For each case, the worst-case assumptions are used. The last case is realized by determining the earliest time a vehicle would enter the intersection area (accelerating case) and the latest time the vehicle would leave the intersection area (deceleration case). If the time intervals of two vehicles for the crossing area intersect, the required safe distance is not given.

**Implementation** The complete calculation of the former summarized safe distances was realized prototypically. That included the situation type detection and projection. The implementation is mainly based on the network graph representation of the given map, to reduce computational effort.

#### 3.5.2 Category-Based

While detecting critical scenarios is an important component of identifying relevant scenarios, detecting novel traffic scenario types can be of interest as well. Novel scenarios, the AV has not seen yet in the development phase and have not been tested as well, might be beneficial to be added to the test catalog for future testing. This section presents an approach to identify novel traffic scenarios and enable categorization. Here, no data-driven approaches are used. A traffic scenario is transformed into a different representation which can be used for novelty detection and categorization.



Like the safety distance of the RSS, this method utilizes the EGO information, the surrounding OBJs, and a map of the surrounding traffic infrastructure. The narrative is to transform the scenario description (EGO, OBJs, map) into a graph representation. The steps generating the graph consists of constructing the infrastructure graph, assigning the EGO and optionally the OBJs to the graph nodes. Finally, the graph is simplified and pruned. In [ZZ22], another graph representation for traffic scenarios is presented. In contrast, there the relative relations between traffic participants are encoded as well.

**Graph Construction – Infrastructure** Transforming the infrastructure into a graph representation is the first step of the scenario representation transformation. Assume, that the infrastructure is present in a format where it consists of lane pieces (e. g., Lanelet [PPJ<sup>+</sup>18]). Further assume that the lane pieces are as large as possible, such that they are only split if the semantic attributes change (e. g., neighbors, speed limit, traffic light), or if the lane enters a splitting/merging situation (e. g., intersection, merging lane). Then, each lane piece is represented as a vertex in a graph, and it is connected to other vertices through directed weighted edges according to the reachability and neighborhood relationship: 1 for directly reachable (same direction neighbors and successor lane pieces) and 2 for neighbors with opposite direction. This way, a weighted directed graph is constructed representing the infrastructure connectivity. The attributes underneath the vertices (e. g., speed limit, length, width, curvature) can be later used for more detailed comparison purposes. Given the graph representations, two infrastructures can easily be compared in terms of connectivity. However, the graph misses the actual constellation of the OBJs over time. Moreover, the graph might be too big to properly describe the scenario.

**Graph Construction – EGO and OBJ Assignment** Transforming the EGO and OBJs into the graph representation is straight forward. Each trajectory point is assigned to the lane pieces the vehicle is currently on. This way a route can be extracted, which is a list of lane pieces without successive repetitions. Each of those lists can be assigned to the graph vertices as well. This way, the EGO and OBJ movement over time is transformed into routes in the graph. Whether to use the OBJs’ routes for analysis or not can be decided dependent on the available size for storing scenarios. Since it is likely to detect more novel graphs when using all OBJs compared to when only using the EGO vehicle, one trade-off could be to use the full EGO route while only using the OBJ starting positions, to reduce the number of novel scenarios and hence the required storage requirements. However, this is a hyperparameter and can be selected based on the scenario catalog size one wants to identify.

**Graph Construction – Simplification and Pruning** In a final step, simplifying, and pruning the resulting graph is performed. The simplification is only required if the lane pieces are not as big as possible, as it was assumed in the infrastructure graph

construction. Simplification aims to recover this assumption, by merging all successive vertices where the attributes do not change. This is required to enable a standardized format for meaningful comparisons between two graphs. Finally, unnecessary parts of the infrastructure graph can be removed. For this purpose, all vertices are selected, which the EGO vehicle is assigned to. Then, all neighboring vertices are added to that set. After that step, all vertices of each intersection, that might be included in the so far collected set are added. Furthermore, all vertices starting and leaving at any included intersection vertex are included. This way, only the infrastructure that is relevant to the EGO vehicle is kept. OBJs that permanently stay outside the selected infrastructure can be discarded. It should be noted that it might be useful to add another look-a-head distance to the EGO's last trajectory point. Vertices within that look-a-head distance can be further included into the previous procedure. This way, cases like arriving at an intersection can be represented as graphs where the intersection is included.

**Scenario Comparison – Graph Comparison** Given a traffic scenario in the former discussed graph representation, it should be identified if it is a novel traffic scenario or not. For this purpose, the known traffic scenarios, represented as graphs as well, are available for comparison. Hence, it is checked if the graph and the routes in it are similar to any stored traffic scenario. In general, graphs can be compared by checking if they are isomorphic. Only if the graphs are similar, the routes can be compared by using the bijection of the isomorphism. If the routes are similar after applying the bijection, also the routes can be interpreted as similar. In order to reduce computational cost, the isomorphism is only carried out if other attributes of two graphs are similar. First, the number of vertices has to be similar. Second, the number of vertices having the same number of incoming and outgoing edges has to be similar. Third, also the length of the routes have to be similar. This enables a more efficient comparison of graphs, as also realized in [FFWSM<sup>+</sup>22].

The graph construction and comparison explained in this section is very similar to the one explained in Chapter 4. The mathematical definition of the graphs and their comparison can be seen in the corresponding chapter.

**Scenario Comparison – Trajectory Comparison** If many traffic scenarios with the same graph representations are collected, they can be compared by analyzing the similarity of the underlying trajectories. Like before, this can be realized for the EGO and all OBJs. For the comparison of the trajectories, the DTW distance weighted by the length of the warping path is used. As features, only the velocity and the acceleration over time (called action profiles in Section 4.2) are used. In the master thesis [Dun22], which was supervised in the context of this dissertation, different trajectory similarity measures were analyzed. DTW applied on acceleration and velocity over time has shown the most promising results for comparing trajectories.

**Discussion** The domain knowledge driven approach to categorize traffic scenarios can be used to detect unknown traffic scenarios on an abstract level. However, it lacks detailed information of the traffic scenario such as the geometry of the infrastructure or the specific trajectories. Such information might be relevant to compare scenarios more detailed but are difficult to include in domain knowledge approaches. Therefore, tackling the traffic scenario comparison with data-driven approaches could lead to a more detailed traffic scenario clustering.



## Chapter 4

# Representation Learning for Identifying Relevant Traffic Scenarios

As shown in the previous chapter, a simple statistical proof of an *Autonomous Vehicle's* (AV's) safety is infeasible. Approaches like scenario-based testing are used for validation. There, the testing of an AV is focused on relevant scenarios, instead of exclusively driving randomly millions of kilometers. Identifying relevant scenarios is required for this approach [JWKW18] (c.f. Figure 4.1). The scenarios can be constructed by domain knowledge or from real world driving data.

In the previous chapter, various approaches have been proposed and summarized how to identify relevant traffic scenarios. However, in summary, no approach was able to handle alone the complex traffic scenarios in terms of detecting unknown scenarios. While the

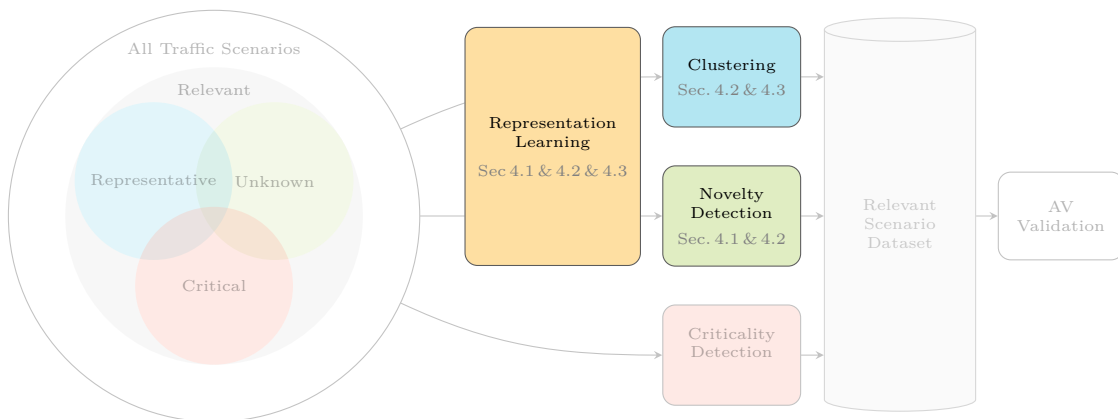


Figure 4.1: Methodological overview of this work and its embedding in the validation process of AVs. Novel methods for *Representation Learning* are introduced in the corresponding sections. The performance is evaluated with novelty detection and clustering.

most existing methods focus on trajectories, some also include infrastructure information. The similarities among traffic scenarios are either defined implicitly e. g., autoencoding regime, or explicitly e. g., DTW (see Section 3.2). Also, the performance of the proposed approaches for outlier detection (LEF, reconstruction-based approaches) is not satisfactory. The presented explicit domain knowledge methods (criticality and category) provide traceability, but fail to express complex similarities and relationships.

As concluded in Section 3.4, simple novelty detection methods can not handle the complex structure of traffic scenarios. Therefore, the objective is to transform traffic scenarios into a representation which can be used for simple novelty detection methods and clustering as well. In this chapter three representation learning approaches are presented which aim to fulfill this task. Instead of relying on approaches without domain knowledge, like autoencoders, domain knowledge is used to improve the performance. The embedding of the representation learning into the overall framework of this work is illustrated in Figure 4.1, where a traffic scenario is first transformed by the representation learning component before being forwarded to the clustering or novelty detection component.

The first approach combines a triplet loss with an autoencoder for road infrastructure images. The domain knowledge is applied through the triplet loss. This approach is extended in the second method to a quadruplet-based autoencoder, such that the dynamic information of the EGO vehicle is covered as well. The third and last presented concept focuses on self-supervised learning and the advantages when using domain knowledge guided augmentations. All three methods generate representations which can be used for tasks like novelty detection or clustering. Therefore, the aim of the presented representation learning methods is to form valuable representations, not to solve the task directly. Overall, introducing domain knowledge into the representation learning approaches improves the performance on the downstream tasks novelty detection and clustering.

### 4.1 Domain Knowledge guided Triplet Autoencoder

The method summarized in this section aids the outlier detection of traffic scenarios, by transforming the traffic scenario into a latent embedding using representation learning guided by domain knowledge. The method is presented in [WBBU21] as well. As discussed in the previous chapter, one meaningful way to enable outlier detection and clustering of traffic scenarios could be to transform the traffic scenarios into more suited representations. As shown in this section, when including domain knowledge into representation learning, the performance can be increased.

A traffic scenario is described by multiple aspects, for example the dynamics and the environment [PEG]. Publications often focus on the dynamics to identify representative and novel scenarios (e. g., [WRZ<sup>+</sup>20], [HBG20], [DARC20], [HGSP20]). Besides dynamics, another crucial component of a scenario is the infrastructure. Here, bird's-eye view images,

representing the infrastructure, are used to detect novel traffic scenarios.

In this section, a method to detect unknown and potentially untested infrastructures is presented. For this purpose, the infrastructure images are projected into a latent space using a deep learning pipeline. As to be shown, a high performance increase is achieved when using the latent space instead of the input space for novelty detection. In the latent space, simple and well-established outlier detection methods can be used to identify novel infrastructures, which indicates that the transformation to the latent space is able to generate strong representations. In order to create this latent space, an autoencoder architecture utilizing metric learning via triplet loss is used. The triplet mining is based on the connectivity of the infrastructures. Through the combination of the autoencoder scheme and the triplet learning, domain knowledge is used for shaping the latent space.

Extensive evaluation of the presented method is performed. For the encoder, state-of-the-art networks such as ViT (Section 2.1.4.5) and ResNet-18 (Section 2.1.2.3) are evaluated. Experiments demonstrate the influence of the triplet loss as well as the autoencoder scheme. The resulting architecture combinations are outperforming the alternative methods. An implementation of the architecture is made publicly available<sup>1</sup>.

The contributions of the research in this area can be summarized as:

1. A new method to detect novel infrastructure images is presented, where the novelty detection is performed in the latent space of a triplet loss-based autoencoder network.
2. Automated triplet mining of road infrastructures without manual labeling is introduced.
3. The performance is evaluated against various methods and shows significant improvements.

In the following of this section, the method is explained. This includes the description of the data generation pipeline and the definition of an infrastructure similarity measure. After that, the training of the triplet autoencoder is defined. The experiments focus on the outlier detection performance in the latent space of the triplet autoencoder. Furthermore, the latent spaces are visualized for additional analysis purposes. Motivated from the latent space visualization, another quantitative analysis of the created representations is introduced, where the similarity of infrastructure images among neighbors in the latent space is considered.

#### 4.1.1 Method

In [WBBU21], an autoencoder network utilizing triplet loss is used to project road infrastructure images into the latent space, where the outlier detection is performed. By using the proposed architecture, expressive latent representations for road infrastructure

---

<sup>1</sup><https://github.com/JWTHI/ViTAL-SCENE>

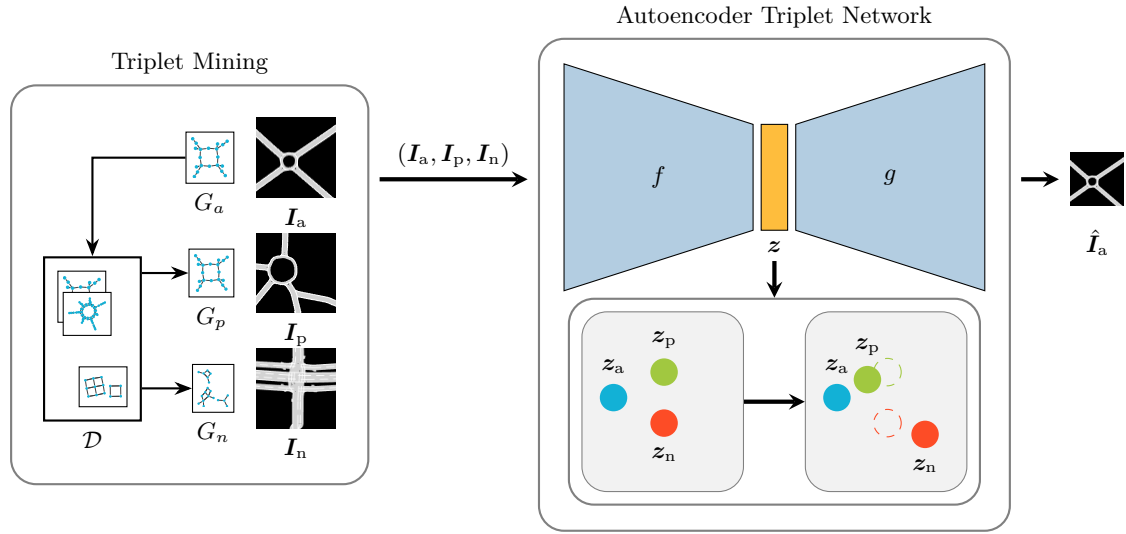


Figure 4.2: Triplet Learning Network: The triplet mining depicts how the triplet is selected based on the graphs  $G$ . Each infrastructure image  $I$  is processed by the network, leading to the latent representations  $z$ . Below the network, the triplet learning objective is illustrated. The infrastructure of the anchor is reconstructed through the decoder  $g$  as  $\hat{I}_a$ .

images are learned. During training, the infrastructure is assessed in two ways. First, the visual appearance of the images, hence the shape of roads etc., is considered via the reconstruction loss. Second, the similarity of infrastructure topologies is taken into account via the similarity of their connectivity graphs, allowing data triplets to be identified. A data triplet consists of three data points: the anchor, a sample similar to the anchor (positive sample), and a sample dissimilar to the anchor (negative sample), which are required for the triplet loss.

This subsection is split into the following parts. First, related work with respect to triplet learning and outlier detection is provided. Second, the data generation and used similarity measure are explained. Third, the realized triplet autoencoder network as well as the used triplet mining is summarized. Last, the unknown traffic scenario detection through novelty detection in the latent space is described.

#### 4.1.1.1 Related Work: Triplet Learning and Outlier Detection

This method is using triplet learning to form the latent space. Triplet networks [SKP15] are used to perform deep metric learning through ranking loss (Section 2.5.2.2). In tile2vec [JWS<sup>+</sup>19], triplet networks are used for geospatial analysis. The sampling is determined based on the distances of the tiles. In [YCS19], an autoencoder network is combined with triplet learning. The difference of this section to [YCS19] lies in the application, the specific network architecture, and the triplet mining.



To aid the detection of outliers, some works are using metric learning. An example is [MRS<sup>+</sup>18], where out-of-distribution data is used for the training.

In the field of deep learning various approaches to detect outliers exist. The most common approach is to use the reconstruction paradigm. For example in f-AnoGAN [SSW<sup>+</sup>19], a GAN is used as generator network, where an additional encoder is trained to learn the mapping from an input image to the GAN’s latent representation (Section 2.3.4.3). An extension of the reconstruction paradigm is using the hidden reconstructions additionally [KSL<sup>+</sup>20] (Section 2.3.4.2).

In this section, standard outlier detection methods are applied in the latent space, namely the LOF [BKNS00] (Section 2.3.1), ABOD [KhZ08] (Section 2.3.1), IF [LTZ08] (Section 2.3.2), and the OCSVM [SWS<sup>+</sup>00] (Section 2.3.3).

#### 4.1.1.2 Data Generation

In this method, a traffic scenario is described only through its static part, the road infrastructure. For this purpose, a black and white image of the infrastructure in bird’s-eye view is generated. Furthermore, a connectivity graph is generated, which will be required to determine the similarity between infrastructure images.

The data generation tool-chain consists of the following steps: a) position selection, b) map data collection, c) image generation and d) graph generation. *Open Street Map* (OSM) [Ope20] is used for the map data. Also, the position selection is realized via OSM.

**Position Selection** Within a search area, all OSM nodes can be considered as positions. For this work the valid positions are restricted to public and car-drivable roads. Identifying the nodes and positions can be realized through the OSM API for example. Each selected position leads to an entry in the dataset, consisting of an image and the corresponding connectivity graph.

**Map Data Collection** For each selected position, it is required to get the associated map data. In this work, the OSM map with a parameterizable bounding box around the location is converted into an OpenDRIVE [D<sup>+</sup>15] map, such that the image generation tool as introduced in [WFBU20] can be used. For the conversion from OSM to OpenDRIVE, the `netconvert` tool of SUMO [LBBW<sup>+</sup>18] is used.

**Image Generation** The image generation is realized as proposed in [WFBU20], introduced in Section 3.3. Hence, the roads are colored gray, lane markings white and the background black. Furthermore, all non-public and non-car accessible roads are not rendered. For each position a grayscale image  $I$  is generated.

**Graph Generation** The graph generation is new with respect to [WFBU20] and is realized as follows. First, the complete OpenDRIVE map is converted into a network graph, using the connectivity and neighbor information. The result is comparable to the routing graph realized in Lanelet2 [PPJ+18]. In the next step, the selected location is assigned to the corresponding node in the graph. Hence, the position on the road is identified. Then, the graph is cropped and simplified using the following rules:

- Use all nodes which can be reached within the time  $t_{\max}$ , given the allowed speed, up to and including all nodes of the first junction (e.g., crossing, roundabout).
- Use all nodes which are neighboring lanes.

The infrastructure graph described here, is very much similar to the infrastructure graph described in Section 3.5.2 of the previous chapter. However, since this method focuses on the static information, the graph is simplified according to the reachability starting from the selected position (center of image in this case).

**Dataset** The above steps are used to generate the dataset  $\mathcal{D} = \{(\mathbf{I}_m, G_m)\}_{m=1}^M$ , where  $M$  is the number of selected positions, and hence the number of resulting images and graphs. The image for the  $m$ -th position is given by the matrix  $\mathbf{I}_m \in \mathbb{R}^{S \times S}$  with  $S$  the selected resolution of the image. Analogous, the graph for the  $m$ -th position is given by  $G_m = (V_m, E_m)$ , where  $V_m$  are the vertices and  $E_m$  the edges of the graph. The actual selected position is not used after the corresponding image and graph are extracted.

#### 4.1.1.3 Similarity Measure

For the triplet-based learning, a notation of being similar or dissimilar is required. In this section, the graphs are used for this purpose. Two cropped road infrastructure areas are considered to be similar if their connectivity graphs are similar. The graphs need to be permuted versions of each other to be similar. This is the case if there exists an isomorphism between the two graphs. Given the graphs  $G_i$  and  $G_j$ , they are similar if there exists a bijection  $p : V_i \rightarrow V_j$  such that  $(u, v) \in E_i \iff (p(u), p(v)) \in E_j$ . Using the notation  $G_i \cong G_j$  for two graphs being isomorphic, the similarity function is defined as

$$s_i(G_i, G_j) = \begin{cases} 1 & \text{if } G_i \cong G_j \\ 0 & \text{else} \end{cases} . \quad (4.1)$$

This similarity measure considers two infrastructures as same, when their connectivity is the same. Within the triplet mining block of Figure 4.2 examples of extracted graphs alongside their images are shown. Contrary, two infrastructures are dissimilar when their connectivity is not the same.

#### 4.1.1.4 Triplet Autoencoder

The main part of this section is the triplet-based autoencoder network. It is used to produce latent representations for given input images of road infrastructures. This section will address the architectural details, the used loss, the triplet mining, and details on the used networks.

Triplet learning realizes metric learning via a ranking loss. The objective is to enforce similarity in the latent space based on three samples, a so-called data triplet. As explained before, each triplet consists of an anchor, a positive sample, and a negative sample. The anchor and the positive sample are similar, according to the used similarity measure. Consequently, the negative sample is dissimilar to the anchor, according to the used similarity measure. For example, in [SKP15], the anchor is an image of a face, the positive sample is another image of the same person, and the negative is an image of another person. The objective of the training is to push the latent representation of the negative sample away from the latent representation of the anchor while pulling the latent representation of the positive sample closer. This way, the metric learning is realized. More details on triplet learning can be found in Section 2.5.2.2.

The triplet learning scheme of this method uses road infrastructure images as input, with the anchor  $\mathbf{I}_a$ , the positive sample  $\mathbf{I}_p$ , and the negative sample  $\mathbf{I}_n$ . Let the encoder  $f$  be a trainable network, realizing the mapping from the input representation to the latent representation  $f : \mathbf{I} \mapsto \mathbf{z}$  with  $\mathbf{z} \in \mathbb{R}^{N_z}$  being the latent representation with dimensionality  $N_z$ . During the training, each sample of the triplet  $(\mathbf{I}_a, \mathbf{I}_p, \mathbf{I}_n)$  is passed through  $f$  separately, leading to the latent triplet  $(\mathbf{z}_a, \mathbf{z}_p, \mathbf{z}_n)$ . During inference, only single samples will be passed through  $f$ . The training of  $f$  is partially realized by using the triplet loss

$$\mathcal{L}_{\text{tri}}(\mathbf{I}_a, \mathbf{I}_p, \mathbf{I}_n) = \max(\alpha + d_{ap} - d_{an}, 0), \quad (4.2)$$

with the squared distance between the anchor representation and the positive sample representation  $d_{ap} = \|f(\mathbf{I}_a), f(\mathbf{I}_p)\|_2^2$ , the squared distance between the anchor representation and the negative sample representation  $d_{an} = \|f(\mathbf{I}_a), f(\mathbf{I}_n)\|_2^2$ , and the margin  $\alpha$ . The triplet loss' objective is twofold, to lower the distance between the positive sample and the anchor  $d_{ap}$  while simultaneously increase the distance between the negative sample and the anchor  $d_{an}$  until  $d_{ap} + \alpha$ . This way the latent representation is forced to follow the definition of similarity as provided by the triplets. Like in [JWS+19], a L2-regularization term is added to the loss, where a weight of 0.01 is used for this additional term.

One key point of the triplet scheme is the triplet mining. While sampling  $\mathbf{I}_a$  is realized by randomly picking an image from the dataset  $\mathcal{D}$ , determining the positive and negative samples is based on the anchor. The similarity of two road infrastructure images is defined through Equation (4.1). Hence, the positive sample is randomly picked out of all samples having the same connectivity as the anchor. Given the graph of the anchor as  $G_a$  the pos-

itive sample  $\mathbf{I}_p$  is picked from  $\{\mathbf{I}_m \mid m \in M \wedge s_i(G_m, G_a) = 1\}$ . Using the same notation, the negative sample  $\mathbf{I}_n$  is picked from  $\{\mathbf{I}_m \mid m \in M \wedge s_i(G_m, G_a) = 0\}$ . The graphs are only used to determine the data triplet, but are not processed by the network.

The selection of the negative sample has a significant influence on the training. Considering the case where the negative is already too far away and hence there is no training contribution for this triplet, since  $d_{an} \geq d_{ap} + \alpha$  would lead to  $\mathcal{L}_{\text{tri}} = 0$ . Such samples are called easy negatives. On the other hand-side having hard negatives, i. e.,  $d_{an} < d_{ap}$ , might lead to bad local minima early in the training [SKP15]. Both types should be prevented. Therefore, the objective of the triplet mining is to sample data points that are called semi-hard negative samples. They are further away than the positive sample but still within the margin  $\alpha$ , such that  $d_{ap} < d_{an} < d_{ap} + \alpha$  holds. More detailed explanation on negative sampling including an illustration is provided in Section 2.5.2.2.

The triplet loss conditioned training might be sufficient to separate different connectivity types, for example roundabout with 4 versus 3 exit roads. However, another objective in this method is to ensure that neighboring points in the latent space have visually very similar input images. The reasoning for this is twofold. First, it is assumed that the shape of the road also has an influence on the novelty of a scenario. Second, it is assumed, the more similar the neighborhood in the latent space is, the more reliable the projection is for unknown data. By this, even within the same connectivity group further refinement is achieved. For this purpose, a decoder  $g$  is introduced to complete the overall architecture. The decoder is used to regularize the latent space, such that it can be used to reconstruct the anchor  $\mathbf{I}_a$ <sup>2</sup>. The underlying assumption is, that for the reconstruction to work, the latent space has to be formed in such away, that the neighbors are not sharing only connectivity-based similarities but also visual similarities. The trainable decoder network  $g$  is used to generate the reconstructed anchor image  $\hat{\mathbf{I}}_a$  by  $g : z \mapsto \hat{\mathbf{I}}$ . The reconstruction loss of the anchor,

$$\mathcal{L}_{\text{rec}}(\mathbf{I}_a) = \|\mathbf{I}_a - g(f(\mathbf{I}_a))\|_2^2 \quad (4.3)$$

is used to enforce the above described objective.

The complete architecture is trained using the loss

$$\mathcal{L} = \mathcal{L}_{\text{tri}} + \mathcal{L}_{\text{rec}}, \quad (4.4)$$

such that both objectives are fulfilled: connectivity-based structure and visual similarity. The overall pipeline of the triplet autoencoder network including the triplet mining can be seen in Figure 4.2. The exemplary graphs are the results for the shown images. This highlights the need for the local refinement by the decoder, since the shown anchor and its positive sample are sharing the same connectivity but have quite different visual

---

<sup>2</sup>In general, all input samples could be reconstructed. However, in this work it was sufficient and computationally less expensive to only use the anchor sample for reconstruction.

appearance. Below the network, the triplet learning process is indicated.

The triplet autoencoder network can be trained on a huge dataset such as OSM, ensuring a strong projection method. After training, the network can be used during inference phase to project new data. This way, it is possible to train the network only once. Furthermore, it allows one to store the data in a compressed format, i. e., the latent representations.

The decoder network is kept fairly simple<sup>3</sup>, in order to limit the complexity of the latent space. The decoder network structure is kept the same for all experiments, only the encoder is varied. For the encoder network the ResNet and ViT are tested and are therefore briefly explained in the following.

**ResNet** In [HZRS16] the widely used ResNet-18, -50 etc. are proposed. Residual networks consist of multiple residual blocks with multiple convolutional layers each. The input to each residual block is added to the output again. Details about ResNets and CNNs can be found in Section 2.1.2.

**Vision Transformer** Most recently, ViTs have been introduced in [DBK<sup>+</sup>21], using the attention mechanism-based transformers as in [VSP<sup>+</sup>17]. ViT is a convolution-free network for image classification, showing state-of-the-art performance.

The transformer usually uses an encoder and a decoder for sequence to sequence modelling, but the ViT is using only the encoder part, as shown in Figure 4.3. First, an image is split into flattened patches which are linearly projected, leading to the input embeddings (marked blue). Then, an additional embedding token (orange) is concatenated to the input embeddings before added to the learnable positional embedding (green). The sum is fed as an input to the transformer. Inside the transformer, a multi-layer multi-head-attention network is realized.

The output token, corresponding to the additional embedding token, is processed through a MLP. Here, in difference to the original ViT as explained in Section 2.1.4.5, the output vector is the latent representation  $\mathbf{z}$  (red), whereas in the original ViT it is used to predict a class label.

#### 4.1.1.5 Novelty Detection

The overall objective of this section is to detect unknown road infrastructures. For this purpose, outlier detection methods can be applied. Let the base dataset  $\mathcal{D}_{\text{base}}$  be the data already known, for example, the scenarios an autonomous driving function has already been tested on. If a data point is tested with respect to its outlierness it is basically

<sup>3</sup>The architecture of the decoder is as follows in PyTorch notation:  $\text{ConvTranspose2D}(N_z, 32, 6, 6) \rightarrow \text{ReLU} \rightarrow \text{ConvTranspose2D}(32, 64, 4, 2) \rightarrow \text{ReLU} \rightarrow \text{ConvTranspose2D}(64, 64, 4, 2) \rightarrow \text{ReLU} \rightarrow \text{ConvTranspose2D}(64, 1, 6, 2) \rightarrow \text{sigmoid}$

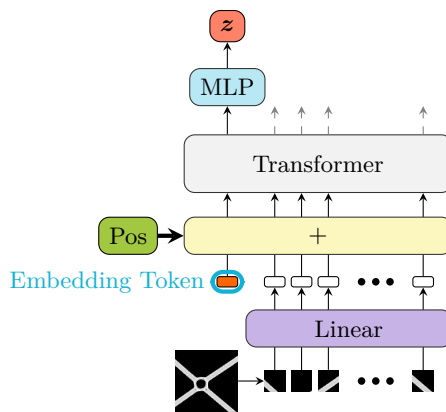


Figure 4.3: Vision Transformer as used in this work. Inspired from [DBK<sup>+</sup>21].

investigated if the point fits into the known data  $\mathcal{D}_{\text{base}}$ . The training data  $\mathcal{D}$ , used to train the networks  $f$  and  $g$ , is not required to be the same as the base data  $\mathcal{D}_{\text{base}}$ .

Provided with the latent infrastructure representation  $z$ , simple and well-established outlier mechanism can be applied directly in the latent space instead of the input space. This way, the enforced similarity utilizing connectivity and shape will be the main influence for the outlier detection.

Within this section various outlier detection methods are used. Interested readers may refer to the corresponding sections in Chapter 2. In the following the used methods are briefly summarized.

**Local Outlier Factor** The LOF [BKNS00] (Section 2.3.1) is analyzing how isolated a data point is with respect to its neighbors. For this the local densities are used.

**Isolation Forest** The IF [LTZ08] (Section 2.3.2) is estimating the outlierness through the number of splits required to isolate a data point, given randomly grown trees. It holds, that the fewer splits the more outlying.

**Angle-Based Outlier Detection** Another approach is ABOD [KhZ08] (Section 2.3.1). The variance of the direction vector's angle from the point under investigation to all other data points is investigated. The higher the variance the lower the outlierness. Here, the fast version of ABOD is used, considering only the  $K$  nearest neighbors of the investigated point.

**One-Class Support Vector Machine** The OCSVM [SWS<sup>+</sup>00] (Section 2.3.3) is the extension of the normal SVM to the problem of outlier detection. The data is mapped to the feature space as by the kernel. In the feature space, the data is separated from the origin via a hyperplane. This way, the decision boundary is meant to encapsulate the data in the input space.

**UMAP-based Local Entropy Factor** In the work [WFBU20], summarized in Section 3.3, the neighborhood of data points is used to define local similarity measures. A point’s outlierness is evaluated based on how well it fits into its neighbors’ neighborhoods using the entropy and the point-wise similarity.

#### 4.1.2 Experiments

The introduced method is evaluated in this subsection. Various architecture realizations and outlier detection methods are compared. This subsection is split into the following parts. First, the details about the data used for the outlier detection and the data used for visual analysis is explained. Second, the analyzed architectures are briefly summarized. The outlier detection performance is discussed in the third part. The various resulting latent space visualizations are shown and discussed in the fourth part. In the fifth subsection, the local visual similarity quality is assessed, proposing another possibility to evaluate the performance of the shown architectures. The results are summarized in the last part.

##### 4.1.2.1 Datasets

The training dataset  $\mathcal{D}$  is decoupled from the base dataset used for the outlier detection  $\mathcal{D}_{\text{base}}$  and the outlier dataset  $\mathcal{D}_{\text{ano}}$ . Both are briefly described in the following. For both the images display a region of size  $100 \text{ m} \times 100 \text{ m}$ , while the reachable time for generating the graphs is selected as  $t_{\text{max}} = 5 \text{ s}$ .

The training dataset  $\mathcal{D}$  consists of  $\approx 70\,000$  pairs of images and graphs. To provide an insight to the dataset, it has been analyzed with respect to rough groups, but those groups are not used in the training. In total, approx. 13 400 highway, 16 900 roundabout, 18 000 crossing, 19 900 single lane, and 1 700 multiple lane non-highway pairs are used. The extraction region for the highway and roundabout pairs is selected to be the complete district of Upper Bavaria in Germany. The extraction region for the remaining types is the city of Ingolstadt in Germany with its adjacent counties.

The base and the outlier dataset are taken from [WFBU20]. Hence, the base dataset  $\mathcal{D}_{\text{base}}$  is used to fit the models of the outlier detection methods. Therefore, the data considered to be known are highway images only. Since, the outlier dataset  $\mathcal{D}_{\text{ano}}$  is taken from [WFBU20] as well, rural and inner-city images are considered unknown.

##### 4.1.2.2 Architectures

In order to investigate the influence of various architecture realizations, different versions of the network were used for the experiments. The architecture of the decoder is kept the same for all the experiments, enabling a fair comparison. The encoder is realized through different networks. Furthermore, the loss function was changed for some experiments,

highlighting the importance of the overall pipeline. Here, the various used options will be briefly summarized.

For all architectures, the following parameters hold: image size  $64 \times 64$ , epochs 200, latent dimensionality 50.

**ResNet-S** A small ResNet, like ResNet-18 but consisting of fewer parameters. Learnable parameters  $\approx 0.3 \cdot 10^6$ .

**ResNet-18** Here the basic implementation of ResNet-18 [HZRS16] is adjusted for single channel inputs. Learnable parameters  $\approx 11.0 \cdot 10^6$ .

**ViT-S** A small version of the ViT. Learnable parameters  $\approx 0.2 \cdot 10^6$ . (patch size  $8 \times 8$ , layers 6, dimension of input embedding  $n_{\text{patches}} \times 64$ , internal MLP dim. 128).

**ViT-L** A large version of the ViT. Learnable parameters  $\approx 6.7 \cdot 10^6$ . (patch size  $8 \times 8$ , layers 20, dimension of input embedding  $n_{\text{patches}} \times 256$ , internal MLP dim. 128).

**Loss Terms** Furthermore,  $\mathcal{X}_{\text{rec}}$  is highlighting architectures where the decoder network is deactivated. Therefore, the loss is only based on the triplet part. The triplet loss is not used for the architectures marked with  $\mathcal{X}_{\text{tri}}$ . Hence, only the reconstruction loss is used.

### 4.1.2.3 Outlier Detection

The various architectures are evaluated with respect to their outlier detection performance. For this, the outlier detection methods LOF, ULEF, OCSVM, and ABOD are applied in the latent space of the resulting network. As baseline methods, the outlier detection methods are also applied in the input space to highlight the performance gain when using the latent representation (last row in Table 4.1). The networks are trained using  $\mathcal{D}$ , while the outlier detection is performed using  $\mathcal{D}_{\text{base}}$  as known and considering the remaining  $\mathcal{D}_{\text{ano}}$  as unknown. Therefore, the novelty detection is applied to the case where only highway images  $\mathcal{D}_{\text{base}}$  are known. Then unknown data points, such as inner-city images,  $\mathcal{D}_{\text{ano}}$  are investigated with respect to their outlierness. If the outlier detection is able to identify that for example inner-city images are outlying with respect to highway images, the task is fulfilled successfully. This evaluation shows the outlier detection capabilities with respect to connectivity classes.

Additionally, the reconstruction-based outlier detection methods f-AnoGAN [SSW+19] (Section 2.3.4.3) and RaPP [KSL+20] (Section 2.3.4.2) are evaluated. For the RaPP, a convolutional autoencoder is trained using  $\mathcal{D}_{\text{base}}$  then the normal reconstruction error is used for outlier detection (ROD) and the simple aggregation along pathway (SAP) as



Table 4.1: Outlier detection performance – AUC.

Architecture			Input	LOF	ULEF	IF	OCSVM	ABOD	ROD	SAP	f-AnoGAN
$f$	$\mathcal{L}_{\text{rec}}$	$\mathcal{L}_{\text{tri}}$									
ResNet-S	$\times$	$\checkmark$	$z$	0.913	0.775	0.935	<b>0.959</b>	0.892	–	–	–
ResNet-S	$\checkmark$	$\checkmark$	$z$	<b>0.954</b>	0.696	0.933	0.950	<b>0.954</b>	–	–	–
ResNet-18	$\checkmark$	$\times$	$z$	0.696	0.776	0.770	0.752	<b>0.784</b>	–	–	–
ResNet-18	$\checkmark$	$\checkmark$	$z$	0.949	0.781	0.917	0.912	<b>0.956</b>	–	–	–
ViT-S	$\checkmark$	$\checkmark$	$z$	0.730	0.689	0.698	0.910	<b>0.920</b>	–	–	–
ViT-L	$\checkmark$	$\checkmark$	$z$	0.900	0.793	0.707	0.937	<b>0.956</b>	–	–	–
Baseline Methods			$\mathbf{I}$	0.446	0.612	0.196	0.247	0.700	0.855	0.845	0.758

introduced in [KSL<sup>+</sup>20]. For the f-AnoGAN the network is also trained only on  $\mathcal{D}_{\text{base}}$ . More details on the training and evaluation in this setting are listed in Section 4.2.1.5.

In Table 4.1, the resulting *Area Under Curve* (AUC) values are shown for the various combinations. The AUC will be 1 if all outliers are detected correctly (see Section 3.3.3 for details on AUC). For each outlier detection method (LOF, ULEF, IF, OCSVM, ABOD) the performance is increased when applied on the latent representations compared to the results when applied on the plain input  $\mathbf{I}$  directly (see Section 3.4). The networks are providing a powerful latent representation, where simple outlier detection methods can perform well. In fact, most combinations (network with a basic outlier detection method) are outperforming other baseline methods such as ROD, SAP, and f-AnoGAN. Using one of the architectures in combination with ABOD is the preferable solution, since it yields the highest performance for all triplet autoencoding-based schemes ( $\checkmark\mathcal{L}_{\text{rec}}$  and  $\checkmark\mathcal{L}_{\text{tri}}$ ) and provides the highest performance overall. The results show that the proposed approach to include domain knowledge in shaping the latent space improves the outlier detection performance significantly in this application.

The relevance of the triplet loss becomes clear when comparing the simple autoencoder architecture (ResNet-18  $\times\mathcal{L}_{\text{tri}}$ ) against the one including the triplet loss (ResNet-18  $\checkmark\mathcal{L}_{\text{tri}}$ ). The use of the triplet loss increases the performance for all architectures remarkably. The influence of the decoder can be identified from ResNet-S  $\times\mathcal{L}_{\text{rec}}$  versus  $\checkmark\mathcal{L}_{\text{rec}}$ . Its contribution to the outlier detection is not as clear as for the triplet loss. Indeed, for some methods the outlier detection is getting slightly worse, however, for some it is getting better. The reason for introducing the decoder is the local visual similarity, which is not represented by the outlier detection analysis, since this is only covering the class oriented scale. For this purpose, another analysis will be carried out in the following subsection. Further comparison can be drawn from the different encoder types when using the triplet loss and the decoder. Because of its stable and high performance only ABOD is considered for further discussions. The ResNet-18 is only slightly better than the smaller ResNet-S. The performance difference for the two ViT versions is more significant. Here, the ViT-L would be the architecture of choice. In conclusion, using the triplet loss as well as the decoder is clearly beneficial, while either the ResNet-S, ResNet-18 or ViT-L can be used

from the outlier detection performance point of view.

#### 4.1.2.4 Latent Space Visualization

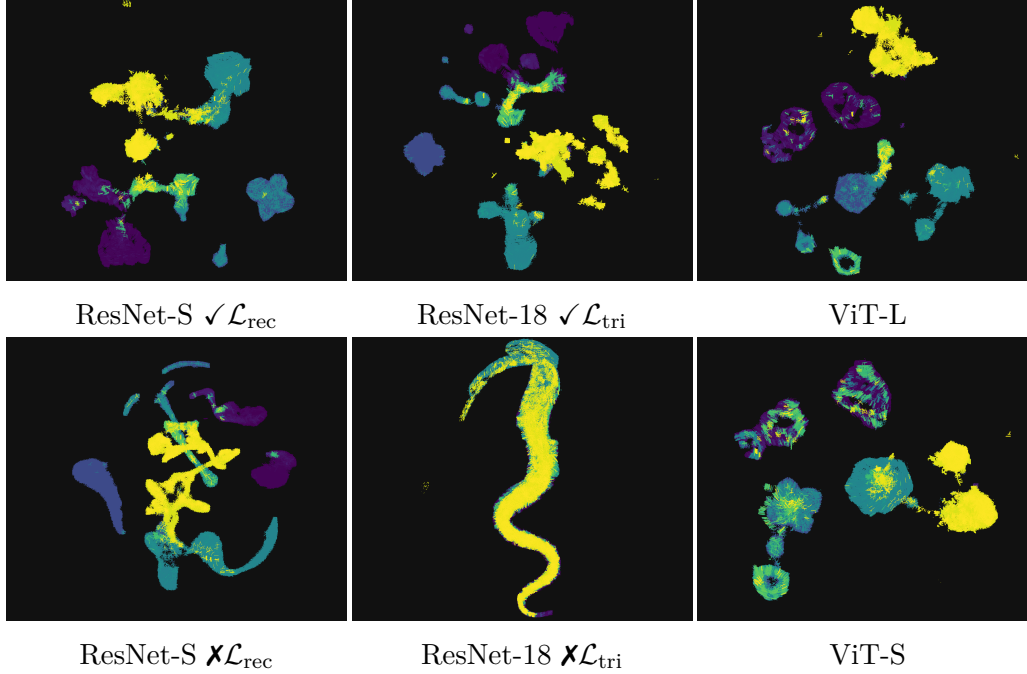


Figure 4.4: UMAP projections of  $\mathcal{D}_z$ . ■ highway, ■ crossing, ■ single lane, ■ multiple lane, ■ roundabout.

Interactive visualization tool SCENATLAS: <https://jwthi.github.io/SCENATLAS/>

Another approach to assess the quality of the latent space is provided through visual investigation. For this purpose, the latent representations  $\mathcal{D}_z = \{z_1, \dots, z_M\}$  are projected into a two-dimensional space using UMAP [MHM18] (Section 2.2.3). In Figure 4.4, the projections of  $\mathcal{D}_z$  for the previously tested architectures are visualized.

On this scale, the difference of ResNet-S  $\checkmark \mathcal{L}_{\text{rec}}$  versus  $\times \mathcal{L}_{\text{rec}}$  is hard to figure out. That difference will be further investigated via the local similarity analysis. For the difference produced by the triplet loss ResNet-18  $\checkmark \mathcal{L}_{\text{tri}}$  versus  $\times \mathcal{L}_{\text{tri}}$  (middle column in Figure 4.4), this scale is sufficient. It is visible, that training the autoencoder without the triplet loss leads to a less separable latent representation. The difference between the ResNet-S  $\checkmark \mathcal{L}_{\text{rec}}$  and ResNet-18  $\checkmark \mathcal{L}_{\text{tri}}$  is mainly in the more diffuse distribution of the roundabouts when using ResNet-18. When comparing the ViT-L against ViT-S, the former one is showing clear advantage, since in contrary to the smaller version, it is able to distinguish between highway and multi-lane. Comparing the architectures ResNet-S  $\checkmark \mathcal{L}_{\text{rec}}$ , ResNet-18  $\checkmark \mathcal{L}_{\text{tri}}$ , and ViT-L, all of them show clear grouping of the analysis classes, and hence, from this perspective are equally well suited. Interested readers may refer to <https://jwthi.github.io/SCENATLAS/>, where the interactive visualization tool SCENATLAS

is provided, which allows to discover the latent spaces more intuitively.

#### 4.1.2.5 Local Similarity Analysis

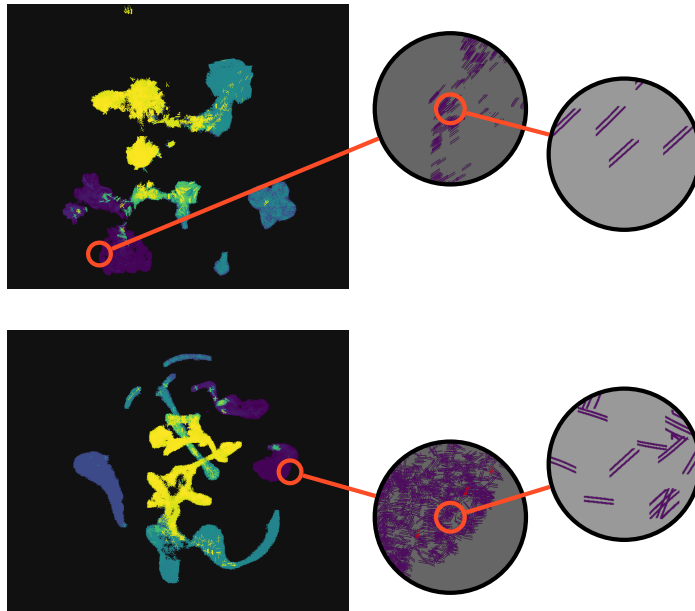


Figure 4.5: Local visual similarity motivation. ResNet-S Up:  $\sqrt{g}$ , Down:  $\mathbf{X}g$ .

As stated above, the main motivation of introducing the decoder is the local visual similarity of the latent space. This property has not yet been analyzed. So far, only the class orientated outlier performance was evaluated, i. e., differentiate highway versus non-highway. Therefore, another analysis is performed for the local scale. In Figure 4.5 the problem is visualized. It is showing two zoom levels of the embeddings from ResNet-S  $\sqrt{\mathcal{L}_{\text{rec}}}$  and ResNet-S  $\mathbf{X}\mathcal{L}_{\text{rec}}$ . Therefore, the embedding using the reconstruction loss  $\sqrt{\mathcal{L}_{\text{rec}}}$  appears to be more visually similar, as can be seen from both magnification levels. In the following this quantity is assessed through numerical evaluation.

Given the latent representations  $\mathcal{D}_z$ , the  $K$  nearest neighbors for the  $i$ -th sample in the latent space are determined, leading to the nearest neighbor set  $\mathcal{K}_i$ . Then, the average distance in the input space ( $\mathbf{I}$ ) between a point and its neighbors is determined. By using

$$d_{\text{local}}(k) = \frac{1}{M} \sum_{m=1}^M \frac{1}{k} \sum_{j \in \mathcal{K}_i} \|\mathbf{I}_m - \mathbf{I}_j\|_2 \quad (4.5)$$

the average distance between the latent space neighbors is evaluated. Since a low value states a high average visual similarity between the  $i$ -th point and its neighbors, it is considered that lower values reflect more meaningful representations. In Figure 4.6 the resulting values are shown for all the architecture variants. Therefore, the reasoning for the

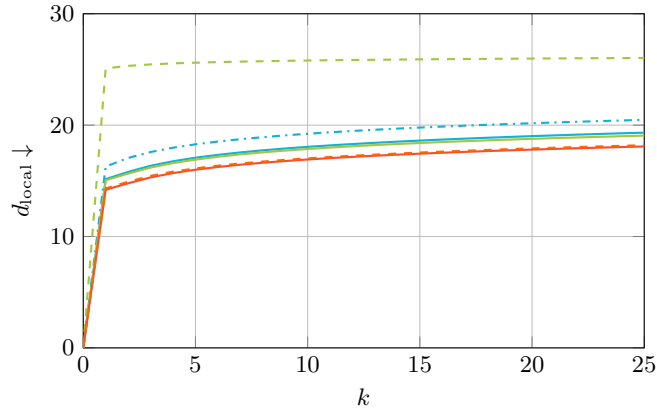


Figure 4.6: Average distance of the data points in the input space to their nearest neighbors corresponding to the respective latent representations. Smaller distance values are better.  $\cdots$ : ResNet-18  $\times \mathcal{L}_{\text{tri}}$ ,  $\text{—}$ : ResNet-18,  $\text{—}$ : ResNet-S,  $\text{- - -}$ : ResNet-S  $\times \mathcal{L}_{\text{rec}}$ ,  $\text{—}$ : ViT-S, and  $\text{- - -}$ : ViT-L

usage of the decoder is clearly provided. Moreover, the usage of the triplet loss is increasing the performance as well. The ResNet-S and ResNet-18 are again on a comparable level, but the ViT based architectures outperform all others. This indicates, that the ViT based method are preferable to identify local, shape based outliers.

#### 4.1.2.6 Summary

Table 4.2: Architecture overview, with AUC results when using ABOD in the latent space.

Architecture			AUC $\uparrow$	$d_{\text{local}}(5)$ $\downarrow$	Nparams
$f$	$\mathcal{L}_{\text{rec}}$	$\mathcal{L}_{\text{tri}}$			
ResNet-S	$\times$	$\checkmark$	0.892	25.61	$0.3 \cdot 10^6$
ResNet-S	$\checkmark$	$\checkmark$	0.954	16.90	$0.3 \cdot 10^6$
ResNet-18	$\checkmark$	$\times$	0.784	18.28	$11.0 \cdot 10^6$
ResNet-18	$\checkmark$	$\checkmark$	<b>0.956</b>	17.07	$11.0 \cdot 10^6$
ViT-S	$\checkmark$	$\checkmark$	0.920	<b>16.00</b>	$0.2 \cdot 10^6$
ViT-L	$\checkmark$	$\checkmark$	<b>0.956</b>	16.12	$6.7 \cdot 10^6$

To sum up this analysis, the important results are gathered in Table 4.2. Here, the results using ABOD are used. The overall best performance is provided by the ViT-L. However, also the small networks ResNet-S  $\checkmark \mathcal{L}_{\text{rec}}$  and ViT-S perform remarkably well.

#### 4.1.3 Conclusion

A method to create meaningful representations of road infrastructures in order to identify novel traffic scenarios based is presented in this section. The introduced pipeline is outper-

forming existing outlier detection methods. It has the additional advantage that it can be trained on a huge dataset (e. g., OSM), such that no retraining is necessary. In contrary, methods relying on the reconstruction paradigm would require retraining when detecting unknown scenarios. This approach presents a possibility to incorporate domain knowledge of a scenario’s static environment for shaping the latent space of a triplet autoencoder. The presented results show that methods like outlier detection can significantly benefit from a latent space shaped in this way.

Another approach to detect unknown infrastructure based on their topology, could be realized through a simple categorization logic using the graphs, as shown in Section 3.5.2. However, the method presented here provides an insight to the relationship between infrastructure types and additionally the local shape similarity, what can not be handled by the approach shown in Section 3.5.2.

In conclusion, the suggested pipeline consists of a connectivity graph-based similarity definition, an autoencoder triplet network with a ViT as encoder, and the ABOD method performing the novelty detection in the latent space. It shows superior performance with respect to its novelty detection capabilities and with respect to the neighborhood similarity evaluation. The interactive visualization SCENATLAS of the latent spaces is provided (<https://jwthi.github.io/SCENATLAS/>) and the code implementing the method is made publicly available.

Nevertheless, the method presented is very limited since it only covers the static part of a traffic scenario. Therefore, the inclusion of dynamic traffic scenario elements is required. Moreover, the latent spaces show potential to also be used for other tasks like clustering. Investigating the clustering performance given the embedded data is yet another open question that is analyzed in the remainder of this chapter.

## 4.2 Domain Knowledge guided Quadruplet Autoencoder

Two important tasks to enable the scenario-based approach are the definition of representative scenarios and the identification of potentially unknown and therefore untested scenarios. Representative scenarios can either be defined manually or automatically from collected data. For the latter one, usually clustering is used to define groups and representatives per group. The task of identifying untested scenarios can be realized through novelty detection like in the previous section or by checking if the scenario fits into an existing class or cluster (e. g., [KWM<sup>+</sup>19, BKBD21]). For the tasks of clustering and novelty detection, a good representation or similarity measure is required. When applied directly on the plain data, the result is unsatisfactory (Section 4.2.2.2). This section presents the approach *Expert-knowledge guided Latent Space for Traffic Scenarios* (Expert-LaSTS) that has been introduced in [WBBU22], which proposes a method to design a representation space for traffic scenarios using domain knowledge constraints. This representation space

can be utilized for the tasks of scenario clustering and of detecting novel scenarios.

The method introduced in this section [WBBU22] extends the findings and method of the previous section ([WBBU21]). While only the static part of a scenario is considered in the previous section, in this section also dynamic parts are taken into account with the inclusion of the EGO dynamics. Under some conditions, it can be considered to be sufficient to represent the dynamic part of a scenario by only the EGO trajectory. Assuming that the EGO is driving in the exact same infrastructure with almost the same constellation of objects surrounding it, it will only behave differently if an object, or another external factor, relevant to the trajectory of the EGO has changed. Such a scenario will be detected, since the EGO trajectory will be different. If the EGO trajectory is the same, that would mean, that if something changed in the constellation and setting, it is not important for the EGO trajectory, hence such scenarios are considered as redundant.

The methodology can be summarized as follows. First, domain knowledge based objectives are formulated. Then, it is shown how to design a loss function and network architecture such that those objectives are fulfilled. To realize the designed loss function, an automatic quadruplet mining process is introduced, that is based on a similarity measure for scenarios. A similarity measure based on the topology graph of the road network and the routes defined by the trajectories is proposed. This way, no manual labeling is required. Overall, the objective is to form a latent space, which hierarchically divides samples into groups, based on infrastructure, EGO route, and EGO trajectory. This latent space shall be well-suited clustering and novelty detection of traffic scenarios. An implementation realizing the presented method is made publicly available<sup>4</sup>.

The resulting latent space is most suitable for the tasks of novelty detection, clustering, and feature stability compared to alternative approaches. The presented method can be applied for the validation of AVs. It can aid the analysis of existing scenario databases or the detection of novel scenarios.

The contributions of the research on domain knowledge guided quadruplet autoencoders for traffic scenario analysis can be summarized as:

1. Definition of a domain knowledge aided loss and architecture to design the latent space as required.
2. Definition of an automatic mining strategy for traffic scenarios.
3. Comprehensive analysis and comparison of various representation spaces.

### 4.2.1 Method

The aim of the presented method, the metric learning network, is to design a latent space by means of domain knowledge to represent traffic scenarios, including dynamic and static

---

<sup>4</sup><https://github.com/JWTHI/Expert-LaSTS>

aspects of the scenario. It is shown how this latent space can be utilized to detect unknown traffic scenarios and to cluster traffic scenarios. Here, a traffic scenario is described by the road infrastructure and the dynamic information of the EGO. This method extends the previous method [WBBU21], presented before, which is limited to the road infrastructure of a traffic scenario.

#### 4.2.1.1 Preliminaries

A traffic scenario  $\mathcal{X} = \{\mathbf{I}, \mathcal{T}\}$  consists of the road infrastructure image  $\mathbf{I}$  and the EGO trajectory  $\mathcal{T}$ . The dataset  $\mathcal{D} = \{(\mathcal{X}_0, G_0, R_0), \dots, (\mathcal{X}_M, G_M, R_M)\}$  consists of  $M$  traffic scenarios and the corresponding graphs  $G$  and routes  $R$ . The individual elements are defined in the following.

**Infrastructure** The road infrastructure is represented as a grayscale bird’s-eye view image  $\mathbf{I} \in \mathbb{R}^{S \times S}$  and a graph representation  $G$ . The graph contains  $N_G$  lane pieces as vertices  $V = \{v_1, \dots, v_{N_G}\}$  and  $N_E$  edges  $E = \{e_1, \dots, e_{N_E}\}$  connecting them. The graph for a scene includes

1. all lanes, which are part of the EGO route,
2. all lanes up to and including the next intersection,
3. all lanes of possible intersections in 1) and 2), and
4. all lanes neighboring any lanes in 1) - 3).

This way, the graph contains mainly the relevant lanes, whereas the image contains all lanes within the defined area. The resulting graph might differ from the one defined in the previous section, since here, the EGO route instead of the reachability from the center is used.

**Trajectory** The trajectory information is represented in two ways. The sequence based representation  $\mathcal{T} = \{[x_1, y_1, t_1], \dots, [x_N, y_N, t_N]\}$  and the route representation  $R = \{r_1, \dots, r_{N_G}\}$ . The construction of  $R$  is realized as follows. For a trajectory point the corresponding vertices are  $v(x_i, y_i)$ , which leads to the vertices sequence for the trajectory as  $\mathcal{T}_R = \{v(x_1, y_1), \dots, v(x_N, y_N)\}$ . The route representation  $R$  is directly linked to  $G$  as,

$$r_n = \begin{cases} 2 & \text{if } v_n \in v(x_1, y_1) \\ 1 & \text{if } v_n \in \mathcal{T}_R \setminus v(x_1, y_1) \\ 0 & \text{else} \end{cases} \quad (4.6)$$

Hence, the vertex on which the trajectory starts is linked to  $r_n = 2$ , all other vertices the trajectory passes lead to  $r_n = 1$ .

The graph generation for the infrastructure and the trajectory is very similar to the concept discussed in Section 3.5.2. There the graphs are used for the detection of traffic scenario categorized and potentially novel traffic scenario categories. However, relying on only the connectivity graphs, information like distances and curvatures of the road, and hence dynamic information as well is not covered. In this method, it is aimed to combine the rough categorization realized by the connectivity graphs with features of the infrastructure and the trajectory.

#### 4.2.1.2 Base Method

In [WBBU21], a triplet autoencoder is used to project the infrastructure image  $\mathbf{I}$  into a latent representation. Using the triplet learning, the latent representations are optimized to represent the topology and the shape of the infrastructure. In the remainder of this section, a brief summary of the method [WBBU21] presented in Section 4.1 is provided.

It is proposed to use the road topologies underlying each image to perform the triplet mining and obtain a positive and negative sample based on the anchor sample. Positive samples have the same graph as the anchor, while negative samples have a different graph. An example: if the anchor is a four-way roundabout, the positive sample would also be a four-way roundabout. However, the shape of the roundabouts are not necessarily the same. The negative sample would be some other topology (e. g., intersection).

The positive sample  $\mathbf{I}_p$  is drawn from the subset  $\{\mathbf{I}_m \mid m \in M \wedge s_i(G_m, G_a) = 1\}$ , representing all samples which have the same topology as the anchor sample, with  $s_i$  as defined in Equation (4.1). Contrary, the negative sample  $\mathbf{I}_n$  is drawn from the remaining samples  $\{\mathbf{I}_m \mid m \in M \wedge s_i(G_m, G_a) = 0\}$ .

The network is a triplet autoencoder consisting of an encoder  $f : \mathbf{I} \mapsto \mathbf{z}$  and a decoder  $g : \mathbf{z} \mapsto \hat{\mathbf{I}}$ , where  $\mathbf{z} \in \mathbb{R}^{N_z}$  is the latent representation. Training the network is performed simultaneously by two approaches: the autoencoding regime and the triplet learning. This combination enables both, the topology based learning through the triplet strategy (Equation 4.3), and a low-level image similarity caused by the autoencoding objective (Equation 4.2).

#### 4.2.1.3 Proposed Method

This section extends the static description from the previous section to a scenario by considering the dynamics of the EGO vehicle. The network architecture is adjusted and the triplet learning is extended to quadruplet learning, called metric learning in the following. The overall concept is depicted in Figure 4.7, which is divided into the quadruplet mining process and the metric learning network.

The aim of this method is to design a latent space, which enables the clustering of scenarios and the detection of novel scenarios. Domain knowledge is used to aid and constrain the training process. For this, the following objectives are formulated:



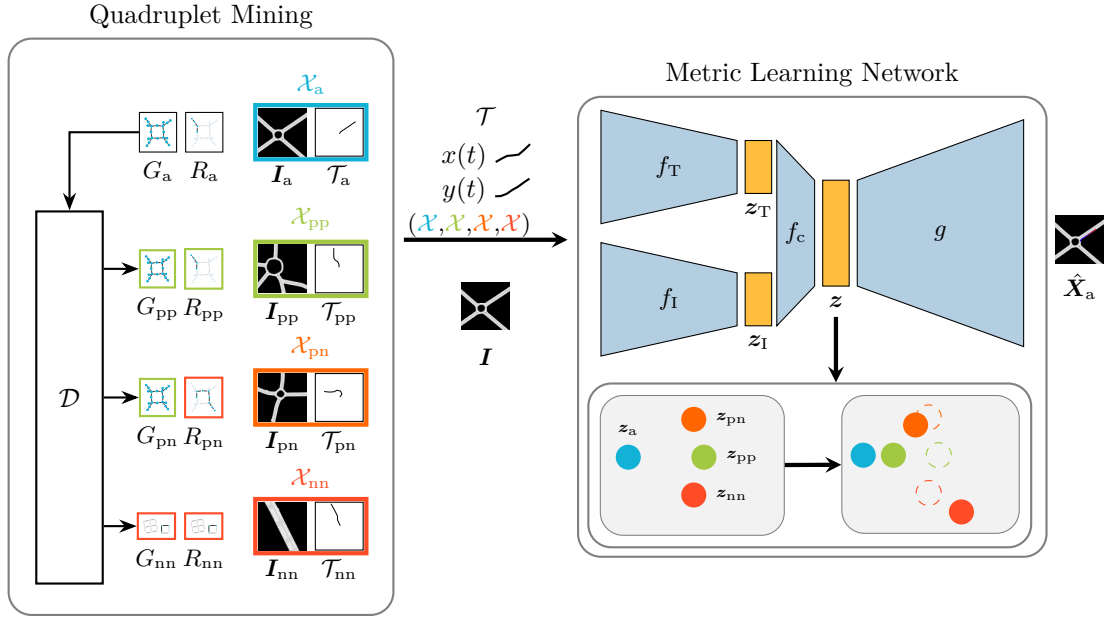


Figure 4.7: Metric Learning Network for Traffic Scenarios: The quadruplet mining depicts how the required scenario quadruplet is selected based on the graphs  $G$  and routes  $R$ . Each scenario  $\mathcal{X}$  consists of an image  $I$  and a trajectory  $\mathcal{T}$ . Each scenario is processed by the network, leading to the latent representations  $z$ . Below the network, the quadruplet learning objective is illustrated. The scenario is reconstructed through the decoder  $g$  into a merged representation  $\hat{\mathcal{X}}$ , combining the infrastructure image and the trajectory.

- A) Scenarios with the same infrastructure and similar trajectories shall be close together in the latent space.
- B) Scenarios with the same infrastructure but different trajectories shall be close but not as close as A).
- C) Scenarios without the same infrastructure shall be farther away than B).
- D) The distance of scenarios according to A) shall be adjusted based on the similarity of the underlying actions.
- E) Neighbors should have high similarities with respect to trajectory and infrastructure features.

The case that two scenarios have a different infrastructure but a similar route, is not tractable in this method, since the proposed route comparison requires a similar infrastructure. The objectives reflect domain knowledge based assumptions about the similarity of scenarios in a hierarchical way. Hence, the latent space should reflect these domain knowledge based hierarchical similarity objectives. In order to achieve the objectives, an automatic quadruplet mining process, the metric learning as well as the network architecture are presented.

**Quadruplet Mining** In Section 4.1 it is shown how the identification of similar infrastructures can be realized through their topology. Hence, distinguishing the cases C) from A) or B) is possible. To realize the required separation between A) and B), and therefore to include the trajectory information to the quadruplet mining process, the mining definitions are extended.

Given the case that two graphs are isomorphic  $G_i \cong G_j$ , hence their infrastructure is the same, two trajectories are considered to be similar, if they share the same route within their graphs. The trajectories  $\mathcal{T}$  are transformed into the route representation  $R$  which are directly linked to the respective graph  $G$  (see Section 4.2.1.1). Two scenarios share the same route if there exists a bijection  $p$  on  $G_i, G_j$  such that  $R_i = p(R_j)$ , which is formulated as  $G_i \cong G_j \mid R_i = p(R_j)$ . The route-based similarity measure is defined as

$$s_r(G_i, G_j, R_i, R_j, ) = \begin{cases} 1 & \text{if } G_i \cong G_j \mid R_i = p(R_j) \\ 0 & \text{else} \end{cases} . \quad (4.7)$$

According to D), just defining two trajectories to be similar is not sufficient, instead it shall be adjusted based on the actions of the trajectories. To allow further fine-tuning in the training, for trajectories sharing a similar route an additional similarity measure  $s_t$  is defined. Let  $\mathbf{A}_i = [\mathbf{a}_{\text{lat}}, \mathbf{a}_{\text{lon}}, |v|]$  be the accelerations and speed per timestamp for the  $i$ -th trajectory. The dissimilarity between two trajectories is then calculated via  $d = d_{\text{DTW}}(\mathbf{A}_i, \mathbf{A}_j) / |\text{DTW}_{\text{seq}}|$ , where  $d_{\text{DTW}}$  denotes the DTW distance and  $|\text{DTW}_{\text{seq}}|$  the warping path length divided by maximum sequence length. The intuition is to compare the trajectories which share the same route on an action level, therefore considering the accelerations and speed. The similarity is calculated with respect to maximum dissimilarity ( $d_{\text{max}}$ ) within all trajectories with the same route, as

$$s_t = 1 - \frac{d}{d_{\text{max}}} . \quad (4.8)$$

The mining of a data quadruplet is realized by randomly sampling an anchor scenario  $\mathcal{X}_a$ . Based on its corresponding graph  $G_a$  and route  $R_a$ , samples for the cases A) - C) are drawn. Hence, three types of scenarios are sampled, where compared to the anchor scenario they are having:

$\mathcal{X}_{\text{pp}}$  the same infrastructure  $G$  and the same route  $R$ ,

$\mathcal{X}_{\text{pn}}$  the same infrastructure  $G$  but a different route  $R$ , and

$\mathcal{X}_{\text{nn}}$  a different infrastructure  $G$ .

The scenario  $\mathcal{X}_{\text{pp}}$ , with similar infrastructure and similar route, is drawn from all scenarios which have the same infrastructure graph and route as the anchor scenario, hence it is randomly picked from  $\{\mathcal{X}_m \mid m \in M \wedge s_i(G_m, G_a) = 1 \wedge s_r(G_m, G_a, R_m, R_a) = 1\}$ . The

scenario  $\mathcal{X}_{\text{pn}}$ , with same infrastructure but different route, is randomly drawn from  $\{\mathcal{X}_m \mid m \in M \wedge s_i(G_m, G_a) = 1 \wedge s_r(G_m, G_a, R_m, R_a) = 0\}$ . Finally, the scenario  $\mathcal{X}_{\text{nn}}$ , with a different infrastructure, is randomly picked from  $\{\mathcal{X}_m \mid m \in M \wedge s_i(G_m, G_a) = 0\}$ . This quadruplet mining process is also visualized on the left side of Figure 4.7.

**Metric Learning** This section introduces the metric learning and the network architecture, such that the objectives **A) - E)** are realized.

As can be seen in Figure 4.7, the network architecture consists of two encoders,  $f_I : \mathbf{I} \mapsto \mathbf{z}_I$  for the image and  $f_T : \mathcal{T} \mapsto \mathbf{z}_T$  for the trajectory. The two intermediate representations are concatenated and then passed through the network  $f_c : [\mathbf{z}_T, \mathbf{z}_I] \mapsto \mathbf{z}$  to create the final latent representation  $\mathbf{z} \in \mathbb{R}^{N_z}$ . Finally, a decoder  $g : \mathbf{z} \mapsto \hat{\mathbf{X}}$  is used to generate a merged representation  $\hat{\mathbf{X}} \in \mathbb{R}^{S \times S \times 2}$  of infrastructure and trajectory.

The data quadruplet  $(\mathcal{X}_a, \mathcal{X}_{\text{pp}}, \mathcal{X}_{\text{pn}}, \mathcal{X}_{\text{nn}})$  is passed through the encoders  $f_I, f_T$ , and  $f_c$ , such that the latent representations  $\mathbf{z}_a, \mathbf{z}_{\text{pp}}, \mathbf{z}_{\text{pn}}, \mathbf{z}_{\text{nn}}$  are generated. The squared distances from the anchor to the samples are defined as  $d_{\text{pp}} = \|\mathbf{z}_a - \mathbf{z}_{\text{pp}}\|_2^2$ ,  $d_{\text{pn}} = \|\mathbf{z}_a - \mathbf{z}_{\text{pn}}\|_2^2$ , and  $d_{\text{nn}} = \|\mathbf{z}_a - \mathbf{z}_{\text{nn}}\|_2^2$ . The objectives **A) - D)** can then be formulated as

$$\text{B) \& C): } d_{\text{nn}} \geq d_{\text{pn}} + \alpha_G, \quad (4.9)$$

$$\text{A) \& B): } d_{\text{pn}} \geq \max\{d_{\text{pp}}, \alpha_T\} + \alpha_R, \quad (4.10)$$

$$\text{A) \& D): } d_{\text{pp}} = (1 - s_t)\alpha_T, \quad (4.11)$$

where  $\alpha_G, \alpha_R$ , and  $\alpha_T$  are margin parameters. Those constraints lead to the following loss formulations

$$\mathcal{L}_G = \max\{\alpha_G + d_{\text{pn}} - d_{\text{nn}}, 0\}, \quad (4.12)$$

$$\mathcal{L}_R = \max\{\alpha_R + \max\{\alpha_T, d_{\text{pp}}\} - d_{\text{pn}}, 0\}, \quad (4.13)$$

$$\mathcal{L}_T = |(1 - s_t)\alpha_T - d_{\text{pp}}|. \quad (4.14)$$

The losses  $\mathcal{L}_G$  and  $\mathcal{L}_R$  are basic triplet losses [SKP15], when optimizing both, it leads to the quadruplet loss as presented in [ZZLZ16]. However, in this work, additionally to the quadruplet loss, the objective is to further refine the similarity between the anchor representation and the representation of the sample with similar infrastructure and similar route, according to  $s_t$ . Like in [JWS<sup>+</sup>19], a L2-regularization term for the embedding vectors is added to the overall loss, with a weight of 0.01, this way the representations are kept from being pushed too far away.

The objective in **E)** is not directly addressed by the former loss definitions, hence another strategy needs to be adopted. For this purpose, the latent representation of a scenario is used to reconstruct the scenario. In this case, the reconstruction generates an image  $\hat{\mathbf{X}}$  with two channels, one channel for the infrastructure as in  $\mathbf{I}$  and one channel

for the trajectory information as in  $\mathcal{T}$ . This way, the network has to connect the image information with the trajectory information. Furthermore, for the decoding to work properly, the neighborhood in the latent space has to share high similarities. To further aid the training, the target reconstruction image  $\mathbf{X}$  only shows the lanes which are part of the graph. Since simple reconstruction might fail due to the high sparsity of the generated output, the reconstruction loss is adopted piece-wise for trajectory, infrastructure, and respective background pixels. The weighted sparse reconstruction loss, which embeds the objective **E**), is defined as

$$\mathcal{L}_{\text{Rec}} = \gamma_{\text{I}}\mathcal{R}(\mathcal{I}_{\text{I}}) + \gamma_{\bar{\text{I}}}\mathcal{R}(\mathcal{I}_{\bar{\text{I}}}) + \gamma_{\text{T}}\mathcal{R}(\mathcal{I}_{\text{T}}) + \gamma_{\bar{\text{T}}}\mathcal{R}(\mathcal{I}_{\bar{\text{T}}}), \quad (4.15)$$

with  $\mathcal{R}(\mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} (\mathbf{X}_{\text{a}}(i) - \hat{\mathbf{X}}_{\text{a}}(i))^2$  the reconstruction error for the pixel subset  $\mathcal{I}$ . The subset  $\mathcal{I}_{\text{I}}$  refers to all ground truth infrastructure pixels,  $\mathcal{I}_{\bar{\text{I}}}$  all remaining pixels in the infrastructure channel. For the trajectory pixels  $\mathcal{I}_{\text{T}}$  and  $\mathcal{I}_{\bar{\text{T}}}$  the logic applies respectively.

Combining all the loss definitions leads the overall loss as

$$\mathcal{L} = \beta_{\text{M}}(\beta_{\text{G}}\mathcal{L}_{\text{G}} + \beta_{\text{R}}\mathcal{L}_{\text{R}} + \beta_{\text{T}}\mathcal{L}_{\text{T}}) + \beta_{\text{Rec}}\mathcal{L}_{\text{Rec}}, \quad (4.16)$$

with the hyperparameters:  $\beta_{\text{M}}$  controlling the influence of the metric learning term,  $\beta_{\text{Rec}}$  controlling the influence of the reconstruction term, and the hyperparameters  $\beta_{\text{G}}$ ,  $\beta_{\text{R}}$ , and  $\beta_{\text{T}}$  controlling the respective loss subterms of the metric learning term. Training the network with  $\mathcal{L}$  aims to realize all the domain objectives **A**) - **D**) as defined in the beginning of this section.

#### 4.2.1.4 Extension

The presented method focuses only on the latent space formed by  $\mathbf{z}$ . Nevertheless, the other latent representations  $\mathbf{z}_{\text{T}}$  and  $\mathbf{z}_{\text{I}}$  could be utilized as well. In this subsection, a straight forward extension of the proposed method is shown. It basically adopts the triplet learning regime introduced in the former section, to the latent representation  $\mathbf{z}_{\text{I}}$  of the infrastructure encoder. Even though it would be possible, the latent representation from the trajectory encoder is not used for any learning objective in this extension.

Given the data quadruplet as defined before, the used infrastructure latent representations are defined as  $\mathbf{z}_{\text{I,a}}$ ,  $\mathbf{z}_{\text{I,pp}}$ ,  $\mathbf{z}_{\text{I,nn}}$ , where  $\mathbf{z}_{\text{I,pn}}$  is not used since only a triplet is required. The distances to the anchor representation are defined as  $d_{\text{I,pp}} = \|\mathbf{z}_{\text{I,a}} - \mathbf{z}_{\text{I,pp}}\|_2^2$  and  $d_{\text{I,nn}} = \|\mathbf{z}_{\text{I,a}} - \mathbf{z}_{\text{I,nn}}\|_2^2$ . The triplet loss for the infrastructure latent representations can be defined as

$$\mathcal{L}_{\text{I}} = \max\{\alpha_{\text{I}} + d_{\text{I,pp}} - d_{\text{I,nn}}, 0\}. \quad (4.17)$$

This loss forces the latent representations  $\mathbf{z}_{\text{I}}$  to follow the infrastructure similarity, as in the previous section. Using this as an extension for the proposed quadruplet autoencoder,

the overall loss becomes

$$\mathcal{L}_{\text{ext}} = \beta_{\text{M}}(\beta_{\text{I}}\mathcal{L}_{\text{I}} + \beta_{\text{G}}\mathcal{L}_{\text{G}} + \beta_{\text{R}}\mathcal{L}_{\text{R}} + \beta_{\text{T}}\mathcal{L}_{\text{T}}) + \beta_{\text{Rec}}\mathcal{L}_{\text{Rec}}. \quad (4.18)$$

This way, both latent representations should be optimized to follow the domain knowledge similarities. This could enable a separate analysis between infrastructure and a complete scenario. Applications benefiting from such an extension would for example be to identify whether the infrastructure or only the behavior inside that infrastructure is unknown. Therefore, it provides the possibility to reason, on what level a traffic scenario is novel.

#### 4.2.1.5 Alternative Approaches

In this section, the objective is to generate representations which are well suited for clustering and outlier detection, as well as showing high similarities between features among neighbors. The proposed quadruplet method aims to solve that task through a combination of metric learning and the reconstruction regime. In order to evaluate the proposed method better, alternative approaches to solve the task are examined. In the remainder of this Section 4.2.1.5, the setting for each alternative approach is briefly explained, while in the following Section 4.2.2 the results of these alternative approaches are put into comparison. All methods relying with alternative metric losses rely on the quadruplet sampling as introduced in the proposed method.

**Classifier** Since the information of similarity between the graphs is available, one can create groups of similar samples, hence, classes. This can be realized on an infrastructure level and on a route level. Therefore, let  $C_{\text{G}}$  be the number of infrastructure classes ( $C_{\text{G}}$  unique infrastructure graphs in the training data), while  $C_{\text{R}}$  is the number of route classes. The encoder part of the proposed architecture is kept. Two classification heads  $g_{\text{class,G}}$  and  $g_{\text{class,R}}$  are plugged after  $f_{\text{c}}$ , hence  $g_{\text{class,G}} : \mathbf{z} \mapsto \{\mathcal{C}_{\text{G},c}\}_{c=1}^{C_{\text{G}}}$  and  $g_{\text{class,R}} : \mathbf{z} \mapsto \{\mathcal{C}_{\text{R},c}\}_{c=1}^{C_{\text{R}}}$ . The overall architecture  $f_{\text{I}}$ ,  $f_{\text{T}}$ ,  $f_{\text{c}}$ ,  $g_{\text{class,G}}$ , and  $g_{\text{class,R}}$  is trained using the cross entropy loss for classification. For this, the two loss terms of the two classification heads are simply summed. The classification heads are both realized through a MLP with two hidden layers. The architecture of the MLP uses  $(N_{\mathbf{z}} - C_{\text{G}} - C_{\text{R}})$  number of neurons in the corresponding layers. The narrative behind this approach is the assumption, that the latent representations have to be expressive such that both classifiers perform well.

**Autoencoder** An approach without the graph-based similarity is when using only the reconstruction regime. Hence, the overall architecture is kept as in the proposed method. But here, only the reconstruction loss is used to train the network:

$$\mathcal{L}_{\text{A,AE}} = \beta_{\text{Rec}}\mathcal{L}_{\text{Rec}}. \quad (4.19)$$

**Contrastive Learning** In the proposed method, the objectives lead to the loss definition which is essentially using two triplet losses. Instead, one could replace the triplet loss terms with a more simple contrastive loss (Section 2.5.2.1). Hence, the relation between a negative and a positive sample is not specifically defined. The loss terms  $\mathcal{L}_G$  and  $\mathcal{L}_R$  are adjusted such that the contrastive loss Equation (2.75) is used. This leads to the adjusted loss as

$$\mathcal{L}_{G,\text{cont}} = d_{\text{pn}}^2 + \max\{\alpha_G - d_{\text{nn}}^2, 0\}, \quad (4.20)$$

$$\mathcal{L}_{R,\text{cont}} = \max\{\alpha_T, d_{\text{pp}}\}^2 + \max\{\alpha_R - d_{\text{pn}}^2, 0\} \text{ and} \quad (4.21)$$

$$\mathcal{L}_{A,\text{cont}} = \beta_M(\beta_G\mathcal{L}_{G,\text{cont}} + \beta_R\mathcal{L}_{R,\text{cont}} + \beta_T\mathcal{L}_T) + \beta_{\text{Rec}}\mathcal{L}_{\text{Rec}}. \quad (4.22)$$

In order to compensate the missing ranking between the infrastructure similarity and the route similarity, the hyperparameters are selected to be  $\beta_R = 10$  and  $\beta_I = 1$ , such that distances between samples of similar route are penalized stronger. The margins are kept as  $\alpha_G = \alpha_R = \alpha_T = 1$ .

**Cross Entropy-Based Contrastive Learning** As demonstrated in Section 2.5.2.3, contrastive learning with cross entropy loss can be interpreted as an approximation of the triplet loss. Therefore, an obvious alternative is to replace the triplet losses by cross entropy-based losses. The cross entropy-based contrastive loss for a single negative instance using dissimilarities is defined as

$$\mathcal{L}_{\text{CE}} = \frac{1}{\beta} \log(1 + \exp(\beta(\alpha + d_p - d_n))). \quad (4.23)$$

Here, only the  $\frac{1}{\beta}$  scaled version of the loss is used in order to keep the similarity to the triplet loss. When adjusting the loss terms to follow the cross entropy-based contrastive loss, the following definitions result:

$$\mathcal{L}_{G,\text{CE}} = \frac{1}{\beta} \log(1 + \exp(\beta(\alpha_G + d_{\text{pn}} - d_{\text{nn}}))), \quad (4.24)$$

$$\mathcal{L}_{R,\text{CE}} = \frac{1}{\beta} \log(1 + \exp(\beta(\alpha_R + \max\{\alpha_T, d_{\text{pp}}\} - d_{\text{pn}}))), \quad (4.25)$$

$$\mathcal{L}_{A,\text{CE}} = \beta_M(\beta_G\mathcal{L}_{G,\text{CE}} + \beta_R\mathcal{L}_{R,\text{CE}} + \beta_T\mathcal{L}_T) + \beta_{\text{Rec}}\mathcal{L}_{\text{Rec}}. \quad (4.26)$$

The influence of the hyperparameter  $\beta$  is analyzed in more detail in the experiments. Like before, the margins are kept as  $\alpha_G = \alpha_R = \alpha_T = 1$ , while the loss terms are weighted with  $\beta_T = \beta_R = \beta_I = 1$ .

**SwAV Similarity-Based** SwAV was presented in a self-supervised setting Section 2.5.3.1. Instead of generating two views which are somehow similar, here, it is investi-

gated if two similar samples defined by the graph similarity (like in the proposed method) can be used for learning appropriate representations as well. Therefore, the metric learning part of the proposed method is adjusted to consist of only SwAV-based losses. As in SwAV, a projection head  $g_{\text{proj}} : \mathbf{z} \mapsto \mathbf{h}$  with  $\mathbf{h} \in \mathbb{R}^{N_h}$  and  $N_h = 512$  is used. Still, the decoder is reconstructing from  $\mathbf{z}$ . The losses based on SwAV from Equation (2.108) are defined as

$$\mathcal{L}_{G,\text{SwAV}} = l(\mathbf{h}_a, \mathbf{q}_{\text{pn}}) + l(\mathbf{h}_{\text{pn}}, \mathbf{q}_a), \quad (4.27)$$

$$\mathcal{L}_{R,\text{SwAV}} = l(\mathbf{h}_a, \mathbf{q}_{\text{pp}}) + l(\mathbf{h}_{\text{pp}}, \mathbf{q}_a) \text{ and} \quad (4.28)$$

$$\mathcal{L}_{A,\text{SwAV}} = \beta_M(\beta_G \mathcal{L}_{G,\text{SwAV}} + \beta_R \mathcal{L}_{R,\text{SwAV}}) + \beta_{\text{Rec}} \mathcal{L}_{\text{Rec}}. \quad (4.29)$$

Therefore, the intuition behind is to break the graph-based similarity into two components. First, the anchor  $\mathcal{X}_a$  and the sample with similar infrastructure but different route  $\mathcal{X}_{\text{pn}}$  is used for one SwAV term, aiming to make their representations very similar. Second, the anchor  $\mathcal{X}_a$  alongside the  $\mathcal{X}_{\text{pp}}$  sample used for another SwAV term, again aiming to make them as similar as possible. As in the contrastive learning setting, the ranking could be compensated by different weighing of the two terms. For further analysis, only  $\mathbf{z}$  is used, while  $\mathbf{h}$  is actually used for the training. Besides the fact that no negative samples  $\mathbf{h}_{\text{mn}}$  are required, prototypes are learned alongside the network training.

**Barlow Twins Similarity-Based** Inspired from the idea to use the SwAV objective in the overall loss to train the network, here, the Barlow Twins loss (Section 2.5.3.2) is used in the same fashion. Instead of generating two similar samples, the similar samples are defined using the mining procedure as in the proposed method. Also, a projection network is used  $g_{\text{proj}} : \mathbf{z} \mapsto \mathbf{h}$ . Let  $\mathbf{R}(\{\mathbf{h}_a, \mathbf{h}_{\text{pn}}\}_B)$  be the correlation matrix as used in Barlow Twins loss Equation (2.112), but here using the projected representation of the anchor  $\mathbf{h}_a$  and the sample with similar infrastructure but different route  $\mathbf{h}_{\text{pn}}$ . The loss terms to train the network are then defined as

$$\mathcal{L}_{G,\text{BT}} = \sum_i (1 - \mathbf{R}_{ii}(\{\mathbf{h}_a, \mathbf{h}_{\text{pn}}\}_B))^2 + \lambda \sum_i \sum_{j \neq i} \mathbf{R}_{ij}(\{\mathbf{h}_a, \mathbf{h}_{\text{pn}}\}_B)^2, \quad (4.30)$$

$$\mathcal{L}_{R,\text{BT}} = \sum_i (1 - \mathbf{R}_{ii}(\{\mathbf{h}_a, \mathbf{h}_{\text{pp}}\}_B))^2 + \lambda \sum_i \sum_{j \neq i} \mathbf{R}_{ij}(\{\mathbf{h}_a, \mathbf{h}_{\text{pp}}\}_B)^2 \text{ and} \quad (4.31)$$

$$\mathcal{L}_{A,\text{BT}} = \beta_M(\beta_G \mathcal{L}_{G,\text{BT}} + \beta_R \mathcal{L}_{R,\text{BT}}) + \beta_{\text{Rec}} \mathcal{L}_{\text{Rec}}. \quad (4.32)$$

Like for the SwAV-based approach, no negative samples are required here.

**VICReg Similarity-Based** The last approach utilizing the similarity definition is based on VICReg (Section 2.5.3.3). Similar to the concepts in combination with SwAV and Barlow Twins, the self-supervised setting typically used in VICReg is not used, rather

the similar samples are defined based on the mining process of the proposed method. With the VICReg loss Equation (2.118), the used loss is defined as

$$\begin{aligned} \mathcal{L}_{G, \text{VICReg}} = \alpha_v \left[ \mathcal{L}_v(\{\mathbf{h}_a\}_B) + \mathcal{L}_v(\{\mathbf{h}_{\text{pn}}\}_B) \right] + \alpha_i \mathcal{L}_i(\{\mathbf{h}_a, \mathbf{h}_{\text{pn}}\}_B) \\ + \alpha_c \left[ \mathcal{L}_c(\{\mathbf{h}_a\}_B) + \mathcal{L}_c(\{\mathbf{h}_{\text{pn}}\}_B) \right], \end{aligned} \quad (4.33)$$

$$\begin{aligned} \mathcal{L}_{R, \text{VICReg}} = \alpha_v \left[ \mathcal{L}_v(\{\mathbf{h}_a\}_B) + \mathcal{L}_v(\{\mathbf{h}_{\text{pp}}\}_B) \right] + \alpha_i \mathcal{L}_i(\{\mathbf{h}_a, \mathbf{h}_{\text{pp}}\}_B) \\ + \alpha_c \left[ \mathcal{L}_c(\{\mathbf{h}_a\}_B) + \mathcal{L}_c(\{\mathbf{h}_{\text{pp}}\}_B) \right] \text{ and} \end{aligned} \quad (4.34)$$

$$\mathcal{L}_{A, \text{VICReg}} = \beta_M (\beta_G \mathcal{L}_{G, \text{VICReg}} + \beta_R \mathcal{L}_{R, \text{VICReg}}) + \beta_{\text{Rec}} \mathcal{L}_{\text{Rec}}. \quad (4.35)$$

**Self-Supervised Methods** The methods SwAV, Barlow Twins, and VICReg can be used as well in their self-supervised setting to generate representations. Hence, no definitions of similarity are used here. The encoder part ( $f_I, f_T, f_C$ ) of the proposed architecture is used for the self-supervised methods, and only the projection head  $g_{\text{proj}}$  is added. The typical augmentations as summarized in Section 2.5.3.4 are not suited for traffic scenarios represented by  $\mathcal{X}$ . Therefore, four domain knowledge guided augmentations are used to generate the necessary transformations. First, with some probability, the *infrastructure image* is *replaced* by another image which only depicts the elements which are part of the infrastructure graph as used in the proposed method. Assumption: two scenarios are the same since the core elements of the infrastructure are still the same. Second, with some probability, the same *random rotation* is applied to the image and the trajectory. Assumption: two scenarios are still the same even though they are rotated versions of each other. Third and fourth, with some probability, the image is *blurred* and *noise* is added to the trajectory points.

The methods used to train the network are the versions of SwAV, Barlow Twins and VICReg as summarized in Section 2.5.3. The analysis is realized based on  $\mathbf{z}$ .

**Dimensionality Reduction** For reference, the analysis is also performed on the data projected with PCA and UMAP to  $N_z$  dimensional space.

## 4.2.2 Experiments

The quality of the latent space constructed by the proposed method is analyzed through various experiments. The analysis is based on four perspectives: 1. Novelty detection: *Given a base set, are new scenarios detected as novel?* 2. Clustering: *Do the latent representations form meaningful clusters?* 3. Feature stability: *Are scenario features of neighboring latent representations similar?* 4. Visualization: *Can the latent space be analyzed through aided visualizations?* In order to evaluate the impact of the various loss terms and possible network variations, different settings are used for all the experiments.



This subsection is structured as follows. First, the dataset is explained. Second, the presented method is compared against alternative approaches. In the third part a comprehensive ablation study is provided. The second and third parts are structured equally: novelty detection, clustering, and feature stability. The visual assessment is shown in part four. The different results are summarized in the last part.

#### 4.2.2.1 Data

The data used to train and analyze the network is generated through simulation. The data generation process is divided in two parts, the infrastructure sampling and the simulation.

**Infrastructure Sampling** To obtain the images of road infrastructures  $I$ , the sampling is realized as in the previous section ([WBBU21]). From OSM [Ope21], random nodes are selected as center for the scenarios. The underlying road infrastructure within the area of  $200\text{ m} \times 200\text{ m}$  per node is extracted as image ( $100\text{ px} \times 100\text{ px}$ ), graph, and as map for simulation. To generate the graphs and images, the tools from [WBBU21] are used.

For generating the training dataset, the district of Upper Bavaria (roundabouts and highways) and the city of Ingolstadt with its adjacent counties (inner city and rural roads) are used as extraction regions. From all that nodes,  $\approx 70\,000$  infrastructures are sub-sampled for training, which builds the training dataset.

The state of Saarland and its state capital Saarbrücken are used to generate the validation dataset. Here,  $\approx 60\,000$  infrastructures are sub-sampled for validation.

**Simulation** To obtain the trajectory of the EGO vehicle  $\mathcal{T}$ , for each of the infrastructures, simulations in SUMO [LBBW+18] are performed. One vehicle (EGO) is inserted at the center position of the scenario, while other vehicles are randomly spawned, such that the scenarios show a rather high traffic load<sup>5</sup>. The scenario is simulated for a total time span of 6 s. The EGO trajectory  $\mathcal{T} = \{[x_1, y_1, t_1], \dots, [x_N, y_N, t_N]\}$  during this timespan is recorded.

The infrastructures as well as the routes are sampled such, that for each scenario, similar infrastructure and route samples are available.

**Groups** For the analysis with respect to groups (clustering and novelty detection), three detail levels are examined. First, a rough level, consisting of the road type categories:

1. single-lane,
2. multi-lane,
3. intersection,

---

<sup>5</sup>To achieve high traffic load, the number of vehicles per lane is estimated based on counting statistics from different road types in Bavaria.

4. intersection entering,
5. roundabout,
6. roundabout entering,
7. highway and
8. highway entering.

The second detail level considers all unique infrastructure graphs in the dataset, leading to 737 groups. Hence, those groups can be used to analyze the performance with respect to the infrastructure.

The third and most detailed level groups the samples by their unique routes combined with their graphs leading to 1692 different groups. It provides insight with respect to the complete scenario.

### 4.2.2.2 Comparison

The proposed method is compared to other approaches as discussed in Section 4.2.1.5. They can be grouped into the following types: methods with a different metric loss, the proposed extension, self-supervised methods, and alternative baseline methods. For the group of methods with a different metric loss, the overall architecture is kept as in the proposed method, only the metric loss is realized differently. The self-supervised methods only use the encoder structure of the architecture and slightly modified augmentations. The alternative methods contain a classifier approach, dimensionality reduction (UMAP and PCA), autoencoder, and an approach where the input data is used directly (plain). The overall performance of all the methods in comparison to the proposed method can be found in Table 4.3. In the following, the results for the aspects novelty detection, clustering, and feature stability is discussed.

**Novel Scenario Detection** Detecting novel scenarios is a crucial task in the validation process of autonomous driving. The latent space designed by the former method suits this need, as shown in this section. Except for the plain approach, either the latent representations  $z$ , or the dimensionality reduced representation (UMAP and PCA) are used for the novelty detection. Here, detecting novel scenarios is realized through outlier detection. Therefore, assuming a base dataset (the already known scenarios), the task is to identify scenarios which do not fit in the base dataset.

The novelty detection is performed as  $n$ -vs-1, where one group (Section 4.2.2.1) is excluded from the base dataset. It is tested, how well this left-out group is detected as novel. This procedure is repeated for all groups. The novelty detection performance is measured using the *Area Under Curve* (AUC). *Angle Based Outlier Detection* (ABOD) is

Table 4.3: Comparison Performance Summary: red indicates worse than the proposed method, green better, and yellow comparable. The subscripts of the  $AUC$  and the  $ACC$  scores stand for: C category level, G graph level, and R route level. The subscripts of the  $\bar{d}$  scores stand for the feature stability with respect to the:  $I$  image,  $T$  trajectory,  $v$  velocity,  $a_{lon}$ ,  $a_{lat}$  longitudinal and lateral acceleration, and  $\psi$  orientation.

Approach	Novelty Detection $\uparrow$			Clustering $\uparrow$			Feature Stability $\downarrow$					
	$AUC_C$	$AUC_G$	$AUC_R$	$ACC_C$	$ACC_G$	$ACC_R$	$\bar{d}_I$	$\bar{d}_T$	$\bar{d}_v$	$\bar{d}_{a_{lon}}$	$\bar{d}_{a_{lat}}$	$\bar{d}_\psi$
<b>Proposed</b>	0.892	0.907	0.865	0.817	0.553	0.479	36.18	0.53	1.97	0.49	0.29	0.10
	Different Metric Loss											
$\mathcal{L}_{A,cont}$	0.875	0.888	0.850	0.624	0.436	0.266	35.37	0.30	0.82	0.44	0.27	0.09
$\mathcal{L}_{A,VICReg}$	0.871	0.865	0.837	0.616	0.480	0.393	36.78	0.51	1.93	0.46	0.29	0.10
$\mathcal{L}_{A,SwAV}$	0.880	0.811	0.775	0.537	0.119	0.096	35.58	0.42	1.59	0.51	0.28	0.09
$\mathcal{L}_{A,BT}$	0.865	0.854	0.835	0.616	0.502	0.428	37.83	0.61	2.11	0.46	0.30	0.10
$\mathcal{L}_{A,CE}, \beta = 1000$	0.890	0.901	0.860	0.716	0.527	0.490	36.11	0.54	2.01	0.50	0.29	0.10
$\mathcal{L}_{A,CE}, \beta = 100$	0.891	0.904	0.859	0.768	0.538	0.484	36.17	0.55	2.02	0.51	0.30	0.10
$\mathcal{L}_{A,CE}, \beta = 10$	0.887	0.902	0.859	0.793	0.547	0.486	36.07	0.51	1.87	0.52	0.29	0.10
$\mathcal{L}_{A,CE}, \beta = 2$	0.885	0.885	0.845	0.774	0.489	0.410	36.21	0.53	1.93	0.53	0.29	0.10
$\mathcal{L}_{A,CE}, \beta = 1$	0.897	0.890	0.849	0.754	0.405	0.367	36.20	0.48	1.77	0.50	0.29	0.10
	Extension											
Stacked	0.886	0.916	0.866	0.682	0.636	0.501	36.06	0.56	2.22	0.53	0.29	0.10
	Self-Supervised Methods											
SwAV	0.203	0.239	0.271	0.324	0.074	0.071	39.73	0.48	0.55	0.44	0.35	0.12
VICReg	0.715	0.623	0.632	0.447	0.072	0.072	39.47	0.51	0.60	0.45	0.34	0.12
Barlow Twins	0.734	0.641	0.651	0.427	0.050	0.065	39.45	0.49	0.59	0.46	0.34	0.12
	Alternative Methods											
Classifier	0.860	0.760	0.731	0.704	0.132	0.097	39.70	0.90	2.41	0.51	0.36	0.12
UMAP	0.613	0.116	0.063	0.613	0.116	0.063	35.96	0.90	5.02	0.67	0.35	0.11
PCA	0.692	0.699	0.665	0.613	0.482	0.406	29.86	1.08	4.63	0.58	0.34	0.11
Autoencoder	0.818	0.758	0.741	0.612	0.093	0.092	35.57	0.41	1.65	0.52	0.27	0.09
Plain	0.500	0.485	0.487	0.265	0.255	0.271	28.80	1.47	5.96	0.74	0.43	0.30

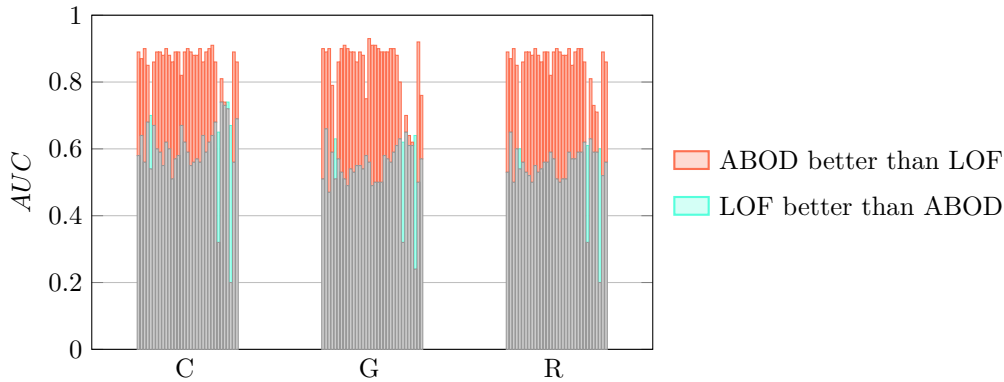


Figure 4.8:  $AUC$  performance ABOD vs. LOF over various representation learning settings. Red/cyan performance loss/gain when using LOF instead of ABOD. Peaks show best overall performance. C category level, G graph level, and R route level.

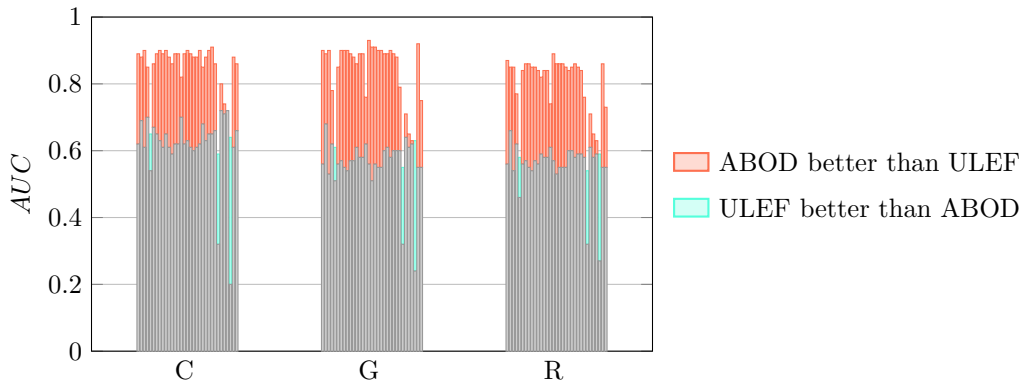


Figure 4.9:  $AUC$  performance ABOD vs. ULEF over various representation learning settings. Red/cyan performance loss/gain when using ULEF instead of ABOD. Peaks show best overall performance. C category level, G graph level, and R route level.

used as the novelty detection method. Using other outlier detection methods, like LOF and ULEF decreased the performance in most of the cases. The novelty detection performance of ABOD compared to LOF and ABOD compared to ULEF can be found in Figure 4.8 and Figure 4.9.

As the results show (Table 4.3), detecting novel scenarios is best realized in the latent space formed by the proposed method or in the latent space formed by the approximated version (see Section 2.5.2.3 about cross entropy-based contrastive loss)  $\mathcal{L}_{A,CE}, \beta = 1, \dots, 1000$ . Apart from them, only the classic contrastive learning approach  $\mathcal{L}_{A,cont}$  reaches comparable performance, when considering all the three detail levels  $C$ ,  $G$ , and  $R$ . Nevertheless, the group of methods with alternative metric loss outperforms all self-supervised and alternative methods. The group of self-supervised methods performs surprisingly weak, especially when compared to a classical autoencoder approach. This might be due to the augmentations, which are rather slight. Interestingly, for all

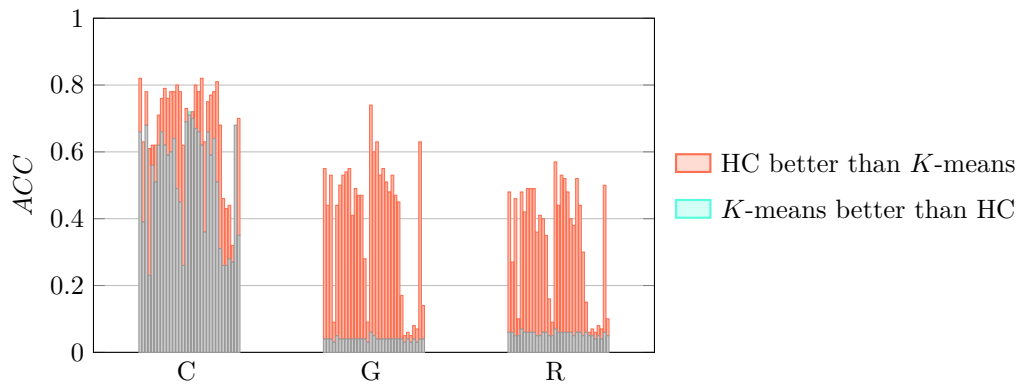


Figure 4.10:  $ACC$  performance HC vs.  $K$ -means over various representation learning settings. Red/cyan performance loss/gain when using  $K$ -means instead of HC. Peaks show best overall performance.

self-supervised methods, the outlier detection method LOF performs significantly better, however, still it falls behind the classical autoencoder approach. In the following Section 4.3, an approach focusing on strong augmentations for traffic scenarios is presented.

**Clustering** Another task in the field of validating AVs is to cluster scenarios into groups. This way, possible representatives for testing per cluster can be defined. In the following, the clustering performance is demonstrated when using the designed latent space. As for the novelty detection, the latent representations  $z$ , or the dimensionality reduced representation (UMAP and PCA) are used for the clustering.

Because of the highly imbalanced number of samples per group (Section 4.2.2.1), agglomerative *Hierarchical Clustering* (HC) suits this task well. As linkage function, the average is used. The clustering performance is stated as accuracy  $ACC$  [YXN<sup>+</sup>10]. For this, the best mapping between the ground truth labels and the predicted labels is determined. Given this mapping the accuracy can be determined. For reference, the results when applying  $K$ -means instead of HC for various representation learning settings are illustrated in Figure 4.10. For the category level C, the  $K$ -means performs worse than the HC but still somehow competitive. On the graph level G and on the route level R, the clustering result of  $K$ -means dramatically decreases compared to HC. This is most likely due to the highly imbalanced nature of the analyzed classes. Hence, in this application it is opted for HC for further experiments.

For the clustering, only the approximated versions with  $\beta = 10$  reached comparable performance to the proposed method (Table 4.3). Compared to the self-supervised methods, the performance gap is even bigger. The same holds for the alternative methods. Hence, for clustering traffic scenarios, the proposed method is best suited and therefore the approach which shows the most promising results for the identification of relevant traffic scenarios.

**Feature Stability** As stated in the design requirements **E**), one of the objectives is that neighbors in the latent space share high similarities with respect to various features. An analysis accessing this is realized in this section.

To analyze the stability within a neighborhood, for each data point, the 15 nearest neighbors in the latent space are considered for the further analysis. The average differences from the data points in focus to their neighbors are determined. For calculating the differences  $d$  various features are used: 1.  $d_I$  image difference (like in the previous section [WBBU21]), 2.  $d_T$  trajectory difference (average displacement), 3.  $d_v$  average velocity difference, 4.  $d_{a_{\text{lon}}}$  average longitudinal acceleration difference, 5.  $d_{a_{\text{lat}}}$  average lateral acceleration difference and 6.  $d_\psi$  average orientation difference. Those values are averaged over the complete dataset, leading to  $\bar{d}_{\dots}$ . The smaller those average values, the more similar the features within the neighborhood, hence the better the objective is fulfilled. It is important to note, that the features **3** - **6** are not part of the input, and are generated additionally during simulation for analysis purposes only.

The results are listed in Table 4.3 column *feature stability*. The performance difference is not as clear as for the novelty detection and the clustering. Mainly, three methods stand out, and outperform the proposed method:  $\mathcal{L}_{A,\text{cont}}$ ,  $\mathcal{L}_{A,\text{SwAV}}$ , and Autoencoder. In general, the performance among the methods with different metric loss compared to the proposed method is very much comparable. The self-supervised methods show strong results for some trajectory features but weak results for the infrastructure feature.  $\mathcal{L}_{A,\text{cont}}$  is the overall best performing method with respect to feature stability. However, the proposed method is performing on a reasonable level.

**Overall Comparison** Since all three analysis aspects are of interest, here the methods are compared considering the overall performance. For this, the comparison is split into the following groups: cross entropy-based contrastive loss, different metric loss, self-supervised methods, and alternative methods. Various figures illustrate the overall performance, the numbers underlying the figures can be found in Table 4.3. The normalization for each of the following figures, is selected such that the overall minimum and maximum value per feature defines the scaled interval.

When comparing the proposed method, to the methods applying its approximated version  $\mathcal{L}_{A,\text{CE}}, \beta = 1, \dots, 1000$ , it becomes clear that they perform very comparable. Considering Figure 4.11, the two best results are for the proposed method and  $\mathcal{L}_{A,\text{CE}}, \beta = 10$ . Nevertheless, the relationship between the two approaches manifests itself in very similar results.

As can be seen in Figure 4.12 In the group of methods with a different metric loss, only the  $\mathcal{L}_{A,\text{CE}}, \beta = 10$  performs comparable to the proposed method.  $\mathcal{L}_{A,\text{cont}}$  performs worse on clustering and novelty detection but achieves good results for feature stability. Therefore, using the triplet loss or the related cross entropy loss  $\mathcal{L}_{A,\text{CE}}, \beta = 10$  is the best

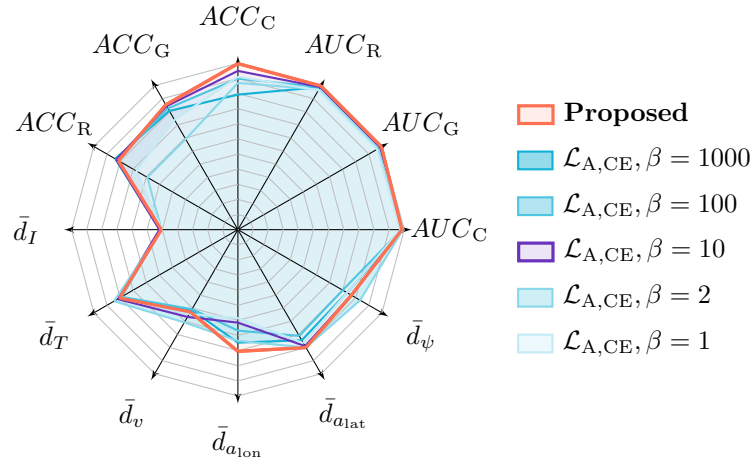


Figure 4.11: Performance Summary: **Proposed** method vs.  $\mathcal{L}_{A,CE}, \beta = 1, \dots, 1000$ .

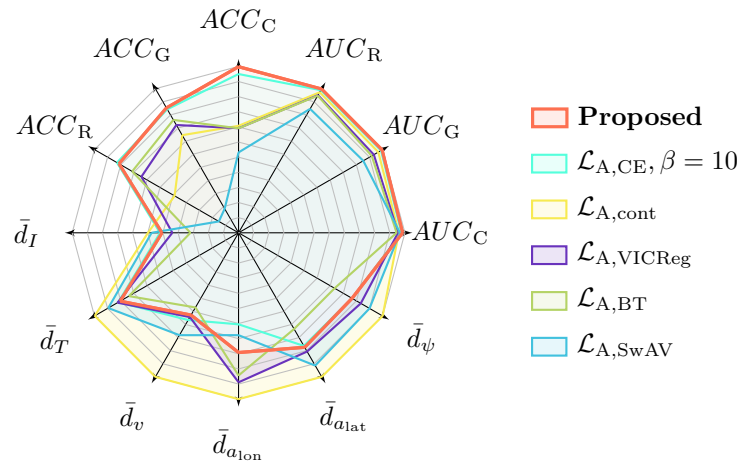


Figure 4.12: Performance Summary: **Proposed** method vs. methods with different metric loss.

choice for training the network, to satisfy most of the objectives.

In this application, the self-supervised methods fail to deliver sufficient results. Only for a few trajectory features they show higher stability than the proposed method (Figure 4.13). With respect to all other measures, the results are worse. This effect might be due to the applied augmentation strategy, which are indeed adjusted but still not very strong.

The proposed method also clearly outperforms all the alternative methods (see Figure 4.14). Only the autoencoder setting shows better results throughout the most feature stability measures.

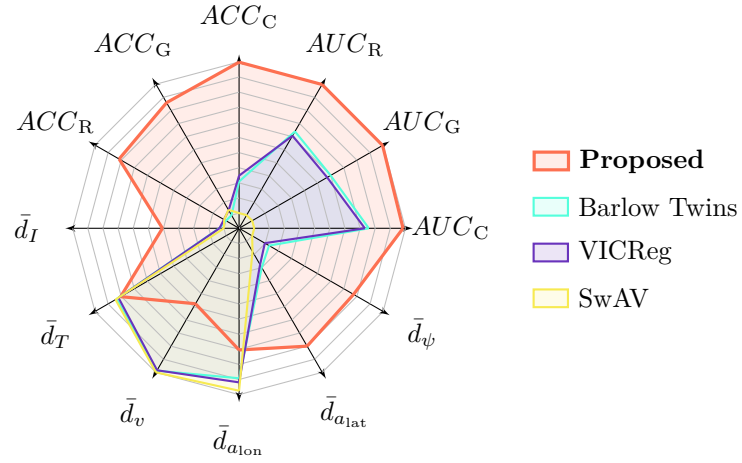


Figure 4.13: Performance Summary: **Proposed** method vs. self-supervised methods.

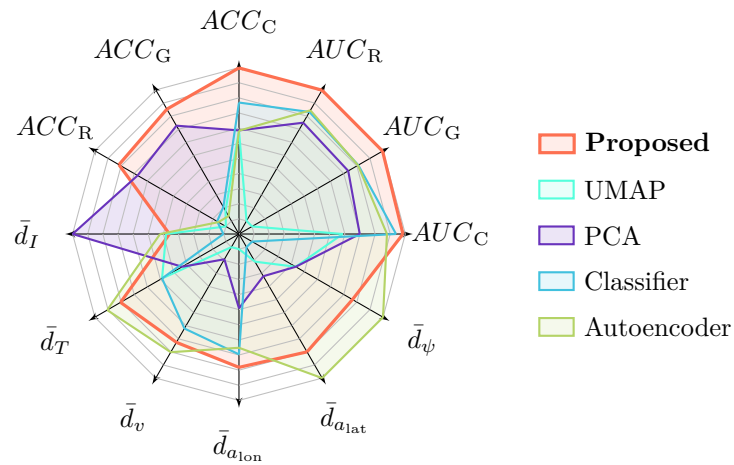


Figure 4.14: Performance Summary: **Proposed** method vs. alternative methods.



Table 4.4: Ablation Performance Summary: orange to red indicates worse than the proposed method, green better, and yellow comparable. The subscripts of the  $AUC$  and the  $ACC$  scores stand for: C category level, G graph level, and R route level. The subscripts of the  $\bar{d}$  scores stand for the feature stability with respect to the:  $I$  image,  $T$  trajectory,  $v$  velocity,  $a_{lon}$ ,  $a_{lat}$  longitudinal and lateral acceleration, and  $\psi$  orientation.

Approach	Novelty Detection 4.2.2.2			Clustering 4.2.2.2			Feature Stability 4.2.2.2					
	$AUC_C$	$AUC_G$	$AUC_R$	$ACC_C$	$ACC_G$	$ACC_R$	$\bar{d}_I$	$\bar{d}_T$	$\bar{d}_v$	$\bar{d}_{a_{lon}}$	$\bar{d}_{a_{lat}}$	$\bar{d}_\psi$
<b>Proposed</b>	0.892	0.907	0.865	0.817	0.553	0.479	36.18	0.53	1.97	0.49	0.29	0.10
$\beta_T = 0$	0.890	0.888	0.844	0.798	0.466	0.348	35.88	0.53	2.12	0.52	0.29	0.10
$\beta_T = 0, \beta_R = 0$	0.891	0.882	0.836	0.781	0.272	0.161	36.40	0.46	1.76	0.51	0.28	0.10
$\beta_M = 0$	0.818	0.758	0.741	0.612	0.093	0.092	35.57	0.41	1.65	0.52	0.27	0.09
$\beta_{Rec} = 0$	0.889	0.928	0.883	0.729	0.731	0.572	39.18	0.75	1.55	0.44	0.35	0.12
$\alpha_G = 10, \alpha_R = 5$	0.896	0.910	0.865	0.712	0.598	0.447	37.11	0.60	2.27	0.50	0.30	0.10
$f_I$ : ViT	0.879	0.901	0.864	0.800	0.529	0.517	36.07	0.54	2.06	0.49	0.29	0.10
$f_T$ : LSTM	0.883	0.898	0.855	0.785	0.550	0.481	35.94	0.56	2.11	0.54	0.29	0.10
random-excl	0.898	0.892	0.841	0.815	0.505	0.390	35.43	0.48	1.81	0.51	0.28	0.09
group	0.854	0.887	0.846	0.627	0.474	0.381	35.32	0.45	1.62	0.51	0.28	0.10
$N_{...} = N_{...} * 2$	0.889	0.910	0.866	0.716	0.633	0.526	35.61	0.48	1.77	0.48	0.28	0.10
$N_z = 32$	0.887	0.900	0.858	0.750	0.526	0.512	36.20	0.54	2.02	0.53	0.29	0.10
$N_z = 16$	0.899	0.894	0.850	0.766	0.469	0.436	36.67	0.61	2.46	0.55	0.29	0.10
$N_z = 8$	0.903	0.881	0.839	0.783	0.447	0.303	36.90	0.57	2.18	0.54	0.30	0.10
$N_z = 4$	0.858	0.794	0.769	0.806	0.166	0.148	37.70	0.44	1.46	0.53	0.29	0.10
$N_z = 2$	0.318	0.319	0.322	0.680	0.050	0.052	39.05	0.86	3.25	0.61	0.32	0.10

### 4.2.2.3 Ablation

**Model Variants** To assess the impact of the various possible settings of the network and the learning process, they are varied and compared.

**Proposed Setting:** The setting which shows overall good performance is as follows:  $f_I$ : ResNet-18 [HZRS16],  $f_T$ : Transformer-Encoder [VSP+17],  $N_I = 64$ ,  $N_T = 16$ ,  $N_z = 64$ ,  $\beta_M = \beta_G = \beta_R = \beta_T = 1$ ,  $\beta_{Rec} = 10$ ,  $\gamma_I = \gamma_T = 5$ ,  $\gamma_I = 10$ ,  $\gamma_T = 20$ ,  $\alpha_G = \alpha_R = \alpha_T = 1$ , and random negative sampling. For the decoder  $g$ , the architecture is kept similar throughout the experiemnts<sup>6</sup> as it is used for regularization purposes of the latent space only.

In the Transformer-Encoder ( $f_T$ ), an embedding token is used like in [WBBU21]. As alternative  $f_T$ , a LSTM [HS97] is used. And as alternative image encoder  $f_I$ , a ViT [DBK+21] with patch-size of 10, dimensionality of 256, MLP dimensionality of 128, 16 layers, and 16 heads is used.

All the other variants used in the ablation, adjust few parameters from the proposed setting. For example, in the variant  $\beta_M = 0$  just the according value is changed, all other values are as stated above.  $N_{...} = N_{...} * 2$  indicates that the dimensionality of all latent

<sup>6</sup>The architecture of the decoder is as follows in PyTorch notation: ConvTranspose2D( $N_z, 64, 6, 2$ )  $\rightarrow$  ReLU  $\rightarrow$  ConvTranspose2D( $64, 128, 8, 2$ )  $\rightarrow$  ReLU  $\rightarrow$  ConvTranspose2D( $128, 128, 8, 2$ )  $\rightarrow$  ReLU  $\rightarrow$  ConvTranspose2D( $128, 128, 8, 2$ )  $\rightarrow$  ReLU  $\rightarrow$  ConvTranspose2D( $128, 64, 6, 1$ )  $\rightarrow$  ReLU  $\rightarrow$  ConvTranspose2D( $64, 2, 6, 1$ )  $\rightarrow$  sigmoid

representations ( $z_I$ ,  $z_T$ , and  $z$ ) is doubled, while for  $N_z = 32$ , only the dimensionality of  $z$  is changed.

For the negative sampling, the following strategies are examined: random, i. e., from all graphs that are different, group, i. e., from all graphs that are different but only inside the same category (highway, etc.) and random-excl, i. e., from all graphs that are different excluding the same category.

**Novel Scenario Detection** In Table 4.4, the results for various model variations are shown. The description in the left column, indicates what parameters are changed compared to the proposed setting.

As one can see, detecting novel scenarios is realized best either with the proposed setting,  $\alpha_G = 10$  and  $\alpha_R = 5$ ,  $N_{...} = N_{...} * 2$ ,  $f_I : \text{ViT}$  or  $f_T : \text{LSTM}$  settings. Hence, the choice of the margin parameters has neglectable effect on the performance.

The performance with double sized latent spaces  $N_{...} = N_{...} * 2$  increases over the proposed setting. When reducing the dimensionality, the performance for the graph and route based novelty detection drops. Hence, either the proposed setting or the setting  $N_{...} = N_{...} * 2$  is preferable for novelty detection.

The importance of each loss term can be seen from Table 4.4 (rows 2-5). The metric learning related losses ( $\mathcal{L}_M$ ,  $\mathcal{L}_G$ ,  $\mathcal{L}_R$  and  $\mathcal{L}_T$ ) are required for good performance.

**Clustering** The clustering accuracies for the model variants are shown in the corresponding columns in Table 4.4. Only the model variants when using ViT or LSTM encoder is achieving comparable results to the proposed setting.

Increasing the size of the latent space  $N_{...} = N_{...} * 2$  has a negative effect on the clustering accuracy for categories but the accuracies for clustering graphs and routes increases. Therefore, using double sized latent spaces might be preferable if accuracy on the category level can be sacrificed.

As for the novelty detection, the metric learning related losses ( $\mathcal{L}_M$ ,  $\mathcal{L}_G$ ,  $\mathcal{L}_R$ , and  $\mathcal{L}_T$ ) are important.

**Feature Stability** The double sized latent space setting  $N_{...} = N_{...} * 2$  achieves better results than the proposed setting. Using only the reconstruction loss ( $\beta_M = 0$ ) does outperform the proposed setting in terms of feature stability. Not using the reconstruction loss ( $\beta_{\text{Rec}} = 0$ ) has a negative effect for the most features. This supports the designed intuition to use the autoencoder regime to achieve feature stability.

#### 4.2.2.4 Visualization

The resulting latent representations can be assessed by visualizing them. Therefore, UMAP is used to project the representation of the training data into a two-dimensional

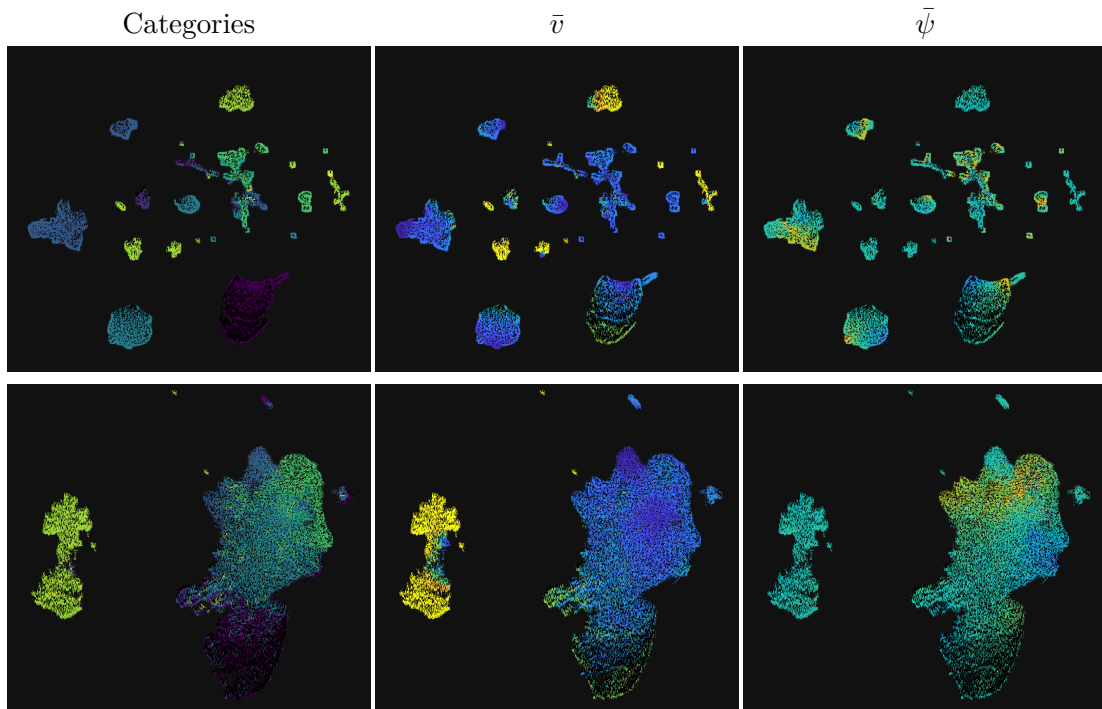


Figure 4.15: UMAP visualizations of the latent spaces of the proposed setting (top) and with  $\alpha_M = 0$  (bottom). *Categories*: **single-lane**, **multi-lane**, **intersection**, **intersection-enter**, **roundabout**, **roundabout-enter**, **highway**, and **highway-enter**.  $\bar{v}$ : **0 m/s ... 42 m/s**  $\bar{\psi}$ :  **$-\pi$  ...  $+\pi$** .

space. In Figure 4.15, the projections of two latent representations are depicted. The upper row shows the projection for the proposed setting, and the bottom row shows the projection for the setting  $\alpha_M = 0$  (turning off the metric learning). In the columns different color codings are used. In the first, the categories as in Section 4.2.2.1 are used. The second shows the average velocity of the trajectory and the last show values related to the orientation of the trajectory. The two latent representations were picked to demonstrate, how the latent representations differ, and how they can be analyzed using the visualization. It becomes clear, that the latent representation of the proposed setting provides well structure behavior in terms of categories, since the various infrastructure types are clearly separated. In contrast, the model without the metric loss fails in separating the categories, instead two big clusters can be seen, one for the highway scenarios and one with all other scenarios. There is a strong relationship between the inner cluster structure and the shown features when using the proposed method. Therefore, the features  $(\bar{v}, \bar{\psi})$  show smooth course within the clusters (e.g., in the lower right cluster, the average speed increases from top to bottom). This is also true for the model without metric loss, but here in a more global scale. The analysis can support in understanding and validating the latent representations. Readers interested in exploring the projections in more detail may refer

to the website published alongside this work <https://jwthi.github.io/Expert-LaSTS/>.

### 4.2.2.5 Summary

The different analysis perspectives highlight the performance with respect to specific tasks. The overall performance is summarized and best model variants are discussed.

The only approach being able to perform at a comparable level as the proposed method is the  $\mathcal{L}_{A,CE}, \beta = 10$ , which has a strong relation to the proposed method (see Section 2.5.2.3 how the cross entropy-based contrastive loss approximates the triplet loss). All the other approaches come with some disadvantages in one or the other perspective. However, a clear trend can be seen, that is including domain knowledge to the learning helps in forming more valuable representations.

As shown in the ablation study, the choice of the encoders for image  $f_I$  and trajectory  $f_T$  have a rather low impact on the overall performance. Increasing the representation spaces dimensionality as  $N_{...} = N_{...} * 2$  improved performance in most of the objectives slightly, however the clustering on group level drops.

### 4.2.3 Conclusion

In this section, a method to design a latent space for traffic scenarios by means of domain knowledge is presented. An automated mining strategy for traffic scenarios is introduced and used to find similar infrastructures and routes. This way, relative similarities as defined by domain objectives can be realized. The results in the emerging latent space outperform alternative approaches on various analysis perspectives, namely detecting novel scenarios, clustering, and feature stability. The ablation study provides deep insight on the impact of various model parameters.

The method presented can be used in the validation process for AVs. More precisely, it can support the analysis of scenarios as well as the detection of representative and novel scenarios.

To include objects surrounding the EGO vehicle can be one possible direction for further research on the proposed method. Also, the performance when using real-world data can be analyzed in a next step.

Self-supervised methods have been used for comparison using domain knowledge inspired augmentations. Analyzing the effect of using automated domain knowledge guided augmentations, when compared to standard *Computer Vision* (CV) augmentations is analyzed in the next section.

### 4.3 Domain Knowledge guided Augmentations for Self-Supervised Learning

In contrast to the previous section, this section focuses on domain knowledge guided augmentations of traffic scenarios rather than specifically defining similarities between traffic scenarios. In various domains like CV and *Natural Language Processing* (NLP), pre-training [ZGL<sup>+</sup>20, RWM21, DCLT19, RWC<sup>+</sup>19] to learn representations is seen as an essential component for solving downstream tasks. Typically, the pre-training is realized through self-supervised training with domain specific augmentations. Applying self-supervised learning for traffic scenarios benefits from domain knowledge guided augmentations as shown in the following. This presents the approach *Expert-guided Augmentation for representation learning of traffic scenarios* ExAgt, which has been summarized in [BWE<sup>+</sup>22] as well. In contrast to the previous sections, the aim of this section is to understand the influence domain knowledge based augmentations for traffic scenarios against standard CV augmentations. Hence, the standard evaluation protocol from the area of self-supervised learning is used. Given a model trained in the self-supervised setting, it is analyzed how well the representations cluster, which can be used for the identification of representative traffic scenarios as required in scenario-based validation of AVs. In contrast to the previous approach, less computational effort is needed, since no similarity measure between traffic scenarios across the training data is calculated. Moreover, not only the EGO and the infrastructure information, but also the surrounding objects (OBJs) are part of the traffic scenario representation used in this chapter.

Most of the state-of-the-art CV models use deep learning models pre-trained on Imagenet [DDS<sup>+</sup>09] to initialize the networks before fine-tuning on the intended task. A similar trend can be seen in NLP, where Word2Vec [MCCD13], BERT [DCLT19], and GPT [RWC<sup>+</sup>19] models are used to initialize task-specific models. In recent years, self-supervised learning methods [LZH<sup>+</sup>21, HMKS19] are proving to be suitable for the task of pre-training in order to learn representations without any human annotation.

In CV, the most popular self-supervised learning methods use cross-view prediction [GSA<sup>+</sup>20, ZJM<sup>+</sup>21, CMM<sup>+</sup>20]. The training of cross-view prediction methods is as follows: the input image is distorted or augmented into two different views. These two views are processed with a shared network to generate two representation vectors, one for each view. The networks objective is to make these representations similar as they are generated from the same input. There are two important components in the cross-view prediction framework: augmentations that are applied to create the two distorted views and a learning objective that makes the representation invariant of distortion. While the learning objective is domain independent, the augmentations to be applied are domain specific. More details on self-supervised learning can be found in Section 2.5.3.

This section focuses on learning representations for traffic scenarios, particularly in-

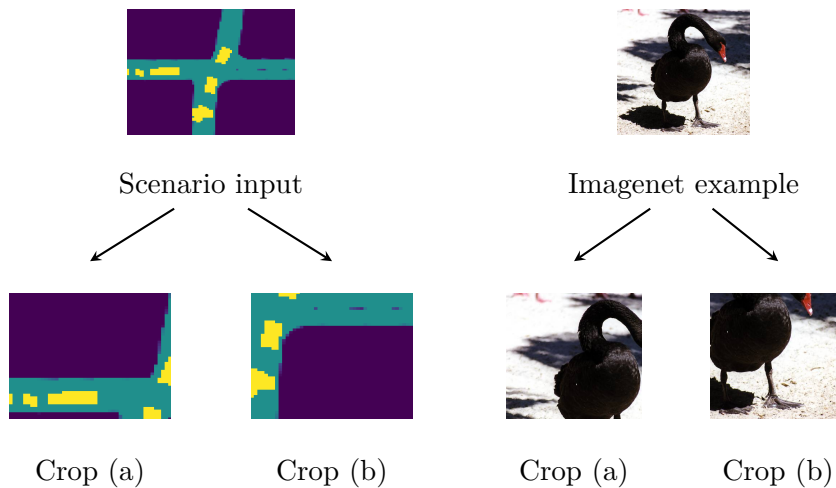


Figure 4.16: Random crop augmentation in traffic scenarios vs Imagenet images.

roducing domain specific augmentations. The traffic scenarios in this section are represented as a sequence of occupancy grids. The data is different from datasets like Imagenet, CIFAR-100 [Kri09], etc., which are used for self-supervised learning methods. The images from Imagenet and CIFAR have central objects of interest e. g., airplanes, dogs. Applying standard augmentations like random crop, gray-scale, etc., [GSA+20] is meaningful, as the distorted views contain a part of the object of interest. As can be seen in Figure 4.16, this is not necessarily the case in traffic scenarios where multiple objects are of interest like traffic participants, infrastructure, and spatio-temporal relations. Therefore, standard CV augmentations are extended with domain knowledge guided augmentations.

This section introduces two augmentations specifically created for traffic scenarios. First, the *connectivity-based* augmentation, which is using the connectivity of the underlying infrastructure and the interaction of traffic participants with the EGO vehicle in a traffic scenario. Second, the *sensor-based* augmentation, simulates sensors in the EGO vehicle with restricted Field of View (FoV) and ranges. The OBJs outside the *Visible Region* (VR) are removed to create a distorted view. Overall the novel augmentations are designed, such that they aid the goal of learning meaningful representations of traffic scenarios.

Methods using the proposed augmentations show superior performance in zero-shot<sup>7</sup> clustering, low-shot<sup>8</sup> supervised learning, and supervised learning, when compared to standard CV augmentations. The resulting model, trained self-supervised with the proposed augmentations can be used for traffic scenario clustering in order to identify representative traffic scenarios for scenario-based testing.

The main contributions of this research on self-supervised learning for traffic scenarios

<sup>7</sup>Clustering using the representation resulting from a network trained only in self-supervised regime.

<sup>8</sup>Fine tune a network that was trained in self-supervised regime with just a few labeled samples.

are as follows:

1. Novel approach to include domain knowledge for augmenting traffic scenarios.
2. Representation learning of traffic scenarios using a cross-view prediction framework.
3. Analysis of the representation space for downstream tasks like clustering, classification, and investigation of the representation space stability.

### 4.3.1 Method

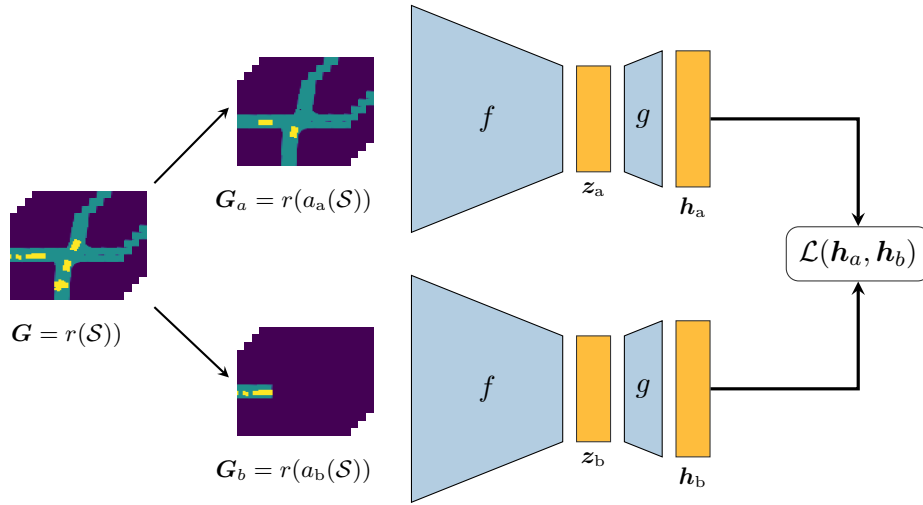


Figure 4.17: Cross-view prediction framework.

#### 4.3.1.1 Preliminaries

A dataset  $\mathcal{D} = \{(\mathcal{S}_1), \dots, (\mathcal{S}_M)\}$  is available, where  $\mathcal{S} = (\mathcal{O}, \mathcal{M})$  is a scenario. The scenario contains an object list  $\mathcal{O} = \{o_1, \dots, o_{M_{\text{obj}}}\}$  with  $M_{\text{obj}}$  objects  $o$  and a map  $\mathcal{M} = \{m_1, \dots, m_J\}$  with  $J$  map elements. Each OBJ  $o = (\mathcal{T}, \mathbf{s}, c)$  contains a trajectory  $\mathcal{T}$ , and information about the object size  $\mathbf{s}$ , and type of the object  $c$ . One map element  $m = (\mathcal{P}, \mathcal{N}, I)$  represents a lane piece. It contains the underlying polygon  $\mathcal{P}$  and connectivity information: the set of neighboring lane pieces  $\mathcal{N}$  and the intersection  $I$  which the lane piece  $m$  is part of. Per scenario, the corresponding sequence of occupancy grids  $\mathbf{G} \in \mathbb{R}^{C \times H \times W}$  can be generated by  $\mathbf{G} = r(\mathcal{S})$ , with  $H \times W$  being the size of the grids over  $C$  timestamps.

Let  $f$  be a trainable network, realizing the mapping from the occupancy grid sequence to the representation vector  $\mathbf{z} = f(\mathbf{G})$  with  $\mathbf{z} \in \mathbb{R}^{N_z}$ . Like in [ZJM<sup>+</sup>21], the representation vector  $\mathbf{z}$  is up-projected with the trainable network  $g$ , leading to the embedding vector  $\mathbf{h} = g(\mathbf{z})$  with  $\mathbf{h} \in \mathbb{R}^{N_h}$ .



As presented in Section 2.5.3, one goal of self-supervised learning is to train the networks  $f$  and  $g$  to generate meaningful representations without the need for any labelled data. The methods discussed in this work use the cross-view prediction framework. The schematics of such a framework is shown in Figure 4.17, where the actual twin architecture is depicted in contrast to the illustrations used in Section 2.5.3 to highlight the augmentations. An input is distorted to generate two views using the augmentations  $a_a$  and  $a_b$ . The encoder network  $f$  uses the two distorted views to produce representations  $z_a$  and  $z_b$ . The representations  $z_a$  and  $z_b$  are up-projected with a projection network  $g$  to get the embeddings  $h_a = g(z_a)$  and  $h_b = g(z_b)$  respectively. A learning objective  $\mathcal{L}(h_a, h_b)$  is defined using the embedding.

The objective is to learn representations that are invariant to distortions applied to the inputs. This constraint is achieved by making the representation of the distorted view of an input similar to another distorted view of the same input. An undesired outcome of this constraint is that all the representations becoming constant, i. e., mode collapse. There are different mechanisms to prevent this collapse by applying constraints in the learning objective  $\mathcal{L}$ . In the following, two of the most recent mechanisms (Barlow Twins and VICReg) are briefly summarized.

**Barlow Twins** [ZJM<sup>+</sup>21] The mechanism used in Barlow Twins is redundancy reduction between the representations of the two distorted views. This is achieved by measuring the cross-correlation matrix between the outputs of  $g(h_a)$  and  $g(h_b)$ , and making it as close to the identity matrix as possible. More details can be found in Section 2.5.3.2.

**VICReg** [BPL21] VICReg uses variance, invariance, and covariance regularization in the learning objective to prevent collapse. A comprehensive explanation of VICReg is provided in Section 2.5.3.3.

#### 4.3.1.2 Domain Knowledge-based Augmentations

Self-supervised learning methods, which are based on cross-view predictions depend on two important components: (a) the learning objective to prevent collapse and (b) possible augmentations to the input. As summarized Section 2.5.3.4, standard augmentations for the cross-view prediction frameworks in CV are random crop, color jitter, gray-scale, Gaussian blur, mix-up, etc. [GSA<sup>+</sup>20, ZJM<sup>+</sup>21, CMM<sup>+</sup>20]. But these augmentations cannot be directly used for domain specific inputs like traffic scenarios. Applying these techniques to traffic scenarios may not aid the learning as exemplified in Figure 4.16. It shows a single occupancy grid from the sequence of occupancy grids representing a scenario and an image from Imagenet. When applying the random crop augmentation on both of these images, the black swan is present in both the images. However, in the occupancy grid, a random crop creates two views, where the connection between both is



lost and can lead to a different interpretation of the scene. This is because CV datasets like Imagenet, and CIFAR-100, have a central object of interest. For domains like traffic scenarios, there are multiple objects of interest like the infrastructure, traffic participants, and the spatio-temporal relations between them. So, only standard CV augmentations might not be sufficient for domain specific inputs.

In order to create meaningful augmentations, the proposed method ExAgt introduces two types of domain specific augmentations. The first augmentation  $a_{\text{con}}$  is based on the connectivity of the underlying infrastructure and traffic participants interaction with the EGO vehicle. The second augmentation  $a_{\text{VR}}$  is based on sensor models which restrict the FoV and range of the EGO vehicle perception.

The standard CV augmentations manipulate only the image  $\mathbf{G}$  as  $\tilde{\mathbf{G}} = a(r(\mathcal{S}))$ , without any additional knowledge. In contrast, ExAgt is leveraging domain knowledge, such that the augmentations help in achieving better performance. Therefore, in ExAgt the augmented occupancy grids are generated by augmenting the scenario information first and then generate the augmented occupancy grids from the augmented scenario. Hence, the augmented occupancy grids are generated by  $\tilde{\mathbf{G}} = r(a(\mathcal{S}))$ , where  $a$  can be  $a_{\text{con}}$ ,  $a_{\text{VR}}$  or  $a_{\cap}$ , which is the combination of  $a_{\text{con}}$  and  $a_{\text{VR}}$ . With the augmented scenarios  $\mathcal{S}_{\text{con}} = a_{\text{con}}(\mathcal{S})$  and  $\mathcal{S}_{\text{VR}} = a_{\text{VR}}(\mathcal{S})$  resulting from the augmentations applied to the scenario  $\mathcal{S}$ , the combined augmentation  $a_{\cap}$  can be defined as

$$\mathcal{S}_{\cap} = (\mathcal{O}_{\text{con}} \cap \mathcal{O}_{\text{VR}}, \mathcal{M}_{\text{con}} \cap \mathcal{M}_{\text{VR}}). \quad (4.36)$$

Both augmentations follow the intuition, that certain information of the scenario can be dropped while maintaining some relevant information of the scenario. Therefore, even though scenarios might look different, they are similar with respect to some aspects. For example, in a crossing situation, not all visible participants are actually of interest for the EGO. As shown in the experiments, such rather simple, but domain knowledge-guided augmentations lead to better performance.

**Connectivity-Based Augmentation** For the connectivity-based augmentation  $a_{\text{con}}$ , some map information as well as some OBJs are dropped. The idea behind this is to retain all elements that are relevant to the EGO or to any OBJ connected to the EGO. For this purpose, the map topology of the scenarios is utilized.

The selection of the map elements and OBJs to be included in the augmented scenario  $\mathcal{S}_{\text{con}} = a_{\text{con}}(\mathcal{S})$  can be summarized as follows. All OBJs which are either directly connected to the EGO or connected to the EGO through a chain of OBJs are included in the augmented object list  $\mathcal{O}_{\text{con}}$ . Connection here means: if they pass the same map element, a neighboring map element or the same intersection. The map elements used for the augmentation  $\mathcal{M}_{\text{con}}$  are all elements, which are passed by the OBJs in  $\mathcal{O}_{\text{con}}$ , and all elements connected to  $\mathcal{M}_{\text{con}}$ , given the previous definition. This way, it is aimed to

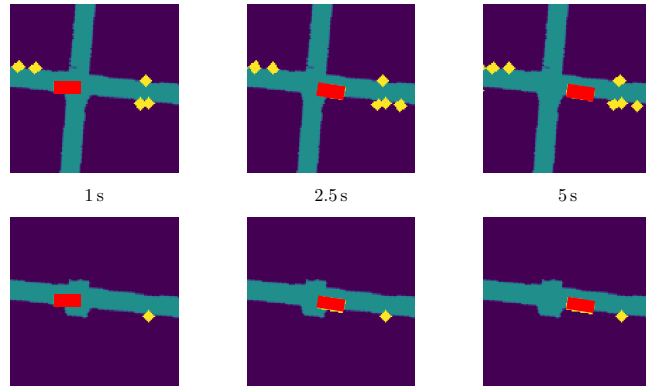


Figure 4.18: Connectivity-based augmentation  $a_{\text{con}}$ : (top) original scenario, (bottom) augmented scenario with selected information from  $\mathcal{M}$  and  $\mathcal{O}$ .

collect most of the relevant OBJs and parts of the infrastructure, which can influence the behavior or driving decisions.

To create the augmented scenario  $\mathcal{S}_{\text{con}} = (\mathcal{O}_{\text{con}}, \mathcal{M}_{\text{con}})$  as described above, the following steps are used. First, the EGO is added to the temporary object set  $\mathcal{O}_{\text{temp}}$ , then all lane elements which are passed by the EGO are gathered in  $\mathcal{M}_{\text{init}}$ . The set  $\mathcal{M}_{\text{conn}}$  contains all neighboring lanes and lanes which are part of the same intersections as the lane pieces in  $\mathcal{M}_{\text{init}}$ . The lane sets  $\mathcal{M}_{\text{init}}$  and  $\mathcal{M}_{\text{conn}}$  are merged to the temporary lane set  $\mathcal{M}_{\text{temp}}$ . After this first run, the process is repeated until the sets  $\mathcal{M}_{\text{temp}}$  and  $\mathcal{O}_{\text{temp}}$  do not change. Therefore, the object set contains all the OBJs which are topologically connected to the EGO. This approach aims to consider possible interacting traffic participants. The complete procedure is also summarized in Algorithm 4.1. An example of a resulting augmented occupancy grid sequence can be seen in Figure 4.18.

Let  $\text{inpolygon}(\mathcal{O}, m)$  be a function returning true if any of the OBJs' trajectory points in  $\mathcal{O}$  lie within the polygon  $\mathcal{P}$  of  $m$ . Hence, if any of the OBJs are passing the polygon, the function returns true. The function  $\text{connected}(\mathcal{M}, m)$  returns true if the element  $m$  is either a neighbor to any element in  $\mathcal{M}$  or the element is part of the same intersection as any element in  $\mathcal{M}$ .

**Sensor-Based Augmentation** Depending on the sensor used in a car, the VR, formed by the FoV and range, varies. Hence, the same scenario can be perceived differently. From [CPS+21], it can be seen that not all the traffic participants around the EGO vehicle do influence the motion planning performance of the EGO vehicle. The second type of augmentations is created by varying the VR of an ideal sensor

For the sensor-based augmentation  $\mathcal{S}_{\text{VR}} = a_{\text{VR}}(\mathcal{S})$ , an ideal sensor fixed at the center of the EGO vehicle is assumed. The sensor is parametrized with: maximum  $\alpha_{\text{max}}$  and minimum  $\alpha_{\text{min}}$  FoV, maximum  $d_{\text{max}}$  and minimum  $d_{\text{min}}$  range. To create a distorted view of a scenario, a random FoV  $\alpha_{\text{VR}}$  and a random range  $d_{\text{VR}}$  is sampled uniformly

---

**Algorithm 4.1** Connectivity-based augmentation  $a_{\text{con}}$ .
 

---

**Input:**  $\mathcal{S}$ 
**Output:**  $\mathcal{S}_{\text{con}}$ 

```

 $\mathcal{O}_{\text{temp}} = \{o_{\text{EGO}}\}$ 
while  $\mathcal{O}_{\text{temp}}$  or  $\mathcal{M}_{\text{temp}}$  changes do
     $\mathcal{M}_{\text{pass}} = \{m \in \mathcal{M} \mid \text{inpolygon}(\mathcal{O}_{\text{temp}}, m)\}$ 
     $\mathcal{M}_{\text{conn}} = \{m \in \mathcal{M} \mid \text{connected}(\mathcal{M}_{\text{pass}}, m)\}$ 
     $\mathcal{M}_{\text{temp}} = \mathcal{M}_{\text{pass}} \cup \mathcal{M}_{\text{conn}}$ 
     $\mathcal{O}_{\text{temp}} = \{o \in \mathcal{O} \mid \text{inpolygon}(o, m) m \in \mathcal{M}_{\text{temp}}\}$ 
end while
 $\mathcal{S}_{\text{con}} = (\mathcal{O}_{\text{temp}}, \mathcal{M}_{\text{temp}})$ 
    
```

---

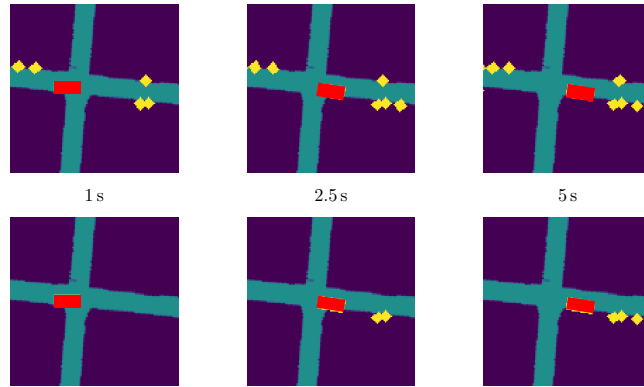


Figure 4.19: Sensor-based augmentation  $a_{\text{VR}}$ : (top) original scenario, (bottom) scenario with sensor parameters restricted to FoV as  $30^\circ$  and range as 25 m.

from  $(\alpha_{\min}, \alpha_{\max})$  and  $(d_{\min}, d_{\max})$ . Traffic participants within the VR are kept in the occupancy grids and all other traffic participants are removed. An exemplary sensor based augmentation can be seen in Figure 4.19.

If an OBJ is inside the VR can be formulated as

$$\text{inVR}(o(t)) = \begin{cases} 1 & \text{if } -\alpha_{\text{VR}} < \alpha_o < \alpha_{\text{VR}} \wedge d_o < d_{\text{VR}} \\ 0 & \text{else} \end{cases}, \quad (4.37)$$

where  $\alpha_o$  is the angle from the OBJ  $o$  to the EGO's line of sight at time  $t$  and  $d_o(t)$  is the distance between the OBJ  $o$  and the EGO at time  $t$ . Given this definition, generating the augmented scenario  $\mathcal{S}_{\text{VR}}$  can be defined as in Algorithm 4.2. Hence, every timeframe is filtered for OBJs which are inside the current VR of the EGO.

### 4.3.2 Experiments

This section discusses the experimental setup and analysis. The analysis of the representation space generated by the self-supervised learning method with and without domain

---

**Algorithm 4.2** Sensor-based augmentation  $a_{VR}$ .

---

**Input:**  $\mathcal{S}$ ,  $\alpha_{VR} = U(\alpha_{\min}, \alpha_{\max})$ ,  $d_{VR} = U(d_{\min}, d_{\max})$ **Output:**  $\mathcal{S}_{VR}$ 

```
 $\mathcal{O}_{temp} = \{o_{EGO}\}$ 
for  $o$  in  $\mathcal{O}$  do
   $o_{temp} = \{\}$ 
  for  $t$  in timestamps do
    if  $inVR(o(t))$  then
       $o_{temp}.append(o(t))$ 
    end if
  end for
   $\mathcal{O}_{temp}.append(o_{temp})$ 
end for
 $\mathcal{S}_{VR} = (\mathcal{O}_{temp}, \mathcal{M})$ 
```

---

knowledge-guided augmentation partially follows typically self-supervised learning experiment protocol (1-3). The representation space is analyzed with respect to the following questions:

1. *Zero-shot clustering*: Is there structure in the representation space formed by self-supervised pre-training?
2. *Linear evaluation*: Is the representation space formed by self-supervised pre-training linearly separable?
3. *Low-shot image classification*: Are few labelled examples enough for fine-tuning the self-supervised network for classification?
4. *Representation space stability*: Are there local neighborhood relations in the representation space?
5. *Ablation*: What is the influence of various augmentations?

#### 4.3.2.1 Dataset

For all experiments, the Argoverse [CLS+19] dataset is used. There are a total of 333 000 scenarios, each one of 5 s length and sampled at 10 Hz. All scenarios have an EGO which is present in the scenario for the complete timespan, OBJs surrounding the EGO, and map information. In order to test downstream tasks like zero-shot clustering and classification, labels for the traffic scenarios are required. To create such labels, data mining strategies are applied for each scenario. A feature vector  $\mathbf{y}$  is extracted for each scenario as

$$\mathbf{y} = [y_{inLeft}, y_{inRight}, y_{inStr}, y_{inChng}, y_{straight}]^T, \quad (4.38)$$

with

- $y_{\text{inLeft}}$ : if the EGO vehicle has taken a left turn in an intersection over the complete scenario
- $y_{\text{inRight}}$ : if the EGO vehicle has taken a right turn in an intersection over the complete scenario
- $y_{\text{inStr}}$ : if the EGO vehicle is going straight in an intersection over the complete scenario
- $y_{\text{inChng}}$ : if the EGO vehicle has done a lane change over the complete scenario
- $y_{\text{straight}}$ : if the EGO vehicle is staying in the same lane over the complete scenario.

Using  $\mathbf{y}$ , the dataset  $\mathcal{D}_1 = \{(\mathcal{S}_1, \mathbf{y}_1), \dots, (\mathcal{S}_M, \mathbf{y}_M)\}$  is formulated and used for testing downstream tasks. The number of classes in  $\mathcal{D}_1$  can be calculated by  $\text{unique}(\mathbf{y}_1, \dots, \mathbf{y}_M)$  leading to 26 classes.

#### 4.3.2.2 Implementation Details

The sequence of occupancy grids  $\mathbf{G}_i$  contains occupancy grids of size  $120 \times 120$  pixels with each pixel corresponding to one meter. The EGO vehicle at time  $t_0$  starts in the pixel position (40, 60). The grids are generated in an EGO-fixed manner. From the total 5s of the scenario 4 grids are uniformly sampled. Hence, dimension of  $\mathbf{G}_i \in \mathbb{R}^{4 \times 120 \times 120}$ . The backbone encoder  $f$  for the self-supervised pre-training is a 3D ResNet-18 [HZRS16], which projects  $f : \mathbf{G}_i \mapsto \mathbf{z}_i$ , where  $\mathbf{z}_i \in \mathbb{R}^{512}$ . The projection network  $g$  is a MLP which does the mapping  $g : \mathbf{z}_i \mapsto \mathbf{h}_i$ , where  $\mathbf{h}_i \in \mathbb{R}^{2048}$ . The networks are trained with both Barlow Twins and VICReg objectives with the augmentation configurations as discussed in the following section. For training, Adam optimizer is used with  $lr = 1e - 4$  and a batch size of 200 for Barlow Twins and a batch size of 480 for VICReg is used for all experiments.

#### 4.3.2.3 Baselines

To compare the representations learned with the domain knowledge-guided augmentations the following baselines are used:

**Random Initialization (Rand. Init.)** The backbone network is initialized randomly with different random seeds.

**Base Augmentations (BaseAgt)** Typical augmentations from CV tasks are used. The grids are augmented by randomly cropping them to  $80 \times 80 \times 4$ , randomly rotating with 2D rotations between  $(-10^\circ, 10^\circ)$ , followed by adding random noise and applying Gaussian blur. The crop size is kept big, such that most of the important scenario information is maintained.

**Domain + Base Augmentations (ExAgt)** The augmentations introduced in this work are combined with the standard augmentations from *BaseAgt*. In one of the views, the  $a_{\text{con}}$  is applied with a probability of 0.7 and the  $a_{\text{VR}}$  is applied with a probability of 0.3. Hence, if both augmentations are used, they are merged as defined in  $a_{\cap}$ . For the other view, the probabilities of  $a_{\text{con}}$  and  $a_{\text{VR}}$  are swapped. The VR is randomly sampled from ( $d_{\text{min}} = 20\text{M}$ ,  $d_{\text{max}} = 100\text{m}$ ) and ( $\alpha_{\text{min}} = 60^\circ$ ,  $\alpha_{\text{max}} = 360^\circ$ ).

**Supervised Training (Sup)** : The labelled dataset is used to train the network  $f$  in a supervised manner.

#### 4.3.2.4 Metrics

The zero-shot clustering performance (question 1) is measured with unsupervised clustering accuracy [YXN<sup>+</sup>10] (ACC). The linear classifier and few-shot classification (questions 2 & 3) are supervised learning experiments, hence, supervised classification accuracy is used. For the representation stability experiment (question 4), the measure as in the previous sections is used. To determine the stability measure, for each data point  $K$ -nearest neighbors in the representation space are chosen. The average differences with respect to features like average velocity and average trajectory displacement from the data point to the  $K$ -nearest neighbors are calculated. The lower the difference is, the more similar the features within the neighborhood, which is an indicator for continuity and stability in the representation space.

#### 4.3.2.5 Experiments with Representations

In this section, experimental results for the clustering and the classification tasks are discussed. Each experiment is repeated five times with different seeds and the average values of the five experiments is presented.

#### 4.3.2.6 Zero-Shot Clustering

This experiment is used to analyze the structure in the representation space without any fine-tuning. The trained network  $f$  is used to extract features  $\{z_1, \dots, z_M\}$  from a validation dataset of size  $M$ . *Hierarchical Clustering* (HC) is performed on the extracted features due to the unbalanced nature of the classes in the dataset.

Table 4.5: Zero-shot clustering accuracy  $\uparrow$ .

Method	<i>Rand. Init.</i>	<i>BaseAgt</i>	<i>ExAgt</i>
Barlow Twins	0.29014	0.3833	<b>0.4588</b>
VICReg	0.29014	0.3294	<b>0.4142</b>

From Table 4.5 it can be seen that when using the domain augmentations along with standard CV augmentations the zero-shot clustering accuracy improves significantly. This shows that the domain knowledge guided augmentations are aiding in learning better representations when compared to using only standard CV augmentations. Therefore, in order to find representative traffic scenarios for scenario-based testing, self-supervised learning in combination with domain knowledge guided augmentations should be preferred over standard CV augmentations.

**Linear Classifier Evaluation** The linear classifier experiment is used to evaluate how linearly separable the traffic scenarios projected by  $f$  are in the representation space. For analyzing this, a linear FC layer is attached to the trained  $f$  and fine-tuned using the complete labelled dataset. During fine-tuning the weights of the network  $f$  are fixed and only the linear FC layer is trained.

Table 4.6: Linear classifier evaluation, classification accuracy  $\uparrow$ .

Method	<i>BaseAgt</i>	<i>ExAgt</i>
Barlow Twins	0.4718	<b>0.4754</b>
VICReg	0.4345	<b>0.4426</b>

The linear classifier evaluation from Table 4.6 shows that ExAgt leads to a comparable performance like standard CV augmentations.

**Few-Shot Classification** The objective of the few-shot classification experiment is to test the performance of the encoder for use in downstream tasks like supervised classification. Subsets of 1% and 10% of the dataset  $\mathcal{D}_l$  with the same class distribution as the validation dataset of Argoverse is selected. The pre-trained self-supervised network is fine-tuned with the 1% and 10% labelled dataset and the classification performance is reported in Table 4.7. In both 1% and 10% settings, the fine-tuned networks which are initialized with pre-trained networks using ExAgt are able to improve the few-shot classification performance compared to BaseAgt. Also, the fine-tuned networks pre-trained using both BaseAgt and ExAgt are able to outperform a supervised network trained from scratch, when using Barlow Twins. This shows that a self-supervised pre-training is essential in cases where labelled data is scarce.

**Representation Space Stability** The representation space stability is used to measure the local neighborhood relations of the data in the representation space, like in the previous section (Section 4.2). To investigate the representation stability, for each data point,  $K$  neighbors in the representation space are considered. The average difference with respect to a selected feature e.g., average velocity and trajectory displacement, between the selected data point to all the  $K$  neighbors is calculated. The average difference for

Table 4.7: Few-shot classification, classification accuracy (higher the better). Bold: best overall, underlined: best augmentation.

Method	% labelled	<i>BaseAgt</i>	<i>ExAgt</i>	<i>Sup</i>
Barlow Twins	1%	0.4653	<b><u>0.4827</u></b>	0.4131
	10%	0.5145	<b><u>0.5453</u></b>	0.4624
VICReg	1%	0.3844	<u>0.3877</u>	<b>0.4131</b>
	10%	0.4439	<u>0.4448</u>	<b>0.4624</b>

each data point is again averaged over the complete dataset to get the feature stability measure  $\Delta_{\dots}$ . Here, only the Barlow Twins method is used for the experiments.

In Figure 4.20,  $\Delta_{\text{velocity}}$ , the average velocity difference,  $\Delta_{\text{traj}_{xy}}$ , the average trajectory displacement,  $\Delta_{\text{mapImage}}$ , the average difference between the bird’s-eye view of the infrastructure image are reported across different  $K$  values. With respect to every  $K$  and every feature (average velocity, trajectory displacement, and map image difference) the stability measure using ExAgt is better compared to BaseAgt. Hence, the network trained with ExAgt has more stable local neighborhood relations.

#### 4.3.2.7 Ablation Study

**Augmentation Study** This study aims to understand and analyze the effect of different augmentations for representation learning of traffic scenarios. For this the following augmentations, besides BaseAgt and ExAgt are considered:

- 40-Crop: All augmentations from BaseAgt with the random crop sized reduced to  $40 \times 40 \times 4$ ,
- BaseAgt+ $a_{\text{VR}}$ : All augmentations from BaseAgt plus the sensor-based augmentation  $a_{\text{VR}}$ ,
- BaseAgt+ $a_{\text{con}}$ : All augmentations from BaseAgt plus the connectivity-based augmentation  $a_{\text{con}}$ .

All these experiments are compared against ExAgt with zero-shot clustering performance. The reduction of zero-shot clustering accuracy from ExAgt, when applied in the Barlow Twins method is shown in Figure 4.21. The reduction when using a reduced crop size of 40 compared to any other combination, underlines the intuition as shown in Figure 4.16. Hence, using small crop sizes with traffic scenarios should be avoided. Another conclusion that can be drawn from Figure 4.21, is that the connectivity-based augmentation  $a_{\text{con}}$  is important for the representation learning. Adding the sensor-based augmentation  $a_{\text{VR}}$  leads to an even better performance, as can be seen from the value of ExAgt.



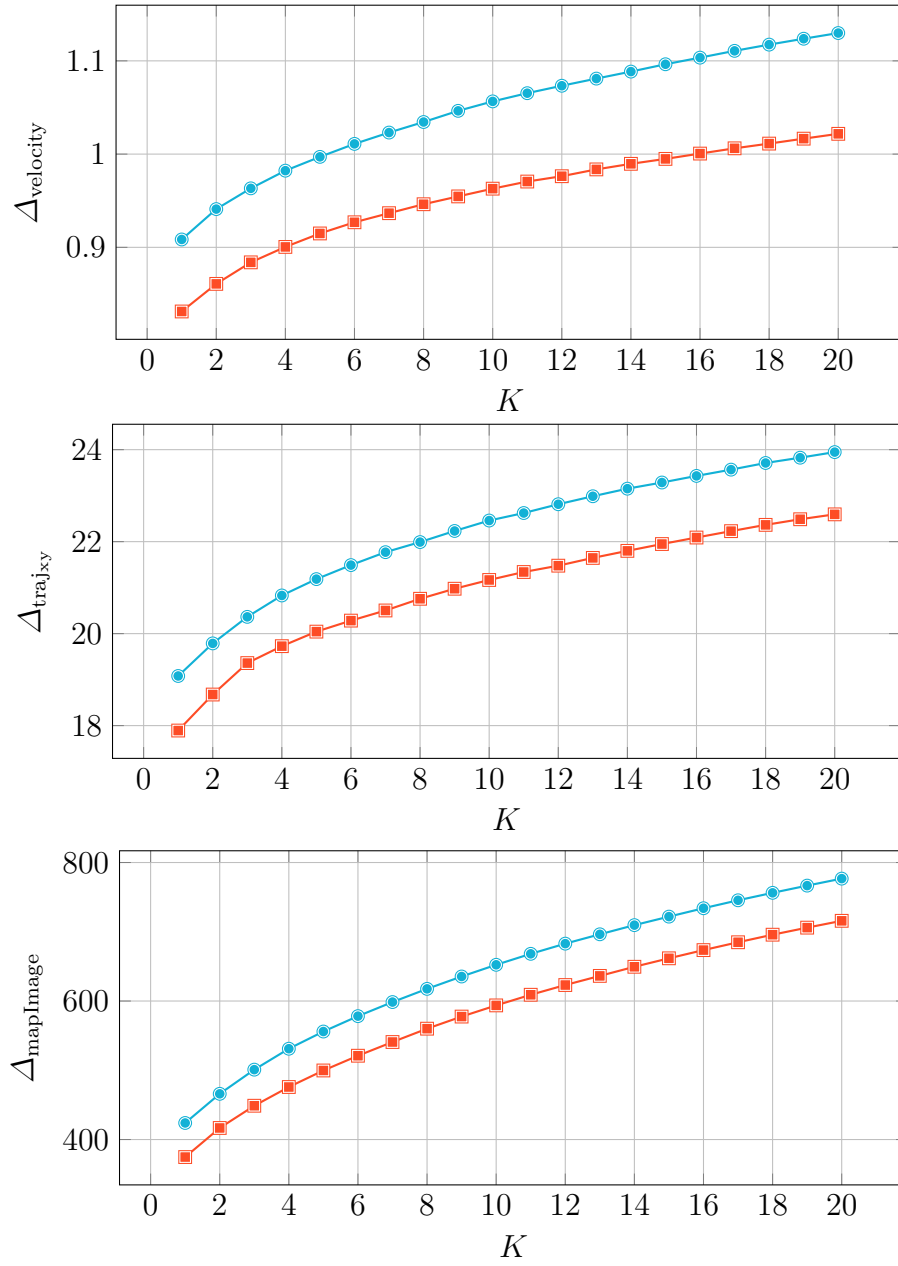


Figure 4.20: Representation space stability with respect to  $\Delta_{\text{velocity}}$ ,  $\Delta_{\text{traj}_{xy}}$ ,  $\Delta_{\text{mapImage}}$  across different  $K$ , when using Barlow Twins. *ExAgt* — and *BaseAgt* —.

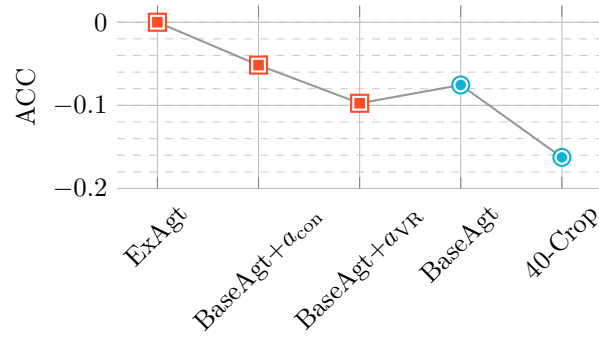


Figure 4.21: Reduction in unsupervised clustering accuracy from ExAgt with different augmentations. —: with domain knowledge; —: without domain knowledge.

**VR Parameter Study** The sensor-based augmentation  $a_{VR}$  is parameterizable. Here, the impact of various VR settings is investigated. For this, unsupervised clustering performance in Barlow Twins is analyzed. The default parameters used in all the experiments are ( $d_{\min} = 20$  m,  $d_{\max} = 100$  m) and ( $\alpha_{\min} = 60^\circ$ ,  $\alpha_{\max} = 360^\circ$ ). In this experiment, the parameters  $d_{\max}$  and  $\alpha_{\max}$  are varied.

Table 4.8: VR parameter study (clustering accuracy ACC  $\uparrow$ ), when using Barlow Twins with ExAgt

Range ( $d_{\min} - d_{\max}$ )	FoV ( $\alpha_{\min} - \alpha_{\max}$ )	ACC
20 – 100	60 – 360	<b>0.4588</b>
20 – 50	60 – 360	0.4069
20 – 100	60 – 120	0.3948
20 – 50	60 – 120	0.3561

In Table 4.8, the clustering performance for the various settings are shown. The setting used as default in this work shows the best overall performance. Also, it can be seen, that both, the FoV and the range are important for the performance. Hence, randomly sampling from a large VR is favorable for representation learning of traffic scenarios.

### 4.3.3 Conclusion

In this section, ExAgt a novel approach to augment traffic scenarios with domain knowledge is presented. Augmentation is a key-component for self-supervised learning methods. The augmentation strategy ExAgt is used in self-supervised learning methods for learning representations of traffic scenarios. The representations learned with ExAgt are compared with the representations learned with standard CV augmentations.

Experiments show that using ExAgt is improving the performance in most of the downstream tasks and leads to better representation space stability. Hence, domain specific augmentations are important for traffic scenario representation learning.

### 4.3. Domain Knowledge guided Augmentations for Self-Supervised Learning

---

In future work, evaluating the pre-trained encoder for tasks like outlier detection and open-set recognition can be explored.



# Chapter 5

## Conclusion and Outlook

Proofing that an AV performs safer than the average human driver through simple statistical test approaches is infeasible, since the AV would need to drive more than 8 billion kilometers (see Section 3.1). Therefore, other testing strategies are required for the validation of AVs. One commonly used concept is the so-called scenario-based validation approach. Scenario-based validation relies on the assumption that it is sufficient to test an AV in only a few relevant scenarios to prove its safety. Such relevant scenarios are typically either created through expert definition (e. g., NCAP) or by identifying them from real-world data. This work presents methods which address the identification of relevant traffic scenarios by specifically detecting representative, unknown, and critical traffic scenarios. The overall framework of this work and its embedding into the validation process of AVs is summarized in Figure 5.1.

Recently, various work aim to solve the tasks to identify representative and unknown scenarios. Typically, clustering is used to find representative scenarios and novelty detection is used to find unknown scenarios. A comprehensive summary of recent approaches is

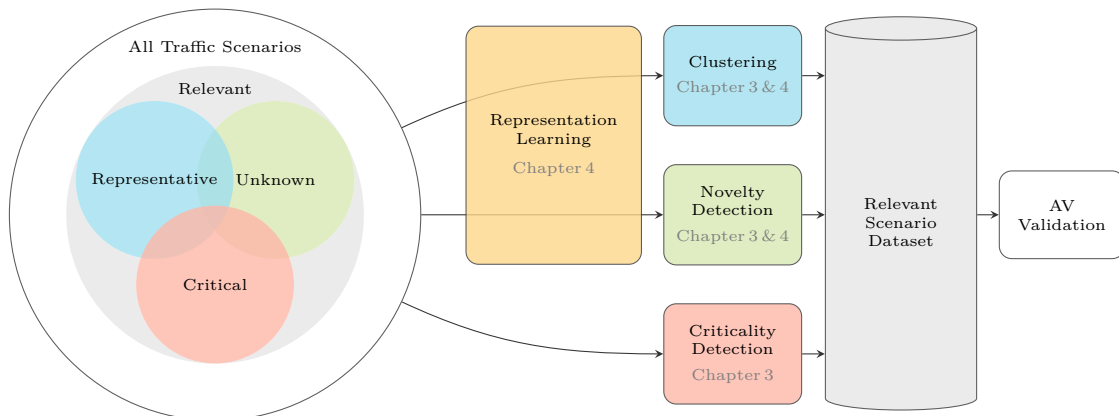



Figure 5.1: Methodological overview of this work and its embedding in the validation process of AVs. : Representation Learning component is only used in Chapter 4.

provided in Section 3.2. Most of the recent methods mainly deal with dynamic information in the form of trajectory data, using well-designed similarity measures alongside projections mechanisms to solve the specific tasks. However, only few incorporate comprehensive infrastructure information. When using both, dynamic and infrastructural information, standard clustering and novelty detection methods struggle to deliver desirable results. Hence, work dealing with combined information typically project the scenario into a representation more suited for clustering and novelty detection. In this work, methods are presented which realize the projection of traffic scenario by the means of representation learning. It is focused on the inclusion of domain knowledge into the representation learning to further strengthen the expressiveness of the resulting representations.

In Section 3.3, a neighborhood-based novelty detection method is presented which is applied on road infrastructure images, in order to detect unknown infrastructures. The method shows high stability with respect to the selected number of neighbors, what is desirable, since in novelty detection the optimal number of neighbors is typically not known. Applied on standard outlier detection benchmark datasets, the novel method shows good performance, comparable to other state of the art neighborhood-based outlier detection methods, while the performance being more robust against the number of neighbors. However, as most vector-based methods, the introduced method struggles with image data.

Other outlier detection approaches are applied to the problem of identifying unknown infrastructure images in Section 4.2.1.5. While, the vector-based approaches do not achieve reasonable results, the outlier detection mechanisms specialized for image data show better performance (see Section 4.2.1.5). However, the established image outlier detection methods do not suit the task of traffic scenario clustering. Therefore, in this work it is focused on transforming the traffic scenario into representations, such that simple and established vector-based outlier detection methods can be used and such that clustering can be applied on the resulting representation.

Approaching the detection of representative and critical scenarios through domain knowledge-based methods, without relying on data-driven concepts, is presented in Section 3.5. Detecting critical scenarios is realized in various ways, from using dynamic information of the EGO vehicle only, up to using all surrounding objects and infrastructure information. Detecting representative scenarios is realized through automatic domain knowledge-based categorization. For this a traffic scenario is transformed into a graph representation, which can be used to easily compare traffic scenarios. The methods using only domain knowledge, can be seen as an additional fallback path to the presented representation learning-based approaches. This way, the risk of overlooking critical scenarios and also unknown scenarios is reduced. However, the domain knowledge-only methods should not be used stand-alone, since they fail to express the complete scenario.

Besides the before mentioned proposed methods and explored approaches, the main contribution of this work are the methods using domain knowledge guided representation

---

learning to solve the task of projecting traffic scenarios into representations suited for clustering and novelty detection of traffic scenarios. In each of the three presented approaches in Chapter 4, the utilization of domain knowledge leads to representations where clustering and novelty detection can be performed more reliable. In the following, each approach is briefly summarized, as well as the respective advantages and disadvantages are discussed. Also, possible further research directions emerging from the approaches are highlighted.

A triplet autoencoder for the projection of infrastructure images is presented in Section 4.1. It utilizes domain knowledge by the way the similarity between infrastructure images is determined. For this, the infrastructure is represented through connectivity graphs, where two infrastructures are considered to be more similar if they share the same graph as if they have a different graph. This way, a representation space is shaped in such a way, that novelty detection can be performed with simple outlier detection methods. As shown, the detection of novel infrastructures with the presented method outperforms other, vector- and image-based novelty detection approaches. Also, the importance of including domain knowledge can be seen when comparing to domain knowledge-free representation learning approaches such as an autoencoder. The main drawback of the presented method, is that it only considers the infrastructure and hence only the static part of a traffic scenario. The method aids as a proof of concept, therefore it is applied on infrastructure data generated from OSM, its application to more detailed maps is yet an open topic. Care needs to be taken about the amount of training data, since to achieve valuable representations even for very uncommon infrastructures, the training data should be very comprehensive.

The triplet autoencoder is extended to the quadruplet autoencoder as shown in Section 4.2. Besides the infrastructure, also the EGO dynamics are used to represent a scenario. The graph-based similarity is extended to include the definitions of routes, such that the domain knowledge similarity objectives can be realized as shown in Section 4.2. The resulting projection leads to expressive representations, which can be used for clustering, novelty detection, and further analysis. As the results highlight, the proposed method leads to the overall best performance in clustering traffic scenarios, detecting novel scenario types, and in showing high feature stability among neighbors. Using only the EGO information is the main drawback of the presented method. The EGO dynamics are considered to be sufficient to represent a scenario in this work, since the trajectory of the EGO already contains its reaction to other traffic participants and the road infrastructure. Obviously, that argumentation only holds for a high level overall system validation concept. If one is interested in validating for example the perception, it might indeed be of interest to detect uncommon constellations and setting of objects surrounding the EGO vehicle. For this purpose, the method as presented in Section 4.2 could be extended to include also the surrounding objects' trajectories. The graph-based similarity measure is able to

handle multiple objects as discussed in Section 3.5. Such a training regime would require a huge amount of data, if one wants to provide multiple similar instances per sample. The presented quadruplet autoencoder method is only evaluated in a proof of concept way, hence simulation data is used for the training and evaluation consisting of  $\approx 70\,000$  and  $\approx 60\,000$ , respectively. Applying the method to real-world data could be realized in a future application oriented work. The quality of the representations formed by the proposed method, mainly depends on the quality and amount of used data. Hence, the larger the dataset the better. However, determining the similarities between scenarios on huge datasets is computationally expensive, considering that the DTW distance between similar scenarios is determined. Determining the graph and route similarity can be realized efficiently, by comparing high level properties of the graphs and routes first, as presented in [FFWSM<sup>+</sup>22]. To reduce the computational effort for the DTW calculation, one could sub-sample the dataset in a balanced fashion, using the graph and route similarity.

Section 4.3 introduced domain knowledge guided augmentations for traffic scenarios to be used in self-supervised learning frameworks. Two different augmentation types are presented, where both manipulate the scenarios' information by dropping information in a domain knowledge guided way. The narrative is that scenarios are still comparable even if less important information is missing. The first augmentation is based on connectivity constraints, which is changing the map and the object list by ignoring non-connected elements from both. The second augmentation is based on the *Visible Region* (VR), hence only the object list is modified while the map stays unchanged. Like before, the aim is to form representations where clustering and novelty detection can be applied. In contrast to the computationally expensive similarity determination of the previous approach, the augmentation can be generated on the fly. Compared to standard CV augmentation, the performance for various downstream tasks improves when using the domain knowledge guided augmentations. However, considering the results from Section 4.2, the self-supervised setting does not yield such powerful representations as the approach where the similarity of traffic scenarios is defined through their graphs and routes. Combining both strategies could be beneficial, and could be further explored in future research.

In summary, various approaches have been presented and explored to apply clustering and novelty detection on traffic scenarios. The presented methods, showed promising performance increase in both tasks, aiding the validation process of AVs.



# List of Acronyms

ABOD	Angle-Based Outlier Detection
ACC	ACCuracy of a clustering
AUC	Area Under Curve
AV	Autonomous Vehicle
BYOL	Bootstrap Your Own Latent
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CV	Computer Vision
DTW	Dynamic Time Warping
EGO	EGO vehicle is the vehicle under test, or the vehicle of focus
ExAgt	Expert-guided Augmentation for representation learning of traffic scenarios
f-AnoGAN	fast Anomaly Detection with Generative Adversarial Network
FC	Fully-Connected
FoV	Field of View
GAN	Generative Adversarial Network
HC	Hierarchical Clustering
IF	Isolation Forest
ISOS	Intrinsic Stochastic Outlier Selection
KNNSOS	Nearest Neighbor Stochastic Outlier Selection
LDOF	Local Distance-based Outlier Factor
LEF	Local Entropy Factor
LOF	Local Outlier Factor

LSTM	Long Short-Term Memory
MLP	MultiLayer Perceptron
NLP	Natural Language Processing
OBJ	OBJ vehicle is in the surrounding of the EGO vehicle
OCSVM	One-Class Support Vector Machine
ODIN	Outlier Detection using Indegree Number
OpenDRIVE	Open Dynamic Road Information for Vehicle Environment
OSM	OpenStreetMap
PCA	Principal Component Analysis
PDF	Probability Density Function
PMF	Probability Mass Function
RaPP	Reconstruction along Projection Pathway
ReLU	Rectified Linear Unit
ResNet	Deep Residual Network
RF	Risk Feeling
RNN	Recurrent Neural Network
ROD	Reconstruction-based Outlier Detection
RSS	Responsibility Sensitive Safety
SAP	Simple Aggregation along Pathway
SimCLR	Simple Contrastive Learning of visual Representations
SimSiam	Simple Siamese representation learning
SOS	Stochastic Outlier Selection
SUMO	Simulation of Urban Mobility
SVM	Support Vector Machine
SwAV	Swapping Assignments between Views
t-SNE	t-distributed Stochastic Neighbor Embedding
THW	Time HeadWay
tLEF	t-SNE graph-based non-sparse Local Entropy Factor
tSLEF	t-SNE graph-based Sparse Local Entropy Factor
TTC	Time To Collision
ULEF	UMAP graph-based non-sparse Local Entropy Factor
UMAP	Uniform Manifold Approximation and Projection

USLEF	UMAP graph-based Sparse Local Entropy Factor
USOS	UMAP graph-based KNNSOS
VICReg	Variance, Invariance, Covariance Regularization
ViT	Vision Transformer
VR	Visible Region



# Bibliography

- [AC15] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.
- [AP02] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer Berlin Heidelberg, 2002.
- [Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2006.
- [BKBD21] Lakshman Balasubramanian, Friedrich Kruber, Michael Botsch, and Ke Deng. Open-set recognition based on the combination of deep learning and ensemble method for detecting unknown traffic scenarios. In *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jul 2021.
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016.
- [BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD 00*. ACM Press, 2000.
- [BMR<sup>+</sup>20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. GPT-3.
- [BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

- [BPL21] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. May 2021.
- [BSS21] Johannes Bernhard, Mark Schutera, and Eric Sax. Optimizing test-set diversity: Trajectory clustering for scenario-based testing of automated driving systems. pages 1371–1378, Indianapolis, IN, USA, 2021. IEEE.
- [Bun18] Statistisches Bundesamt. Verkehrsunfälle 2017. *Verkehr, Fachserie 8 Reihe 7*, 2018.
- [BWBD21] Lakshman Balasubramanian, Jonas Wurst, Michael Botsch, and Ke Deng. Traffic scenario clustering by iterative optimisation of self-supervised networks using a random forest activation pattern similarity. In *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jul 2021.
- [BWE<sup>+</sup>22] Lakshman Balasubramanian, Jonas Wurst, Robin Egolf, Michael Botsch, Wolfgang Utschick, and Ke Deng. Exagt: Expert-guided augmentation for representation learning of traffic scenarios. *2022 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022.
- [BWE<sup>+</sup>23] Lakshman Balasubramanian, Jonas Wurst, Robin Egolf, Michael Botsch, Wolfgang Utschick, and Ke Deng. Scenediffusion: Conditioned latent diffusion models for traffic scene prediction. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3914–3921, 2023.
- [CH20] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. November 2020.
- [CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [CLD<sup>+</sup>21] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2021.
- [CLS<sup>+</sup>19] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

- 
- [CMM<sup>+</sup>20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020.
- [CMS<sup>+</sup>20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.
- [CPS<sup>+</sup>21] Long Chen, Lukas Platinsky, Stefanie Speichert, Błażej Osiński, Oliver Scheel, Yawei Ye, Hugo Grimmett, Luca Del Pero, and Peter Ondruska. What data do we need for training an av motion planner? In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1066–1072. IEEE, 2021.
- [Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [CZS<sup>+</sup>16a] Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, jan 2016.
- [CZS<sup>+</sup>16b] Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. Supplementary Material for On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. <http://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/>, 2016. Accessed: 2019-06-27.
- [D<sup>+</sup>15] Marius Dupuis et al. *OpenDRIVE – Format Specification Rev1.4*, 1.4 edition, 2015.
- [DARC20] Andreas Demetriou, Henrik Alfvåg, Sadegh Rahrovani, and Morteza Haghiri Chehreghani. A deep learning framework for generation and analysis of driving scenario trajectories. *cs.CV*, 2020.
- [DBK<sup>+</sup>21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [DDS<sup>+</sup>09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [DKN<sup>+</sup>06] T. A. Dingus, S.G. Klauer, V. L. Neale, A. Petersen, S. E. Lee, J. Sudweeks, M. A. Perez, J. Hankey, D. Ramsey, S. Gupta, C. Bucher, Z. R. Doerzaph, J. Jermeland, and R.R. Knipling. The 100-car naturalistic driving study, phase ii – results of the 100-car field experiment. resreport DOT HS 810 593, National Highway Traffic Safety Administration, 2006.
- [Dun22] Max Dunger. Entwicklung, evaluierung und implementierung vontrajektorienvergleichsmethoden zur Ähnlichkeitsbestimmung vonverkehrsszenarien. mathesis, HTWG Konstanz, March 2022. Unpublished.
- [DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. 2016.
- [DVR<sup>+</sup>19] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Image anomaly detection with generative adversarial networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 3–17. Springer International Publishing, 2019.
- [EKS<sup>+</sup>96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
- [FBBA18] Laura Fraade-Blanar, Marjory S. Blumenthal, and James M. Anderson. *Measuring Automated Vehicle Safety: Forging a Framework*. RAND CORP, 2018.
- [FFWSM<sup>+</sup>22] Alberto Flores Fernández, Jonas Wurst, Eduardo Sánchez Morales, Michael Botsch, Christian Facchi, and Andrés García Higuera. Probabilistic traffic motion labeling for multi-modal vehicle route prediction. *Sensors*, 22(12), 2022.
- [FHK<sup>+</sup>16] Ludwig Fahrmeir, Christian Heumann, Rita Künstler, Iris Pigeot, and Gerhard Tutz. *Statistik*. Springer Berlin Heidelberg, 2016.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.



- 
- [GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
- [GSA<sup>+</sup>20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent—a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.
- [HBG20] Nick Harmening, Marin Biloš, and Stephan Günnemann. Deep representation learning and clustering of traffic scenarios, 2020.
- [HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.
- [HFW<sup>+</sup>20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.
- [HGSP20] Florian Hauer, Ilias Gerostathopoulos, Tabea Schmidt, and Alexander Pretschner. Clustering traffic scenarios using mental models as little as possible. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- [HKF04] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. IEEE, 2004.
- [HKWS19] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019.
- [HMKS19] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in Neural Information Processing Systems*, 32, 2019.
- [HRC21] Fazeleh Hoseini, Sadegh Rahrovani, and Morteza Haghiri Chehreghani. Vehicle motion trajectories clustering via embedding transitive relations. pages 1314–1321, Indianapolis, IN, USA, 2021. IEEE.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.
- [ISO21] Iso 21448 - road vehicles — safety of the intended functionality (sotif), 2021.
- [Jan12] Jeroen Janssens. scikit-sos. <https://github.com/jeroenjanssens/scikit-sos>, 2012.
- [JHPvdH12] Jeroen Janssens, Ferenc Huszár, Eric Postma, and Jaap van den Herik. Stochastic outlier selection. techreport 2012-001, Tilburg centre for Creative Computing, February 2012.
- [JWKW18] P. Junietz, W. Wachenfeld, K. Klonecki, and H. Winner. Evaluation of different approaches to address safety validation of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 491–496, Nov 2018.
- [JWS<sup>+</sup>19] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3967–3974, jul 2019.
- [KBG<sup>+</sup>97] Andre Kleyner, Shrikar Bhagath, Mauro Gasparini, Jeffrey Robinson, and Mark Bender. Bayesian techniques to reduce the sample size in automotive electronics attribute testing. *Microelectronics and reliability*, 37:879–884, 1997.
- [KhZ08] Hans-Peter Kriegel, Matthias S hubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. ACM Press, 2008.
- [KKSZ09] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. LoOP: local outlier probabilities. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*. ACM Press, 2009.
- [KME<sup>+</sup>22] Friedrich Kruber, Eduardo Sánchez Morales, Robin Egolf, Jonas Wurst, Samarjit Chakraborty, and Michael Botsch. Micro- and macroscopic road traffic analysis using drone image data. *Leibniz Transactions on Embedded Systems*, 8(1), 2022.

- 
- [KP16] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182 – 193, 2016.
- [KR87] Leonard Kaufmann and Peter Rousseeuw. Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416, 01 1987.
- [Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [KSL<sup>+</sup>20] Ki Hyun Kim, Sangwoo Shim, Yongsub Lim, Jongseob Jeon, Jeongwoo Choi, Byungchan Kim, and Andre S. Yoon. Rapp: Novelty detection with reconstruction along projection pathway. In *International Conference on Learning Representations*, 2020.
- [KVPF20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.
- [KWB18] F. Kruber, J. Wurst, and M. Botsch. An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.
- [KWBC23] Friedrich Kruber, Jonas Wurst, Michael Botsch, and Samarjit Chakraborty. Unsupervised random forest learning for traffic scenario categorization. In *TBA*. TBA, 2023.
- [KWG<sup>+</sup>20] Jonas Kerber, Sebastian Wagner, Korbinian Groh, Dominik Notz, Thomas Kühbeck, Daniel Watzenig, and Alois Knoll. Clustering of the scenario space for the assessment of automated driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- [KWM<sup>+</sup>19] F. Kruber, J. Wurst, E. S. Morales, S. Chakraborty, and M. Botsch. Unsupervised and supervised learning with the random forest algorithm for traffic scenario clustering and classification. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2463–2470, 2019.
- [KYK<sup>+</sup>08] Takayuki Kondoh, Tomohiro Yamamura, Satoshi Kitazaki, Nobuyuki Kuge, and Erwin Roeland Boer. Identification of visual cues and quantification of drivers’ perception of proximity risk to the lead vehicle in car-following situations. *Journal of Mechanical Systems for Transportation and Logistics*, 1(2):170–180, 2008.

- [LBBW<sup>+</sup>18] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.
- [LBCP<sup>+</sup>20] Hussam Lawen, Avi Ben-Cohen, Matan Protter, Itamar Friedman, and Lihi Zelnik-Manor. *Compact Network Training for Person ReID*, page 164–171. Association for Computing Machinery, New York, NY, USA, 2020.
- [LBD<sup>+</sup>90] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.
- [LBR<sup>+</sup>18] Jacob Langner, Johannes Bach, Lennart Ries, Stefan Otten, Marc Holzapfel, and Eric Sax. Estimating the uniqueness of test scenarios derived from recorded real-world-driving-data using autoencoders. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2018.
- [LGO<sup>+</sup>19] Jacob Langner, Hannes Grolig, Stefan Otten, Marc Holzapfel, and Eric Sax. Logical scenario derivation by clustering dynamic-length-segments extracted from real-world-driving-data. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*. SCITEPRESS - Science and Technology Publications, 2019.
- [LLP07] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer Berlin Heidelberg, 2007.
- [LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, dec 2008.
- [LZH<sup>+</sup>21] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.
- [Mö7] Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.
- [MB20] Wolfgang Utschick Michael Botsch. *Fahrzeugsicherheit und automatisiertes Fahren*. Hanser Fachbuchverlag, 2020.

- 
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [MHM18] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, February 2018. arXiv: 1802.03426.
- [MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.
- [MMCS11] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.
- [MRS<sup>+</sup>18] Marc Masana, Idoia Ruiz, Joan Serrat, Joost van de Weijer, and Antonio M Lopez. Metric learning for novelty and anomaly detection. In *British Machine Vision Conference (BMVC)*, 2018.
- [OK12] Patrick O’Connor and Andre Kleyner. *Practical reliability engineering*. John Wiley & Sons, 2012.
- [OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.
- [Ope19] OpenStreetMap contributors. Data from 25th october 2019 via Overpass API. <https://www.openstreetmap.org>, 2019.
- [Ope20] OpenStreetMap contributors. Data from october 2020 via Overpass API. <https://www.openstreetmap.org>, 2020.
- [Ope21] OpenStreetMap contributors. Data from March 2021 via Geofabrik. <https://www.openstreetmap.org>, 2021.
- [PEG] Research project pegasus. <https://www.pegasusprojekt.de/en/>. Accessed: 2020-03-31.
- [PPJ<sup>+</sup>18] Fabian Poggenhans, Jan-Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, November 2018.

- [RPG<sup>+</sup>21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.
- [RPL<sup>+</sup>20] Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access*, 8:87456–87477, 2020.
- [RRB<sup>+</sup>21] Lennart Ries, Philipp Rigoll, Thilo Braun, Thomas Schulik, Johannes Daube, and Eric Sax. Trajectory-based clustering of real-world urban driving sequences with multiple traffic objects. pages 1251–1258, Indianapolis, IN, USA, 2021. IEEE.
- [RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, 29(2):427–438, jun 2000.
- [RWC<sup>+</sup>19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [RWM21] Vignesh Ramanathan, Rui Wang, and Dhruv Mahajan. Predet: Large-scale weakly supervised pre-training for detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2865–2875, October 2021.
- [SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
- [SG17] Erich Schubert and Michael Gertz. Intrinsic t-stochastic neighbor embedding for visualization and outlier detection. In *Similarity Search and Applications*, pages 188–203. Springer International Publishing, 2017.
- [SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.
- [Soh16] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [SSS17] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017.

- 
- [SSW<sup>+</sup>17] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Lecture Notes in Computer Science*, pages 146–157. Springer International Publishing, 2017.
- [SSW<sup>+</sup>19] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 54:30–44, 2019.
- [SWS<sup>+</sup>00] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.
- [SY14] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA’14, page 4–11, New York, NY, USA, 2014. Association for Computing Machinery.
- [SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [SZK12] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, 28(1):190–237, dec 2012.
- [TCcFC02] Jian Tang, Zhixiang Chen, Ada Wai chee Fu, and David W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining*, pages 535–548. Springer Berlin Heidelberg, 2002.
- [TDBM20] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020.
- [TDR<sup>+</sup>20] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020.
- [TKI20] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multi-view coding. In *European conference on computer vision*, pages 776–794. Springer, 2020.

- [TLZM16] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web - WWW 16*. ACM Press, 2016.
- [vdM14] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014.
- [vL07] Ulrike von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.
- [VSP<sup>+</sup>17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.
- [WBBU21] Jonas Wurst, Lakshman Balasubramanian, Michael Botsch, and Wolfgang Utschick. Novelty detection and analysis of traffic scenario infrastructures in the latent space of a vision transformer-based triplet autoencoder. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.
- [WBBU22] Jonas Wurst, Lakshman Balasubramanian, Michael Botsch, and Wolfgang Utschick. Expert-lasts: Expert-knowledge guided latent space for traffic scenarios. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.
- [WFBU20] Jonas Wurst, Alberto Flores Fernández, Michael Botsch, and Wolfgang Utschick. An entropy based outlier score and its application to novelty detection for road infrastructure images. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- [WLK<sup>+</sup>20] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.
- [WRZ<sup>+</sup>20] Wenshuo Wang, Aditya Ramesh, Jiacheng Zhu, Jie Li, and Ding Zhao. Clustering driving encounter scenarios using connected vehicle trajectories. *IEEE Transactions on Intelligent Vehicles*, pages 1–1, 2020.
- [WSK<sup>+</sup>22] Julian Wiederer, Julian Schmidt, Ulrich Kressel, Klaus Dietmayer, and Vasileios Belagiannis. A benchmark for unsupervised anomaly detection in multi-agent trajectories. In *2022 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022.
- [Wur18] Jonas Wurst. Unsupervised learning of image features based on relational clustering. mathesis, Technische Hochschule Ingolstadt, 2018.



- 
- [WXYL18] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018.
- [XW05] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.
- [XZC<sup>+</sup>21] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021.
- [YCS19] Yao Yang, Haoran Chen, and Junming Shao. Triplet enhanced AutoEncoder: Model-free discriminative network embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.
- [YXN<sup>+</sup>10] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, oct 2010.
- [ZFYZ21] Jinxin Zhao, Jin Fang, Zhixian Ye, and Liangjun Zhang. Large scale autonomous driving scenarios clustering with self-supervised feature extraction. March 2021.
- [ZGL<sup>+</sup>20] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845, 2020.
- [ZHJ09] Ke Zhang, Marcus Hutter, and Huidong Jin. A new local distance-based outlier detection approach for scattered real-world data. In *Advances in Knowledge Discovery and Data Mining*, pages 813–822. Springer Berlin Heidelberg, 2009.
- [ZJM<sup>+</sup>21] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. March 2021.
- [ZRF<sup>+</sup>18] H. Zenati, M. Romain, C. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *Proc. IEEE Int. Conf. Data Mining (ICDM)*, pages 727–736, November 2018.
- [ZZ22] Maximilian Zipfl and J. Marius Zöllner. Towards traffic scene description: The semantic scene graph. In *2022 25th International Conference on Intelligent Transportation Systems (ITSC)*, November 2022.

- [ZZLZ16] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

# Appendix A

## Appendix

### A.1 Cross Entropy-based Contrastive Loss as Mutual Information Maximization

Under some constraints and assumptions, the contrastive loss  $\mathcal{L}_{ce}$  maximizes a lower bound on the mutual information [OLV18]. The cross entropy loss can be interpreted as a special case of the categorical cross entropy loss, for only the positive pair case. Hence, the cross entropy learning can be interpreted as an assignment problem, such that a sample  $\mathbf{x}$  should be assigned to its positive representative instead of assigning it to any other example. For this purpose, all possible pairs for one data point are arranged in a set, as  $\mathcal{P} = \{(\mathbf{x}, \mathbf{p}_0), (\mathbf{x}, \mathbf{p}_1), \dots, (\mathbf{x}, \mathbf{p}_K)\}$ . The first pair includes the positive representation  $\mathbf{p}_0$ , while all other pairs include the  $K$  negative examples. The cross entropy-based contrastive loss can be reformulated using the categorical cross entropy as [TKI20]

$$\mathcal{L}_{ce} = E_{\mathcal{P}|\mathbf{x}} \left\{ -\log \left( \frac{\exp\left(\frac{1}{\tau}s(\mathbf{x}, \mathbf{p})\right)}{\sum_{k=0}^K \exp\left(\frac{1}{\tau}s(\mathbf{x}, \mathbf{p}_k)\right)} \right) \right\} \quad (\text{A.1})$$

$$= E_{\mathcal{P}|\mathbf{x}} \{-\log(\hat{p}(\mathbf{p}|\mathbf{x}))\}. \quad (\text{A.2})$$

Assume that the sample  $\mathbf{x}$  and the only positive example  $\mathbf{p}_0$  are sampled from a joint distribution  $p(\mathbf{x}, \mathbf{p})$ , while all other examples are independent of  $\mathbf{x}$ , therefore  $p(\mathbf{x})p(\mathbf{p})$ . The optimal probability  $p(\text{pos} = 0|\mathcal{P})$  for the former loss, that the positive pair is indeed

in position 0, can then be defined as [TKI20]

$$p(\text{pos} = 0|\mathcal{P}) = \frac{p(\mathbf{x}, \mathbf{p}_0) \prod_{j \neq 0} p(\mathbf{x})p(\mathbf{p}_j)}{\sum_{k=0}^K p(\mathbf{x}, \mathbf{p}_k) \prod_{j \neq k} p(\mathbf{x})p(\mathbf{p}_j)} \quad (\text{A.3})$$

$$p(\text{pos} = 0|\mathcal{P}) = \frac{\frac{p(\mathbf{x}, \mathbf{p}_i)}{p(\mathbf{x})p(\mathbf{p}_i)}}{\sum_{k=0}^K \frac{p(\mathbf{x}, \mathbf{p}_k)}{p(\mathbf{x})p(\mathbf{p}_k)}}, \quad (\text{A.4})$$

which will be one for the positive example  $i = p$  while all other examples are independent of  $\mathbf{x}$ . While it is zero for all negative examples  $i \neq p$ .

As shown in the following, minimizing the loss maximizes a lower bound of the mutual information between the samples and the representations  $I(\mathbf{x}, \mathbf{p})$ . To show this, the risk of the generalized optimal cross entropy is considered:

$$R_{\text{opt}} = -\mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathcal{P}|\mathbf{x}} \left\{ \log \left[ \frac{\frac{p(\mathbf{x}, \mathbf{p}_0)}{p(\mathbf{x})p(\mathbf{p}_0)}}{\sum_{k=0}^K \frac{p(\mathbf{x}, \mathbf{p}_k)}{p(\mathbf{x})p(\mathbf{p}_k)}} \right] \right\} \right\} \quad (\text{A.5})$$

$$= -\mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathcal{P}|\mathbf{x}} \left\{ \log \left[ \frac{1}{1 + \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} \sum_{k \neq 0} \frac{p(\mathbf{x}, \mathbf{p}_k)}{p(\mathbf{x})p(\mathbf{p}_k)}} \right] \right\} \right\} \quad (\text{A.6})$$

$$= \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathcal{P}|\mathbf{x}} \left\{ \log \left[ 1 + \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} \sum_{k \neq 0} \frac{p(\mathbf{x}, \mathbf{p}_k)}{p(\mathbf{x})p(\mathbf{p}_k)} \right] \right\} \right\} \quad (\text{A.7})$$

$$\approx \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathcal{P}|\mathbf{x}} \left\{ \log \left[ 1 + \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} K \mathbb{E}_{\mathbf{p}} \left\{ \frac{p(\mathbf{x}, \mathbf{p})}{p(\mathbf{x})p(\mathbf{p})} \right\} \right] \right\} \right\} \quad (\text{A.8})$$

$$= \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathcal{P}|\mathbf{x}} \left\{ \log \left[ 1 + \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} K \mathbb{E}_{\mathbf{p}} \left\{ \frac{p(\mathbf{p}|\mathbf{x})}{p(\mathbf{p})} \right\} \right] \right\} \right\} \quad (\text{A.9})$$

$$= \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathbf{p}_0|\mathbf{x}} \left\{ \log \left[ 1 + \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} K \right] \right\} \right\} \quad (\text{A.10})$$

$$\geq \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathbf{p}_0|\mathbf{x}} \left\{ \log \left[ \frac{p(\mathbf{x})p(\mathbf{p}_0)}{p(\mathbf{x}, \mathbf{p}_0)} K \right] \right\} \right\} \quad (\text{A.11})$$

$$= \log(K) - \mathbb{E}_{\mathbf{x}} \left\{ \mathbb{E}_{\mathbf{p}_0|\mathbf{x}} \left\{ \log \left[ \frac{p(\mathbf{x}, \mathbf{p}_0)}{p(p(\mathbf{x})p(\mathbf{p}_0))} \right] \right\} \right\} \quad (\text{A.12})$$

$$= \log(K) - \mathbb{E}_{\mathbf{x}, \mathbf{p}} \left\{ \log \left[ \frac{p(\mathbf{x}, \mathbf{p})}{p(p(\mathbf{x})p(\mathbf{p}))} \right] \right\} \quad (\text{A.13})$$

$$= \log(K) - I(\mathbf{x}, \mathbf{p}). \quad (\text{A.14})$$

Therefore, minimizing the cross entropy loss would maximize the lower bound on the mutual information, since  $I(\mathbf{x}, \mathbf{p}) \geq \log(K) - R_{\text{opt}}$ . The major assumptions for this connection are: that all negative samples are independent of the current sample  $\mathbf{x}$  and

that  $K \rightarrow \infty$  negative samples are used, such that the approximation in Eq. (A.8) becomes valuable and the influence of the missing positive can be ignored. However, in [TDR<sup>+</sup>20] it is shown and concluded that if the negative sampling is not realized independently, the connection to the mutual information vanishes. Moreover, it is shown that a high mutual information does not necessarily lead to better representations. It is argued that the connection of the cross entropy-based contrastive loss to the metric loss (triplet loss) could be better to explain the recent success of methods using cross entropy-based contrastive loss. Hence, it is proposed that knowledge from metric learning domain could also improve cross entropy-based contrastive loss.

## A.2 Statistical Theory

### A.2.1 Statistical Moments

In this section, two statistical moments for discrete and continuous random variables are shown. Also, the formal description of the underlying functions are given. The whole section is based on [FHK<sup>+</sup>16] and [Bis06].

#### A.2.1.1 Discrete Random Variables

Let  $x$  be a discrete random variable with its realization  $x \in \mathbb{X}$ , where  $\mathbb{X}$  is the set of all possible values. The probability of  $x = x$  is defined by  $P(x = x)$ , then the so-called *Probability Mass Function* (PMF) can be written as

$$p(x) = P(x = x). \quad (\text{A.15})$$

The following explanations hold for the case where  $\mathbb{X} = \{x_1, \dots, x_I\}$  with  $x_1 < x_2 < \dots < x_k \leq x_p < \dots < x_I$ . The *Cumulative Distribution Function* (CDF) of the discrete random variable  $x$  is

$$P(x \leq x_k) = \sum_{i=1}^k p(x_i). \quad (\text{A.16})$$

The expectation – the first statistical moment – of the discrete random variable  $x$  is defined as

$$E_x\{x\} = \mu_x = \sum_{i=1}^I x_i p(x_i). \quad (\text{A.17})$$

The variance – the second centered statistical moment – based on the expectation is formulated as

$$\text{Var}_x\{x\} = \sigma_x^2 = \sum_{i=1}^I (x_i - \mu_x)^2 p(x_i). \quad (\text{A.18})$$

### A.2.1.2 Continuous Random Variables

Consider the random variable  $x$  as continuous, therefore  $x \in \mathbb{R}$ . In this case the distribution of  $x$  is given by the *Probability Density Function* (PDF)

$$p(x) = \lim_{dx \rightarrow 0} \frac{P(x - dx < x \leq x)}{dx}. \quad (\text{A.19})$$

The CDF for continuous random variables is defined as

$$P(x \leq x_P) = \int_{-\infty}^{x_P} p(x) dx. \quad (\text{A.20})$$

The expectation and the variance of continuous random variables follow

$$E_x\{x\} = \mu_x = \int_{-\infty}^{\infty} xp(x) dx \quad (\text{A.21})$$

and

$$\text{Var}_x\{x\} = \sigma_x^2 = \int_{-\infty}^{\infty} (x - \mu_x)^2 p(x) dx. \quad (\text{A.22})$$

## A.2.2 Probability Distributions

Following the separation between discrete and continuous random variables, first, the discrete distributions (Bernoulli, binomial and Poisson) are explained. Second, the continuous distributions (normal and beta) are discussed.

### A.2.2.1 Bernoulli Distribution

Let  $x$  be a binary random variable, such that  $x \in \{0, 1\}$ . The probability of  $x$  being 1 is  $P(x = 1) = \mu$  and hence  $P(x = 0) = 1 - \mu$ . Based on this, the Bernoulli distribution is defined by the PMF

$$p(x|\mu) = \mu^x(1 - \mu)^{1-x} \quad (\text{A.23})$$

and the notation  $x \sim \text{Bern}(\mu)$ .

The expectation and variance of a random variable with a Bernoulli distribution are

$$E_x\{x\} = \mu \quad (\text{A.24})$$

$$\text{Var}_x\{x\} = \mu(1 - \mu) \quad (\text{A.25})$$

### A.2.2.2 Binomial Distribution

Let a Bernoulli trial be a trial where its outcome can be modeled by a Bernoulli distribution. If such a trial is repeated  $n$  times independently, the binomial distribution is used

to model the distribution of  $m$  trials leading to  $x = 1$  as

$$p(m|n, \mu) = \binom{n}{m} \mu^m (1 - \mu)^{n-m} \quad (\text{A.26})$$

$$= \frac{n!}{m!(n-m)!} \mu^m (1 - \mu)^{n-m} \quad (\text{A.27})$$

with  $m \in \mathbb{N}_0$  and  $m \leq n$ . Therefore, a binomial distributed random variable is written as  $m \sim \text{Bin}(n, \mu)$ .

A binomial distributed random variable has the expectation and the variance,

$$E_m\{m\} = n\mu \quad (\text{A.28})$$

$$\text{Var}_m\{m\} = n\mu(1 - \mu). \quad (\text{A.29})$$

The CDF of a binomial distributed random variable is

$$P(m \leq k) = \sum_{i=0}^k \binom{n}{i} \mu^i (1 - \mu)^{n-i}. \quad (\text{A.30})$$

### A.2.2.3 Poisson Distribution

Another discrete distribution is the Poisson distribution. Like in the binomial distribution the amount of certain outcomes is modeled. However, for the Poisson distribution, a fixed interval is considered (time or space), with  $\lambda$  the average number of events within the corresponding interval. The Poisson distribution models the probability of observing  $m$  events within an interval through the PMF

$$p(m|\lambda) = \frac{\lambda^m}{m!} e^{-\lambda}, \quad (\text{A.31})$$

with  $m \in \mathbb{N}_0$  and  $\lambda > 0$ . Following the introduced notations, a Poisson distributed random variable is written as  $m \sim \text{Poi}(\lambda)$ .

For a Poisson distribution, the expectation and variance are

$$E_m\{m\} = \text{Var}_m\{m\} = \lambda. \quad (\text{A.32})$$

Although, the Poisson distribution differs from the binomial distribution, the Poisson distribution is a limit case of binomial distribution. Therefore, if  $n \rightarrow \infty$  and  $\mu \rightarrow 0$  is very small, the binomial distributed random variable  $m$  can be approximated through the Poisson distribution, where  $\lim_{\substack{n \rightarrow \infty \\ \mu \rightarrow 0}} \text{Poi}(n\mu) \approx \text{Bin}(n, \mu)$  holds.

Moreover, the Poisson distribution can be approximated by a normal distribution if  $\lambda \geq 30$ . Therefore,  $m \sim \mathcal{N}(\lambda, \lambda)$  (Sec. A.2.2.4).

#### A.2.2.4 Normal Distribution

One of most important distribution is the normal or Gaussian distribution. Given the normal distributed continuous random variable  $x$ , with  $x \in \mathbb{R}$  its PDF is given by

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (\text{A.33})$$

with  $\mu \in \mathbb{R}$  and  $\sigma^2 > 0$ . The compact notation is  $x \sim \mathcal{N}(\mu, \sigma^2)$ .

The expectation and the variance of the normal distribution yield

$$E_x\{x\} = \mu \quad (\text{A.34})$$

$$\text{Var}_x\{x\} = \sigma^2. \quad (\text{A.35})$$

A special type of the normal distribution is the standard normal distribution, with  $\mu = 0$  and  $\sigma^2 = 1$ . If  $x$  is an arbitrary normal distributed variable, then the standardized random variable

$$z = \frac{x - \mu}{\sigma} \quad (\text{A.36})$$

is standard normal distributed. Where the standard normal distribution can be written as

$$p(z) = \varphi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}. \quad (\text{A.37})$$

The CDF of the standard normal distribution and of the normal distribution can not be expressed by elementary function. In case of the standard normal distribution, using Eq. (A.20) leads to

$$P(z \leq z_P) = \Phi(z_P) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_P} e^{-\frac{z^2}{2}} dz. \quad (\text{A.38})$$

Some typically required value pairs of the CDF are listed in Tb. A.1. Due to the normal distribution's symmetry,  $z_P = -z_{1-P}$  holds.

Table A.1: Typical value pairs for the CDF of the standard normal distribution.

$P(z \leq z_P)$	0.50	0.75	0.90	0.95	0.975	0.99
$z_P$	0.00	0.67	1.28	1.64	1.96	2.33

Another important property is based on the symmetric lower bounded CDF

$$P(-z_P \leq z \leq z_P) = \frac{1}{\sqrt{2\pi}} \int_{-z_P}^{z_P} e^{-\frac{z^2}{2}} dz. \quad (\text{A.39})$$

The typically used value pairs for this case are shown in Tb. A.2



Table A.2: Typical value pairs for the lower bounded CDF of the standard normal distribution.

$P(-z_P \leq z \leq z_P)$	0.50	0.683	0.90	0.95	0.954	0.99	0.997
$z_P$	0.67	1.00	1.64	1.96	2.00	2.57	3.00

In order to determine the values of the CDF for a normal distribution the quantity

$$x_P = \mu + \sigma z_P \tag{A.40}$$

along with the appropriate table has to be used.

### A.2.2.5 Beta Distribution

The beta distribution is a distribution for continuous random variables. The beta distributed random variable  $x$  with

$$x \in \begin{cases} [0, 1] & \text{for } a, b \geq 1 \\ (0, 1) & \text{else} \end{cases}, \tag{A.41}$$

is described by the PDF

$$p(x|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}. \tag{A.42}$$

The fraction is the inverse of the beta function, and it is used to normalize the function. More precisely, it is based on the gamma function

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du, \tag{A.43}$$

which yields

$$\Gamma(x) = (x-1)! \tag{A.44}$$

if  $x \in \mathbb{N}$ . The short notation of a beta distributed random variable is  $x \sim \text{Beta}(a, b)$ .

The parameters  $a$  and  $b$  have a strong influence on the shape of the distribution. As a special case,  $a = b = 1$  yields the uniform distribution.