Free University of Berlin
Department of Mathematics and Computer Science

Master's Thesis

Impact of Domain-Based Data Sampling Approaches on the Performance of Object Detection Models.

– In the Context of Automated Driving

Simon Ruber
Matr. 5499549

First Reviewer (Supervisor):
Prof. Dr. Daniel Göhring (Freie Universität Berlin)

Second Reviewer:
Prof. Dr. Tim Landgraf (Freie Universität Berlin)

November 20, 2022

Acknowledgements

I would like to thank Hanno Stage, research associate at *Forschungszentrum Informatik (FZI) Berlin*, for the opportunity to write this thesis in the research group *Embedded Systems and Sensors Engineering* and for his expert guidance and professional advice throughout all steps while working on this thesis. Further on, I want to thank Prof. Dr. Daniel Göhring and Prof. Dr. Landgraf for their professional advice and supervision.

Abstract

Deep learning-based models have become essential for the progress in object detection, but they are heavily dependent on the data with which they are trained. The optimal constitution of this training data to support the model learning the patterns of the data is not yet fully understood. One subtopic of this area is the relevance of image domains which is investigated within this thesis. Image domains and their associated classes influence the visual attributes of images and therefore lead to systematic differences between domain classes (e.g., time of the day or weather). This work proposes a process to evaluate the relevance of image domain classes and to measure their impact on object detection models.

Further on, the impact of image domain-based sampling on the model performance is evaluated. The BDD100K dataset was used as the data source for the experiments. Cleaning and label validation processes were developed to prepare the dataset. The relevance of an image domain class and the impact of domain-based sampling were tested with the YOLOv5s-P6 model. Twelve image domain classes, belonging to three image domains (*weather*, *time of the day* and *scene*) were investigated. Ten out of twelve image domain classes are considered relevant for the performance of the object detection model.

Three model groups, trained with stratified sampled data, were tested against models trained with randomly sampled data. Stratified sampling was not superior in any of the conducted comparisons. Instead, the object size distribution in the training data of the models showed significant impact on the model performances.

Contents

List of Figures

List of Tables

Acronyms

ANN	Artifical Neural Network
AP	Average Precision
API	Application Programming Interface
AUC	Area Under the Curve
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CV	Cross-Validation
DETR	Detection Transformer
DNN	Deep Neural Network
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HAD	Highly Automated Driving
IDC	Image Domain Class
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IMU	Inertial Measurement Unit
IoU	Intersection over Union
IS	Iterative Stratification

LP	Label Powerset
mAP	mean Average Precision
MLC	Multi-Label Classification
MLD	Multi-Label Dataset
MOTS	Multiple Object Tracking and Segmentation
NMS	Non-Maximum Suppression
OSM	OpenStreetMap
PCA	Principal Component Analysis
R-CNN	Region based Convolutional Neural Network
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
t-SNE	T-distributed Stochastic Neighbor Embedding
TN	True Negative
TP	True Positive
UMAP	Uniform Manifold Approximation and Projection
YOLO	You Only Look Once

1 Introduction

This thesis is mainly motivated by four factors. First, Highly Automated Driving (HAD) has been one major research topic in the automotive sector in recent years. Among other benefits, HAD promises to increase road safety by avoiding crashes, to increase independence by allowing people incapable of steering a car by themselves to travel independently, and to free up time by removing the burden of steering a car while traveling [33].

Second, Deep Neural Networks (DNNs) have become essential for the progress in object detection (among other computer vision tasks such as semantic- or instance segmentation) over the last years. This performance boost enables the application of object detection in critical environments, such as HAD. However, understanding the model's behavior and, even more important, its limitations is a topic of high relevance. Even though DNN-based models show the ability to generalize to unseen situations, this skill is limited [23].

Third, compared to classical object detection approaches, the performance of DNN-based models heavily depends on the amount and quality of data during the training phase. For example, feature maps in a Convolutional Neural Network (CNN) are not predefined but learned from the data [31]. This leads to the question of how to optimally collect and assemble training datasets to improve performance and robustness further.

Finally, there exists a lack of research on how to take image domains into account during this process of gathering and sampling data for model training and evaluation. Image domains and their associated classes influence images' visual attributes, and systematic differences between domain classes can be found. Examples are the time of the day and weather (see Section 2.4.2). The impact of image domains is not yet thoroughly researched, and published experiments are short on details about how the model comparisons have been performed. Therefore, this thesis shall help to close the existing gap.

Two subsequent questions must be answered to evaluate the impact of image domains on an object detection model. First, which image domains are of relevance for such a model? An answer to this question could help to focus the data-gathering efforts, and relevant image domain gaps in a dataset could be identified. Second,

can image domain-based sampling be used to improve the model performance and its robustness once the relevant image domains are identified?

The impact of a data attribute, such as an image domain, on an arbitrary model can not be answered in general. Instead, a repeatable process will be described to identify relevant Image Domain Classes (IDCs) and evaluate their impact on object detection models. The vision is that once this process is defined, it can be applied to different datasets and models and support the gathering and assembling of datasets in a specific use case.

The research questions that are investigated in this work are:

1. Which image domains are relevant for the training data of an object detection model?
2. What impact does image domain-based stratified sampling of a dataset have on the performance of an object detection model?

The object detection task has been chosen since, in automated driving, awareness of surrounding objects is crucial for the safe and comfortable behavior of the car. As a specific example, the BDD100K driving dataset [28] and the YOLOv5 object detection model, [34] have been chosen.

The work is split up into four sections. The section Preliminaries describes the fundamentals of object detection, model validation, data sampling methods, and significance tests. Also, the selected dataset is introduced, and an overview of its characteristics is provided.

Within the Method section, the developed methods for the dataset analysis and cleansing are described, followed by a detailed explanation of the two performed experiments: the image domain relevance test and the comparison of different domain-based sampling approaches.

In the Evaluation section, the results of data cleansing and the executed experiments are listed, interpreted, and set into the context of inevitable limitations.

The last Section, Summary & Future Work, sums up the most critical parts of the thesis and reflects on possible next steps and prospects to further improve and extend this thesis.

2 Preliminaries

Before detailing the part of the contribution, the relevant preliminaries are described. The fundamentals of object detection and DNNs, sampling methods, validation methods, evaluation metrics, significance tests, the selected dataset, and image transformation techniques are presented. Furthermore, the thesis will be set in the context of related work.

2.1 Object Detection

Object detection is one of the fundamental tasks in computer vision. Object detection can help to gain a detailed understanding of an image by providing information about objects visible in the image. It is defined as the combined task of identifying the 2D-location of an object in an image (object localization) and categorizing the object into defined classes (object classification). Besides detecting objects in static images, it can also be applied to image series. However, in the scope of this work, only the case of static image object detection is considered. The difficulty of this task lies in the diversity of the images and objects themselves [24]. Even if one takes a simple object, it can be seen from different viewpoints and angles, in various positions, scales, or colors, and in changing lighting conditions and different environments [8]. Additionally, effects like partial occlusion of objects can come into play as another aspect of complexity. The traditional approach of detecting objects in images is divided into three sub-tasks [24]:

1. Informative region selection,
2. feature extraction and
3. object classification

When identifying regions of interest, it has to be considered, that the objects might appear in any size and position in the image. Therefore when using techniques like the sliding window approach, an exhaustive number of candidate windows have to be processed, which is computationally expensive. In the sliding window approach, smaller areas of the image (windows) are defined. The sub-images encapsulated by the windows are evaluated at different positions (sliding) whether they contain an object. Instead of using all possible window sizes, a predefined set of candidate

windows reduces the computational cost, but for the risk of potentially missing objects [24].

Once the regions are defined, visual features have to be extracted from them. Various feature extractors have been developed over the years, to name a few, Hough Transform [1], or Haar-like features, as used in the Viola-Jones algorithm [4]. However, the complexity and diversity of objects and images make it challenging to develop universal feature extractors [24]. Lastly, a classifier is used to categorize the objects based on the extracted features.

This traditional approach with shallow region proposal algorithms and feature extractors is considered not state-of-the-art since it is limited in multiple aspects. Besides the expensive task of processing many candidate windows, the manual design of robust feature extractors is challenging. DNNs helped to mitigate the identified problems and advanced the progress in object detection [24].

2.1.1 Artificial Neural Networks in Object Detection

Ansari [25] explains the general idea of Artifical Neural Networks (ANNs). They are designed to mimic some functionalities of brain cells in a simplified manner by artificially replicating neurons and their network-like connections. Inspired by the processes in the brain, an artificial neuron receives weighted input signals and generates an output based on the applied (activation) function. Artificial neurons can form layers that receive inputs from the preceding layer and forward their processed output to the succeeding layer based on their connections and weights As soon as multiple layers are "hidden" between the input and output layers, a network can be called *deep* even though there is no single definition of the required number of hidden layers. The parameters (weights and biases) of an ANN have to be learned. This happens by training the model with input data and evaluating an appropriate error function. Based on the result, the weights and biases are updated in a backward pass (backpropagation) [25].

Numerous different network architectures have been developed over the last years. One of the most fundamental and important in the context of computer vision and object detection is CNN. CNNs help to solve the problem of finding

robust features by generating self-learned feature maps during training. Due to the nature of convolutions, in comparison to fully connected networks, fewer connections and, therefore, fewer weights are used between subsequent layers since neurons are only connected to a small set of adjacent neurons (the receptive field). Besides the convolution, additional transformations like pooling can be applied [24].

CNNs provide substantial advantages over traditional (shallow) feature extraction methods. Among those are multiple convolutions of an image that enable the model to learn features on different levels of abstraction and a wide field of reception. Since the features are learned from the training data, the features do not have to be designed manually. Further on, the deep architecture has an increased expressiveness compared to manually built feature extractors [24].

Object detection models based on DNNs can be divided into three different model types as shown in Figure 1.

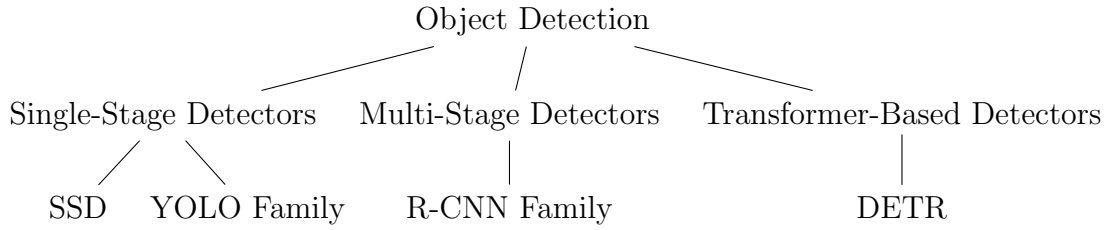

Figure 1: Overview of object detection architectures with examples (leaves)

Multi-stage detectors like the Region based Convolutional Neural Network (R-CNN) [14] follow a similar procedure as the traditional shallow object detection, while single-stage detectors such as You Only Look Once (YOLO) perform localization and classification of objects in one single pass of the network [18]. Transformer-based models such as the Detection Transformer (DETR) [26] model use a CNN-based feature extractor combined with a encoder-decoder transformer and a feed-forward network to generate the predictions. Transformer-based models reached state-of-the-art status in the well-known MS-COCO Challenge [40].

Multi-Stage Detectors Modern multi-stage object detectors use a region proposal framework to bridge the problem of object detection to image classification. Even though there are multiple region proposal-based model families, the multi-stage process will be explained with the example of the R-CNN family. Figure 2 shows a schematic drawing of a multi-stage architecture.

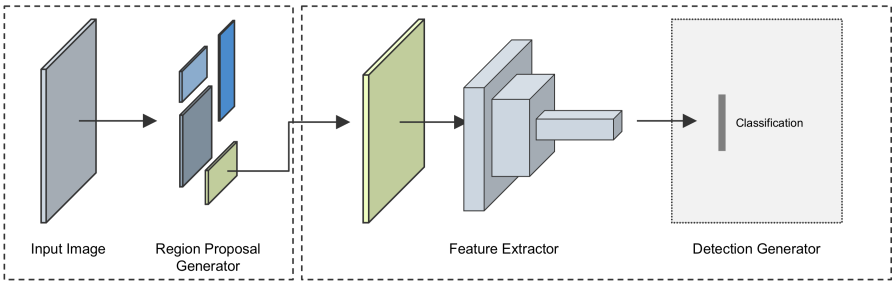

Figure 2: Schematic drawing of a multi-stage object detection architecture. Region-proposals are generated in the first stage. Afterward, features are extracted for each region and object classification is performed.

R-CNN The R-CNN model was initially developed after the success of AlexNet (a CNN-based image classification model) in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [14]. An additional model stage was added to utilize the proven performance of CNNs in image classification in the object detection task. This first stage identifies areas of interest and creates region proposals for the image. Doing so helps to solve the efficiency problem of the sliding window approach. The applied technique of *selective search* uses grouping and saliency cues to reduce the search space and provides approximately 2k candidate windows for an image. Afterward, in the second stage, a CNN is used to extract a feature vector for each of the proposed regions. Finally a Support Vector Machine (SVM) classifier is applied to classify the image regions based on the extracted features and the regions are refined to become the final bounding boxes. The model's advantages are the efficiency and performance improvement compared to the existing state-of-the-art models at the time of development [14]. However, there are also some disadvantages, such as no actual end-to-end training due to the multi-stage architecture or the requirement to pass a fixed-size region proposal to the detection stage [24].

Single-Stage Detectors The class of single-stage detectors contains multiple model families. Two prominent families are presented. Single-stage detectors have one design pattern in common: The avoidance of the region proposal stage. Therefore, bounding box prediction and object classification have to be trained and performed in one unified step [24]. Figure 3 shows a schematic drawing of a single-stage architecture.

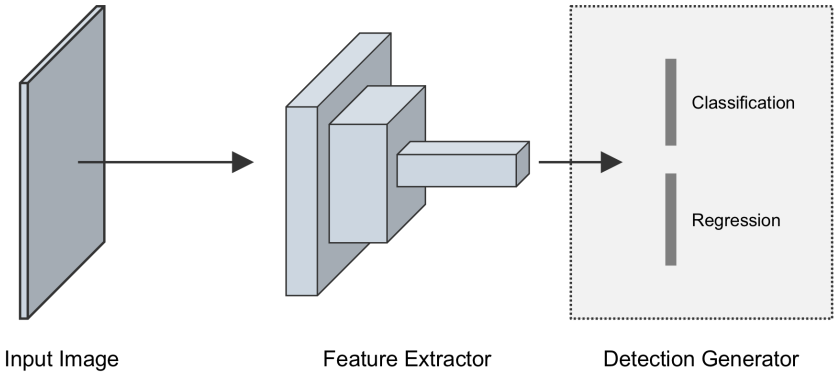

Figure 3: Schematic drawing of a single-stage object detection architecture. The input image is used as a whole during feature extraction. Afterwards, bounding box regression and classification are performed in one step.

YOLO: Instead of regions, the *YOLO* framework uses the complete image for object detection. This is achieved by using a single CNN. Afterward, the predictions are generated based on confidence score thresholds by applying Non-Maximum Suppression (NMS). The general idea of the model is that the image is divided into a grid. Each grid cell has two major prediction tasks. First, it is predicted whether the center of one or multiple bounding boxes lies within the grid cell, and second, if a bounding box center is found, the class of the corresponding object has to be predicted [24]. YOLO can be trained in a simple end-to-end training process, and its inference speed allows for an application in real-time environments. Also, its generalization capabilities are superior to models such as R-CNN [18]. However, the shortcomings of *YOLO* are difficulties in detecting small objects [24]:

After the initial development of YOLO (YOLOv1), further improvements have been introduced over multiple iterations, such as a change of the network's archi-

tecture to a fully convolutional network [20].

Further modifications and extensions have been made that lead to new models: YOLOv2 and v3. Afterward, the original authors of YOLO stopped their development due to privacy concerns and the fear of military applications of their models. However, development was continued by multiple people simultaneously, and different versions up to YOLOv7, PP-YOLOE, and YOLOX have been published. Tracking the improvements is difficult due to the number of releases and modifications that have been published. One major version that is widely used and still under open and active development is YOLOv5 [34]. YOLOv5 is based on YOLOv3 and completely implemented with PyTorch, which makes it easily accessible, and the performance and robustness of YOLOv5 were improved over multiple iterations. Further on, multiple model sizes are maintained, which increases the flexibility to adjust the model size based on the available hardware and requirements [34].

SSD: The Single Shot MultiBox Detector (SSD) [17] framework uses a different approach for detection, compared to YOLO's image grid. The detection is based on a convolutional "base network," which is used as a feature extractor. Afterward, a set of predefined anchor boxes are applied on multiple additional convolutional layers. The layers are designed to decrease the size of the feature map from layer to layer, to handle different object scales. The anchor boxes around each anchor have different sizes and aspect ratios and act as detectors. [25] The advantages of SSD are that SSD300 is slightly faster than YOLOv1 and SSD300 and SSD500 have higher accuracy than YOLOv1 [20]. The difficulties are similar to YOLO since the performance on small objects allows for improvements [24].

Transformer Based Models The transformer-based models are the first detectors that frame the object detection problem as a direct-set prediction. This differs from other approaches(single-stage or multi-stage detectors) since no post-processing step such as NMS is used [26]. Therefore the DETR architecture is proposed as the first fully end-to-end object detector [30]. Transformer-based models have become state-of-the-art in object detection and dominate the leaderboard in benchmark challenges such as the MS-COCO Challenge [40]. Figure 4

shows a schematic drawing of a transformer based architecture.

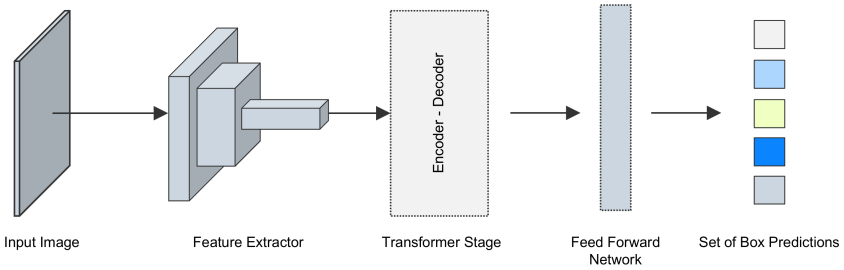

Figure 4: Schematic drawing of a transformer based object detection architecture. First, image features are extracted and passed to an encoder-decoder transformer. The generated embeddings are used in a feed forward network to perform a direct set prediction.

DETR: The DETR framework consists of three main components. First, a CNN backbone to extract features from the fed input image, second the transformer encoder-decoder, and third a feed-forward network to generate the final bounding box coordinates and class predictions. Neither region proposals nor grid/anchor boxes are used in the architecture. Therefore, the model does not depend on the quality of such predefined techniques [26]. The performance of transformer models is comparable with highly optimized multi-stage detectors, but the models are not yet applicable to real-time environments due to slower inference [26].

2.1.2 Imbalance in Object Detection

Multiple imbalances directly related to the objects are known. When decomposing the object detection task into object localization and object classification, one can see two potential sources of imbalance. In the task of object localization, a bounding box has to be positioned and scaled. In both sub-tasks, imbalances can exist. The positions of objects and the size/ratio of the sizes can be imbalanced over the classes, meaning some classes can be more diverse in their scale or position than others. These imbalances are known as *scale imbalance* and *spatial imbalance* [22].

In the classification task, *class imbalance* is a well-known problem in the do-

main of image classification. However, in object detection, two types of class imbalance exist. The first one is known as a foreground-to-background imbalance. Nearly infinitely many bounding boxes that do not contain an object exist, while the number of actual positives is small. In addition, one also has to deal with foreground imbalance, meaning an imbalance in the instance count of different classes [22].

Besides the object-related imbalances, an additional type exists, referred to as *objective imbalance*. This imbalance is related to the learning process of an object detection model and describes imbalances in the loss functions. Since object detection combines multiple tasks, multiple loss functions are usually used to describe the level of fit. Possible loss functions are classification loss or bounding box loss. If those loss functions are combined but not balanced to be on the same magnitude, one loss and, therefore, one sub-task can dominate the optimization [22].

2.2 Sampling Methods

Validating a model in the supervised-learning setting requires splitting the available labeled data into multiple sets. When doing so, one needs to generate representative sub-samples of the dataset.

Many sampling methods have been developed over the years. Some of them can be applied to many datasets, such as simple random sampling. In contrast, others are computationally expensive or restricted to a certain type of data [9]. Only sampling techniques relevant to this thesis will be introduced to limit this introduction.

Simple Random Sampling One of the simplest but commonly used methods is simple random sampling. The probabilistic sampling method assigns equal probabilities to all samples. Assume, a dataset D of size $n = |D|$ is used to sample a subset T of size $m = |T|$, each sample $x \in D$ has the probability $p(x \in T) = \frac{n}{m}$ of being selected. This method has, in general, a low bias, is easy to implement and is computationally cheap. However, its limitations are that, in the case of non-uniformly distributed data or when $m \ll n$, it is prone to distort the repre-

sentation of the original dataset. Therefore, the method is known for having high variance [9].

Stratified Random Sampling Stratified sampling generally uses additional intrinsic information from the data to generate a sample that appropriately represents the structure of the dataset. Therefore, before sampling, the dataset is clustered based on the so-called *strata*, and samples are selected from each cluster. Different methods exist to select the samples. In the case of stratified *random* sampling, the samples are selected at random by applying simple random sampling for each cluster individually. The size, of which a cluster is represented in the sample is called *quota* and can be chosen based on different rules. Commonly used options to determine the quota are *equal allocation* and *proportional allocation* [9].

In equal allocation, each cluster adds the same number of samples to the sub-sample. For a Dataset D of size $n = |D|$ with q clusters C_i with $i = 1, \ldots, q$, a sub-sample T of size $m = |T|$ is drawn. Each cluster is represented with size $n_i = \frac{m}{q}$ in the sub-sample. When applying proportional allocation, the number of samples per cluster is based on the cluster-size. Each cluster C_i is represented with size $n_i = \frac{m}{n} \frac{|C_i|}{\sum_{j=1}^{q} |C_j|}$ in the sub-sample [9].

Stratified random sampling is advantageous when cluster-like structures exist in the dataset since care is taken to represent all clusters appropriately [9].

Multi-Label Stratification The above paragraph has focused on the principle of simple stratification with a single set of clusters/strata. However, the generalization of this approach to multiple sets of strata is not trivial. The problem is known as Multi-Label Stratification in the domain of Multi-Label Classification (MLC), where Multi-Label Datasets (MLDs) have to be split in a stratified manner.

Label Powerset Transformation One approach to deal with multiple labels is based on dataset transformations. The single-label-multi-class transformation, also known as Label Powerset (LP) transformation, can be applied to transform a multi-label problem into a multi-class problem. The transformation is performed by combining all labels of one image into one single label. This is performed for all images. Each unique label set is added as a class to the new LP.

In the case of multiple independent single-class labels, the number of classes of the new LP is upper-bound by 2^l, with l being the number of original labels [12]. In the case of multi-class labels, the actual label powerset for a dataset with l labels, each with c_i classes, is upper-bound by Equation 1.

$$|\text{LP}| \leq \prod_{i=1}^{l} c_i \tag{1}$$

After applying the LP transformation, the standard single-label stratification approach can be applied to generate a stratified sub-sample.

In the case of large l, the powerset transformation can lead to a situation where $|LP| \approx n$. This would render the transformation useless regarding the stratification since each cluster would contain only a few or even a single element [11].

Iterative Stratification Iterative Stratification (IS) is an alternative multi-label stratification approach proposed by Sechidis, Tsoumakas, and Vlahavas [11]. Compared to LP-based stratification, IS uses a relaxed ruleset. It can be used to fill k-folds of size r_i. The algorithm works iterative and distributes the samples of one label per iteration. It starts with distributing the samples assigned with the rarest label among the folds. The rationale behind this greedy approach is that rare labels have to be distributed appropriately at first since created imbalances might not be adjustable later on. In contrast, the distribution of frequent labels can still be corrected in case of imbalances. In each iteration, only samples that are assigned with the currently selected label are distributed. The selection of the fold that receives a sample is based on the existing delta for the given label. The fold which is missing the most samples of the current label is chosen first. In the case of ties, among those folds, the fold which misses the most samples is selected. In case of further ties, one of those folds is selected randomly. Again a greedy approach is chosen to minimize the existing deltas [11].

An important difference compared to the LP transformed stratification is that each label is treated individually, while the powerset stratification works on complete label sets. The advantage of the iterative approach is that the labels are

distributed in a meaningful way over all subsets, even in the case of many different labels, which would lead to sparse clusters in the LP-based stratification (few examples per strata). The disadvantage of the given approach is that interdependencies between labels are not addressed. This can lead to a distorted distribution of label pairs in the folds.

2.3 Model Evaluation

Designing and building a model with high generalization and prediction capabilities is one of the main targets when applying machine learning techniques. For the subset of supervised learning problems, a model is trained to predict an unknown target function which is approximated by learning from the given training data and its labels. After completing the training stage, a model should have two skills, predicting correct outputs for data coming from the training dataset and generalizing its predictions to unseen data. These objectives are at some point contrary since improving the model performance on the training data can lead to an overfit, which reduces the generalization capability. The model is starting to learn the training examples by heart instead of learning general patterns. Stopping training too early can lead to overall bad performance on both the training data and unseen data. This trade-off is also known as Bias-Variance Dilemma. To properly balance the two demands, validation techniques can be applied. One common family of techniques is called *cross-validation* [9].

After estimating the skill of a model using a validation technique, one needs to be able to compare these skill estimates of multiple models with each other. Significance tests can be used to perform a comparison. They are introduced at the end of this section.

2.3.1 Validation Methods

During model training, its predictions for the training data are computed, and the deviation between the predictions and the underlying ground-truth is used to optimize the model (training loss). This evaluation loop provides a measure for the first part of the two model objectives: the prediction capability for data coming from the training set. However, the model's generalization capability is still

unknown. Unseen data has to be used to get an estimate for this second objective. This data is typically referred to as validation dataset. Validation data is split-off before training starts. Therefore, the model cannot use this data for parameter optimization. The validation process introduces at least two new problems: First, how is such a data split performed? And second, how can one ensure a robust estimate of the model's generalization capabilities?

Generally, a validation method defines the number and sizes of the sub-samples used for training and validation and the sampling technique to split the data.

Cross-Validation Cross-Validation (CV) combines a whole family of methods to validate a model's performance. In its most simple form, the dataset is split into two parts by applying simple random sampling. One part is used for training, and the other is used to evaluate the generalization performance of the model [9].

Hold-Out Cross-Validation is also known as early-stopping cross-validation or train / validation / test split. The dataset is split into three mutually disjoint sets using simple random sampling. The training set T_{tr}, the validation set T_v and the test set T_t. T_{tr} is used for model training, while T_v is periodically used to evaluate the model performance on unseen data. Training is stopped when one condition is met, either the required model performance on T_v is reached, or the model does not improve its performance on T_v any further. This simple technique helps to avoid overfitting the training data but at the cost that the validation set T_v influences the training (decision of stopping). Therefore, to avoid biasing the performance estimate, a third independent set T_t has to be used to evaluate the model on unseen data. As an advantage of this method, the size of each sub-set can be chosen flexibly as long as $n_{tr} + n_v + n_t \leq n$ holds [9].

K-Fold Cross-Validation K-fold CV generalizes the basic CV principle to multiple training and validation splits. This approach aims to compute a robust estimate of the model performance by measuring repetitively. This method is especially useful when the dataset is small, and the standard hold-out approach would require too much data for validation and testing. Using simple random sampling, the dataset T is split into k mutually disjoint, equally sized subsets

(folds). Each subset is used $k-1$ times for training and one time for validation. For each iteration, all folds, except one, are combined to form the training dataset, while the remaining fold is used to validate the model performance. The same procedure of early-stopping as in hold-out cross-validation is applied, but due to several measurements, a dedicated testing set is omitted [9].

One disadvantage of k-fold CV for $k > 2$ is that the training datasets are not independent anymore. While the validation sets are still independent, the repeated use of each fold within the training data leads to the dependence of the training sets. This can bias the variance estimate and increase the Type I error rate of significance tests. The skill estimate itself stays unbiased [7].

RxK-Fold Cross-Validation Repeated k-fold CV generalizes the validation process even further. Here, the previously described k-fold cross-validation is repeated r times. In this case, the training and test sets overlap over multiple repetitions. Within one repetition, all k-folds are mutually disjoint [6]. The problem of dependent training sets remains, while another source of dependence is introduced since testing sets also overlap over multiple repetitions.

Stratified RxK-Fold Cross-Validation The above introduced general method of rxk fold cross validation can also be combined with different sampling techniques such as stratified random sampling, or in the multi-label case, e.g., LP transformed stratified random sampling can be applied. The validation mechanics stay the same, while the method to generate the folds is changed.

Random Resampling Random resampling repetitively divides the dataset into two disjoint sets. A training set T_{tr} of size n_{tr} and a validation / test set T_v of size n_v. The size of the subsets is not fixed, but usually, the training set is larger than the validation set. In each repetition, the sub-sets are drawn using the simple random sampling method. Datasets overlap over multiple repetitions. The model is trained using the training set, and the final skill estimate is performed on the validation set. This validation method is simple but at the drawback of dependence of the training and the test sets [3].

2.3.2 Evaluation Metrics

A skill estimate is required to validate and compare models. The metrics depend on the task of the model. In object detection, multiple performance measures have been developed or adapted. Common metrics are *Precision*, *Recall*, *Average Precision* (*AP*), and *mean Average Precision* (*mAP*). Different variants of AP are widely used across the object detection community as primary performance indicator to compare models [27] (e.g. in the Common Objects in Context (COCO) detection challenge).

Basic Performance Measures Knowledge about the composition of a *confusion matrix* (see Table 1) is fundamental for the understanding of advanced metrics in object detection.

		Ground-Truth		
		Positive	Negative	Total
Predictions	Positive	TP	FP	$TP + FP$
	Negative	FN	TN	$FN + TN$
	Total	$TP + FN$	$FP + TN$	N

Table 1: Exemplary confusion matrix

The following definitions are taken from [27]:

- True Positive (TP): correctly detected ground-truth bounding box.
- False Positive (FP): incorrectly detected object without the existence of a ground-truth bounding box or non-sufficient overlap with existing ground-truth bounding box.
- False Negative (FN): undetected existing ground-truth bounding box.
- True Negative (TN): no detection for an area where no bounding box exists in the ground-truth data.

The last measure (TN) is not used in object detection since infinitely many TNs exist. Whether a detection is correct or incorrect is based on the similarity between the predicted bounding box B_{pr} and the ground-truth bounding box B_{gt}. The similarity is measured using the Intersection over Union (IoU), also known as the Jaccard-Index. The formula to compute the IoU is given in Equation 2.

Categorization of predictions into TPs, FPs and FNs is performed based on one threshold t for the IoU. For IoU $\geq t$ the prediction is categorized as correct, if IoU $< t$ the prediction is categorized as incorrect [27].

$$\text{IoU} = \frac{\text{area}(B_{pr} \cap B_{gt})}{\text{area}(B_{pr} \cup B_{gt})}, \quad \text{with} \quad \text{IoU} \in [0, 1] \tag{2}$$

Precision can be derived from the above-introduced measures. It provides a measure of how good the actual predictions of a model are by comparing the TPs against all predictions. Therefore, precision P is the ratio of correct predictions among all predicted bounding boxes as shown in Equation 3 [27].

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{with} \quad \text{P} \in [0, 1] \tag{3}$$

Recall Recall, also known as *sensitivity*, can as well be derived using the basic measures from above. It provides the ratio of correct predictions among all ground-truth bounding boxes. This indicates how much of all relevant objects a model detects. Equation 4 shows the formula to compute the recall [27].

$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{with} \quad \text{R} \in [0, 1] \tag{4}$$

Precision-Recall Curve Since both precision and recall provide only an isolated view of the model performance; both measures can be combined into the precision-recall curve to create a more general measure of the model performance. Figure 5 shows exemplary precision-recall curves for multiple object classes. The confidence score, which describes the confidence of the model that a given prediction is correct, is used to generate this curve. The predictions are ordered descending by confidence. Generally, when considering only predictions with high confidence, a higher precision but a lower recall is expected. The precision-recall curve is expected to be (monotonically) decreasing when the predictions are sorted descending by confidence.

Average Precision The perfect detector is expected to find all objects (Recall = 1) and to predict correctly (Precision = 1). The area under this ideal precision-

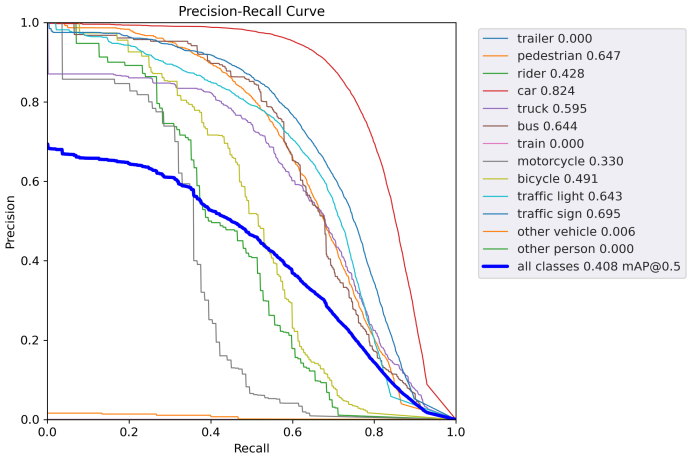

Figure 5: Precision-recall curves for multiple object classes. Predictions with an IoU$\geq$ 0.5 are considered as TP. The precision is evaluated for 101 recall values and interpolated.

recall curve is, in this case, maximized to 1.

$$\mathrm{AUC}_{\mathrm{ideal}} = 1, \quad \text{with} \quad \mathrm{AUC} \in [0, 1] \tag{5}$$

A good detector is expected to have a high Area Under the Curve (AUC) since this indicates a good precision/recall ratio for a wide range of confidence scores. In actual applications, the precision-recall curve is not smooth but "zig-zagged". Therefore to compute an estimate for the AUC, interpolation methods are used [27]. One interpolation method which is implemented in the COCO Application Programming Interface (API) for evaluation is the 101-point interpolation. Therein, the precision value is measured at 101 recall levels: $[0 : 0.01 : 1]$ [36]. The 101 measurements are interpolated by taking the maximum precision at or above a given recall level. This transforms the "zig-zagged" line into a step-like function. The enclosed area below the curve can be computed by summing over all segments, as shown in Equation 6 [27].

$$\mathrm{AP}_{101} = \frac{1}{101} \sum_{R \in \{0, 0.01, \ldots, 0.99, 1\}} \mathrm{P}_{\mathrm{interp}}(R), \tag{6}$$

with

$$\mathrm{P}_{\mathrm{interp}}(R) = \max_{\tilde{R}:\tilde{R}\geq R} \mathrm{P}(\tilde{R}) \tag{7}$$

One last step is necessary to obtain the mAP. Until now, all introduced metrics are computed on the object class level. The AP of multiple object classes can be combined into one metric by computing the mean. See Equation 8 for the mAP computation with N classes [27].

$$\mathrm{mAP} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i \tag{8}$$

The AP and mAP measures, as introduced, are considering one IoU threshold and all object sizes. Further variants exist, such as the mAP@.50:.05:.95 which averages the performance over 10 IoU thresholds or $\mathrm{mAP}^{\mathrm{SMALL}}$ which only considers objects $< 32^2$px size [27].

2.3.3 Significance Tests

Model validation methods are used to estimate the skill of a specific model. Comparison of these skill estimates can be performed by applying statistical significance tests. The introduced tests have been selected based on their application within the thesis.

Wilcoxon's Signed Rank Test This test is the non-parametric pendant to the paired Student's t-test and therefore does not depend on the assumption of normally distributed data. The test can be either a one-sample or two-sample test using matched pairs. The test investigates whether the location of the median θ is equal to a given constant. In the two sample test n paired measurements $(X_1, Y_1), \ldots, (X_n, Y_n)$ are transformed into their differences $D_i = X_i - Y_i$ and the median of the differences is investigated. The null hypothesis and the two-sided alternative are shown in Equation 9 [10].

$$H_0 : \theta = 0 \quad \text{vs.} \quad H_1 : \theta \neq 0 \tag{9}$$

The differences are ranked based on their absolute values $|D_i|$. The ranks are between 1 (for the smallest difference), and n (for the largest difference). In general, ties are not expected due to the continuity assumption. However, in applications, non-zero difference ties are typically handled by assigning an average rank to the tied differences. Zero differences are also not to be expected due to the continuity assumption. If they occur, they are included in the ranking process but are removed when computing the test statistic. The ranks are aggregated into the test statistics R^+, R^- and R as shown in Equations 10 to 12, which can be used interchangeably [10].

$$\mathrm{R}^+ = \sum_{D_i>0} \mathrm{rank}(|D_i|) \tag{10}$$

$$\mathrm{R}^- = \sum_{D_i<0} \mathrm{rank}(|D_i|) \tag{11}$$

$$\mathrm{R} = \mathrm{R}^+ - \mathrm{R}^- \tag{12}$$

Assumptions as defined in [10]:

1. The differences D_i have to be mutually independent.
2. The distribution F from which D_i is sampled has to be continuous and symmetric about the median.

The null hypothesis H_0 is rejected in the two-sided case at significance level α if $R^+ \geq w_{\alpha/2}$ [10] (or $R^- \geq w_{\alpha/2}$). Critical values w_α are tabulated or can be computed by permuting the rank signs [10].

Wilcoxon's Rank Sum Test The Rank Sum Test is the non-paired variant of Wilcoxon's Signed Rank Test. Therefore the Null-hypothesis and two-sided alternative are identical to Equation 9. Also, the same assumptions are valid. The test statistic is defined for two sample-groups G_1 and G_2 in [10] as:

$$W = \sum_{i=1}^{N} iV_i \quad \text{with} \quad V_i = \begin{cases} 1, & \text{if } i\text{-th smallest sample} \in G_1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

For tied observations, the mean rank can be computed. The test, as introduced, is not robust against variance differences among the two sample groups. However, a variant exists that can be used in that case [10]. The null hypothesis is rejected when W exceeds a critical value $w_{\alpha/2}$ in the two-sided case.

Hotelling's T^2 Test This test can be seen as the multivariate variant of Student's t-test. One-sample and two-sample variants exist. The two-sample test investigates whether the multivariate means of two distributions are equal (two-sided):

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 \quad \text{vs.} \quad H_1 : \boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2 \tag{14}$$

Assumptions In the case of p measured variables, the following assumptions are made for the Hotelling's T^2 test, as defined in [5]:

1. The two samples are drawn from a p-dimensional normal distribution $N_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $N_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ respectively.
2. The samples are drawn independently.
3. The covariance matrices are equal $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$ but can be unknown.

The T^2 test statistic is defined in characteristic form as shown in Equation 15 and provides a standardized distance measure for the two sample mean vectors $\bar{\boldsymbol{y}}_1$ and $\bar{\boldsymbol{y}}_2$ [5].

$$T^2 = (\bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2)' \left[\left(\frac{1}{n_1} + \frac{1}{n_2} \right) \boldsymbol{S}_{pl} \right]^{-1} (\bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2), \tag{15}$$

with $\boldsymbol{S}_{pl}$ being the unbiased estimator for the covariance matrix $\boldsymbol{\Sigma}$,

$$\boldsymbol{S}_{pl} = \frac{1}{n_1 + n_2 - 2} (\boldsymbol{W}_1 + \boldsymbol{W}_2), \tag{16}$$

$$\boldsymbol{W}_1 = \sum_{i=1}^{n_1} (\boldsymbol{y}_{1i} - \bar{\boldsymbol{y}}_1)(\boldsymbol{y}_{1i} - \bar{\boldsymbol{y}}_1)' = (n_1 - 1)\boldsymbol{S}_1, \tag{17}$$

$$\boldsymbol{W}_2 = \sum_{i=1}^{n_2} (\boldsymbol{y}_{2i} - \bar{\boldsymbol{y}}_2)(\boldsymbol{y}_{2i} - \bar{\boldsymbol{y}}_2)' = (n_2 - 1)\boldsymbol{S}_2, \tag{18}$$

and $\bar{\boldsymbol{y}}$ being the mean vectors.

$$\bar{\boldsymbol{y}}_1 = \sum_{i=1}^{n_1} \frac{\boldsymbol{y}_{1i}}{n_1}, \quad \bar{\boldsymbol{y}}_2 = \sum_{i=1}^{n_2} \frac{\boldsymbol{y}_{2i}}{n_2} \tag{19}$$

$(n_1 - 1)\boldsymbol{S}_1$ and $(n_2 - 1)\boldsymbol{S}_2$ are unbiased estimators of $(n_1 - 1)\boldsymbol{\Sigma}_1$ and $(n_2 - 1)\boldsymbol{\Sigma}_2$. As derived in [5].

H_0 is rejected when $T^2 \geq T^2_{\alpha,p,n-1}$. The critical value $T^2_{\alpha,p,n-1}$ can be computed from the T^2 distribution for a given significance level α, the number of variables p and the adjusted number of samples $n - 1 = n_1 + n_2 - 1$.

Assumption-Violation If the assumption of normality is violated, the same approach as for the Student's t-test can be applied. Based on the multivariate central limit theorem, for large n, $\bar{\boldsymbol{y}}$ is approximately normal distributed even if the sample $\boldsymbol{y}$ is not normally distributed [5]. In the univariate case, $n-1 = 30$ is widely accepted as the threshold to assume normality of the mean. However, to achieve a good approximation of the standard normal distribution in the multivariate case, n depends on p. Rencher proposes a method in [5] to estimate the required size of n based on the univariate case, by keeping the ratio of $T^2_{\alpha,p,n-1}$ and $T^2_{\alpha,p,\infty}$ approximately equal to the univariate case. For $\alpha = 0.05$:

$$\frac{T^2_{0.05,1,30}}{T^2_{0.05,1,\infty}} \approx \frac{4.171}{3.841} \approx 1,086 \overset{!}{=} \frac{T^2_{0.05,p,n-1}}{T^2_{0.05,p,\infty}} \tag{20}$$

E.g. for $p = 4$, $n - 1 \geq 90$ and for $p = 10$, $n - 1 \geq 200$ is required to remain in the large sample size domain.

When violating the assumption of equal covariance matrices, either a correction as defined in [2] can be performed or the sample sizes n_1 and n_2 have to be kept equal to obtain a robust version of Hotelling's T^2 Test.

Further methods, such as bootstrapping or permutation-based variants, can be applied, e.g., to derive a non-parametric p-value.

2.4 Data

Multiple open-source datasets are suitable for image-based object detection in the context of automated driving. A comparison of a subset of existing driving datasets is presented in Table 2. At least two components are necessary to use an image dataset for model training and validation: (1) the images and (2) additional annotations accompanying them. The annotations contain information about the visible objects in an image. Typically they are provided in the form of bounding box labels which contain details about the class, size, and position of an object. Figure 6 shows example images annotated with the provided bounding box labels.

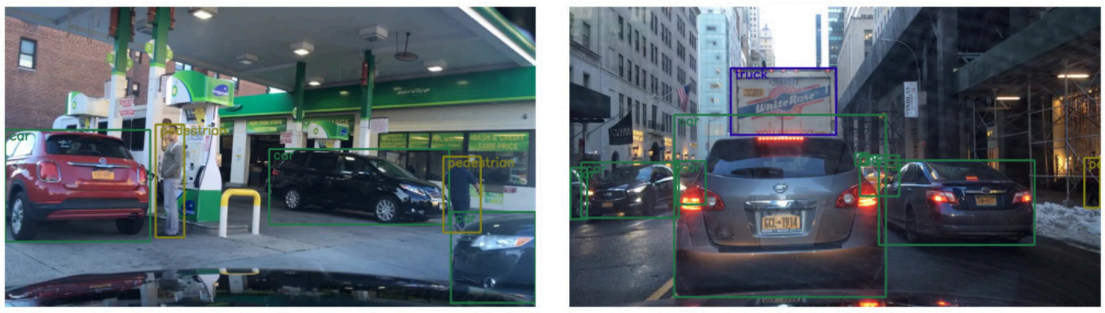

Figure 6: Example images from the BDD100K dataset with bounding box annotations.

	KITTI	**CityScapes**	**Mapillary**	**BDD100K**
# Train / val img	7481	3475	20000	80000
# Test img	7518	1525	5000	20000
Multiple locations	No	Yes	Yes	Yes
Multiple weather	No	No	Yes	Yes
Multiple seasons	No	Yes	Yes	Yes
Multiple times of the day	No	No	Yes	Yes
Multiple scene types	Yes	(No)	Yes	Yes
Multiple recording sources	No	No	Yes	Yes

Table 2: Comparison of driving dataset for object detection, based on [41], [13], [19] and [16]

.

For the comparison in Table 2, only the part of each dataset that is suitable for object detection is considered. The size of the datasets varies significantly from approximately 3'500 labeled images in CityScapes, up to 80'000 in BDD100K. CityScapes does contain more images, but since it is not directly built for 2D-object detection, only for 3475 images, instance-based annotations are provided, which can be transformed into bounding boxes. In addition, some indicators for the diversity of the datasets are given in Table 2. The most diverse dataset regarding the location is the Mappillary dataset since it is recorded at multiple globally distributed locations. In contrast, the KITTI dataset is recorded at only one location, the metropolitan area of Karlsruhe.

For *weather*-, *time of the day*- and *recording source*-diversity, two groups exist. The more diverse datasets regarding these indicators are Mappillary and BDD100K. CityScapes and KITTI provide only daytime images in non-adverse weather situations, recorded with a small set of recording systems.

In this context, a *scene type* describes the surrounding scene and not the driving situation itself. CityScapes is, as to be expected, recorded in urban environments only. The other datasets provide additional imagery, e.g., from highway-, countryside-, and, in the case of Mapillary, even off-road scenes.

Regarding the Mapillary dataset's recording scheme, one must mention that not all images are exclusively recorded with driving vehicles. But, *street-level* data is collected, which is recorded by different sources (e.g. pedestrians) [19]. The images provided in the other datasets are solely recorded with vehicles.

2.4.1 BDD100K Dataset

The BDD100K dataset will be introduced in more detail since it is the dataset used within this thesis's practical part. The dataset contains images/videos and annotations for multiple computer vision tasks such as 2D-object detection, lane marking, driveable area detection (vehicle localization), semantic instance segmentation, and Multiple Object Tracking and Segmentation (MOTS). Therefore, the dataset is suitable for multitask learning [28].

Dataset Collection and Split The dataset comprises 100k video recordings, collected and uploaded crowd-source based. The videos are provided by thousands

of riders and are recorded during approximately 50k different rides [28]. All videos have been recorded with the rear-camera of an iPhone 5 [42] which provides a video resolution of 1280x720px at 30fps. Each video clip is 40 seconds long. The keyframe at the 10th second of each video is taken to transform the video recordings into images. The videos are mainly recorded in five broader areas: San Francisco, the Bay Area, Berkeley, New York [28], and Israel/West Bank.

The dataset is split into three chunks. The largest junk, the training dataset, contains 70k images, the validation dataset contains 10k images, and the test dataset contains 20k images. For the training and validation datasets, additional data such as bounding box annotations are published [28]. As usual, the annotations for the testing dataset are not publicly available for benchmarking reasons. All classes labeled in the BDD100K dataset are shown in Table 11.

Metadata Besides the basic data necessary for 2D-object detection, additional metadata is available. Among other attributes, these are:

- Global Positioning System (GPS) based geographic coordinate, speed- and course measurements,
- Inertial Measurement Unit (IMU) based acceleration data
- the timestamp at which the image/video was recorded
- an anonymized ID of the rider who recorded the image

In addition to metadata directly recorded with the imagery, additional image labels are provided. The labels and their corresponding classes are shown in Table 3. The distribution of the classes in the BDD100K dataset will be discussed in the evaluation section (see 4.1.2).

Label	**Classes**
Weather	Clear, Partly Cloudy, Overcast, Rainy, Snowy, Foggy, Undefined
Scene	City Street, Highway, Residential, Gas Station, Tunnel, Parking Lot, Undefined
Time of the Day	Daytime, Night, Dawn/Dusk, Undefined

Table 3: Image labels and the corresponding classes in BDD100K

2.4.2 Image Domains

The term *image domain* has to be defined to lay the ground for upcoming sections of this work. Yu et al. and Romera et al. describe that images differ in their properties between domains, meaning that within a domain, a certain level of similarity can be expected. This domain difference / domain gap might affect the performance of an object detection or semantic segmentation model [28], [23].

Further on, models trained on a non-diverse dataset (diversity regarding the image domains) are prone to overfit the characteristics of their training data, as shown in [28].

One can conclude that within one domain, some visual attributes of an image are bound or at least systematically biased towards a specific peculiarity. This means the visual appearance of an image is influenced or even dominated by the domain it comes from.

To name the most prominent domain example within the driving context: time of the day and especially the differences between the two domain classes, daytime and night, are both visually dominant and heavily influential regarding the performance of an object detection / semantic segmentation model [28], [23].
The term *image domain* is therefore not bound to a closed set of characteristics but rather loose. Therefore, the parameters influencing an image are heavily context-dependent and have to be defined with the data at hand or at least with a clear vision of how the application context of a model will be. For this thesis, the image domains investigated are shown in Table 3, but further domains exist, such as the season, the country, or the environmental situation (e.g., topography), to name a few.

In a previous section, object detection imbalances have already been mentioned. However, the image domain distribution introduces another potential source of imbalance. For example, datasets such as CityScape or KITTI cover only a small subset of the existing domain diversity. However, even more diverse datasets, such as BDD100K, do not achieve to represent all the domains in sufficient size (this will be shown in the evaluation section).

2.5 Image Transformation

The transformation from image matrices to vectors and their distance determination requires a set of methods that are explained in the following part.

2.5.1 Image Featurization

Dimension reduction techniques can be applied to ease the comparison of images. For example, one non-linear technique used in [39] is the CNN-based image featurization. A pre-trained ResNet model, of which the last layers (fully connected block) are cut off, is used to generate a dense representation of an image, the feature vector. This reduces the image size to 512×1. The open-source implementation used in this thesis is available under [37].

2.5.2 Cosine Similarity

The similarity of the feature vectors can be measured by measuring the vector distance. The applied measure to compare images in [39] is the *cosine similarity*. However, other measures exist. By design, the cosine similarity does compare the similarity in the angles and not the magnitude of the vector, in comparison to, e.g., the Euclidean distance. The formula used to compute the *cosine similarity* is given in Equation 21. k measures the similarity of two row vectors x and y by computing the L2-normalized dot product. For identical vectors $k = 1$ and for vectors pointing in the opposite directions $k = -1$.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}\boldsymbol{y}^\top}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}, \quad \text{with} \quad k \in [-1, 1] \tag{21}$$

2.5.3 UMAP

Uniform Manifold Approximation and Projection (UMAP) is a technique to reduce the dimensions of a dataset. Its approach is based on constructing and simplifying a k-neighbor graph of the data. Its main objective is to preserve local distances of the data. However, compared to other dimension reduction techniques such as T-distributed Stochastic Neighbor Embedding (t-SNE) it also preserves more

of the global structure while being computationally more efficient. Three axioms are made that are fundamental to UMAP. (1) That there exists a manifold on which the data is uniformly distributed. (2) This manifold is locally connected, and (3) The topological structure of this manifold shall be preserved. This manifold is approximated with a k-neighbor graph and simplified to a low-dimensional representation. This representation is used to reduce the dimensions of the data [21].

2.5.4 PCA

PCA is a linear non-parametric method to transform the dimensions by changing the basis such that the signal-to-noise ratio is maximized for each dimension and the redundancy is minimized (correlation of dimensions). This is achieved by maximizing the variance and minimizing the covariance, which is a covariance matrix diagonalization. For an m dimensional dataset, this is achieved by iterative selecting m orthogonal vectors p_i in the direction of the maximum possible variance. The requirements of maximizing the variance for each vector p and the orthogonality lead to an ordered set of m vectors with descending variance. The vectors form a new basis for the dataset [15].

2.5.5 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) can be used for data clustering purposes. The fundamental assumption of a GMM is that the data is drawn "from a mixture of a finite number of Gaussian distributions with unknown parameters" [32]. When providing the number of clusters (components) (e.g. by visual inspection or due to an information criterion) the parameters of the Gaussian distributions can be approximated iterative by applying an expectation-maximization algorithm. After initializing the components (e.g., with k-means or randomly), the algorithm consists of two steps. First, the probability of each data point to be generated by the components is computed. Second, the parameters of the components are updated such, that the likelihood is maximized for the determined assignment [32].

2.6 Related Work

Besides this thesis, further publications investigate the effects of image domains in object detection and related computer vision tasks.

In the paper released with the BDD100K dataset, image domain effects were briefly investigated. This was done by splitting the data based on one domain, e.g., time of the day. Then, multiple models were trained using the prepared datasets, e.g., containing only daytime, non-daytime, or random samples. Each training set included 30k images. After training, the models were tested on three different test sets. The first set only included images from the daytime class, the second only included images from non-daytime classes, and the third set was the default validation dataset. All three models were evaluated on all test datasets, and the mAP was measured. The evaluation results in Table 4 show that all three models perform better on daytime images than on non-daytime images. However, the daytime model and random model are on par for the daytime dataset and the default validation dataset, while the random model is superior for the non-daytime dataset. The non-daytime model outperforms the daytime model on the non-daytime dataset but is inferior in all remaining comparisons. The experiment is identically performed for the scene type *city street* [28].

It is important to note that the paper is short on the specific experiment design. However, it remains unclear whether the instance distribution is comparable for all training sets and how training and testing were conducted in detail. Also, how the distribution of the other image domains was handled is not explained. It is concluded that both experiments show significant results, meaning the investigated domain classes are relevant for the performance of the object detection model, even though it is not stated how the decision about significance has been made [28]. The original result table of the daytime experiment is shown in Table 4.

Train \ Test	Daytime	Non-Daytime	Validation
Daytime 30k	30.6	23.6	28.1
Non-Daytime 30k	25.9	25.3	25.6
Random 30k	29.5	26.0	28.3

Table 4: mAP@.50:.05:.95 estimates of the daytime impact experiment performed in the BDD100K paper [28].

The authors of the CityScapes dataset have performed a similar experiment applied to semantic segmentation. They split their dataset in a stratified manner, based on four criteria: (1) size of the cities, (2) geographic east-west location, (3) geographic south-north location, (4) time of the year at recording [16].

The effects of extreme values of these characteristics are investigated by training a semantic instance segmentation model on the default validation dataset. This set contains 500 images and represents the diversity of the data within the four characteristics. The model is evaluated on subsets of the test dataset, which are designed to expose the model to extreme values of the four groups [16].

As a result, the effects of the four characteristics are small (variation of approx 1.5% in model performance) except for "time of the year" for which a larger effect is observed (3.9% performance increase on the *end of the year* subset). It is hypothesized to be "[...] due to softer lighting conditions in the frequently cloudy fall" [16]. Again, the experiment design is only briefly described, and handling of possible confounding factors, such as an imbalanced instance distribution, is not explained.

Besides investigating the impact of domain-based data sampling, some publications take a different approach to the problem of domain discrepancy. Romera et al. observe a performance decay of semantic segmentation models during nighttime. They propose a method of domain adaption by applying a Generative Adversarial Network (GAN) on the images to transform them from day to night and vice versa. This allows for two possibilities to tackle the domain gap. Either the amount of well-labeled night images during training can be increased (day-to-night transformation), or the multi-domain-class problem can be reduced to the daytime domain during inference (night-to-day transformation) [23].

3 Method

The Method section introduces all methods that have been developed in the practical work of the thesis. Often a combination of existing techniques was applied to achieve a specific objective. First, an overview of the *Label Database* generation will be given. This database is the pivotal point for steps like sampling or evaluating the image domain distributions. Afterward, the design of the two main experiments is explained.

3.1 Dataset Preparation

The BDD100K dataset is available in the form of several directories and files, and for the task of object detection, three parts of the dataset have been used. (1) the images, (2) the label annotations, and (3) the info files. The entire file and folder structure of the dataset can be found in the Appendix, Figure 33.

3.1.1 Label Database

To be in the position of performing quick and versatile analyses, the provided metadata, split over thousands of files, was stored in one single database. The key of this database is the unique image id.

Data from the bounding box labels (version 2020) was extracted in the first step. The labels provide details about all objects visible in the images (object class and 2D-bounding box). In addition, the metadata about the weather, scene type, and time of the day was added for each image. The bounding box information was aggregated into object counts for each image. The aggregation was performed on class and size levels. The classes, labeled in the BDD100K dataset are shown in Table 11. The size thresholds applied are taken from the COCO detection challenge. Objects whose bounding box area is $Bb_{\text{area}} < 32^2$ pixels are categorized as *small*. For bounding boxes $32^2 \leq Bb_{\text{area}} < 96^2$, the objects are categorized as *medium* and for bounding boxes $96^2 \leq Bb_{\text{area}}$, the objects are categorized as *large* [27].

In the second step, data stored in the info files was added. Those files have been

generated together with the original video recordings. Therefore, data for many attributes is available in a time-series format. Each file contains information about the ride (rideID), location (latitude, longitude, speed, course, accuracy), start- and end timestamps, and details about the acceleration (x,y,z). Since the time-series data is provided in different granularity, the keyframe (at the 10th second) does not necessarily match the sensor data resolution. Therefore, the sensor data points were matched by taking the closest data point based on the time difference between the data point's timestamp and the keyframe's timestamp. Finally, the actual keyframe timestamp was computed using the start timestamp provided in the info file and a time offset provided in the label file.

3.1.2 Duplicate Image Identification

After the initial data collection, the data quality had to be verified. The independence of the images is essential to obtain an unbiased estimate of the model performance. Therefore a process has been established to identify duplicate and dependent images. An image is categorized as duplicate when it is an identical copy of another image in the dataset. An image is categorized as dependent whenever it is strongly related to another image by, e.g., the same driving situation & position. The second condition cannot be transformed into a clear rule but rather into a range where cut-off values have to be defined.

The implemented data cleaning process is shown in Figure 7. The applied ruleset to remove images is shown in Table 5. As an alternative to the cumbersome cleaning process, one could also remove all images except one per rider and remove images taken in close-by geographic positions. However, this would dramatically shrink the size of the dataset and is therefore not preferred.

Before comparing the images with each other, they were transformed from their

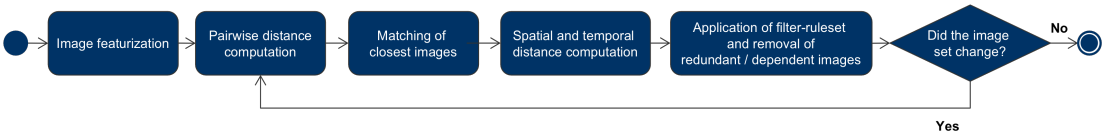

Figure 7: Iterative process to identify and remove redundant images. The image distances (visual, spatial, and temporal) are computed, and images are compared based on a predefined ruleset.

original shape $1280 \times 720 \times 3$ to 512×1 vectors. In the second step, pairwise distances are computed between the feature vectors. The applied distance measure is the *cosine similarity*. Step three matches the most similar image to each image. Until this step, the process is taken from Stage et al. in [39].

These steps unavoidably lead to approx. 80k matches. Further rules have to be applied to distinguish between actual duplicate or dependent images and only similar-but-different images.

Therefore in step four, the spatial- and temporal distances between the images and their closest matches are computed. Afterward, a filter-ruleset is applied (see Table 5). Each rule is applied independently. This means that for an image to be selected for removal by a rule, e.g., rule one, it has to be recorded with a maximum spatial distance of 1 m and a maximum temporal distance of 1 day with respect to its matched image. If these criteria are met, the image is marked as duplicate or dependent and is removed from the database. The idea behind the multi-rule approach is that one has the flexibility to restrict one dimension tightly while releasing the other dimensions, which might be fuzzy due to imperfections in the data collection process.

- Rule 1 is designed to detect images recorded at the same location.
- Rule 2 shall identify images recorded within a short timeframe.
- Rule 3 filters out visual duplicates.
- Rule 4 is a relaxed combined variant of rules 1 and 2 for images taken by the same rider.
- Rule 5-A and 5-B are relaxed variants of rule 3 for images taken by the same rider.

Rule	**Spatial dist**	**Temporal dist**	**Cosine sim**	**Same ride**	**Night**
1	$\leq$ 1m	$\leq$ 1day	-	-	-
2	$\leq$ 1000m	$\leq$ 10sec	-	-	-
3	-	-	$\geq$ 0.98	-	-
4	$\leq$ 1000m	$\leq$ 10min	-	Yes	-
5-A	-	-	$\geq$ 0.94	Yes	No
5-B	-	-	$\geq$ 0.95	Yes	Yes

Table 5: Filter-ruleset to identify redundant images. All conditions of a rule have to be matched. Images that are matched by a rule are removed from the dataset.

Finally, to deal with duplicate / dependent series of images, the whole process is applied in an iterative manner. After each iteration, the size of the image set is compared with the size after the preceding iteration. If the set size does not change, no further dependent images have been identified, and the process terminates.

3.1.3 Label Verification

The quality of the image domain labeling is of particular relevance for this work since the impact of image domain-based stratified sampling is investigated. Individual processes are necessary for each image domain to verify the labeling quality.

Time of the Day The labels are verified by computing additional *time of the day* labels based on different data sources. Afterward, the labels are compared with each other.

Elevation of the Sun Based Label Based on the image timestamp T_{image} and spatial location L_{image}, a *time of the day* label can be assigned by estimating the elevation of the sun. The process is shown in Figure 8. The astronomical definitions of dawn T_{dawn}, sunrise T_{sunrise}, sunset T_{sunset} and dusk T_{dusk} are used to map a tuple of $(T_{\text{image}}, L_{\text{image}})$ on the provided daytime categories [29]. The key timestamps T_i are computed based on L_{image} and the day of recording. Afterwards, T_{image} is mapped on the categories. The mapping is shown in Equation 22.

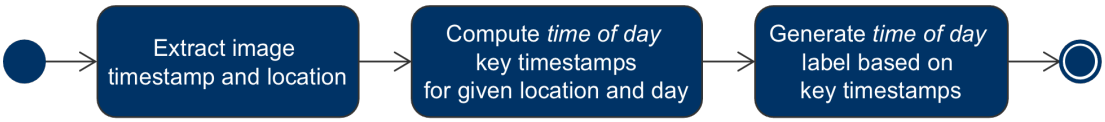

Figure 8: Process to generate a *time of the day* label, based on the elevation of the sun.

$$\phi(T_{\text{image}}) = \begin{cases} \text{night}, & T_{\text{image}} < T_{\text{dawn}} \\ \text{dawn/dusk}, & T_{\text{dawn}} \leq T_{\text{image}} < T_{\text{sunrise}} \\ \text{daytime}, & T_{\text{sunrise}} \leq T_{\text{image}} < T_{\text{sunset}} \\ \text{dawn/dusk}, & T_{\text{sunset}} \leq T_{\text{image}} < T_{\text{dusk}} \\ \text{night}, & T_{\text{dusk}} \leq T_{\text{image}} \end{cases} \tag{22}$$

An open-source implementation is used to compute the critical timestamps for the elevation of the sun [29].

Cluster Based Label After image featurization, the 512×1 feature vectors can be further reduced by applying classical dimension reduction techniques. When applying UMAP, one can see that two clusters are formed. They separate daytime from night quite clearly (see Figure 21). This situation can be exploited to generate a second label. Once the UMAP reduction is performed, a clustering algorithm such as GMM can be applied. The discovered clusters are used as additional daytime and night labels. The cluster-based labeling process is shown in Figure 9.

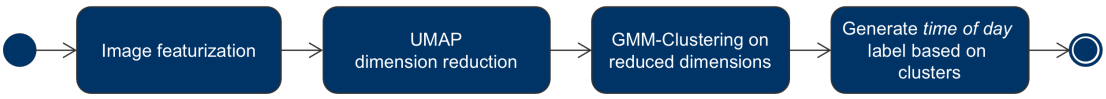

Figure 9: Process to generate a *time of the day* label based on visual image clusters.

Identification of Incorrect *time of the day* Labels The new labels can be used to generate a list of candidate images that are probably incorrectly labeled. This list is generated by comparing the original labels with the cluster-based and elevation of the sun-based labels. The original *time of the day* label was visually inspected, and the vast majority of the labels were verified as correct. Therefore it is taken as the baseline label. Images are categorized as incorrectly labeled candidates if both generated labels differ from the original. The complete relabeling process is shown in Figure 10.

The ruleset shown in Table 6 is applied to reduce the candidate list. All conditions have to be met by an image, otherwise the image is removed from the candidate list.

With rules 1 and 2, dawn/dusk or undefined labeled images are removed. The dawn/dusk transition period is relatively short. Therefore, a slight inaccuracy in the timestamp or location can render the generated label incorrect. In addition, the clustering-based label only distinguishes between daytime and night due to the two-clusters formed by the data.

Rule 3 removes tunnel images since it is impossible to verify a *time of the day* label without exposure to natural illumination.

Rule	**Description**
1	Original time of the day label is "daytime" **OR** "night"
2	Computed elevation of the sun label is "daytime" **OR** "night"
3	Scene type **is not** "tunnel"

Table 6: Ruleset to remove ambiguous images from the candidate list of incorrectly labeled time of day images. All conditions have to be matched by an image to stay on the candidate list.

Finally, the remaining candidate images are transformed to the LAB color space. The first channel L provides the lightness value of a pixel $(l_{i,j})$ and the latter two channels A and B provide color values. The average lightness $\bar{l}$ per image is computed as the mean value over all pixels per image. For an image with width w and height h, $\bar{l}$ can be computed as shown in Equation 23. Afterward, the images are sorted by $\bar{l}$. Visual inspection is used to define cut-off values $\bar{l}_{\text{crit,d}}$ and $\bar{l}_{\text{crit,n}}$. Daytime images of the candidate list with $\bar{l} < \bar{l}_{\text{crit,d}}$ are relabeled to night images. Night images with $\bar{l} > \bar{l}_{\text{crit,n}}$ are relabeled to daytime images.

$$\bar{l} = \frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} l_{i,j} \tag{23}$$

Scene type The second image domain that was validated is the scene type. Again, the approach of identifying discrepancies between the originally provided and an additionally generated label is used. The applied method is based on the spatial location to identify the scene type. An existing implementation to query an OpenStreetMap (OSM) API, based on a (latitude, longitude) tuple is adapted

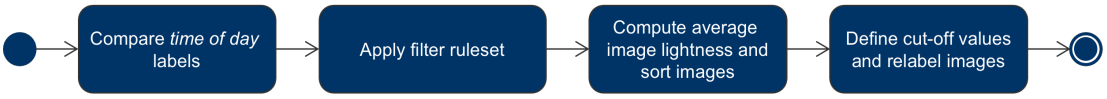

Figure 10: Process to relabel daytime/night images. The labels are compared, and the candidate list is reduced based on predefined rules. Images are sorted by their perceptual lightness, and cut-off values are defined visually.

[38]. For each image, the street type of the closest street within a distance of $20m$ is returned. In addition, the information if a tunnel or bridge is found within this radius is returned. To also identify scene types like gas stations or parking lots, *amenities* are searched within the distance of $200m$. If one is found, the image is flagged accordingly. The process is shown in Figure 11.

The generated data points are compared with the existing labels to validate the scene label. Unfortunately, the street type categories differ between BDD100K and OSM. A mapping as shown in Table 10 is required, to match the street/scene types. The class *city street* which is the largest class in the scene domain (see Figure 18) can not be mapped since no corresponding class exists in the OSM data tags. This gap renders automated validation useless. Therefore, the efforts are focused on the manual verification of minority classes.

Gas Stations By applying the rules shown in Table 7, one can generate a list of potential gas station images such that they can be inspected visually. Rule 1 selects all images for which a gas station was found in proximity. Rule 2 selects those images recorded on a *service* street, which is often chosen as the street type for links between gas stations and highways [35]. Rule 3 matches those BDD100K scene types that could be easily confused with gas stations.

Rule	Description
1	OpenStreetMap *gas station* found with 200m radius
2	OpenStreetMap street type is *service*
3	BDD100K scene type is *undefined* **OR** *parking lot*

Table 7: Filter-ruleset to identify gas station images. All rules have to be matched at once.

Tunnel Incorrectly labeled tunnel images can be identified similarly to gas station images. A filter ruleset is applied to generate a candidate list of potential tunnel images. Rule 1 selects the images that are flagged as *tunnel* based on the OSM results. Rule 2 selects all images for which a mismatch in the two labels is found. The resulting list of images is inspected manually.

Rule	Description
1	OpenStreetMap *tunnel* found within 20m radius
2	BDD100K Scene type is $\neq$ *tunnel*

Table 8: Filter-ruleset to identify tunnel images. All rules have to be matched at once.

Parking Lot Incorrectly labeled parking lot images are identified with the same approach. The filter rules can be found in Table 9. Rule 1 selects only those images that are flagged as recorded in proximity to a parking area. Rule 2 selects the images not yet labeled as a parking lot and can be potentially confused with parking spaces.

Rule	Description
1	OpenStreetMap *parking lot entrance* **OR** *parking space* found within 200m radius
2	Scene type is *undefined* **OR** *tunnel* **Or** *gas stations*

Table 9: Filter-ruleset to identify parking lot images. All rules have to be matched at once.

Data-tag	OpenStreetMap Keys	BDD100K Scene
highway	motorway, trunk, primary, secondary, tertiary	highway
highway	residential	residential
highway	service	-
tunnel	yes, avalanche protector	tunnel
amenity	fuel, charging station	gas stations
amenity	parking, parking space, parking entrance	parking lot

Table 10: Possible mapping from OpenStreetMap keys to BDD100K scene types.

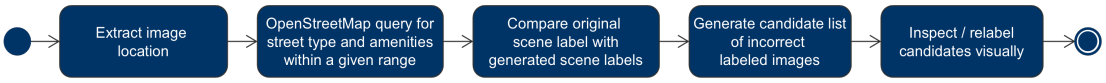

Figure 11: Relabeling process of scene images. The image location is extracted and compared to available OpenStreetMap information. Different rules are applied to identify possible scene types like gas stations, tunnels, or parking lots. Eventually, the images are inspected and relabeled manually.

3.2 Image Domain Class Relevance Test

This experiment is designed to answer the first research question: Which image domains are relevant for the training data of an object detection model? Therefore, the image domains relevant for the performance of an object detection model shall be identified. Since a domain as a whole is very diverse, the domain classes are investigated directly. It is assumed, that images within one IDC differ visually from images that are not part of this specific IDC. Therefore, the first step is to measure the magnitude of these visual differences. Afterward, the impact of the IDCs is tested in trained object detection models and compared against baseline performances.

3.2.1 Experiment Design

The relevance of an IDC is tested by the impact of the presence/absence of the IDC. This idea follows the question, does it have an impact on the model performance

if the IDC is represented in the training data? The hypotheses are:

- H_0 = Absence of the selected IDC in the training dataset **does not have** an impact on the model performance.
- H_1 = Absence of the selected IDC in the training dataset **does have** an impact on the model performance.

The end to end process is shown in Figure 12. The process steps will be explained one by one.

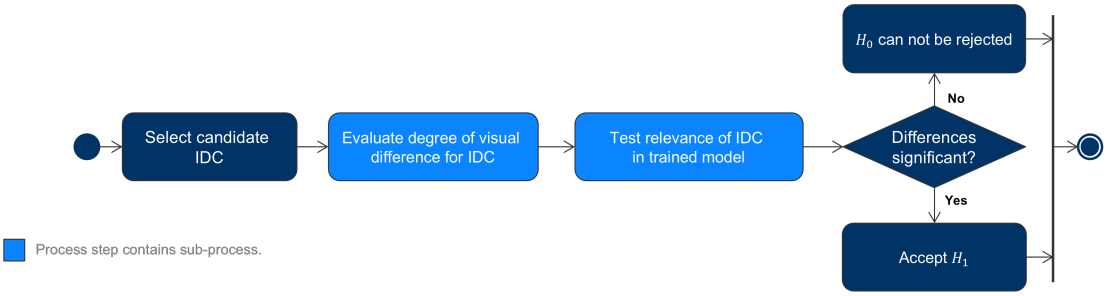

Figure 12: Process to identify relevant image domain classes. A selected image domain class is tested for visual differences to obtain a first indication. Afterward, the class is tested by comparing the model performances of trained object detection models.

Evaluation of Visual Differences The first step, after selecting a candidate IDC, is to evaluate visual differences. This is done by comparing the feature vectors of the images, which are generated by applying the image featurization technique explained in the preliminaries. The images are split based on one IDC. Two groups are formed (1) images part of the IDC, and (2) images not part of the IDC. The vector means of both groups are compared by performing Hotelling's T^2 test. The resulting T^2 test statistic is a distance measure for the two mean vectors. Since the feature vectors encode visual attributes of the images, the T^2 statistic is interpreted as the mean visual difference of the image groups and, therefore, as the visual difference introduced by the IDC. The image domains not tested in the given test are fixed to a single class in both groups (the majority classes for the data containing images of the investigated IDC) to avoid the confounding impact of other image domains. Therefore, only the investigated IDC varies between the groups. For example when testing the IDC *night*, the two groups are different

regarding the *time of the day* domain. One group only contains *night* images, the other group only contains *daytime, dawn/dusk, undefined* images. The other domains are fixed for both groups. *Scene* is fixed to *city street* and *weather* is fixed to clear, since those are the majority classes for images with the time of the day label *night*.

Since the required assumptions of normal distributed data and equal covariance matrices can not be guaranteed, two precautions are taken. First, the test is performed on a sample size such that the large sample approximation holds. Second, the samples of both groups are generated such that $n_1 = n_2$. Therefore the test can be considered robust against inequalities of the covariance matrices. Further on, the test is performed in a permutation-based manner. This approach ensures a non-parametric p-value since it is generated based on data permutations.

As explained in the Preliminaries, the required sample size to assume the normality of the mean vectors depends on the number of variables. Principal Component Analysis (PCA) is applied to reduce the required sample size. This method is chosen since it only performs a centered rotation of the coordinate system and ensures to maintain the most of the variance with the minimum number of dimensions. However, it does not distort the distances in the data [15]. After applying the transformation, a proposal for the number of variables is generated by identifying the knee in the *explained variance ratio*-plot. The proposed number of variables is used to determine the required sample size based on the approach, shown in Equation 20.

Once the preparation regarding dimension reduction is completed, the IDCs are evaluated using Hotelling's T^2 test. This subprocess is shown in Figure 13.

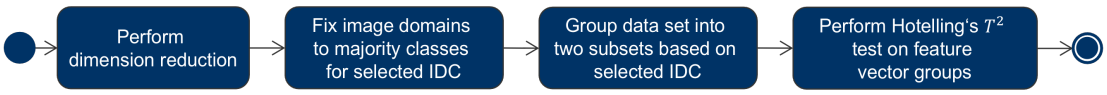

Figure 13: Subprocess to identify visual differences. The difference is evaluated by performing Hotelling's T^2 test on reduced feature vectors.

The process can be repeated for all IDCs and the resulting T^2-values are used to prioritize the IDCs for the next experiment steps. The validity of the obtained prioritization and whether this step can be used to identify a cut-off point is investigated in the evaluation section (see Section 4.2.3).

IDC Relevance Evaluation Method For each relevance test, two sets of models are compared. One set is trained on datasets including the investigated IDC, and the other set is trained on datasets excluding the investigated IDC. Both model sets are validated on unseen validation data, including the investigated IDC. Afterward, the performance is compared. An IDC is said to be relevant for the object detection model if the performance of the two model groups significantly deviates on the validation subset that consists of images of the same IDC.

The subject of interest in this experiment is the impact of a specific IDC. Therefore in an ideal world, both models would use identical training datasets, with the only difference being the presence/absence of the investigated IDC in one dataset. Of course, this is not possible with actual field-recorded image data. However, multiple aspects must be monitored to match this requirement as closely as possible.

1. The investigated IDC should be represented in a significant size (in one of the two datasets).
2. Both datasets need to be of comparable size.
3. Both datasets should contain the same images (except for the investigated IDC).
4. Both datasets should have a comparable number of instances per object class.
5. Both datasets should represent the diversity of the original dataset.

To draw further conclusions about the impact across the IDCs, the share of the investigated IDC in the sample should be comparable across the experiments.

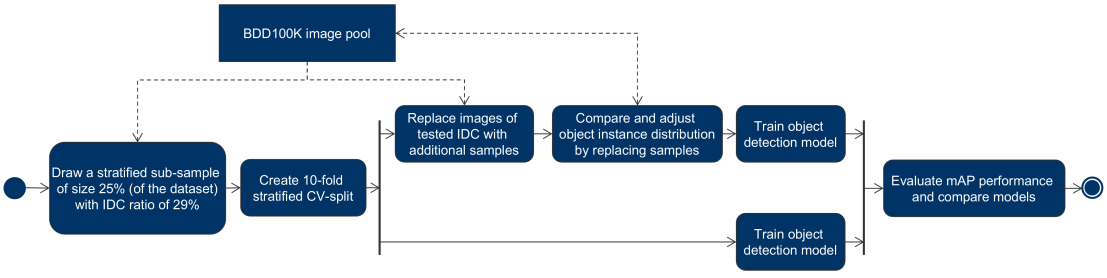

Figure 14: Subprocess to test image domain classes for relevance. Models are trained on different data splits that either include or exclude the tested class. Afterward, the model performances are compared.

To stick with those requirements, the process shown in Figure 14 was defined. The ratio of the investigated IDC r_{IDC} is fixed to 29% and is comparable over multiple experiments. The drawn sample represents the diversity of the original dataset due to stratified sampling. The used strata are the LP transformed multi-class labels, which encode the image domain classes. The sample size ratio r_{sample} of 25% of the dataset ensures enough flexibility for later adjustments. The ratio of the investigated IDC and the sample size ratio r_{sample} interact with each other and depend on the IDC size n_{IDC} and the dataset size $n_{dataset}$. Equations 24 and 25 provide bounds for the parameter choices.

$$r_{sample}\ r_{IDC}\ n_{dataset} \leq n_{IDC} \tag{24}$$

$$(1 - r_{sample}(1 - r_{IDC}))n_{dataset} \geq n_{IDC} \tag{25}$$

After the sample is drawn, a 10-fold stratified CV split is created. This allows for multiple comparisons of the models, which is necessary to obtain a robust result. For the validation strategy and the number of folds contradictory requirements exist. On the one hand, the number of folds should be maximized to obtain many comparable measurements, which can help to obtain a more robust estimate of the performance difference. On the other hand, since the size of the data used for all folds is fixed by r_{sample}, using more folds reduces the size per fold. This can increase the variance for each estimate since the measurement is taken on a smaller validation set. No ideal applicable validation strategy exists. Based on Bouckaert and Frank, a 10-fold split was chosen. While the 10-fold split introduces a variance bias due to overlapping training folds [3] other strategies such as the 5x2-split have shown lower replicability [6]. The preferable validation methods based on the literature are inapplicable in real-world deep-learning scenarios with limited resources. E.g., the corrected 10 $\times$ 10-CV and 100$\times$resampling method lead to a training time increase by factor 10.

After step two, the process splits into two branches. In the lower branch, the 10-fold CV split is used as is, to train ten different models. In the upper branch, the folds are adjusted. All images of the tested IDC are replaced in a stratified manner with images from other classes of the same image domain. These samples

are drawn from the remaining pool of unused images.

Since the preceding step can affect the object instance distribution of the sample, the resampled part is again adjusted, by replacing images one by one to approximate the original object instance distribution. The images are selected based on the instance count delta. The object class that deviates the most (relative measure) from the count in the unmodified fold is adjusted by replacing images with only a few instances of the class against images with many or vice versa. The process is performed image-wise, and the instance count deltas are updated after each iteration until the deviation is smaller than a given ϵ. In the applied method, the parameter $\epsilon = .10$ is chosen such, that extreme object instance deviations are reduced, while some flexibility is kept to avoid modifying the complete sample and therefore to change the stratification too much. The pseudo-code is presented in Algorithm 1.

These steps ensure adherence to three requirements: First, comparable fold size. Second, similar object instance distribution, and third comparable image domain distribution of both 10-fold CV splits. An example of the sampling process for the IDC *night* is provided in the appendix in Figure 34. Finally, both model sets are trained followed by validation on the unmodified validation folds. The mAP is used as performance indicator and computed for multiple validation fold subsets. The decision about the relevance of an IDC is taken based on performance differences on validation images of the investigated IDC. The computed mAP measures are compared visually and by applying Wilcoxon's Signed Rank Test at significance level $\alpha = .05$. This test is chosen since the normal distribution of the mAP measure can not be assumed, and due to the small sample size of only ten comparisons, one cannot use the large sample size approximation. In addition, the measurements are clearly paired since the models are evaluated on the same ten validation folds (and trained on very similar training sets).

When executing the experiments, specific training parameters must be chosen, and a model must be selected. YOLOv5 includes a whole variety of scaled model architectures from *n* (*nano*) up to *xl* (*extra-large*). The model chosen for the experiments is the YOLOv5s-P6. The size *small* was chosen since its performance characteristics offer a good trade-off between performance and resource consumption. Its performance is between *nano* and *medium*, while its training time is only

Algorithm 1 Sampling Algorithm to adjust the instance distribution

```
procedure adj_folds(folds_incl_idc, folds_excl_idc, obj_classes, unused_imgs)
    eps ← .10                                   ▷ Threshold (can be lowered iterative)
    classes ← obj_classes                       ▷ Defined object classes
    n ← number_of_folds
    i ← 0
    img_pool ← unused_imgs                      ▷ Remaining image pool
    while i < n do
        fold_incl ← folds_incl_idc[i]
        fold_excl ← folds_excl_idc[i]
        max_inst_delta, max_delta_class ← comp_delta(fold_incl, fold_excl)
        while |max_inst_delta| > eps do
            if max_inst_delta > 0 then          ▷ Instances missing in fold excl. IDC
                sort(fold_excl, by=max_delta_class, how=ascending)
                sort(img_pool, by=max_delta_class, how=descending)
            else                                ▷ Too many instances in fold excl. IDC
                sort(fold_excl, by=max_delta_class, how=descending)
                sort(img_pool, by=max_delta_class, how=ascending)
            end if
            new_image ← img_pool[0]
            img_pool ← img_pool[1 :] + fold_excl[0]
            fold_excl ← fold_excl[1 :] + new_image
            max_inst_delta, max_delta_class ← comp_delta(fold_incl, fold_excl)
        end while
        i ← i+1
    end while
end procedure
```

Algorithm 2 Instance Delta Computation

```
procedure comp_delta(fold_incl, fold_excl, classes)
    max_inst_delta ← 0
    max_delta_class ← None
    for cl in classes do
        cur_delta ← fold_incl[cl].count() − fold_excl[cl].count()
        cur_delta ← cur_delta/fold_incl[cl].count()
        if |cur_delta| > |max_inst_delta| then
            max_inst_delta ← cur_delta
            max_delta_class ← class
        end if
    end for
return (max_inst_delta, max_delta_class)
end procedure
```

slightly increased in comparison to *nano* (see Appendix in Table 17 and Figure 35). The models are trained from scratch for 85 epochs to balance performance improvement with training time. The resulting model of the last epoch is taken. The P6 model is chosen since the input resolution for these models is 1280x1280px. Therefore, the images can be used in their native resolution. Other available model groups process 640x640px images, which lead to a significant performance decrease during tests.

Details about the training setup are provided in the appendix. The hardware setup of the machine that was mainly used, can be found in Table 14. The customized hyperparameters can be found in Table 15.

3.3 Domain-based Data Sampling Test

After identification of the relevant IDCs, the impact of image domain based sampling shall be measured to answer the second research question: What impact does image domain-based stratified sampling of a dataset have on the performance of an object detection model? Therefore, again, the performances of trained models are compared with each other.

3.3.1 Experiment Design

As a preparatory step, the image domain classes are relabeled to reflect the gained knowledge about the relevance of the classes. Relevant IDCs remain as individual classes, while non-relevant IDCs are grouped together as one class within the image domain. IDCs with too few samples are kept since no decision about their relevance was made.

For all sampling methods that are going to be compared, a 10-fold CV split is formed. The used part of the data is increased to 50% since, unlike to the first experiment, no dataset adjustments have to be made post-sampling.

The baseline performance (*Baseline*) is obtained by ignoring the existence of IDCs. The naive sampling method of simple random sampling is applied to generate the training folds.

The first alternative sampling approach (*A1*) is the straightforward application of the approach used in the previous experiment: LP-based stratified sampling.

The second alternative (*A2*) applies the same approach but also tries to maximize the object instances to isolate the impact of IDC based sampling from instance count deviations. The maximization is done by selecting the images for each strata based on their instance count.

The third alternative *A3* is based on a different sampling approach, the IS, which also ensures stratified sampling but is less restrictive. The advantage is that sparse clusters are avoided at the cost of not adjusting for inter-image-domain dependencies. The testing process is shown in Figure 15

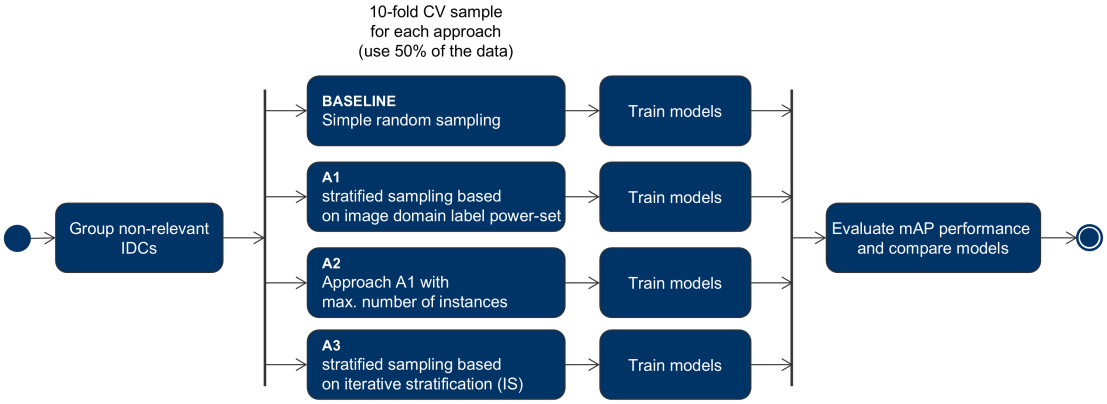

Figure 15: Process to evaluate the impact of image domain based sampling. Three stratified sampling strategies are compared against random sampling in their impact on the model performance.

To ensure that all models are evaluated on unseen data, only the original training set of the BDD100K is used to sample the training folds. Therefore, the validation dataset is kept for model comparison. Based on this validation dataset, multiple subsets are drawn with random and stratified sampling strategies. The used validation datasets are:

- Complete validation dataset
- Random sampled subset of the validation dataset
- LP-Stratified sampled subset of the validation dataset based on the IDC distribution in the validation dataset
- LP-Stratified sampled subset of the validation dataset based on the IDC distribution in the training dataset
- IS sampled subset of the validation dataset

The models of approach *A1* to *A3* are compared against the *Baseline* models. The measurements can not be paired since the training sets are independent for each model group and the validation datasets are identical for all models. The performance is compared visually and by applying Wilcoxon's Rank Sum test. $\alpha = 0.05$ is chosen as significance level.

The models are all trained with identical training parameters which can be found in the Appendix in Table 16. The same model (YOLOv5s-P6) as in the previous experiment is used. Due to a changed sample size and a changed training objective, in comparison to the IDC relevance tests, the models are trained for 150 epochs. This parameter was defined based on training experiments conducted in advance.

4 Evaluation

First, the identified characteristics and issues of the dataset will be explained. Afterward, the performed experiments are evaluated. Finally, this chapter concludes with the limitations of the work identified during the design and execution of the experiments.

4.1 BDD100K

The BDD100K results and evaluation section consists out of the analysis of the dataset characteristics and the performed cleansing and relabeling steps.

4.1.1 General Findings

Some first findings were made while preparing the *Label Database*. Since each image label does not only contain bounding boxes but also additional information such as scene type, weather, or time of the day, each image should be accompanied by a label. However, this is not the case. For 137 images in the training dataset, no labels are provided. In addition, one info file is missing. Since the actual labeling rules are unknown, the images were removed from the dataset. Therefore only 79 862 of 80 000 images remain in the combined training and validation dataset.

For the remaining images, the object instance distributions were compared. Figure 16 provides an overview of the object count for all object classes in all sizes. One can see that only ten images do contain zero objects. The peak of this right-skewed distribution is located at 15 object instances (3354 images), and the median is located at 17 object instances.

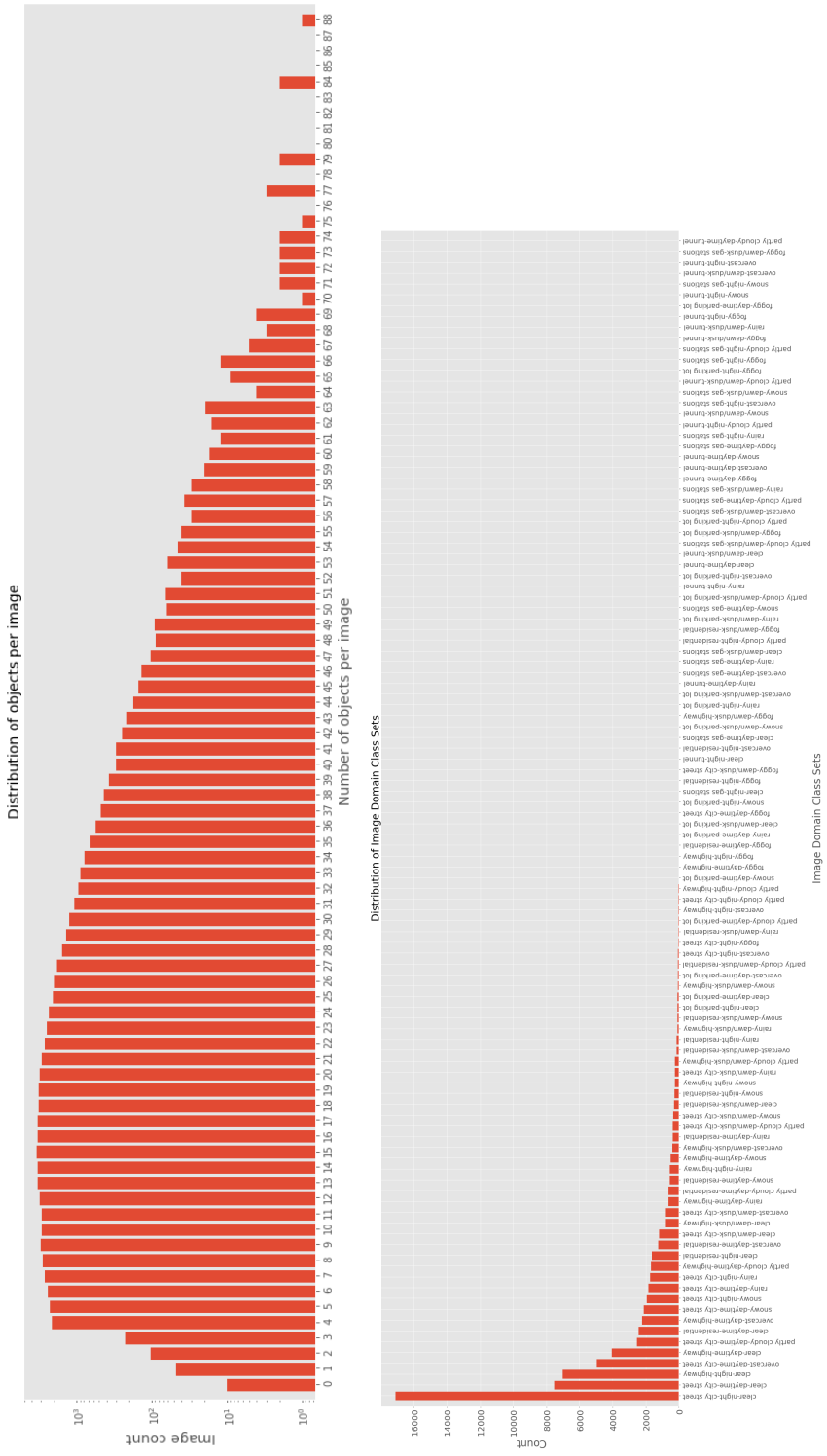

Figure 16: Distributions for images from the BDD100K dataset. Left: Distribution of the object count per image, counted for all object classes and sizes. Right: Distribution of the LP-transformed image domain class sets.

Class	**per img median**	**per img mean**	**total**
bicycle*	0	0.10	8075
bus	0	0.17	13528
car*	9	10.06	797651
motorcycle*	0	0.04	3469
other person	0	0.00	211
other vehicle	0	0.01	885
pedestrian*	0	1.32	104644
rider*	0	0.07	5173
train*	0	0.00	143
trailer	0	0.00	73
truck*	0	0.40	31844
traffic light*	1	2.69	213481
traffic sign*	3	3.42	271495
total	**17**	**18.30**	**1450672**

Table 11: Object classes and instance counts per class for all sizes. (*) indicates that a class is considered in performance evaluation.

When considering the instance distributions at the object class level, one can see in Table 11 that the number of instances per image heavily depends on the class. Multiple cars are commonly seen in one image, but most of the other classes are rare. The most extreme difference among the classes considered relevant for evaluation by the BDD100K authors is between *car* and *train*. In total, 797 651 car labels and only 143 train labels are found within all training and validation images.

Within the existing info files, 80 images are missing location data. These images remain unverified in those steps where location data is necessary (e.g., to compute spatial distances) but are kept in the dataset.

Regarding the location distribution, one must mention that the vast majority of the images was recorded in New York City, with approximately 62 735 images. In Northern California (San Francisco, Berkeley, and Bay Area), 9194 images were recorded. In addition, 6719 images were recorded in Israel / West Bank. These three areas account for 99.3% of the images in the BDD100K dataset. The ground truth for this ratio is 79 197, which is the number of cleaned training and validation images with location data available. The mentioned areas are shown

in Figure 17. The remaining images are recorded in other regions of the United States of America.

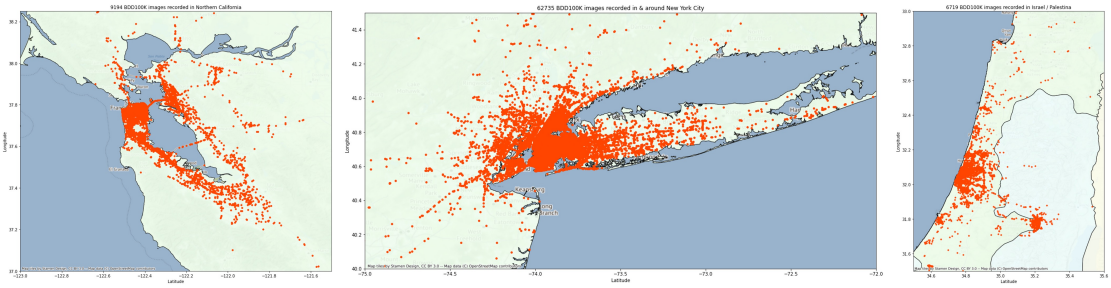

Figure 17: Main recording areas of the BDD100K training and validation videos. Left to right: San Francisco including Bay Area and Berkeley (9194), New York (62 735), Israel and West Bank (6719)

4.1.2 Image Domain Distribution

Figure 18 shows the distribution of the three image domains. The *time of the day* domain is distributed among three classes, plus one minority class *undefined*, which contains examples where no assignment can be made, e.g., in the tunnel. Daytime and night are within the same magnitude, while the transition phase (dawn/dusk) only constitutes a minor part.

The *scene* domain distribution is heavily skewed. *City street* is the majority class with a share of more than 62% within the domain. Highway and residential represent approximately 25% and 12% of the data, while the other four classes only represent approximately 1% of the data.

Also the *weather* domain distribution is skewed, but not as extreme as the *scene*. The class *clear* represents approximately 53% of the data. *Overcast* and *undefined* account for 13% and 12% of the data, while *snowy*, *rainy* and *partly cloudy* are in the range of 7.5% to 8%. *Foggy* is visible on approximately 2‰ of the images.

Besides the independent image domain distributions, one can also look at the different combinations of the three image domains. Under the application of the LP transformation, one can investigate the distribution for the concatenated label sets. Figure 16 shows the distribution of the transformed IDC sets. One needs to

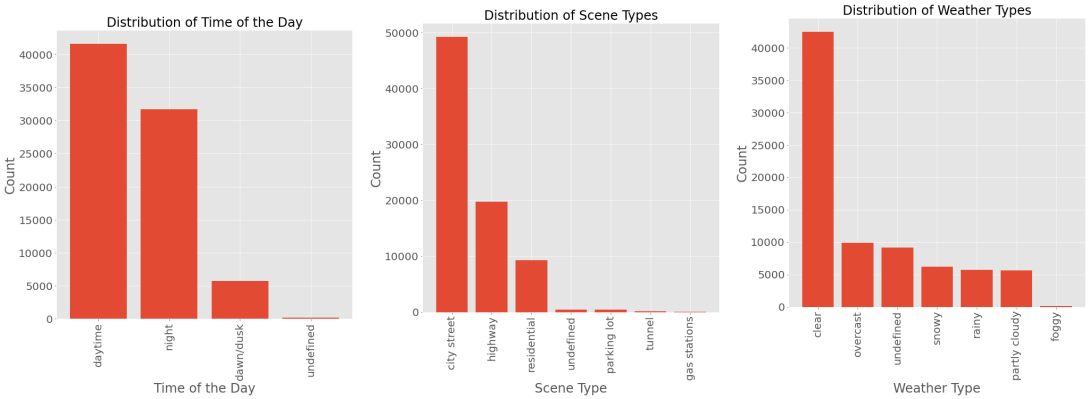

Figure 18: Image domain distribution of the BDD100K training and validation images. Left: Time of the day, center: Scene, right: Weather

mention that the *undefined* classes have been removed in the figure since they do not specify a clear state of an image domain but are a loose bin for uncategorizable images. The plot shows that a few ubiquitous class combinations represent the majority of the data. The four most common class sets represent approximately 52.5% of the dataset, and the eleven most common class sets represent 77.1%. On the other hand, the 28 rarest class sets in the dataset are assigned to less than ten samples each. Twenty-four class sets are not represented at all in the dataset.

Even though the BDD100K dataset is diverse compared to other driving datasets and shows multiple non-standard/adverse situations, most of this dataset is contributed by a small subset of the available states. For example, in the four most common class sets, three sets show clear weather, the scene type is city street in three cases, and only day and night are represented.

Important to name is that not all IDCs are independent of each other. For example, deciding whether the sky is cloudy or overcast during nighttime is difficult. Also, the weather situation in a tunnel is most often not defined. This reduces the set of possible combinations, or at least it biases the distribution of the combinations because some combinations can be observed only in rare situations. For example, the weather is still perceivable when driving in an avalanche protector.

4.1.3 Duplicate Images

The process shown in Figure 7 is applied to the dataset to identify and remove duplicate or dependent images. The thresholds of the ruleset provided in Table 5 have been determined based on a visual inspection of the image set. By applying this process, 492 images were removed from the training dataset and 94 images from the validation dataset. Thereof only three image pairs show actual duplicates with a cosine similarity of 1. Figure 19 shows two of these examples.

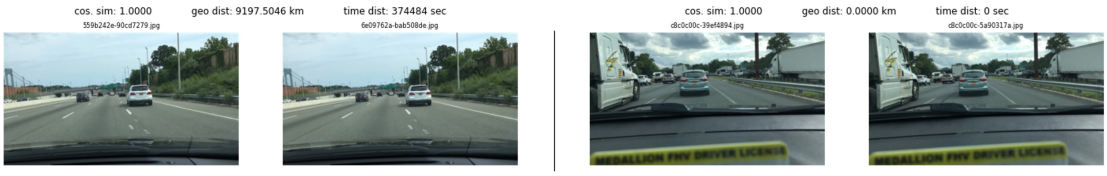

Figure 19: Duplicate image pair examples within the BDD100K training and validation images.

Besides duplicate images, also dependent image pairs have been identified. Examples are shown in Figure 20. One can see that the term *dependent image* is fuzzy. While some pairs show nearly identical images (top row), others are recorded at the same location, but the situation changed (center row). The examples shown at the bottom row are recorded at different locations but in similar situations.

The effectiveness of this process was iteratively verified by inspecting the remaining image pairs and fine-tuning the ruleset. Since the most similar remaining image pairs show independent images, one can conclude that the heavily dependent pairs have been identified. One must mention that the process's pivotal point is the pairwise image matching based on their cosine similarity. Duplicates and multiple duplicates of the same image are detected due to this similarity-based matching and the iterative approach. However, if dependent images are visually more different from their dependent image than another independent image, one cannot detect this dependent pair.

After removing the identified images, the dataset and label database was reduced to 79 277 images.

Figure 20: Dependent image pair examples within the BDD100K training and validation images. Top: nearly duplicates, center: similar location but different situation, bottom: different location but similar situation.

4.1.4 Label Validation

The second dataset modification step is aimed at the quality improvement of the IDC labels. Validation processes are proposed for two out of three image domains: *time of the day* and *scene type*. Validation of the *weather* label was, in coordination with the supervisor at FZI Berlin, excluded from the scope of this work.

Time of the Day Labels The necessary process steps to validate the *time of the day* labels, are shown in Figures 8, 9 and 10. The steps are (1) computation of the *time of the day* label based on the elevation of the sun, (2) computation of the cluster-based *time of the day* label, and (3) application of filter rules, label comparison and relabeling. As explained in the methods section, only the daytime and night labels were validated since the transition phase dawn/dusk is present only during a short period of the day. Relabeling them might be prone to errors since small inaccuracies in the timestamp, spatial location, or sun elevation estimation could already lead to a wrong label suggestion. In addition, the clustering process only identifies two clusters (daytime and night). Therefore, no cluster-based label could be generated for the dawn/dusk phase. The clustering results, including the

relabeled images, are shown in Figure 21. In total, 152 images originally labeled as daytime were identified as actual night images, and 251 images originally labeled as night were identified as actual daytime images. Figure 22 shows examples of incorrect labeled images.

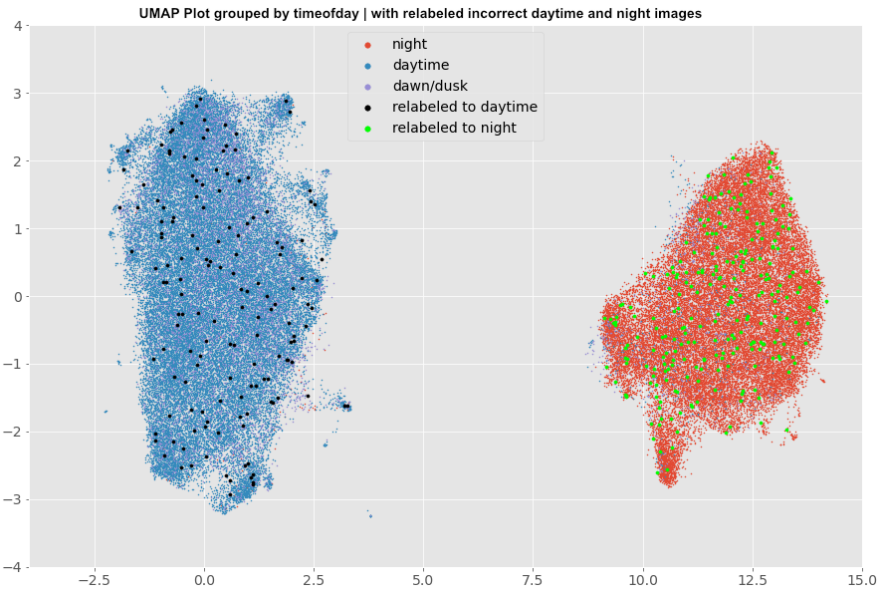

Figure 21: UMAP-reduced feature vectors of the BDD100K training and validation images, color based on time of the day label. Relabeled images are highlighted in a different color.

Instead of applying the automated filter ruleset, which is used to increase the precision of this method, one could also manually verify the candidate list of suspicious images, which is generated based on differences between the computed labels and the original label. In this manual approach, one could also verify dawn/dusk labels. In 7500 cases the elevation of the sun-based generated label is different from the original. This would reduce the manual inspection effort significantly since only approximately 9.4% of the total image dataset would need to be inspected. However, to do so, clear rules must be provided about distinguishing between dawn/dusk and daytime or night.

Scene Labels To identify incorrect scene labels, the process presented in Figure 11 is applied for the classes: gas station, tunnel and parking lot.

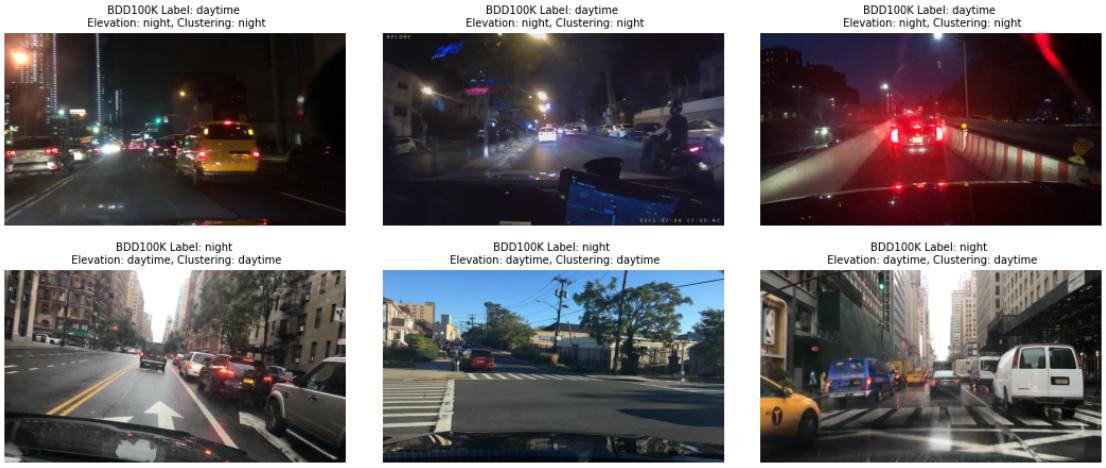

Figure 22: Example images with the incorrect time of the day label. Top row: incorrect daytime label, bottom row: incorrect night label.

Gas Stations In total, only 33 images are labeled with scene type *gas stations*. one can find four additional samples by applying the filter rules shown in Table 7. After visual inspection of the candidate images, four examples are relabeled to *gas stations*. Distinguishing a gas station image from a parking lot image is complex. E.g., the top center image in Figure 23 shows a parking space at a gas station. A clear ruleset needs to be developed to reduce ambiguity.

Tunnel 44 incorrectly labeled images were found that actually show a tunnel scene by applying the filter rules of Table 8. This is approximately 28% of the existing tunnel images in the BDD100K dataset. Therefore either the label quality regarding *tunnel* is poor compared to other evaluated IDCs or the applied manual method is superior over the more automated processes. Figure 23 (center) shows examples of incorrect labeled tunnel images.

Parking Lot Seven images not labeled as parking lot images were identified by manual review of the candidate list after applying the filter rules shown in Table 9. The problem with identifying parking lot images is the ambiguity of the label. It is difficult to visually identify whether the car is actually parking on a static image. The bottom left example of Figure 23 shows a difficult labeling situation. The car is (probably) still on a public road, but a parking lot is visible

in the image. Which label should be assigned? Therefore, the categories need to be described such that clear decision rules exist.

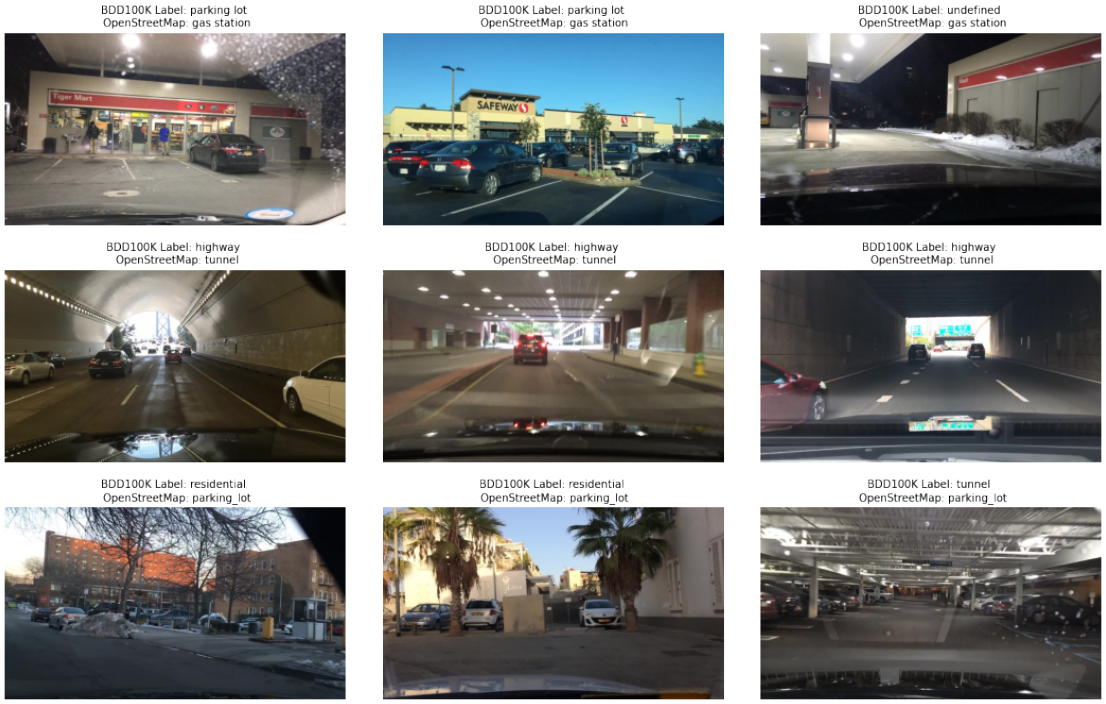

Figure 23: Example images with incorrect scene labels. Top row: incorrect gas station images, center row: incorrect tunnel images, bottom row: incorrect parking lot images.

4.2 Image Domain Relevance

The following part describes how the image domain classes were tested for relevance. The idea behind this test is to identify whether an image attribute that one can assign to images is relevant for the performance of an object detection model. This shall help to evaluate which image characteristics must be considered when designing or extending an object detection image dataset. The first research question shall be answered with this test: Which image domains are relevant for the training data of an object detection model? The relevance of an IDC in the context of this work is defined as follows. An IDC is relevant if its absence during training leads to a significant performance difference on images of the same IDC.

The designed process tests the effect of the absence of a specific IDC on the model performance. With the conducted comparison, the performance differences between models which did see samples of the IDC during training and models which did not see any samples of this IDC are investigated. The overall test steps are shown in Figure 12.

The performance measure used to compare the models is mAP for IoU$\geq$ 0.5. This measure is commonly used in object detection. It reflects precision and recall and is also sensitive to performance differences in different classes. However, it also has some shortcomings. In the context of automated driving, not single metrics need to be optimized, but the car's behavior needs to be safe and reliable. Therefore minor differences in the mAP performance metrics due to class confusion or slightly decreased IoU that do not lead to changes in the behavior are not that critical. Since the IoU threshold is set to 0.5, which is at the lower end of the scale, this is to some extent reflected.

4.2.1 Visual Differences – Test Results

A pre-assessment can be used to generate the first indication about visual differences and, therefore, indicate the relevance of an IDC. It will be evaluated whether the results of this test are consistent with the actual performance differences measured in the IDC relevance tests. The sub-process to identify visual differences is shown in Figure 13.

As explained, the required sample size depends on the number of variables included in the test. The multivariate approximation of the large sample size domain is based on the univariate case, and its behavior when applying it to hundreds of variables is not known. To reduce the number of dimensions, one can perform PCA on the 512 feature variables of the image vectors. The optimum trade-off between explained variance and added dimensions can be found by identifying the knee of the explained variance ratio plot, shown in Figure 24.

The first four components are used in the visual difference analysis in the given case. Equation 20 is used to determine the minimum number of samples with $p = 4$. Based on Equation 27 one can look up the minimum number of samples to apply the large sample size approximation in [5]. For $n \geq 90$ this condition is

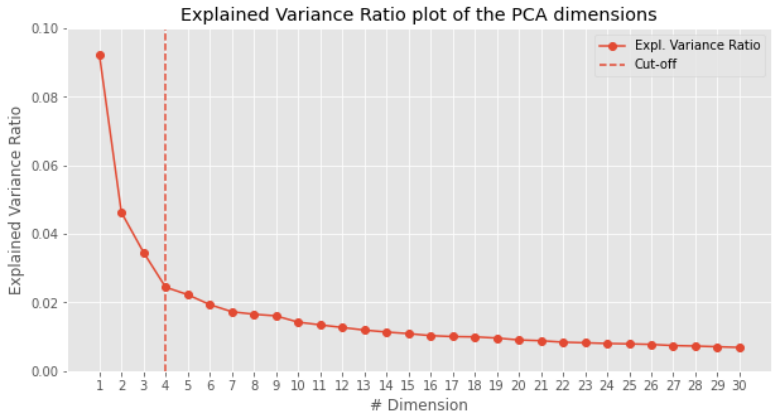

Figure 24: Explained variance ratio for the first 30 components of the PCA transformed image feature vectors. The knee is found by visual inspection.

fulfilled.

$$1.086 \overset{!}{=} \frac{T^2_{0.05,4,n-1}}{T^2_{0.05,4,\infty}} \Leftrightarrow 1.086 \times T^2_{0.05,4,\infty} \overset{!}{=} T^2_{0.05,4,n-1} \tag{26}$$

$$\Rightarrow 1.086 \times 9.488 = 10.304 \overset{!}{=} T^2_{0.05,4,n-1} \tag{27}$$

After determining the minimum sample size, Hotelling's T^2 test was performed with the 13 IDCs that fulfill the sample size requirement. Two samples are drawn per test, one with images from the IDC and the other one with images not from the IDC. The two samples are fixed in the other two image domains to isolate the effect of the IDC. This shall help to reduce confounding factors. The results for the IDCs are shown in Figure 25.

Each T^2 value provides a distance measure for the mean vectors of the two groups tested. In the plot, one can see two magnitude levels for the T^2 value. *Night* and *daytime* are the two classes that differ the most from the other classes within their image domain. This is expected since the lighting situation has a dominant effect on the overall visual appearance of an image. Besides that, the other eleven variables only show minor mean differences, while two are not significant at the level $\alpha = 0.05$.

Based on these results, the expectation is, that *night* and *daytime* show the

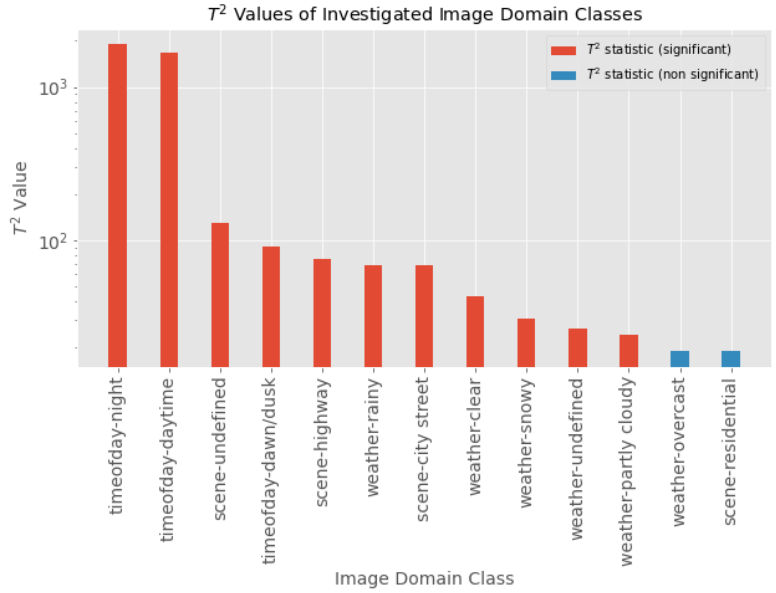

Figure 25: Log scaled plot of the T^2 statistic for all investigated IDCs, ordered by value. The image domains not affected by a test were fixed to the majority classes.

strongest performance difference in the following test, while the other IDCs show smaller but comparable performance difference, except for *overcast* and *residential* for which no significant performance difference is expected.

4.2.2 Image Domain Relevance Test – Test Results

To test relevance of the IDCs, the process shown in Figure 14 is applied. Sampling tests performed upfront showed that, by selecting $r_{sample} = 25\%$ and $r_{IDC} = 29\%$, the lower and upper bound requirements defined in the Equations 24 and 25 can be fulfilled for most of the classes. For twelve out of thirteen classes that were tested for visual difference in the previous step, the requirements derived from Equations 24 and 25: $5748 \leq n_{IDC} \leq 65\,205$ are fulfilled. The tested IDCs are listed in Table 12. The only class that is not represented in sufficient size is the scene class *undefined*.

At first, some important preliminary information is provided, which needs to be considered while interpreting the following results.

- The model, which includes the tested IDC, sees approximately 29% of this IDC in the training data.

- The model which excludes the tested IDC does not see a single image of this IDC in the training data.
- This difference also affects the distribution of the other IDCs of the same image domain in the sample since replacing 29% of the training data with images from other IDCs increases their share in the training data.
- An IDC is relevant if its presence/absence leads to a significant performance difference on images of the same IDC.
- The sampled training datasets are, due to resource limitations, small in the domain of deep learning.

The results will be presented and discussed image domain-wise. All Y-axes are fixed to the same scale to improve the comparison of the effect sizes. For each IDC, the mAP performance is compared on different subsets of the corresponding validation dataset. These are the complete validation fold (*full*), one subset for each IDC of the image domain, and a subset containing all images except those of the tested IDC.

The mAP is computed by averaging the class-wise AP value for IoU$\geq$ 0.5 for all object classes that are present on more than 1500 images in the training dataset. This decision is based on a recommendation of the YOLOv5 authors, which provide this guideline for the training dataset size [34]. This is fulfilled for the object classes *pedestrian*, *car*, *truck*, *bus*, *bicycle*, *traffic light* and *traffic sign*. A summary of the results is provided at the end of this section in Table 12.

Time of the Day The result plots will be explained once in more detail to introduce them to the reader.

All result plots for the image domain *time of the day* are shown in Figure 26. Each plot shows the median differences of the mAP for IoU $\geq$ 0.5. The median is computed based on the paired performance differences between all ten models (10-fold CV) of both model groups. Each plot shows the test result for one of the three tested IDCs. The median mAP difference is plotted to indicate the strength of the effect, while the color encodes whether the result is significant at the significance level $\alpha = 0.05$. Wilcoxon's Signed Rank test is performed to test for differences. This test is chosen since it does not require normal data distribution and is designed for paired measurements. The test assumes independence of the

sample pairs, which is true for the validation datasets. However, it is violated for the training datasets (each fold is used $k-1$ times for training). This violation can bias the variance estimate of the distribution and has to be taken as a limitation. The performance estimate remains unbiased.

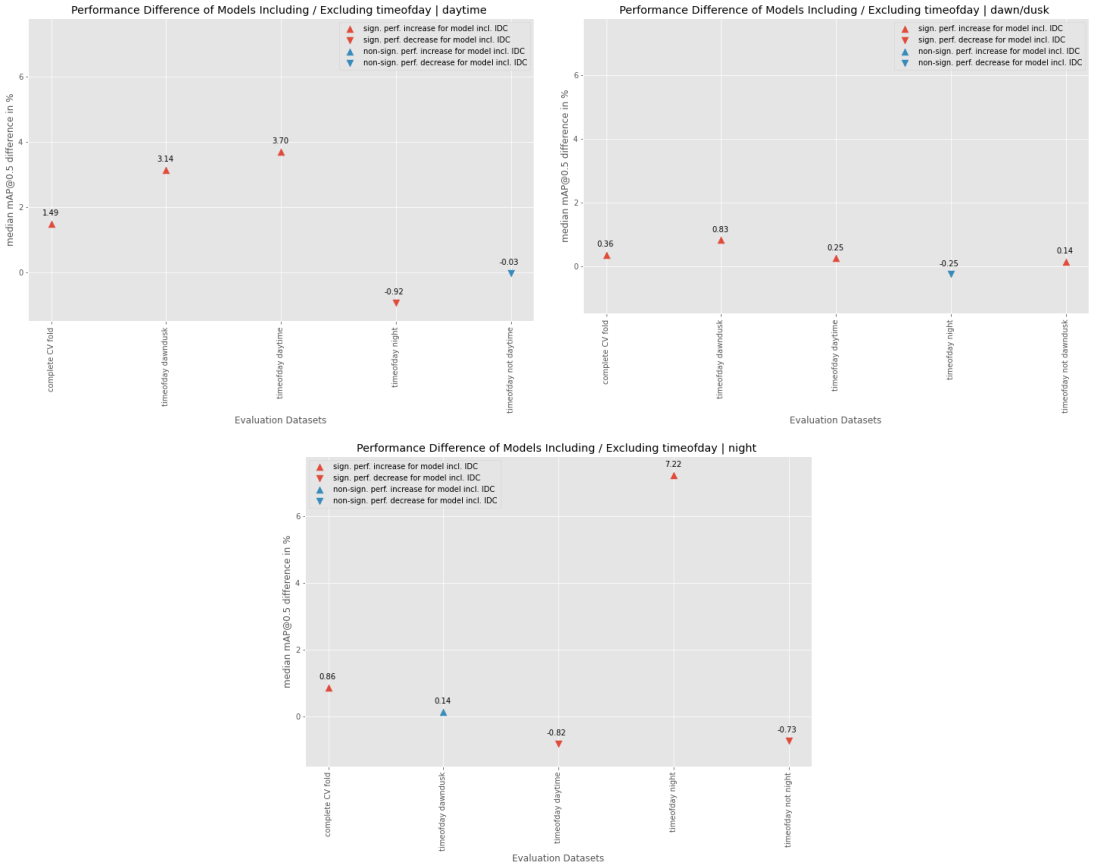

Figure 26: Time of the day relevance test results. Top left: daytime, top right: dawn / dusk, bottom: night

Daytime results: The models trained on images including the daytime IDC perform significantly better on the daytime validation images (+3.70%), therefore *daytime* is a relevant IDC. Further on, also the performance on *dawn/dusk* is influenced significantly positive (+3.14%). The performance on *night* is significantly worse (-0.92%) than for the comparison models which did not see *daytime* during training. The overall performance on *non-daytime* is on par for both model groups. One has to consider that for the model that did not see *daytime* during

training, the share of *night* is larger than for the model that saw *daytime* in the training data. Therefore the positive effect on *dawn/dusk* can be due to a similarity between *daytime* and *dawn/dusk*. In contrast, the negative effect seen for *night* can be because the comparison model saw more night images during training.

Dawn/Dusk results: The models which saw dawn/dusk images during training performed significantly better on dawn/dusk images (+0.83%). Therefore *dawn/-dusk* is relevant. Further on, the performance on *daytime* is also significantly improved (+0.25), while the performance on *night* is not significantly different. Overall on *non-dawn/dusk* an improvement is measured. This result is in line with the *daytime* results and supports the existence of similarity effects between daytime and dawn/dusk images. At the same time, the absence of a negative effect on night images shows that some similarities exist here.

Night results: The performance difference between the models that trained on night images and those that did not train on night images is the largest within this experiment (+7.22%). Therefore, *night* is relevant. For *dawn/dusk*, the presence or absence of night images does not make a difference in performance, while a negative effect is seen for daytime (-0.82%) if night images are part of the training data. This also explains the negative effect on non-night images (-0.73%). Again a similarity between *night* and *dawn/dusk* can be seen. The negative effect on *daytime* can again be because the model which did not see *night* in training did see more daytime image samples.

The results for the image domain *time of the day* show that in the given experiment, *dawn/dusk* acts as a transition between night and daytime, while it might be closer to daytime. However, one needs to mention that this similarity's strength is not predefined and depends on the applied labeling rules. When defining custom quota for the strata, one should be aware of the similarity between *daytime* and *dawn/dusk*.

Scene For the scene type, only *highway*, *city street* and *residential* have been tested, since the other IDCs are not represented in sufficient size. The result plots are shown in Figure 27.

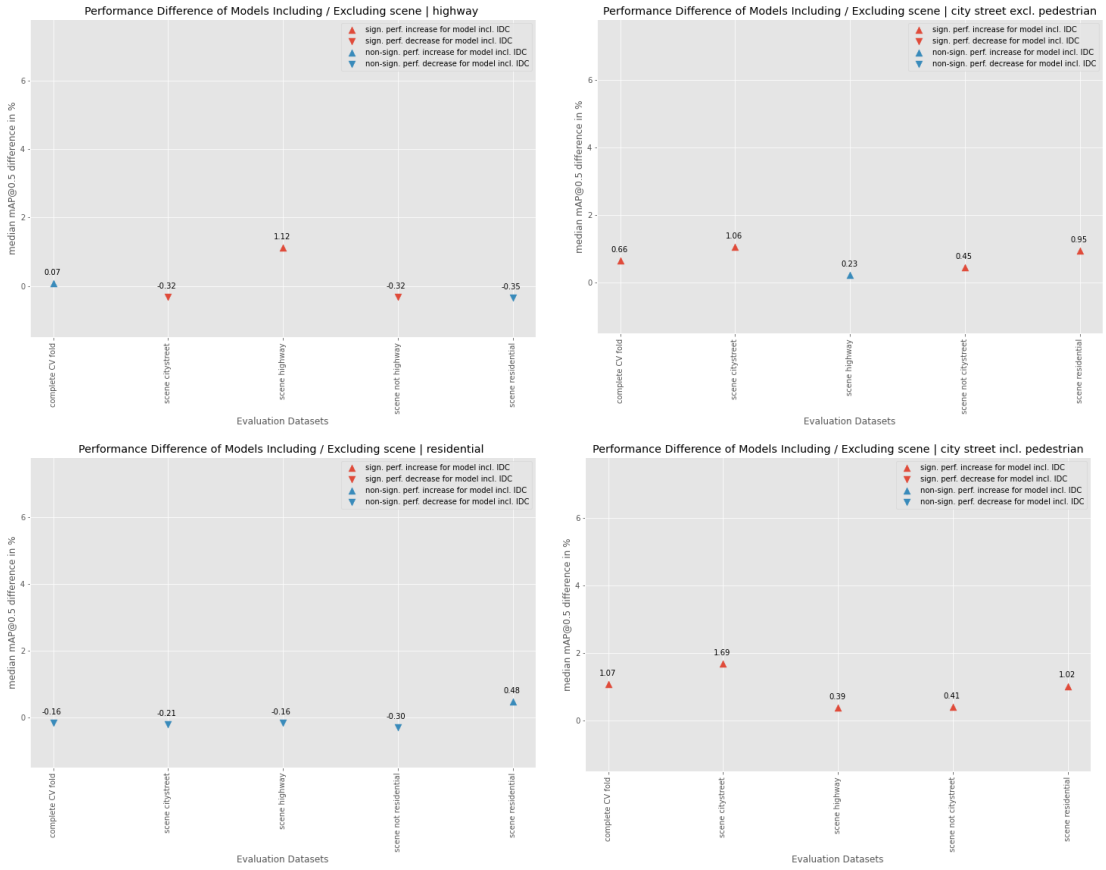

Figure 27: Scene type relevance test results. Top left: highway, top right: city street (excl. pedestrians), bottom left: residential, bottom right: city street (incl. pedestrians)

Highway results: The models trained with highway images perform significantly better (+1.12%) on highway validation images than those trained without highway images. Therefore, *highway* is relevant. Replacing city street and residential images with highway images seems to significantly worsen the performance on *city street* (-0.32%) and also a tendency is noticeable on *residential* (-0.35% non-significant). The constraint is, as explained, that adding *highway* to the dataset unavoidably leads to fewer city street and residential images in the training set. Therefore one can only conclude that learning on highway images does not generalize to city street and residential images as good as learning directly on these IDCs.

City street results: First, one has to mention that city street images show statistically more pedestrians than *non-city street*. This significant difference led to the fact that even though the instance distribution was adjusted, the deviation for the pedestrian class was larger than the accepted $\epsilon = 10\%$. Therefore two performance measures are computed, one including the pedestrian class and one excluding pedestrians. City street images as part of the training data lead to a significantly improved validation performance on city street images (+1.06% excl. pedestrian / +1.69% incl. pedestrian). Therefore, *city street* is a relevant IDC. In addition, training on city street images also leads to a better performance on residential images (+0.95% / +1.02%) and at least to comparable performance on highway images (+0.23% non-sign. / +0.39% sign.). The comparison of both results (including and excluding pedestrians) shows that adding the class to the mAP computation does increase the performance difference. This supports the hypothesis that the number of instances is relevant for the model's performance since the model trained with city street images did see more pedestrians and shows an increased performance among all validation subsets. However, a general offset by adding the pedestrian class is not to be expected, since the differences are evaluated. Therefore, the increase in the differences can be associated with the imbalanced instance distribution.

Again some similarity between residential and city street classes can be assumed. One reason for this could be overlapping labels. In addition, similarity in the scene composition (buildings next to the street, parking cars) can be expected. The absence of a negative effect on *highway* could be due to the higher visual complexity of city street images since more complex scenes are expected to appear at a higher rate than in highway images (e.g., intersections, pedestrians, different view angles and occlusion). Therefore, even though the city street scenes differ from highway scenes, the model can generalize to the highway class to some extent.

Residential results: Residential images in the training data do not significantly increase performance on residential validation images. Therefore, residential is not a relevant IDC. In addition, neither the performance on *city street* nor on *highway* is significantly different under the presence or absence of residential images. The

assumed similarity between city street and residential images might not be the relevant factor when interpreting the positive effect of *city street* on *residential*.

When accepting the fact that *city street* images have a positive impact on other IDCs, one needs to keep in mind that for the *highway* and *residential* tests, also city street images were present in the training data. Therefore, the effect explained in the preliminary information of this section has to be considered. For example, removing residential images automatically leads to more city street images in the training data. This biases the result when the effects of the IDCs are not isolated.

Weather Weather is the domain for which most tests have been conducted. The result plots are shown in Figure 28.

Clear results: Adding clear weather images to the training data does not significantly improve performance in any of the subsets. Therefore, *clear* is not relevant based on the used definition of relevance. A significant decrease can be seen for *overcast*, *snowy* and *undefined*, while for *partly cloudy* and *rainy* the performance is not significantly worse. Clear is the largest class within the weather domain in the dataset. In addition, clear is often chosen as the weather label for night images, even when the weather situation is barely recognizable.

Rainy results: Rainy images in the training data lead to a significant performance improvement on rainy validation images (+1.16%). Therefore *rainy* is relevant. Adding rainy images leads to a significant performance decrease (-0.56%) on partly cloudy validation images. For all other classes, the presence or absence of *rainy* does not make a difference. Even though by intuition, *rainy* might be correlated with an overcast sky, no positive effect can be found for rainy images on overcast. Therefore, the dominant visual difference for rainy images might not be the sky and the typical lighting situation of an overcast sky but rather raindrops on the windshield or reflections due to wet roads/cars.

Snowy results: Snowy images in training significantly improve the performance on the snowy validation set (+0.95%). Therefore, *snowy* is relevant. The second positive significant effect is seen on rainy images, which is even stronger (+1.20

%). All other classes do not show significant differences. The observed correlation between *snowy* and *rainy* is not reciprocated.

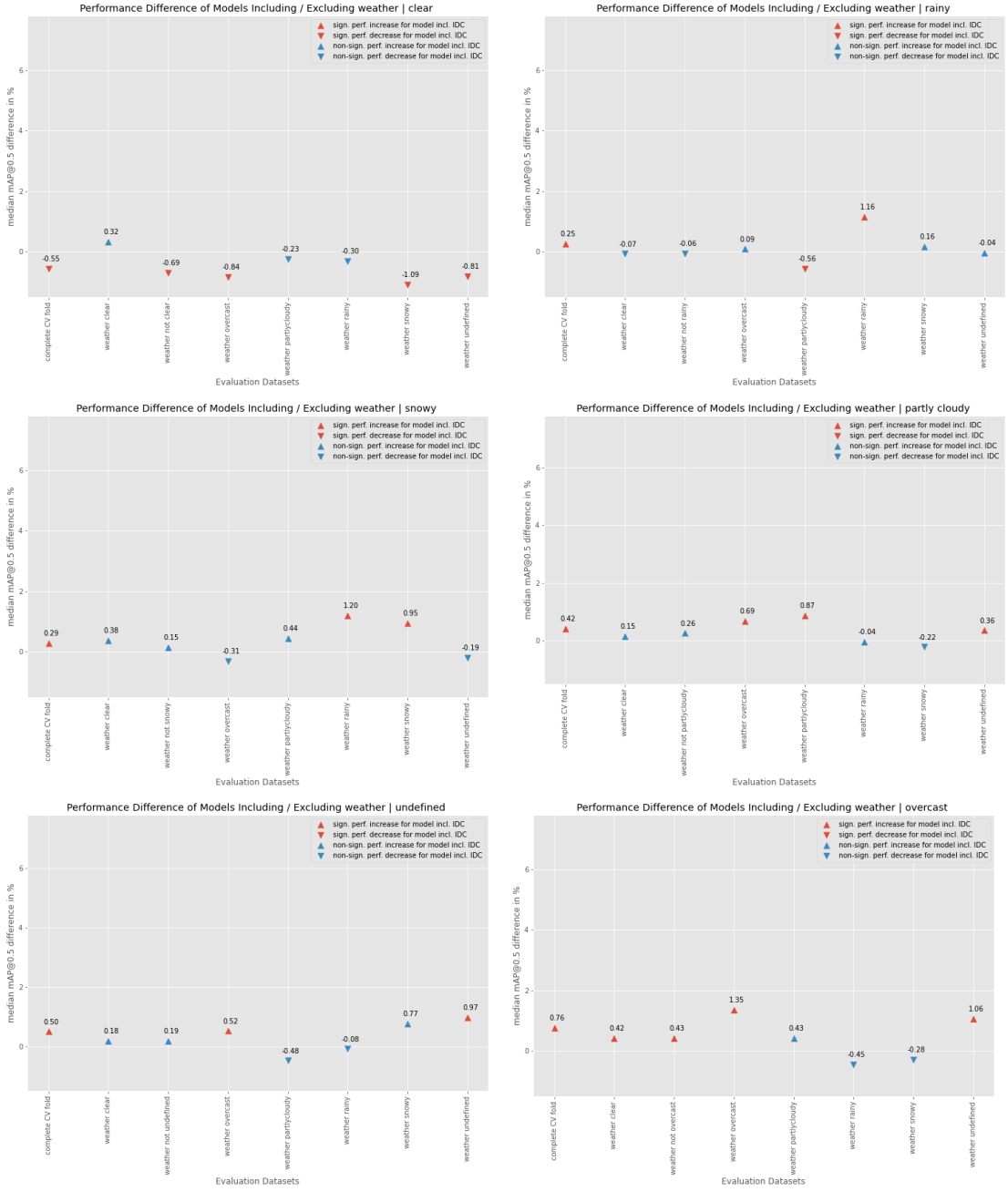

Figure 28: Weather relevance test results. Top left: clear, top right: rainy, center left: snowy, center right: partly cloudy, bottom left: undefined, bottom right: overcast

Partly cloudy results: The models that saw partly cloudy images during training performed significantly better on the partly cloudy validation set (+0,87%). Therefore *partly cloudy* is relevant. Additional positive significant effects are observable on the overcast (+0.69%) and undefined (+0.36%) validation sets. Labeling *partly cloudy* and *overcast* probably leads to overlapping labels. Therefore a certain similarity between those classes is to be expected. The positive effect on *undefined* could also be due to a similarity between partly cloudy and undefined weather, which shows in many cases overexposed skies.

Undefined results: Undefined images in training lead to a significant performance improvement on images of class undefined. Therefore, *undefined* is relevant. Undefined images do have a positive significant effect on *overcast* (+0.52%). No other significant effects are observable. When manually inspecting the weather class undefined, one can find many images with overexposed (white) skies which can be due to a cloudy sky and a specific lighting situation. Therefore a certain similarity for at least some images can be expected between the classes undefined and overcast.

Overcast results: The models that trained on data including overcast images performed significantly better on the overcast validation set (+1.35%) than the comparison models. Therefore, *overcast* is relevant. Adding overcast images to the training data does have a significant positive effect on *undefined* (+1.06%) and *clear* (+0.42%), while no other significant effects can be observed.

The class *clear* was found to be the only non-relevant weather IDC, at the same time it is the largest class within this image domain. However, when considering the negative effects of *clear* on *overcast*, *snowy* and *undefined*, one could still be interested in actively controlling the ratio of clear weather images like for example limiting the share of the class, when defining the strata size manually. The correlation between *overcast* and *undefined* is reciprocated and could be due to some similarities between the classes. In addition, this similarity is also supported by the results of *partly cloudy* and *clear*, where *overcast* and *undefined* are affected in the same direction and magnitude.

4.2.3 Result Consistency

The effect sizes estimated with the visual difference are partially consistent with the results obtained from the actual performance comparison. The results are summarized in Table 12. *Daytime* and *night* show by far the largest effect sizes in both tests. However, one of the two non-relevant classes (*clear*) shows a significant difference in Hotelling's T^2 test. *Overcast*, which has not shown a significant visual difference in the T^2 test, is relevant for the model performance and shows an effect size comparable to the other IDCs. Therefore the visual difference test can be conducted to identify large differences. However, the size of the differences can not be transferred directly to the model performance as measured in this experiment.

Image Domain	**Class**	**Visual Differences Significant**	**IDC is Relevant**
Time of day	Daytime	Yes	Yes
Time of day	Dawn/Dusk	Yes	Yes
Time of day	Night	Yes	Yes
Scene	Highway	Yes	Yes
Scene	City Street	Yes	Yes
Scene	Residential	No	No
Scene	Undefined	Yes	-
Weather	Clear	Yes	No
Weather	Rainy	Yes	Yes
Weather	Snowy	Yes	Yes
Weather	Partly Cloudy	Yes	Yes
Weather	Undefined	Yes	Yes
Weather	Overcast	No	Yes

Table 12: Result table for the visual difference and IDC relevance tests. (Scene class *undefined* was not tested for IDC relevance due to too small sample size.)

4.3 Impact of Domain-Based Data Sampling

The following section describes and discusses the results of the experiment introduced in Section 3.3. Its purpose is to identify the impact of image domain-based stratified data sampling methods on object detection models. The research question that guides this experiment is: What impact does image domain-based stratified sampling of a dataset have on the performance of an object detection model? Of course, this question can only be answered for the given model, dataset, and chosen stratification techniques, but the process can be reused for different use cases.

The process to test the impact of domain-based stratified data sampling methods is shown in Figure 15. Again a 10-fold split was chosen to increase the robustness of the performance measure compared to a single estimate at the cost of generating dependent training datasets. The ratio of data used during training was increased to 50% of the complete dataset, to utilize more of the available data while keeping enough flexibility for the different sampling strategies. The main metric used to compare the model performances is mAP for IoU ≥ 0.5. However, also the averaged mAP for IoU levels from 0.5 to 0.95 is used. Validation datasets with different sampling strategies are chosen to test for performance differences based on the data structure. The prediction results of different sampling strategies for one example image are shown in Figure 29.

Figure 29: Example detections for models trained with four different sampling strategies. Top left: Model trained with random sampled data. Top right: Model trained with LP-based stratified data. Bottom left: Model trained with LP-based stratified data with maximized instances. Bottom right: Model trained with iterative stratified data.

4.3.1 Test Results

The performance of each model group on the validation datasets is compared against the performance of the baseline model group to evaluate the impact of different sampling strategies. The following plots are similar to the result plots introduced for the image domain relevance tests in Section 4.2.2. However, they are generated using Wilcoxon's Rank Sum test since the evaluation results are not paired.

LP-Based Stratified Sampling The first comparison is performed between the model group trained with randomly sampled data (*Baseline*) and the model group trained with LP-stratified data (*A1*). The results are shown in Figure 30 (top left). The model performances are compared on five different validation sets. The left-most data point shows the performance on the complete BDD100K validation dataset. No significant difference is found between the *Baseline* models and the

A1 models. All other data points show subsets of the validation dataset based on different sampling strategies. A significant difference can be found for only one of them, the random sample. Even though the median mAP performance difference is small (-0.42%), the *Baseline* models perform better on the random sample than the *A1* models. This result is surprising since one would not expect a specific repeatable structure of a random sample due to its randomness. The right-most data point shows the performance difference on a validation dataset that is generated with LP-based stratified sampling based on the IDC distribution of the training data. This means the sampling strategy is identical to the sampling strategy used to generate the training data for the *A1* models. However, no significant performance difference is found.

When investigating further on this performance difference on the random sample, the performance can also be measured when split by different object sizes (Figure 30 bottom). Both models perform similar for small objects, but the *Baseline* models are slightly superior for medium and large objects.

In addition, the mAP can be measured over different IoU thresholds from 0.5 to 0.95 (Figure 30 top right), here a similar result can be found, with the exception that also for the complete validation set, a slight performance difference exists. Again, the *Baseline* models perform slightly better.

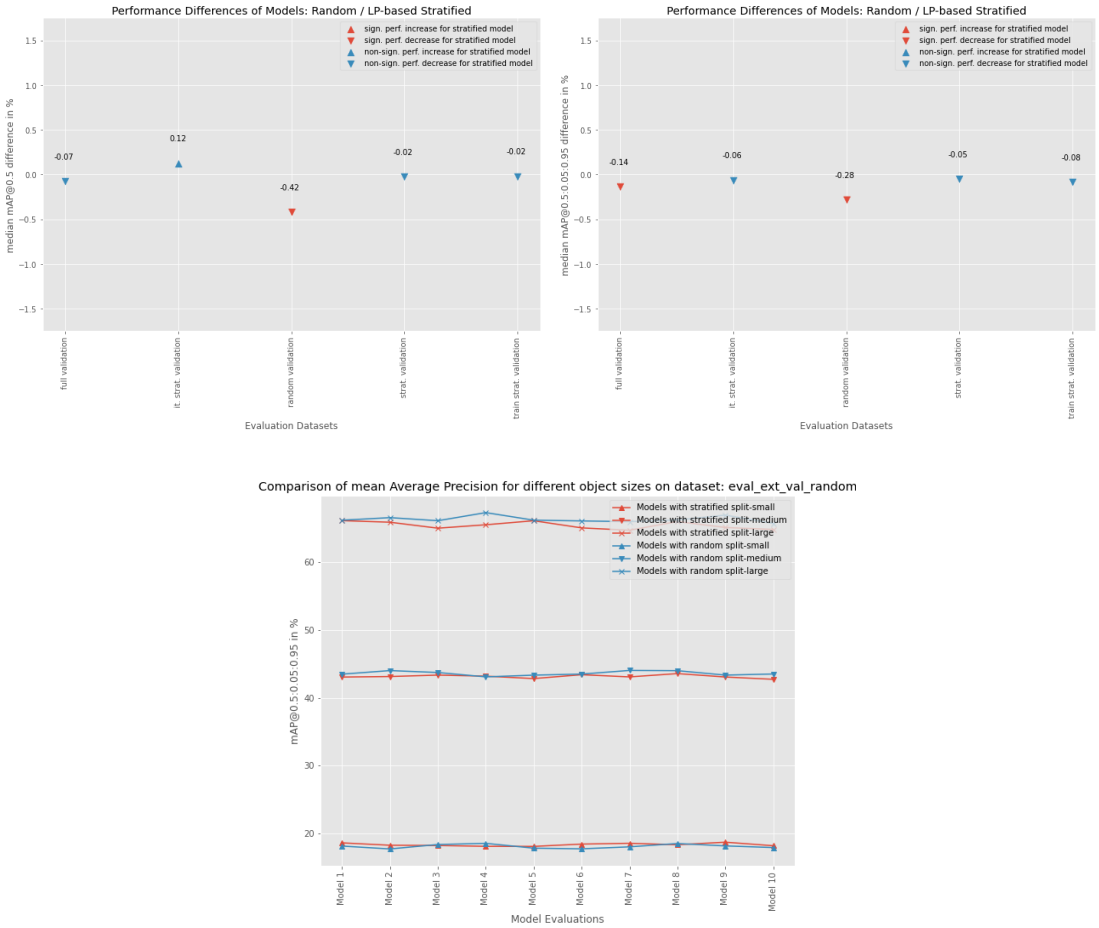

Figure 30: Sampling impact result comparison of models trained with LP-based stratified sampled data and randomly sampled data on different validation datasets. Top left: for IoU ≥ 0.5. Top right: averaged for IoUs from 0.5 to 0.95. Bottom: on the random sample validation set split for object sizes.

LP-Stratified Sampling with Maximized Instances The second comparison is performed between the models trained with randomly split data (*Baseline*) and the models trained with LP-stratified data for which the object instance count is maximized (*A2*). The result shown in Figure 31 (top left) is clear. The *Baseline* models outperform the *A2* models on all validation datasets. The largest difference is seen on the random sample dataset (-1.45 % median mAP difference). A similar but not as strong result is obtained for the comparison over different IoUs (0.5 to 0.95) (top right). Here the largest difference is -1.26 % median mAP difference.

If one groups the mAP by object size, as shown in Figure 31 on the bottom, one can see more detailed how the model performance differs. The *A2* models outperform the *Baseline* models on small objects by a small margin. Nevertheless, the effect is reversed on medium and large objects. Table 13 shows the object size distribution for the different sampling strategies. One can see that the data sampled with strategy *A2* consists out of images with more objects. However, the number of small objects increases disproportionately. Therefore, even though the total number of object instances is higher for the model trained on stratified data, the performance is worse. Therefore, not only the number of instances, but also the balance of instance sizes seems to be important for the model performance.

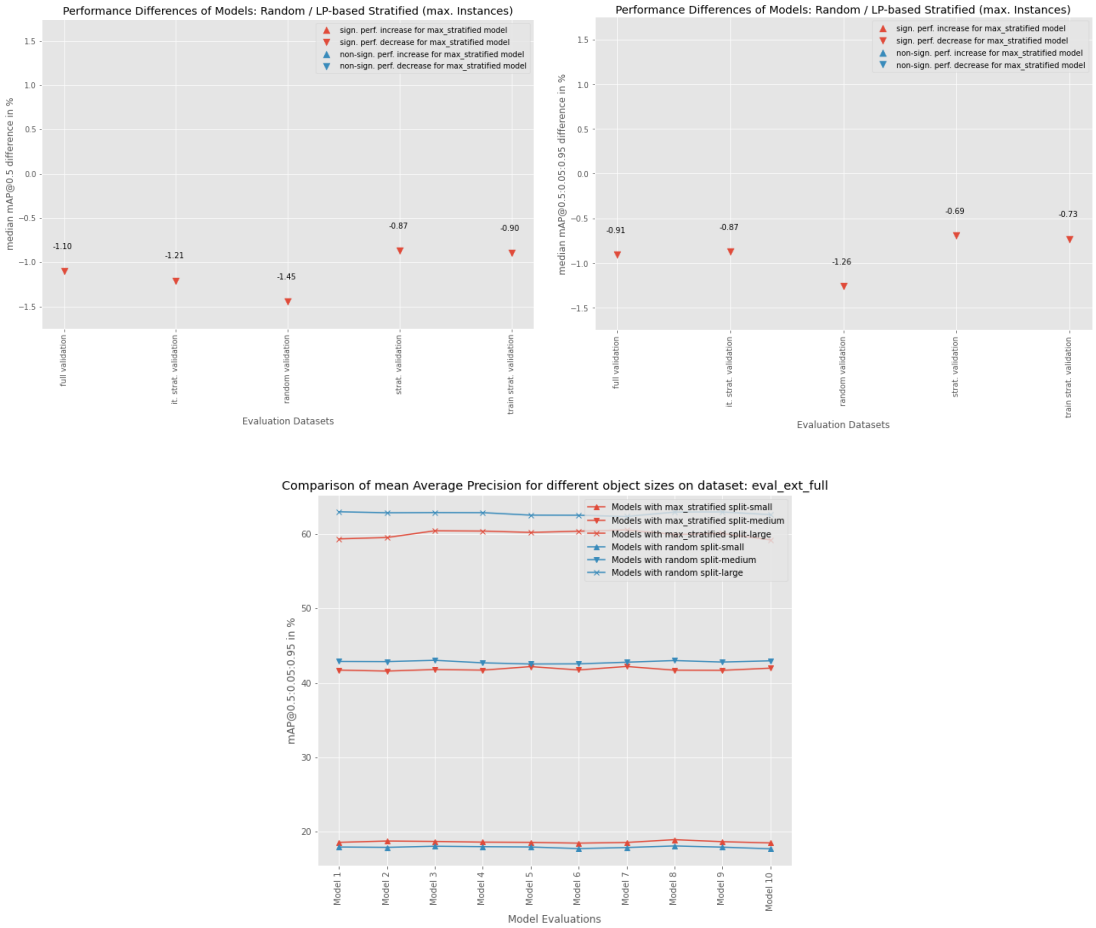

Figure 31: Sampling impact result comparison of models trained on LP-based stratified data with maximized object instances and models trained on randomly sampled data. Top left: comparison over the validation datasets for IoU ≥ 0.5. Top right: comparison over the validation datasets for different IoUs from 0.5 to 0.95. Bottom: mAP comparison for object sizes small, medium, and large on the complete validation dataset.

Object sizes	***Baseline*** **Random**	***A1*** **LP-based stratified**	***A2*** **LP-based stratified max. instances**	***A3*** **Iterative stratified**
Small	10.12	10.12	14.37	10.12
Medium	5.82	5.83	7.50	5.83
Large	2.31	2.32	2.62	2.32

Table 13: Comparison of the mean object instance count per image and size for the training samples generated by the four different sampling strategies.

Iterative-Stratified Sampling The third comparison is performed between the model group trained with iterative stratified data (*A3*) and the model group trained with random data (*Baseline*). The results are shown in Figure 32 (top left) and are similar to the *Baseline* - *A1* comparison. No significant differences are found on the complete validation dataset or the iterative stratified sample of the validation dataset. However, the performance on the random validation sample shows a slight significant difference (-0.21% median mAP difference) (*Baseline* models show a slightly improved performance compared to the *A3* models). On none of the stratified validation samples, a performance difference was found. When comparing the performance based on different IoUs, also on the full validation dataset a slight performance difference can be found (-0.10% mAP median difference).

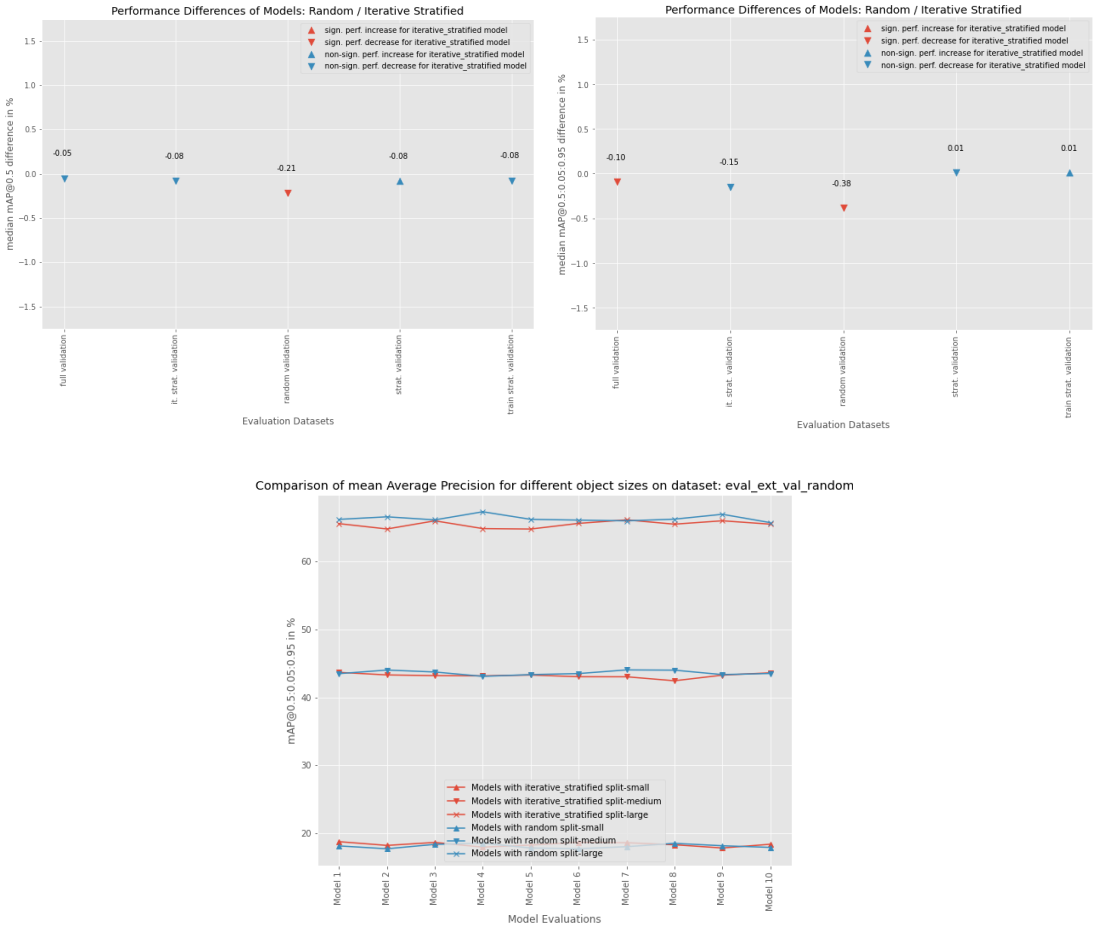

Figure 32: Sampling impact result comparison of models trained with iterative stratified sampled data and randomly sampled data on different validation datasets. Top left: for IoU ≥ 0.5. Top right: averaged for IoUs 0.5 to 0.95. Bottom: mAP comparison for object sizes small, medium, and large on the complete validation dataset.

Result interpretation Two key findings can be extracted from this experiment. First, the applied image domain-based sampling techniques do not lead to superior model performance on the evaluated validation sets. In fact, on one subset, the randomly drawn validation sample, the performance of all models trained with stratified data is slightly worse than that of the models trained with random data when evaluating with the IoU ≥ 0.5. When averaging over multiple IoU-thresholds from 0.5 to 0.95, the performance differences become smaller for the first two comparisons, but the overall trend is maintained. However, except for the *A2*

approach with maximized object instance counts, the median differences are very small ($< 0.5\%$ mAP).

Second, maximizing the object instances leads to side effects impacting the model performance. The intuitive rationale that more object instances lead to extended training and, therefore, to a better performance was found in the IDC relevance tests for the class *city street* and the object class *pedestrian*. However, the ratio between object sizes is shifted towards smaller objects when forcing to maximize the object instance count with the applied sampling strategy. This leads to an improvement in the model's mAP performance on small objects but also to a decline on medium and large objects.

As a result, the applied stratified sampling strategies are not superior to random sampling in any of the tested cases but the performance differences between the *Baseline, A1* and *A3* are very small. A stratified sampling technique can therefore be used to ensure, that all relevant image domain classes are part of the training dataset without the cost of major performance decline. For random sampling an even image domain class distribution can not be guaranteed, especially in the case of small clusters.

What remains is to develop a method to transfer the knowledge, gained in the IDC relevance tests, into a sampling technique, such that the model performance increases. The approach of selecting the strata size based on the existing IDC distribution did not lead to a significant impact.

The constant performance difference of the model with maximized instance counts shows that other effects, such as the object size distribution, have to be monitored and controlled to further reduce confounding factors of the conducted experiments.

5 Summary & Future Work

The following section recaps the most important results of the thesis and provides an overview about possible next steps.

5.1 Summary

In total, three major steps were completed within this thesis:

1. The BDD100K dataset was cleaned, and the existing *scene* and *time of the day* labels were validated.
2. Image domain relevance tests were conducted to identify IDCs that show a significant impact on the performance of an object detection model.
3. The impact of domain-based sampling approaches was investigated and compared to the naive method of random sampling.

Cleansing and validating the dataset disclosed multiple issues. The selected dataset is not as diverse as expected, and previously unknown deviations from the official documentation exist, namely, the unmentioned recording locations in Israel / West Bank. The label validation process showed that even though the label quality is high, there is a significant amount of wrong time of the day and scene labels, and more subtle cases exist for which clear labeling rules need to be documented and published to refer to a unified baseline.

The image domain relevance tests show that ten of the twelve tested IDCs are relevant for the object detection model. Even though the strongest effects were found for those classes that differ visually the most (daytime and night), also other classes show relevant effects but on a smaller scale (except for residential scenes while clear weather shows significant but only negative effects). The correlation between the visual difference assessment and the IDC relevance test is limited. Both tests identified the two magnitude levels consistently, but the visual difference test can not be used as implemented to distinguish between significant and non-significant IDCs. The two classes *clear* and *overcast* were incorrectly classified by the visual difference analysis.

The assumption that instance distribution differences are important and should be monitored is supported by the differing results for the city street class when

including or excluding the imbalanced pedestrian class. See Figure 27.

The domain-based sampling experiments show no significant performance differences when comparing models trained on LP-based stratified or iterative stratified datasets with models trained on randomly sampled data, with one exception. The models trained with stratified data showed a slightly inferior performance on the randomly sampled validation sub-set. See Figures 30 and 32. Therefore, these sampling techniques can be used to ensure coverage of all relevant image domain classes while maintaining a comparable performance level to models trained with randomly sampled data.

In contrast to these results, a strong effect was observed for changes in the object instance balance. The models trained on LP-based stratified data, for which the number of object instances was maximized, showed a declined mAP performance compared to the models trained with randomly sampled data. Maximizing the objects leads to a disproportionate increase of small objects compared to medium and large objects. Even though the models were trained with more instances of all object sizes, a negative effect on the model's performance was observed for medium and large objects. See Figure 31.

5.2 Future Work

Numerous improvements can be added, and the scope of the work can be extended. Some ideas on how one could proceed are mentioned in this section. The first set of potential next steps is concerned with label quality. Improved validation routines could be developed, such as validating scene types by using the actually driven trajectories and not only single location points. This approach could also ease the identification of parking cars. In addition, the scene label should be validated on maps that reflect the historic street and amenity network at the recording time, while in the used implementation the current state of OpenStreetMap was used. The weather data validation was not included in this work. The label validation would require clear rules for categorizing the images and a data source that delivers granular weather data for a given location and time. Further on, the time of the day label verification could be improved by including the *dawn/dusk* validation with additional rules.

Another potential improvement would be a more detailed comparison of the generated data splits. In the current implementation, the object instance distribution is compared in the data splits, but the distribution of the object sizes is not monitored or controlled. Also, additional image domains, such as seasons or locations, can be included and monitored or even tested. Finally, one could investigate whether the hypothesis formulated in the CityScapes paper regarding a performance increase during fall [16] is also supported with BDD100K data on the object detection task.

Since only 12 of the 18 labeled image domain classes have been investigated, developing techniques or extending the dataset such that all IDCs can be investigated would also be a possibility to extend the thesis.

One large and probably the most important field of further improvement is to work on the result transfer of the IDC relevance tests to a (stratified) sampling strategy. The sampling techniques applied in this work did not lead to a significant improvement in the model performance. Therefore, a strategy to determine the optimal image domain class distribution for the training data needs to be developed and tested. Challenges are the large solution space of the optimization problem, the interaction of IDC dimensions, and resource-consuming model training.

References

[1] Richard O Duda and Peter E Hart. "Use of the Hough transformation to detect lines and curves in pictures". In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

[2] G. A. F. Seber. *Multivariate Observations*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1984. ISBN: 9780471881049.

[3] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Computation* 10.7 (Oct. 1998), pp. 1895–1923. ISSN: 0899-7667. DOI: 10.1162/089976698300017197. URL: https://doi.org/10.1162/089976698300017197.

[4] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[5] Alvin C Rencher. *Methods of Multivariate Analysis, Second Edition*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2003. ISBN: 9780471461722.

[6] Remco R. Bouckaert and Eibe Frank. "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 3–12. ISBN: 978-3-540-24775-3.

[7] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (Dec. 2006), pp. 1–30. ISSN: 1532-4435.

[8] Pedro F. Felzenszwalb et al. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.

[9] Zuzana Reitermanová. "Data Splitting". In: *WDS'10 Proceedings of Contributed Papers: Part I - Mathematics and Computer Sciences*. 2010, pp. 31–36. ISBN: 978-80-7378-139-2.

[10] Shakhawat Hossain Ejaz S. Ahmed Enayetur Raheem. "International Encyclopedia of Statistical Science". In: Wiley Series in Probability and Statistics. Springer Berlin Heidelberg, 2011. Chap. Wilcoxon-Signed-Rank Test and Wilcoxon–Mann–Whitney Test, pp. 1656–1659. ISBN: 9783642048975.

[11] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. "On the Stratification of Multi-label Data". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Dimitrios Gunopulos et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–158.

[12] Francisco Charte et al. "A First Approach to Deal with Imbalance in Multi-label Datasets". In: *Hybrid Artificial Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, Sept. 2013, pp. 150–160. ISBN: 978-3-642-40845-8. DOI: 10.1007/978-3-642-40846-5_16.

[13] A Geiger et al. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[14] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[15] Jonathon Shlens. "A Tutorial on Principal Component Analysis". In: *CoRR* abs/1404.1100 (2014). arXiv: 1404.1100. URL: http://arxiv.org/abs/1404.1100.

[16] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

[17] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: https://doi.org/10.1007%5C%2F978-3-319-46448-0_2.

[18] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[19] Gerhard Neuhold et al. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[20] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[21] Leland McInnes et al. "UMAP: Uniform Manifold Approximation and Projection". In: *The Journal of Open Source Software* 3.29 (2018), p. 861.

[22] Kemal Oksuz et al. "Imbalance Problems in Object Detection: A Review". In: *CoRR* abs/1909.00169 (2019). URL: http://arxiv.org/abs/1909.00169.

[23] Eduardo Romera et al. "Bridging the Day and Night Domain Gap for Semantic Segmentation". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1312–1318. DOI: 10.1109/IVS.2019.8813888.

[24] Zhao ZQ et al. "Object Detection With Deep Learning: A Review". In: *IEEE Transactions on Neural Networks Learning Systems* 30 (2019), pp. 3212–3232.

[25] Shamshad Ansari. *Building Computer Vision Applications Using Artificial Neural Networks: With Step-by-Step Examples in OpenCV and TensorFlow with Python*. Jan. 2020. ISBN: 978-1-4842-5886-6. DOI: 10.1007/978-1-4842-5887-3.

[26] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 213–229.

[27] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[28] Fisher Yu et al. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[29] Simon Kennedy. *Astral v2.2 Python Package*. 2021. URL: https://github.com/sffjunkie/astral.

[30] Zhiqing Sun et al. "Rethinking Transformer-based Set Prediction for Object Detection". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 3591–3600.

[31] Ran Zhang et al. "DDE process: A requirements engineering approach for machine learning in automated driving". In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*. 2021, pp. 269–279. DOI: 10.1109/RE51729.2021.00031.

[32] scikit-learn developers. *Gaussian mixture models*. 2022. URL: https://scikit-learn.org/stable/modules/mixture.html#gaussian-mixture. (accessed: 13.10.2022).

[33] *Highly automated technologies, often called self-driving cars, promise a range of potential benefits.* 2022. URL: https://coalitionforfuturemobility.com/benefits-of-self-driving-vehicles/. (accessed: 01.11.2022).

[34] Glenn Jocher et al. *ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations*. Version v6.2. Aug. 2022. DOI: 10.5281/zenodo.7002879. URL: https://doi.org/10.5281/zenodo.7002879.

[35] *Key:amenity - OpenStreetMap Wiki*. 2022. URL: https://wiki.openstreetmap.org/wiki/Key:amenity. (accessed: 23.09.2022).

[36] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context. Pycocotools*. Sept. 15, 2022. URL: https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools.

[37] Christian Safka. *Image 2 Vec with PyTorch*. 2022. URL: https://github.com/christiansafka/img2vec.

[38] Hanno Stage. *OpenStreetMap API query application developed at FZI*. 2022.

[39] Hanno Stage et al. “Towards a Data Engineering Process in Data-Driven Systems Engineering”. In: *IEEE International Symposium on Systems Engineering* (2022).

[40] *MS-COCO test-dev object detection benchmark leaderboard*. URL: https://paperswithcode.com/sota/object-detection-on-coco. (accessed: 07.07.2022).

[41] Fisher Yu. *BDD100K: A Large-scale Diverse Driving Video Database*. URL: https://bair.berkeley.edu/blog/2018/05/30/bdd/. (accessed: 03.09.2022).

[42] Fisher Yu. *Image Sensor Specification #143*. URL: https://github.com/bdd100k/bdd100k/issues/143. (accessed: 17.09.2022).

A Supplementary Figures

```
|_ images
    |_ 100k
        |_ train
            |_ 0a0a0b1a-7c39d841.jpg
                    ⋮
        |_ val
            |_ b1c9c847-3bda4659.jpg
                    ⋮
        |_ test
            |_ cabc30fc-e7726578.jpg
                    ⋮
|_ labels
    |_ det_20
        |_ det_train.json
        |_ det_val.json
|_ info
    |_ 100k
        |_ train
            |_ 0a0a0b1a-7c39d841.json
                    ⋮
        |_ val
            |_ b1c9c847-3bda4659.json
                    ⋮
```

Figure 33: File structure of the BDD100K dataset

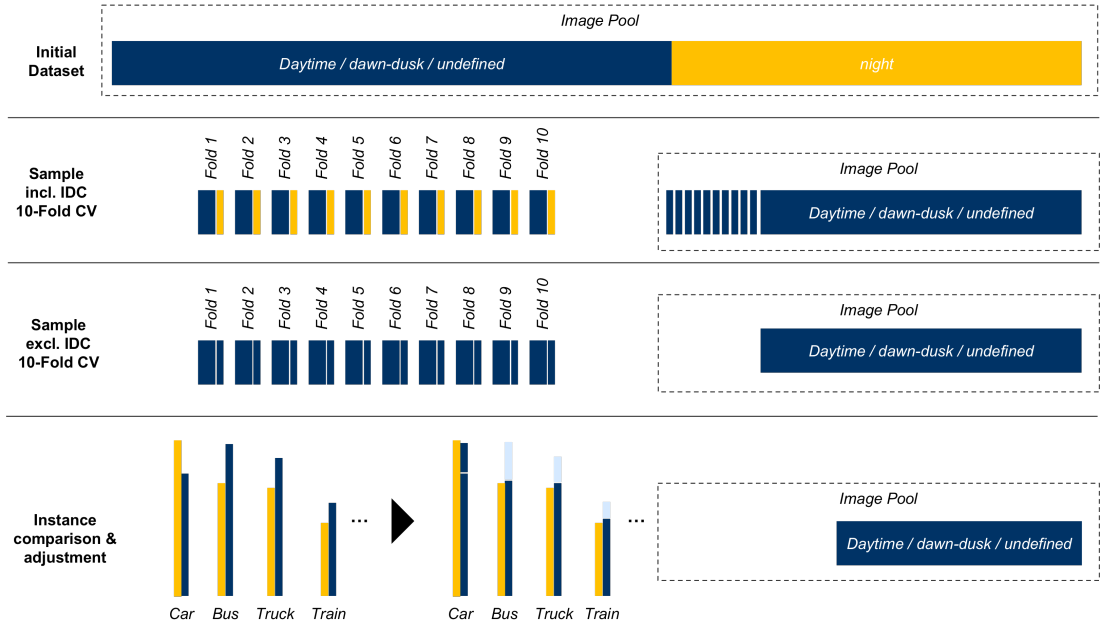

Figure 34: Exemplary sampling process for IDC night. In the first step, a 10-fold sub sample of size 25% is drawn. The share of night images in this sample is 29%. Afterwards by using the remaining image pool, the modified sample is generated by replacing the night images with non-night images. Eventually the object instances in both 10-fold sub samples is compared and adjusted by replacing images in the modified sample with images of the remaining image pool.

B Experiment Parameters

Hardware Setup

	Machine Parameters
CPU	2x AMD EPYC 7742 64-Core Processor
RAM	16x 32GB DDR4 RAM
GPU	4x Nvidia A100 PCIe 40GB
Disk	2x 447.1 GB NVMe SSD in raid 1

Table 14: Hardware specifications of the machine used for training and validation.

Training Parameters

Hyperparameter	**Value**
Initial learning rate	0.01
Final learning rate ratio	0.5892
Momentum	0.937
Optimizer weight decay	0.0005
Warmup epochs	3.0
Warmup momentum	0.8
Warmup bias learning rate	0.1
Box loss gain	0.05
Class loss gain	0.5
Class BCELoss weight	1.0
Object loss gain obj	1.0
Obj BCELoss weight	1.0
IoU training treshold	0.20
Anchor-multiple threshold	4.0
Number of anchors	3
Focal loss gamma	0.0
Hue aug.	0.015
Saturation aug.	0.7
Value aug.	0.4
Rotation degree aug.	0.0
Translation aug.	0.1
Scale aug.	0.5
Shear aug.	0.0
Perspective aug.	0.0
Flip up-down prob.	0.0
Flip left-right prob.	0.5
Mosaic prob.	1.0
Mixup prob.	0.0
Segment copy paste prob.	0.0
Parameter	**Value**
Batch size	16
Epochs	85
Data Caching	RAM
Frozen layers	0
Train from scratch	Yes

Table 15: Training parameter settings for IDC relevance test

Hyperparameter	Value
Initial learning rate	0.01
Final learning rate ratio	0.2674
Momentum	0.937
Optimizer weight decay	0.0005
Warmup epochs	3.0
Warmup momentum	0.8
Warmup bias learning rate	0.1
Box loss gain	0.05
Class loss gain	0.5
Class BCELoss weight	1.0
Object loss gain obj	1.0
Obj BCELoss weight	1.0
IoU training treshold	0.20
Anchor-multiple threshold	4.0
Number of anchors	3
Focal loss gamma	0.0
Hue aug.	0.015
Saturation aug.	0.7
Value aug.	0.4
Rotation degree aug.	0.0
Translation aug.	0.1
Scale aug.	0.5
Shear aug.	0.0
Perspective aug.	0.0
Flip up-down prob.	0.0
Flip left-right prob.	0.5
Mosaic prob.	1.0
Mixup prob.	0.0
Segment copy paste prob.	0.0

Parameter	Value
Batch size	32
Epochs	150
Data Caching	RAM
Frozen layers	0
Train from scratch	Yes

Table 16: Training parameter settings for domain-based sampling impact test

	nano	**small**	**medium**	**large**
Time per epoch [sec.]	195	253	455	664
Time per 85 epochs [hours]	4.60	5.97	10.74	15.68

Table 17: Training time of IDC relevance test for different model scales (training on single GPU with batch size 16)

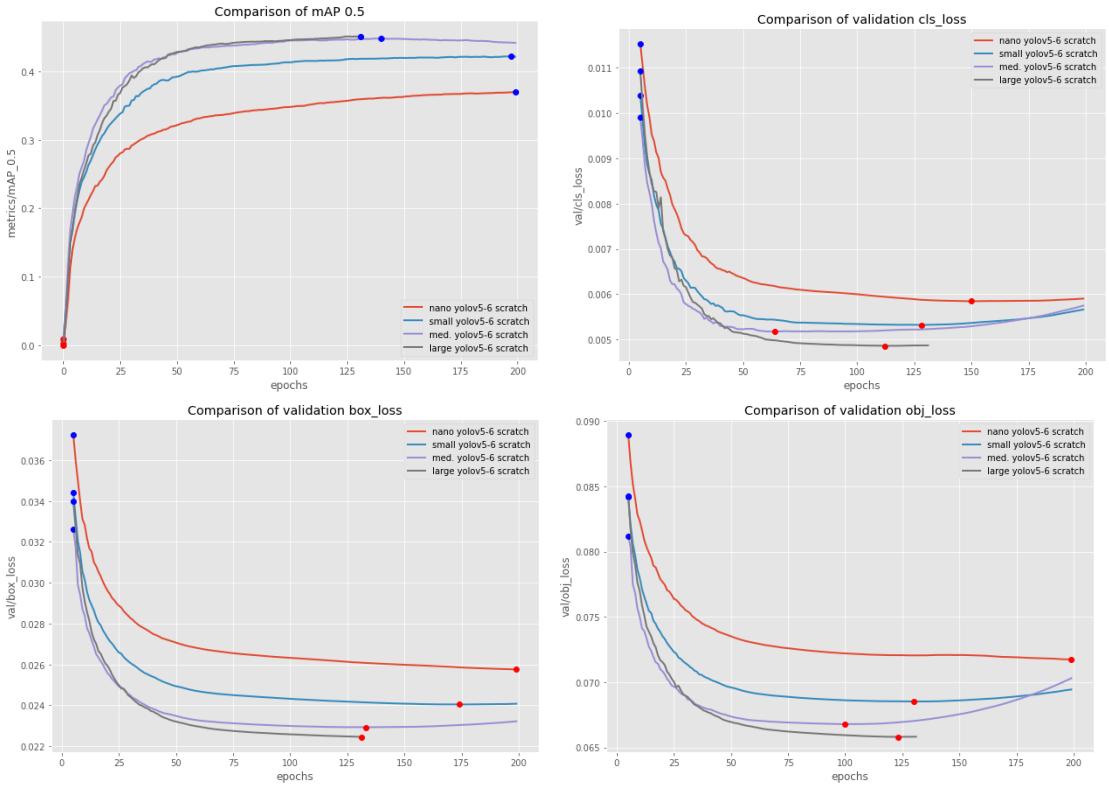

Figure 35: Training tests of different model sizes