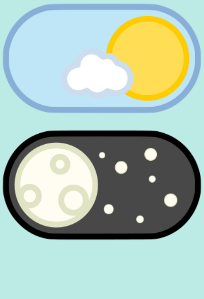
A CONCEPT FOR DEPLOYMENT AND EVALUATION OF UNSUPERVISED DOMAIN ADAPTATION IN COGNITIVE PERCEPTION SYSTEMS

Karlsruher Institut für Technologie

DOCTORAL THESIS



MARK SCHUTERA

" The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say. "

- J. R. R. Tolkien,
The Lord of the Rings

10.5445/IR/1000162895
[KITopen]

A Concept for Deployment and Evaluation of Unsupervised Domain Adaptation in Cognitive Perception Systems

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN
(Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

M.Sc. Mark Schutera

Tag der mündlichen Prüfung:	29. September 2023
Hauptreferent:	apl. Prof. Dr. Markus Reischl
Korreferenten:	Prof. Dr. Veit Hagenmeyer
	Prof. Dr. Stefan Elser

Kurzfassung

Jüngste Entwicklungen im Bereich des tiefen Lernens ermöglichen Perzeptionssystemen datengetrieben Wissen über einen vordefinierten Betriebsbereich, eine sogenannte Domäne, zu gewinnen. Diese Verfahren des überwachten Lernens werden durch das Aufkommen groß angelegter annotierter Datensätze und immer leistungsfähigerer Prozessoren vorangetrieben und zeigen unübertroffene Performanz bei Perzeptionsaufgaben in einer Vielzahl von Anwendungsbereichen.Jedoch sind überwacht-trainierte neuronale Netze durch die Menge an verfügbaren annotierten Daten limitiert und dies wiederum findet in einem begrenzten Betriebsbereich Ausdruck. Dabei beruht überwachtes Lernen stark auf manuell durchzuführender Datenannotation. Insbesondere durch die ständig steigende Verfügbarkeit von nicht annotierten großen Datenmengen ist der Gebrauch von unüberwachter Domänenanpassung entscheidend. Verfahren zur unüberwachten Domänenanpassung sind meist nicht geeignet, um eine notwendige Inbetriebnahme des neuronalen Netzes in einer zusätzlichen Domäne zu gewährleisten. Darüber hinaus sind vorhandene Metriken häufig unzureichend für eine auf die Anwendung der domänenangepassten neuronalen Netzen ausgerichtete Validierung. Der Hauptbeitrag der vorliegenden Dissertation besteht aus neuen Konzepten zur unüberwachten Domänenanpassung. Basierend auf einer Kategorisierung von Domänenübergängen und a priori verfügbaren Wissensrepräsentationen durch ein überwacht-trainiertes neuronales Netz wird eine unüberwachte Domänenanpassung auf nicht annotierten Daten ermöglicht. Um die kontinuierliche Bereitstellung von neuronalen Netzen für die Anwendung in der Perzeption zu adressieren, wurden neuartige Verfahren speziell für die unüberwachte Erweiterung des Betriebsbereichs eines neuronalen Netzes entwickelt. Beispielhafte Anwendungsfälle des Fahrzeugsehens zeigen, wie die neuartigen Verfahren kombiniert mit neu entwickelten Metriken zur kontinuierlichen Inbetriebnahme von neuronalen Netzen auf nicht annotierten Daten beitragen. Außerdem werden die Implementierungen aller entwickelten Verfahren und Algorithmen dargestellt und öffentlich zugänglich gemacht. Insbesondere wurden die neuartigen Verfahren erfolgreich auf die unüberwachte Domänenanpassung, ausgehend von der Tag- auf die Nachtobjekterkennung im Bereich des Fahrzeugsehens angewendet.

Abstract

Recent developments within the application of deep learning allow cognitive perception systems to draw knowledge about the domain of operation purely data-driven. The supervised learning approach to deep learning is further accelerated by the emergence of large-scale labeled datasets and progressively powerful computers. As a result, supervised learning approaches have shown unprecedented capabilities within perception tasks, for example, object detection in a wide range of applications, such as autonomous driving. However, supervised trained models are limited by the available labeled data, finding expression in a finite domain of operation. With supervised learning heavily relying on manual labor, and the ever-increasing availability of unlabeled datasets, the extensive use of unsupervised domain adaptation approaches is crucial for the advancement of model deployment. Available unsupervised domain adaptation approaches are usually not suitable for achieving model deployment in the adapted domain. Moreover, existing metrics are often not sufficiently considered or inadequate for unsupervised domain adaptation of models under deployment. The major contribution of the present thesis is a new concept for the training and validation of unsupervised domain adaptation methods involved in the continuous deployment of perception models. Based on a conceptualization of domain shift types, the a priori available knowledge representations within a supervised trained model are capitalized to enable further unsupervised domain adaptation on unlabeled data. To address the need for continuous deployment of neural networks, three novel unsupervised domain adaptation approaches are specifically developed to enable the extension of a neural network's domain of operation. Employing exemplary automotive perception tasks, it is established how a combination of both the novel unsupervised domain adaptation approaches and the newly proposed domain adaptation metrics can contribute to handling continuous adaptation and deployment based on unlabeled data. To compound the contribution, the implementations of all developed approaches and algorithms are demonstrated and made publicly available. In particular, the novel approaches were successfully applied to unsupervised domain adaptation from day to night object detection in autonomous driving.

Preface

The work in your hands originates from my activities and time as a doctoral student in cooperation between Research and Development at ZF Friedrichshafen AG and the Institute for Automation and Applied Informatics at Karlsruhe Institute of Technology. During the last months and years, I experienced encouragement and support from many directions; for this, I am very grateful and appreciative. With the following lines, and without guarantee for completeness, I dare the attempt to deliver the appropriate acknowledgment:

To my most radiant beacon, apl. Prof. Dr. Markus Reischl, who allowed me to merge academic research with industrial application and conduct research in such an interesting and extensive field, for the unfailing support, constructive discussions, and guidance for my research in general, this thesis in specific, and all things beyond.

To my consistent ignition, Prof. Dr. Veit Hagenmeyer, who followed my research with great interest, for the excellent feedback, contributing to the success of this work, especially on the home stretch.

To my persistent fire accelerator, Prof. Dr. Stefan Elser, who helped me find my place and pace both as a doctoral researcher at ZF Friedrichshafen AG and as an adjunct lecturer at Ravensburg-Weingarten University, for the friendship and continuous commitment.

To my initial spark apl. Prof. Dr. Ralf Mikut, who already during my studies at KIT, opened the field of machine learning to me, for the excellent academic and technical guidance and for advising me at the outset of this journey on so many a question.

To my fellow students, colleagues, and Ph.D. students at KIT, the ZF Innovation Hub in Sunnyvale, and ZF Friedrichshafen AG, who kindly received and constantly welcomed me in their hallowed halls for their unobstructed support by sharing ideas, projects, coffee, and motivation. Especially, I want to thank Prof. Dr. Christian Pylatiuk, Stefan Bühler, Christian Witte, Jo-

hannes Bernhard, Jonas Schmidt, Mostafa Hussein, Marcel Schilling, and Luca Rettenberger for the collaborations and shared endeavors.

To my industrial supervisor, Dr. Jochen Abhau, who gave me a chance to prove myself, for granting me support, freedom, space, and opportunities to drive research and to grow within ZF Friedrichshafen AG.

To my discussion partners, Dr. Daniel Plencner and Dr. Daniel Schmitt, who regularly made me rethink concepts, challenge ideas, and be innovative, for being the extraordinarily smart, curious, and nice people they are.

To my hidden supervisors, Dr. Niklas Goby and Tobias Mindel, whom I appointed myself without them knowing, for readily sharing knowledge, listening, generating drive, and getting me started by believing in me, adding perspective to my ideas, and disrupting my beliefs.

To my closest companions on this odyssey, Christian Herzog, Tim Härle, Frank Hafner, Anja Fessler, Florian Wilhelmi, and Hendrik Vogt, which care that I stay grounded, on the wall, and my feet, for 24/7 autonomous driving, diamond hands, going the extra mile and discussing the ultimate questions of life, the universe, and everything. To my friends near and far, who hold me dear and let me know that they are around, for accepting me for who I am and for who I am not.

To the woman who accompanied me for a part of this journey, for love, another perspective, and for enriching my life in so many ways. To my sister, who at times rescued my self-esteem, reminding me that things are just fine, that I am doing fine, and that a few breaks are okay. To my beloved parents, who broke social coercion and afforded time for me to put the pedal to the metal, for their love, patience, and unconditional endorsement.

Friedrichshafen, December 2022 *Mark Schutera*

Nomenclature

Alphabetical Symbols

$\mathbb{A}$	Set
$\mathbb{E}$	Expectation
$\mathbb{R}$	Set of Real Numbers
$\mathscr{D}$	Domain
$\mathscr{G}$	Complete Bipartite Graph
$\mathscr{N}(\mathbf{x};\mu,\Sigma)$	Gaussian Distribution over $\mathbf{x}$ with Mean μ and Covariance Σ
$\mathscr{P}_{\mathscr{G}}(u_i)$	Parent Nodes of u_i in $\mathscr{G}$
$\mathscr{T}$	Task
$\mathbf{A}$	Matrix
$\mathbf{a}$	Vector
$\mathbf{b}$	Bias Vector
$\mathbf{C}$	Correspondence Tensor
$\mathbf{I}$	Identity Matrix
$\mathbf{M}$	Mapping Tensor
$\mathbf{R}$	Covariance Matrix
$\mathbf{V}$	Vertex Cover
$\mathbf{W}$	Weight Tensor
a	Scalar
bel	Posterior Belief
Cov	Covariance
$f,\ g$	Functions
FN	False Negatives
FP	False Positives
fps	Frames per Second
h	Hidden Layer, Function
$i.i.d$	Independent and Identically Distributed
$J()$	Cost Function
KL	Kullback-Leibler Divergence
l	Layer
o	Object

$O()$	Time Complexity (Order of the Function)
$p()$	Probability Distribution
px	Pixels
std	Standard Deviation
T	Threshold
t	Time Stamp
TN	True Negatives
TP	True Positives
u	Unit, Neuron, Cell, Node
Var	Variance
w	Scalar Weight
x	State, Input
z	Layer Activation State

Greek Symbols

α	Rotation Angle
Δ	Step, Difference
δ	Difference Operator
ϵ	Learning Rate
μ	Mean Value
∇	Partial Derivative Matrix
∂	Partial Derivative
σ	Standard Deviation
τ	Weight Updates, Number of Computational Operations
θ	Set of Parameters, Weights

Superscripts

$\hat{y}$	Model Prediction, Estimated Output
$\overline{a}$	Prior
$\mathbf{A}^\top$	Transpose of Matrix **A**
$\tilde{y}$	Ground Truth, meaning the Empirical Distribution
$\tilde{y}'$	Pseudo Ground Truth
a^*	Optimal form of a
a'	Derivative, Mapping

Subscripts

$\mathbf{A}_{:,j}$	Column j of Matrix $\mathbf{A}$
$\mathbf{A}_{i,:}$	Row i of Matrix $\mathbf{A}$
a_i	Element i of Vector $\mathbf{a}$
$A_{i,j}$	Element i, j of Matrix $\mathbf{A}$
a_{init}	Initial Value

Other Symbols

$*$	Scalar Product
.	Placeholder
#	Cardinality
_	Empty Element
$\downarrow$	Defined, Decreasing
$\rightarrow$	The process of annotation
$\mathbf{A} \odot \mathbf{B}$	Hadamard Product (Element-wise Product)
$\uparrow$	Undefined, Increasing

Contents

1 Introduction

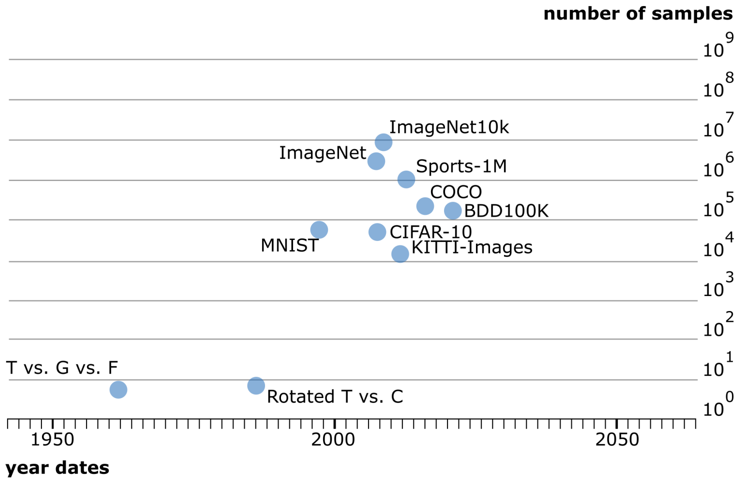
1.1 Emanation and Significance of the Work

1.1.1 Deep Learning - a Paradigm Shift

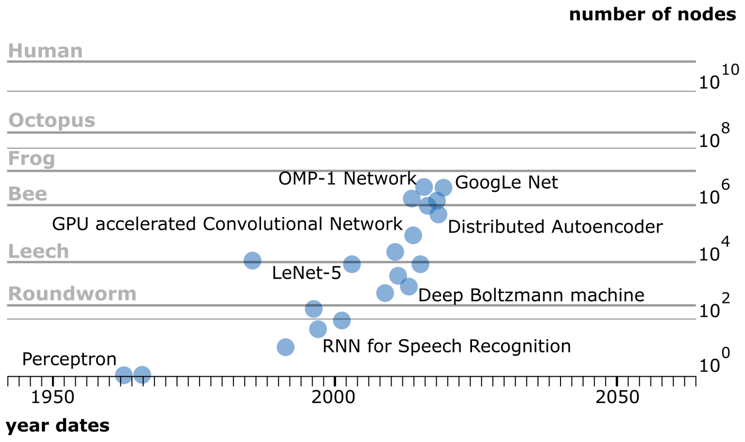
Deep learning has evoked a paradigm shift within cognitive perception systems in recent years. Deep learning allows machines to learn from experience given by annotated data. As such, deep learning enables machines to be aware of the world as a set of hierarchical concepts, with each concept being defined by a set of more fundamental concepts [17]. Drawing knowledge about the world, purely data-driven, allows the machine to build up and learn about these concepts and their relation to each other. Represented as a graph, this graph is deep, with a lot of layers - giving reason to naming this approach - deep learning [18].

If a system is to behave intelligently, it needs to inherit a considerable amount of knowledge about the world or the perceived environment and its underlying structures and interrelationships. Much of this knowledge is subjective, and as far as humans are concerned, most actions are instead based on intuitive premises. When it comes to artificial intelligence, one faces the challenge of mapping this idiomatic knowledge so a computer can access and process it. The earlier methods attempted to hard-code knowledge into statements structured by formal languages and logical operators, also known as the knowledge-based approach.

As the number of labeled data increases in terms of the number of available datasets and samples per dataset (see Fig. 1.1a), another approach became applicable: Deep learning strives to depict system behavior by means of abstract representations, which are in turn obtained and defined by less abstract ones. In deep learning approaches, these representations are depicted within neural networks. Those representations are then evaluated on how well they correspond with reality, being readjusted if these representations do not perform well, hence called the data-driven approach. Deep learning is further propelled by progressively powerful computers or processing units, with significantly more memory, faster computation, ever-increasing available data,



(a) Development of dataset sizes used in machine learning, size in the number of samples per dataset over time [18]. Datasets are depicted as blue dots. The figure intends to depict the development of datasets only, and the central datasets will be introduced in detail (see Sec. 1.4 and 5.3.1).



(b) Development of neural network size on a logarithmic scale over time. Neural networks are depicted as blue dots and are put in relation to biological neural network sizes present in the fauna [18].

Figure 1.1

and enhanced and increasingly sophisticated annotation of datasets. As of today, the number of units deployed meaningfully within neural networks doubles approximately every 2.4 years (see Fig. 1.1b), reaching the same number of neurons as being present in a human brain approximately in 2050 [18]. Modern machine learning and deep learning methods are essential for autonomous perception systems.

1.1.2 Significance of the Work

Neural networks within perception systems, solving tasks such as object detection, semantic segmentation, and others, are central to a wide range of applications such as autonomous driving, biomedical computer vision, and many more. On the way to market, perception systems have to deal with different versions, variants, and development stages. Since it is neither economical nor realistic to develop a mature, finalized software system, the systems will have to evolve in line with extending requirements by improving performance, adding entirely novel functionalities, and adapting to newly faced domains and environments outside of the initial operational design domain (ODD). Therefore, the initial systems are only developed for a limited range of applications, which can then be gradually expanded.

The principle of continuous development and deployment is a state-of-the-art software development approach when improving edge devices such as smartphones. As is, this procedure is not easily transferable to state-of-the-art deep learning methods. Here, desired changes, extensions, and adaptations to an existing component, such as additional classes, differing sensor technology, sensor position, advancements in the sensor technology, or supplementing differing domains, while adhering to the safety requirements, often trigger another renewed, complete training and validation cycle followed by deployment. Thus, there is a definite need to develop new methods and processes that make adaptation more seamless, such as continuous learning or shared feature spaces.Most commonly, deep learning approaches perform well when trained supervised, following the mutual assumption: The training samples and the samples received during application are within the same domain. However, supervised learning requires a dataset of annotated samples for training and validation. Labeling is a time and cost-consuming process. Thus, there is a definite need to develop further methods and procedures that make manual supervision unnecessary - this work will concentrate on unsupervised domain adaptation.

Methods for the efficient extension and transformation of existing deep learning modules are developed to resolve the aforementioned problem statements. Furthermore, these methods get autonomous perception systems ready for the challenges of a repetitive emergence of new domains and the shift from datasets to datastreams when deploying neural networks. The following introduction sets out to build the foundation on which the contributions of this thesis are able to unfold. Initially, the supervised deep learning pipeline is presented, together with the general concepts of deep learning. Second, object detection as the central task of this thesis is outlined, followed by the associated deep learning architectures for object detection. Subsequently, the current status of large-scale datasets is described, together with the theoretical foundation of label policies and manual annotation. Further, the capabilities to evaluate deep learning models are pointed out, which later serve as a gateway to a novel evaluation strategy that is customized to and targets domain adaptation. Concluding, current concepts and the state-of-the-art of transfer learning are fanned out, building the groundwork for the novel unsupervised domain adaptation concept and approaches, which are defined objectives of this thesis.

1.2 The Connectionist Approach to Mind

In order to understand cognitive perception systems, the prevalent concept of connectionism first needs to be understood. Within cognitive science, the mind was traditionally advocated within a computational frame. While representing mental states as strings of symbols [19], mental processes were understood as computations on those symbols. Over time, computationalism was superseded by the connectionist approach, which proposes an understanding of the mind as activations unfolding within a network structure. A network consists of computationally simple nodes, also termed units connected by weighted edges, which transmit activations through the network and hence enable interaction between nodes, by which intelligent behavior emerges [20]. This approach provokes comparisons to the nervous system, leading to analogies and the interdisciplinary borrowing of terms, such as *neuron* for the nodes and *neural* in general. However, the resemblance is soon exhausted, considering that biological neural networks are unmatched in their complexity and far from being understood completely [21, 22].

Today's machine learning algorithms mainly build upon linear algebra, probabilistic, information theory, and numerical optimization and do not necessarily resemble any biological counterpart [18, 23].

In machine learning, networks, also called models, are deployed in order to depict a predictive function f^* by determining a set of parameters θ which optimally (θ^*) approximates $f^*(\mathbf{x};\theta^*)$, while $\mathbf{x} \in \mathbf{X}$ represents an input sample, associated with a specific domain $\mathscr{D}$. The probability distribution over the input samples $\mathbf{X}$ is referred to as the marginal probability distribution $P(\mathbf{X})$.

Definition 1.1 ***(Domain).*** *A domain $\mathscr{D}$ consists of a feature space $\mathscr{X}$ and a marginal probability distribution $P(\boldsymbol{X})$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \in \mathscr{X}$ with an index (here n) depicting the number of samples within the domain. Domain coverage is defined as the extent to which $P(\boldsymbol{X})$ is represented by $\boldsymbol{X}$. A domain sample $\boldsymbol{x}_1$ comes with an associated ground truth $\tilde{\boldsymbol{y}}_1$. The tilde is introduced to differentiate between the actual, true, ground truth $\boldsymbol{y}$, and the empirical estimation $\tilde{\boldsymbol{y}}$, recognizing that any available ground truth is an estimate of some sort. The source domain dataset is denoted as $\mathscr{D}_S = \{(\boldsymbol{x}_{S_1}, \tilde{\boldsymbol{y}}_{S_1}), \ldots, (\boldsymbol{x}_{S_n}, \tilde{\boldsymbol{y}}_{S_n})\} = \{\boldsymbol{X}_S, \tilde{\boldsymbol{Y}}_S\}$. The target domain dataset is denoted as $\mathscr{D}_T = \{(\boldsymbol{x}_{T_1}, \tilde{\boldsymbol{y}}_{T_1}), \ldots, (\boldsymbol{x}_{T_m}, \tilde{\boldsymbol{y}}_{T_m})\} = \{\boldsymbol{X}_T, \tilde{\boldsymbol{Y}}_T\}$ [24].*

Neural network architectures (see Figure 1.2) are cast into multiple layers. The output signal, or prediction, is denoted as $\hat{\mathbf{y}}$, and the hidden layer as $\mathbf{h}$. Each layer, in turn, consists of multiple nodes, the so-called units. The dimensionality of the layers l is coined depth, while the dimensionality of the units is referred to as the width of a layer. While g is the activation function of the hidden layer, $\mathbf{W}^{(l)}$ is the weight matrix for the layer and $\mathbf{b}$ depicts the bias for the layer, with input variable $\mathbf{x}$. The nodes, or units, are denoted following the weight matrix notation with weights $w_{jk}^{(l)}$, which represents the weight of unit k in layer $l-1$ to the unit j in layer l [18]. In a single layer feed-forward architecture, this corresponds to

$$\hat{\mathbf{y}} = f(\mathbf{x};\theta) = g^{(l)}(\mathbf{W}^{(l)\top}\mathbf{x} + \mathbf{b}^{(l)}). \tag{1.1}$$

For further architectural details and deep learning fundamentals, see Appendix A.2.

The convolution of a layer's parameters span a feature space $\mathscr{H}^{(l)}, \mathbb{R}^{|\theta^{(l)}|}$, mapping an input vector $\mathbf{x}$ into a feature representation, within that feature

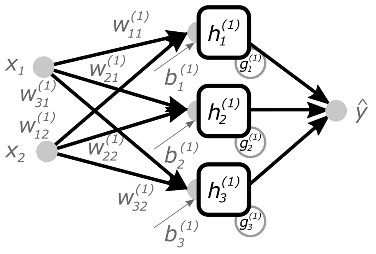


Figure 1.2: Overview of a neural network with input $\mathbf{x}$, hidden layer $\mathbf{h}$ and output signal $\hat{\mathbf{y}}$ with the according notation. Exemplary the unit notation is depicted with respect to unit $h_j^{(l)}$: Including the mapping by a weight $w_{jk}^{(j)}$, the bias $b_j^{(l)}$ and the activation function $g_j^{(l)}$.

space. Feature values of intermediate layers usually are not directly interpretable; however, they provide an internal representation, and their feature spaces are hence referred to as latent feature spaces. In the case that feature representations, meaning a layer's parameters, are common for at least two otherwise separate neural networks, the feature space in question is called a shared feature space. Another term for feature space is feature embedding.

Despite consisting of linear models only, a multi-layered feed-forward neural network, with a finite number of units, is able to represent any given function, as proven by the universal approximation theorem [25]. However, this is not to be understood as a universal guarantee that the corresponding set of parameters is found.

1.2.1 The Supervised Deep Learning Pipeline

In recent years, supervised learning has emerged to be the most successful concept (see Fig.1.3) for pattern recognition. If one hopes to carry over the capabilities of supervised deep learning into a different paradigm, it is indispensable to first scrutinize the cornerstones of the current paradigm: The common supervised deep learning pipeline comprises the approach selection, which covers architectural decisions, as well as the configuration of the training process, the validation concept, and test strategy.

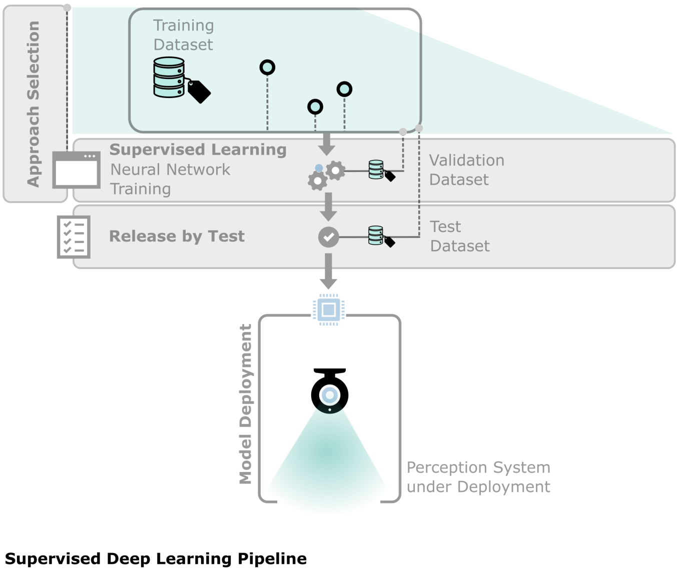


Figure 1.3: A deep learning pipeline, in general, consists of a training dataset, including samples (cyan dots) and associated labels (indicated as black circles around the samples), which is then used for supervised training the neural network. The neural network, in turn, is iteratively evaluated on the validation dataset with respect to defined key performance indicators. Reaching a required threshold on the key performance indicators qualifies the neural network for deployment on the test dataset, which finally allows releasing the model for deployment on the actual application.

In detail, the pipeline is built around data labeling and preprocessing, neural network training, validation, and testing. In supervised deep learning, there is a single large-scale dataset, which in turn separates into a training, validation, and test dataset.

Training, Validation and Testing

The neural network's parameters θ converge during training, determined based on a dataset which consists of the training samples $\mathbf{x}$, and their associated ground truth $\tilde{\mathbf{y}}$ while being guided by an objective function and optimization algorithm. Side-by-side, the training process is monitored by repeatedly evaluating the current state of the model on the validation dataset. Once the training process is optimized as mirrored by the validation result, the model is conclusively evaluated on the test dataset and released on passing. The resulting and final model $f(\mathbf{x};\theta)$ is ready to be used for inference.

Inference

A machine learning life-cycle can be separated into two potentially recurrent and synchronous, distinct capabilities - training and inference. Once the model has converged and has been trained successfully, with respect to an upfront defined set of metrics evaluated on a test dataset, the model is ready for deployment. In this second phase, the neural network deduces a prediction $\hat{\mathbf{y}}$ from a so far unseen, sample $\mathbf{x}$, drawn during deployment. Within a production environment in which the neural network is integrated, these predictions are used as actionable outputs for other components of the overall system.

1.2.2 Learning in Neural Networks

Learning or training of a neural network is realized by adjusting the neural network's current parameters θ, consisting of weights $\mathbf{W}$ and biases $\mathbf{b}$ with the objective to reduce a predefined loss. Usually, this loss is based on ground truth data and a linked cost or objective function, which is, at its core, an optimization task. The number of parameters available in the neural network is termed capacity. The ground truth data is provided either offline: Weights are updated on a set of training samples or online: Weights are updated after each training sample [26]. An exemplary learning rule (see Equation 1.2) is defined by the Hebbian theory [27], in which weights are updated according to the activations (see Sec. A.2) of the predecessor node x_k and the successor node h_j which enclose the weight w_{jk}, and a learning rate ϵ (see Sec. A.2),

$$\Delta w_{jk} \sim \epsilon \, x_k \, h_j . \tag{1.2}$$

Due to the nonlinearity of neural networks with multiple layers and hence a high-dimensional feature space, most objective functions (see Sec. A.2) are non-convex. Therefore gradient-based optimizers are iteratively deployed for the training of neural networks (see Sec. A.2 and Sec. 1.2.3).

1.2.3 From Optimization to Learning

During gradient-based optimization, an objective function's $J(\theta)$ loss is minimized by means of determining θ. However, the overriding priority is the model performance on previously unobserved data, assuming an independent and identical distribution ($i.i.d$) with respect to the training data. This demand for generalization distinguishes machine learning from optimization. The ability to generalize is expressed by the validation loss or test loss, which depicts a previously unobserved input sample's expected error. In machine learning, the objective is thus twofold.

- Minimize the training loss (optimization),
- Minimize the generalization gap between training loss and validation loss (regularization).

Improving model generalization trails along with the law of parsimony, as expressed by Occam's razor [28] and statistical learning theorems of the Vapnik-Chervonenkis dimension [29]: A decrease in model capacity or an increase in the number of training samples decreases the generalization gap's margin. Nevertheless, a decrease in model capacity might also lead to underfitting. The machine learning objectives are ultimately an interplay of optimization and regularization.

1.2.4 Regularization for Neural Networks

Regularization aims to increase the performance of a machine learning approach by decreasing the error regarding previously unobserved data, bridging the generalization gap (see Section 1.2.3). Regularization strategies are manifold, while in general, the strategies function by inflicting constraints on,

- the objective function $J(\theta)$ (soft parameter constraint).
- the parameter set θ (hard parameter constraint).

Constraints can implicitly contribute prior knowledge about the problem or force the architecture into a model with less capacity in order to support generalization. Regularization is a trade-off between increased bias of the expectation $\mathbb{E}$ of an estimator $\mathbb{E}(f(\mathbf{X};\theta) - f^*(\mathbf{X};\theta^*))$ to yield reduced variance of the estimator $Var(\theta)$ [18].

1.3 Object Detection with Neural Networks

This thesis revolves around object detection, which in turn is inseparably embedded into the concepts and pipelines. Therefore, the key considerations on object detection are outlined. Machine learning, and in particular neural networks, are a central building block to enable machines to perceive and relate to their environment [30] based on input from sensory hardware, such as imaging sensors. Computer vision constitutes a sub-type of machine perception, which in particular includes methods to process images into numerical and symbolic information [31].

Definition 1.2 ***(Perception).*** *Machine perception is the capability of a computer system to interpret and transfer sensor inputs into a representation, which consecutively enables a machine to understand and react to its environment [32].*

1.3.1 The Task of Object Detection

Within computer vision, neural networks are deployed for different kinds of pattern recognition and for different tasks. The most important examples of tasks in computer vision today are classification, object detection, and segmentation.

Definition 1.3 ***(Task).*** *A task $\mathscr{T}$ consists of a label space $\mathscr{Y}$ and an objective prediction function $f(.)$, denoted by $\mathscr{T} = \{\mathscr{Y}, f(.)\}$. $f(\boldsymbol{x})$ can be written as $P(\boldsymbol{y}|\boldsymbol{x})$ [24]. A model that takes on a certain task $\mathscr{T}$, is denoted $\theta_{\mathscr{T}}$.*

An object is defined as a spatial coherent semantic concept or pattern affiliated with a class. The task of object detection involves detecting and locating object instances of potentially multiple classes in a sensor measurement, such as camera images. These measurement samples serve as an input $\mathbf{x}$ for the object detector. The output $\hat{\mathbf{y}}$ of an object detector includes the class affiliations c, as well as the location of the objects in the input sample [33].

1.3.2 Object Detection Architectures

Recently a variety of high-performing, high-paced object detectors have been introduced. In general, an object detector, for the most part, follows a complementary approach of: Establishing bounding box hypotheses, generating

Model	Pace [ms]	mAP
ssd_mobilenet_v1_coco	30	21
ssd_inception_v2_coco	42	24
faster_rcnn_inception_v2_coco	58	28
faster_rcnn_resnet50_coco	89	30
rfcn_resnet101_coco	92	30
faster_rcnn_resnet101_coco	106	32
faster_rcnn_inception_resnet_v2_atrous_coco	620	37
faster_rcnn_nas	1833	43

Table 1.1: Pace and mean average precision (mAP) for different object detectors, representative for state-of-the-art object detection architectures. Implementation and evaluation are according to the Tensorflow Detection model zoo [34]. The pace is determined on a 600x600 px image on an Nvidia GeForce GTX TITAN X GPU. The models are trained on the COCO data set (see Sec. 1.4.1) and evaluated according to the COCO mAP, meaning $mAP@[.5:.05:.95]$ [35].

features for each bounding box, and classifying the instances according to the extracted features. In the following, three object detectors, which are representative of state-of-the-art object detection architectures, are assessed in detail[1]. These object detectors form the baseline neural networks for benchmarking the novel approaches in the course of this work. In particular, the RFCN object detector demonstrates a representative trade-off between inference pace and performance (see Tab. 1.1).

SSD: Single Shot MultiBox Detector [36] comprises a feed-forward CNN with multiple layers. The first layers make use of an architecture used for image classification, such as the Visual Geometry Group (VGG) feature extractor [37], the MobileNet architecture [38] or the Inception architecture [39], reduced by the classification layers. Subsequent layers then realize the detection functionality. The detection layers are CNN feature layers decreasing in size. The detection layers evaluate a set of predefined default bounding boxes of different aspect ratios at each location in several feature maps with different scales. Based on these multiple feature maps rather than a single

[1] Tensorflow Object Detection Model Zoo

one, the architecture is able to discretize the class confidence and the shape offset. The endpoints of the feature maps, meaning the predicted bounding boxes with the according class-scores, are finalized by a non-maximum suppression.

One-Stage: Region-based Convolutional Neural Network [40] is based on region proposals (2000 per sample) by selective search, followed by a fully connected CNN that predicts bounding box hypotheses and classification scores concurrently at each position in the input image. Concluding, similar regions are combined into larger ones, producing the final candidate region.

Two-Stage: Region-based Fully Convolutional Network [41] follows a two-stage object detection strategy. The first stage is based on a Fully Convolutional Network (FCN), such as the ResNet-101 [42] architecture, and is used for feature map calculation and region proposal. The region proposal is based on the R-CNN architecture introduced by Girshick et al. [40] and substituted by a Region Proposal Network (RPN). The second stage then executes region classification based on the position-sensitive max-pooled feature maps and the previously proposed regions of interest.

1.4 Large-Scale Datasets and Labeling

The supervised deep learning pipeline is driven by large-scale datasets, which bring together data samples with associated labels. Within this thesis, a main point of contact when attempting to shift from a supervised to an unsupervised paradigm is to unravel the bond and interconnection between annotated large-scale datasets and the training process itself. For this, it is necessary to appreciate the current role of large-scale datasets and to understand common label policies.

1.4.1 Large-Scale Datasets

Deep learning approaches and, in particular, supervised learning approaches are based on feature representation learning driven by large-scale annotated datasets (see *Deep Learning a Paradigm Shift* in Sec. 1.1.1). Further, autonomous perception systems build upon an extensive perception system with different sensor modalities, including cameras, RADARs, and LiDARs. Hereby, cameras are the cheapest and most commonly used sensor modality for object detection [43]. Consequently, available camera sensor samples and their respective labels (see Tab. 1.2) take on a pivotal role in deep learning development.

Dataset	**Images**	**Classes**	**Labels**
COCO dataset [35]	328 k	91	2.5 M
Pascal VOC challenge [44]	11.53 k	20	27.45 k
MNIST [45]	70 k	10	70 k
ImageNet [46]	14 M	> 27	1.03 M

Table 1.2: Representative extract of state-of-the-art object recognition datasets, including COCO, Pascal VOC, MNIST, and the ImageNet database, providing a context on the size and specifications of large-scale datasets for supervised deep learning.

Common Objects in Context (COCO) dataset[2] [35] pursues progress in the development of object recognition in the context of scene understanding, provided by Microsoft in 2014. The benchmark comprises common objects in their natural context, hence COCO. Within 328 k images, 91 object classes and 2.5 M instances are labeled with per-instance segments. Intersection-over-Union is used as the evaluation metric.

PASCAL Visual Object Classes (VOC) challenge[3] [44] is a benchmark for object classification, object detection, object segmentation, and action recognition, mainly contributed by the University of Oxford in 2010. The benchmark comprises consumer photographs collected from the Flickr platform, with high variability in the pose, illumination, and occlusion state. Starting in 2014 the benchmark now includes 11.53 k images, with 27.45 k bounding-box-labeled instances distributed over 20 object classes. The main evaluation metric is the average classification precision.

Modified National Insitute of Standards and Technology (MNIST)[4] [45] is a large-scale digit recognition dataset provided by Yann LeCun at the

[2] COCO: http://cocodataset.org/

[3] PASCAL VOC: http://host.robots.ox.ac.uk/pascal/VOC/

[4] MNIST: http://yann.lecun.com/exdb/mnist/

Courant Institute of Mathematical Sciences at New York University in 2010. The dataset comprises 28×28 px images of digits. 60 k training images and 10k test images are labeled with the digit class information. Error rate or accuracy is used as an evaluation metric.

ImageNet[5] [46] facilitates developing advanced, large-scale content-based image search and image understanding algorithms and providing critical training and benchmarking data for such algorithms, such as the AlexNet [47]. Served by the Stanford Vision Lab in 2009, ImageNet is a large-scale ontology of images organized according to the WordNet hierarchy, with each node comprising thousands of images. ImageNet comprises 14.19 M images, whereas 1.03 M are annotated into 27 high-level object classes. The evaluation metrics are the Top-1 and Top-5 average classification precision.

High-level functions of autonomous vehicles heavily depend on stable and reliable image-based perception capabilities built on visible light camera sensors. Especially object detection architectures (see Tab. 1.1) heavily rely on pretrained feature extractors, also known as backbones. The above-introduced and outlined image datasets are deployed for training feature extractors in the course of this work.

1.4.2 Image Label Policies

Label policies specify how samples $\mathbf{X}$ are annotated ($\rightarrow$) or, in other words, linked to a ground truth $\tilde{\mathbf{Y}}$, which depicts the understanding and knowledge of a system that the model needs to learn. The effort to establish the association between a sample and a label is termed annotation effort; in the most naive assumption, this is depicted as the number or cardinality of samples $\#\tilde{\mathbf{y}}$, which need to be labeled. As machine learning applications become increasingly sophisticated, so do the labels, their formats, and their annotation efforts [48].

Image Label Annotations

The basic label annotation (classification) assigns a single class information to a specific data sample. A more complex label annotation assigns a bounding box for a set of objects within a data sample (localization). A Bounding Box (BB) is a cuboid or, in 2D, a rectangle, hence a convex bounding volume. A bounding volume for a set of objects is a closed volume that

[5] ImageNet: http://www.image-net.org/

completely contains the union of the objects in the set. Usually, the bounding box is aligned with the coordinate system's axes, and it is then known as an axis-aligned bounding box (AABB) [49]. The classification and localization information can be merged to a further label annotation (detection). More advanced detection annotations can take place on the level of abstraction of a pixel instead of a bounding box, meaning the annotation carries the class information for each pixel (semantic segmentation). Temporal information in image sequences is annotated by assigning object IDs on the instance level or time stamps on the sample level. In principle, any kind of information can be mapped to a label vector, further extending the information content of the label annotation and its associated data sample.

Image Label Format
Information captured within a label annotation can come in different forms of representation. Usually, these reflect and fall in line with the deployed neural network architecture and training process requirements. The single value format consists of either a digit $\tilde{y} = 1$, a string $\tilde{\mathbf{y}} = class\_1$, or similar types and usually reflects the association of a sample $\mathbf{x}_i$ with a class or some other quantity $\tilde{y}_i$ that the model needs to predict. The label format is also implicitly determined by the neural network architecture, and especially the output layer of a neural network (see $\hat{\mathbf{y}}$ in Fig 1.2). In particular, for classification tasks, it is common to predict a probability distribution instead of a single value. Therefore, the label format of $\tilde{\mathbf{y}}$ needs to mirror the size of the output layer, which is done by one-hot encoding, also known as an indicator variable. Exemplarily for a class "1" label in a binary classification setting, the indicator variable is the one-hot encoded vector,

$$\mathbf{x}_i \rightarrow \tilde{\mathbf{y}}_i = [0, 1]. \tag{1.3}$$

1.5 Evaluation

The supervised deep learning pipeline builds upon iterative training, which at the same time is monitored through validation and finally released by testing. Validation and testing can be considered as the two pillars of evaluation, which facilitate the learning and deployment of deep learning models. It is to be expected that evaluation has to evolve when shifting from supervised learning to unsupervised domain adaptation. Initially, this requires an understanding of the currently predominant evaluation concepts and metrics, which are given in the following.

1.5.1 Object Detection Metrics

The task of object detection (see Sec. 1.3) is two-fold, including localization and classification. The object detection metrics reflect this [48]. The Intersection-over-Union (IoU) determines the localization performance of detection by taking the set A of predicted object pixels and the set B of true object pixels and calculates their intersection,

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|},\ IoU \in [0,1]. \tag{1.4}$$

This is done for each of the associated object instances by an assignment algorithm [50, 51]. An IoU-based threshold p is used to determine whether a prediction is considered a True Positive (TP),

$$IoU(A,B) \geq p. \tag{1.5}$$

In benchmarks [44], p commonly defaults to: $p = 0.5$. The metrics are then denoted as $metric@[p]$. Additionally, a classification confidence threshold c is defined and incorporates the classification performance as a second condition for a prediction to be considered a TP. The confidence equals the maximum probability value of the prediction $\hat{\mathbf{y}}$.

To quantify object detection performance, a set of metrics is defined. The precision is calculated based on TP and False Positives (FP) and is a measure of the accuracy of the predictions,

$$P = \frac{TP}{TP+FP},\ P \in [0,1]. \tag{1.6}$$

The recall is calculated based on True positives (TP) and False Negatives (FN) and is a measure for the rate of the predictions or the share of relevant instances that the model retrieved,

$$R = \frac{TP}{TP+FN},\ R \in [0,1]. \tag{1.7}$$

The F_1 score is calculated based on the precision and the recall and depicts the trade-off between precision and recall,

$$F_1 = 2\,\frac{P\,R}{P+R},\ F_1 \in [0,1]. \tag{1.8}$$

The correlation between precision (P) and recall (R), the selection of the IoU threshold p, and the confidence threshold c is holistically displayed as precision-recall curves (PRC) [52], which are similar to receiver operating characteristic (ROC) curves - mirroring the central matter of optimizing and trading off recall and precision with respect to performance and functional requirements for operation. Further, a model's performance deviation between two domains $\mathscr{D}_S$ and $\mathscr{D}_T$ is called the domain gap.

1.5.2 Visualizing Feature Spaces

Deep neural networks in perception and computer vision not only attend to complex inputs and outputs but themselves introduce high-dimensional non-linear behaviors. Interpretability is thus limited [53, 54] due to the non-intuitive intermediate connections between layers of a neural network. To increase qualitative explainability and intuition, it is necessary to visualize feature spaces across all layers. This enables analysis and discussion of observations in low-dimensional human-readable and human-interpretable spaces.

A domain is depicted by a specific distribution of its data samples. Neural networks are trained on data with a particular distribution $\mathscr{X}_S$, lacking knowledge about domains $\mathscr{X}_T$ on which they have not been trained. The two domains can be thought of as separate clusters with an inter-cluster distance, domain divergence [55], or domain discrepancy. Dimensionality reduction algorithms can visualize these feature spaces [56] through matrix factorization such as the principal component analysis (PCA) [57], or by graph-based methods such as UMAP [58], or tSNE (see Alg. A.2) [59].

While PCA is deterministic, it is limited to two or three principal components when used in visualization. The graph-based methods are stochastic and based on clustering, thereby making sure that the embedding preserves the pattern in the data during dimension reduction. UMAP preserves local and global structures, while tSNE is capable of preserving local structures between adjacent data samples.

1.6 Transfer Learning and Domain Adaptation

In machine learning, it is assumed that the training data and the data received during inference are drawn from the same distribution and domain ($i.i.d$). However, this assumption is regularly violated, due to deployment shifts and domain shifts in general, when facing real-world applications, such as autonomous driving or biomedical perception systems.
In search for current approaches and concepts, which are able to transfer knowledge between domains, to build upon this Section turns to transfer learning and its sub-types.

1.6.1 Overview Transfer Learning

A transfer learning setup could consist of a task, such as object detection on a target domain but only having the required labeled data to train on a source domain. The two domains might differ in the feature space $\mathscr{X}$ or might have a different data distribution $P(\mathbf{X})$. Transfer learning approaches aim at transferring knowledge acquired in the source domain into the target domain.

For knowledge transfer, a multitude of strategies is available [60]. These are distinguished with respect to which weights are constant or adjusted during transfer learning. When *fine-tuning* or retraining a model, all weights are updated while training on the target domain; the knowledge transfer constituent resides in the initialization of the weights based on the source domain training. In *feature extraction* pre-trained weights remain constant and are utilized for feature extraction on the target domain. This is especially useful in case the feature spaces $\mathscr{X}$ of the input samples of the source and target domain are similar. The resulting feature vector can then be used as a base for further training subsequent layers specifically for the target domain task.

Definition 1.4 ***(Transfer Learning).*** *Given a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and learning task $\mathscr{T}_T$, transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{D}_S \neq \mathscr{D}_T$, or $\mathscr{T}_S \neq \mathscr{T}_D$ [24].*

Transfer learning is an active field of research. Transfer learning is distinguished concerning the domain and task conditions at hand. In the following, the partitions of transfer learning are identified. Unsupervised and self-supervised domain adaptation are described in detail, focusing on domain adaptation as a transductive transfer learning subfield. Finally, current metrics for domain adaptation are outlined.

1.6.2 Transfer Learning Settings

Transfer learning is to be understood with respect to the concerned domains and their association with a task setting and is accordingly categorized (see Fig. 1.4).

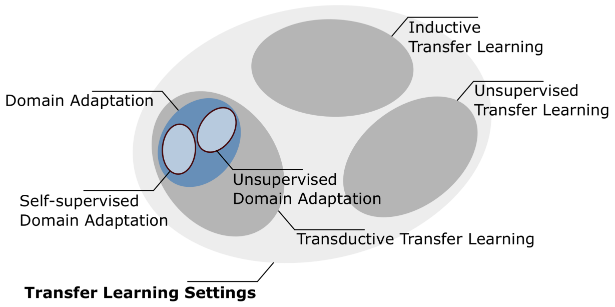

Inductive Transfer Learning
Domain Adaptation
Unsupervised Transfer Learning
Unsupervised Domain Adaptation
Self-supervised Domain Adaptation
Transductive Transfer Learning
Transfer Learning Settings

Figure 1.4: Overview of the different transfer learning settings: Inductive transfer learning, unsupervised transfer learning, and transductive transfer learning, including its sub-setting domain adaptation. Adapted from [2].

In the inductive transfer learning setting, the task on the target domain is different from the task on the source domain. Labeled data is available in the source domain and the target domain. An example of the inductive transfer learning setting is a task changing from classification to object detection. Thus, inductive transfer learning can be understood as task adaptation:

Definition 1.5 ***(Inductive Transfer Learning).*** *Given a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and learning task $\mathscr{T}_T$, inductive transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{T}_S \neq \mathscr{T}_D$ [24].*

In the unsupervised transfer learning setting, similar to the inductive transfer learning setting, the target task is different from the source task. However, unsupervised transfer learning addresses cases in which there are no labels available in either the target domain or the source domain. An example of the unsupervised transfer learning setting is a task changing from classification to object detection, while no labels are available in either domain.

Definition 1.6 ***(Unsupervised Transfer Learning).*** *Given a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and corresponding learning task $\mathscr{T}_T$, unsupervised transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{T}_S \neq \mathscr{T}_D$ and $\mathbf{Y}_S$ and $\mathbf{Y}_T$ are not observable [24].*

In the transductive transfer learning setting, the source and target tasks are the same or closely related, while the source and target domains are different. In this situation, no labeled data is available in the target domain. However, labeled data is available in the source domain. The transductive transfer learning setting sees a constant task, for example, object detection, yet a shift in the domain, while no labels in the target domain are available.

Definition 1.7 ***(Transductive Transfer Learning).*** *Given the source domain $\mathscr{D}_S$ and a corresponding learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and a corresponding learning task $\mathscr{T}_T$, transductive transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{D}_S \neq \mathscr{D}_T$, and $\mathscr{T}_S = \mathscr{T}_D$. Besides, some unlabeled target domain data must be available at training time [24].*

1.6.3 Unsupervised Homogeneous Domain Adaptation

Transductive transfer learning can be further distinguished with respect to a homogeneous and heterogeneous case. The former case is also known as domain adaptation.

Definition 1.8 ***(Homogeneous and Heterogeneous Transfer Learning).*** *Given a source domain $\mathscr{D}_S$, with a corresponding feature space $\mathscr{X}_S$ and a target domain $\mathscr{D}_T$, with a corresponding feature space $\mathscr{X}_T$, in the homogeneous transfer learning case the source domain and the target domain share a feature space $\mathscr{X}_S = \mathscr{X}_T$. While in the heterogeneous transfer learning case, the source domain and the target domain have diverging feature spaces $\mathscr{X}_S \neq \mathscr{X}_T$ [61].*

As mentioned in Subsection 1.6.2, transductive transfer learning can be formulated as a homogeneous domain adaptation problem. Within homogeneous domain adaptation problems, the task in source and target domain is the same $\mathscr{T}_S = \mathscr{T}_T$, and so are the feature spaces of the domains $\mathscr{X}_S = \mathscr{X}_T$, and the marginal probability distributions of the input data are different, $P(\mathbf{X}_S) \neq P(\mathbf{X}_T)$ [24]. Domain adaptation aims at learning a representation that is domain invariant.

Definition 1.9 ***(Unsupervised Homogeneous Domain Adaptation).*** *Given a source domain $\mathscr{D}_S$ and a corresponding learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and a corresponding learning task $\mathscr{T}_T$, unsupervised homogeneous domain adaptation aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$, where $\mathscr{D}_S \neq \mathscr{D}_T$, $\mathscr{X}_S = \mathscr{X}_T$, $P(\boldsymbol{X_S}) \neq P(\boldsymbol{X_T})$, and $\mathscr{T}_S = \mathscr{T}_T$. In addition, unlabeled target domain data must be available at training time [24].*

Homogeneous domain adaptation is partitioned into a supervised, semi-supervised, weakly-supervised, and unsupervised case. The cases range from the supervised case, supplying labeled target data, to the unsupervised case, in which only unlabeled target data is available. Unsupervised homogeneous domain adaptation can also be categorized by type (see Tab. A.12) and a neural network's architectural elements. Discrepancy-based approaches, which build on fine-tuning an existing source domain model, also cover feature extraction approaches (see Sec. 1.6.1). Adversarial-based approaches follow a generator-discriminator architecture. Reconstruction-based approaches ensure intra-domain representation, as well as indistinguishable inter-domain representations [61, 62]. For an overview of the approaches discussed in the following, refer to Tab. A.12.

Generative Models (see a) in Fig. 1.5) combine a generator model θ_G that learns to produce samples, with a discriminator model θ_D, that side-by-side learns to distinguish generated from real samples. Based on source samples or a randomly drawn feature vector, synthetic target samples are generated while preserving the labels of the source samples, which can then be used for training on the target domain [63, 64]. Advanced architectures introduce additional design choices to guide the generator module's training process. Additional losses are defined by additional discriminators [65], and conditions [66, 67]. Parameter-sharing constraints are introduced to favor a shared latent feature space of the source and the target domain [68].

Encoder-Decoder Reconstruction architectures (see b) in Fig. 1.5) [69] combine the encoder network θ_E for representation learning [70] with a decoder network θ_D for the auxiliary input data reconstruction task [71, 72]. The most common example for these architectures are variational autoencoders [73]. The actual training task can then be trained directly, cross-domain, on the shared latent representation [74, 75].

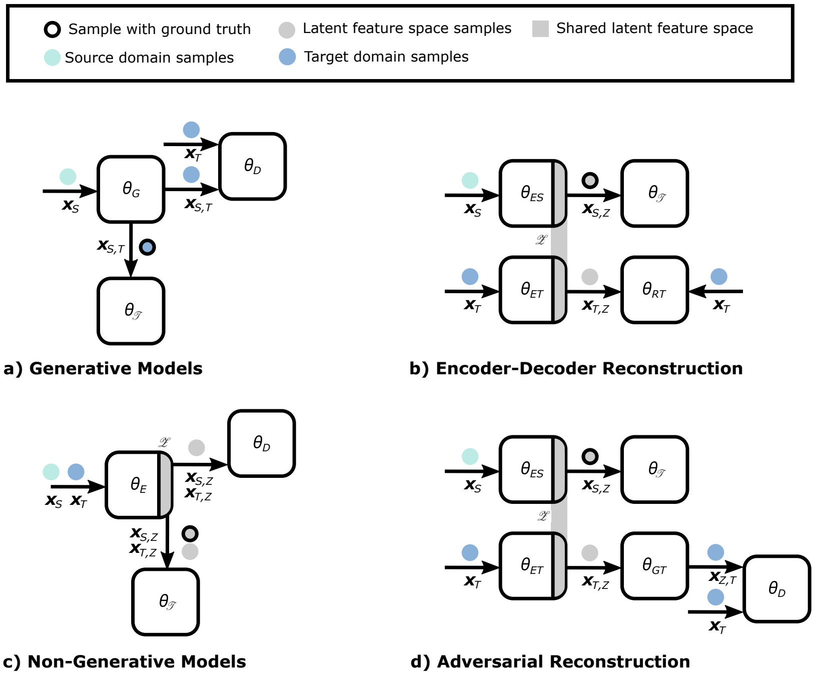


Figure 1.5: Overview of state-of-the-art architectures for unsupervised domain adaptation. Input samples (colored dots) are source domain $\mathbf{x}_S$ (cyan) and respectively target domain samples $\mathbf{x}_T$ (blue). Samples within a shared latent feature space (the gray area) are depicted $\mathbf{z} \in \mathscr{Z}$. Encoded or translated samples are indexed with the domains that capture the transition, such as $\mathbf{x}_{S,T}$ representing a translation from the source to the target domain. A dot with a bold stroke represents a sample with available ground truth $\tilde{\mathbf{y}}$. The architectures comprise different neural network architectural elements such as generative networks θ_G, discriminator networks θ_D, encoder networks θ_E, decoder or reconstruction networks θ_R, and the task network $\theta_{\mathscr{T}}$.

Non-Generative Models (see c) in Fig. 1.5) aim at directly learning an indistinguishable inter-domain representation, termed shared feature space, by approaches such as domain-confusion loss [76], or by explicit parameter-sharing [77]. The model can then be trained using the source domain labels and deployed for inference on the target domain [62].

Adversarial Reconstruction (see d) in Fig. 1.5) combines the notion of auxiliary reconstruction with learning indistinguishable inter-domain representations with GANs and their adversarial loss [77, 78]. The adversarial loss is the difference between the generated [79], and original sample within a domain [80], determined by a discriminator.

1.6.4 Self-Supervised Domain Adaptation

Transductive transfer learning (see Subsection 1.6.2) can be formulated as a domain adaptation problem, in particular as homogeneous domain adaptation [24] if the tasks in source and target domain are the same $\mathscr{T}_S = \mathscr{T}_T$, and so are the feature spaces of the domains $\mathscr{X}_S = \mathscr{X}_T$. Further, the marginal probability distributions of the input data are different for the two domains, $P(\mathbf{X}_S) \neq P(\mathbf{X}_T)$. As is common in supervised learning, self-supervised domain adaptation follows an annotation-based mode of learning [62, 81]. In self-supervised domain adaptation, a model is trained on the target domain data X_T along with a pseudo label $\mathbf{Y}'$ [82]. Pseudo labels are generated based on knowledge from $\mathscr{D}_S$ and $\mathscr{T}_S$ cast into a model's parameters $\theta(.)$. Equation 1.9 is exemplary for class-specific pseudo labels,

$$\mathbf{y}'_i = \begin{cases} 1, & \text{if } i = \arg\max_i \ \theta(\mathbf{x}_T), \\ 0, & \text{else.} \end{cases} \tag{1.9}$$

Definition 1.10 ***(Self-Supervised Homogeneous Domain Adaptation).***
Given a source domain $\mathscr{D}_S$ and a corresponding learning task $\mathscr{T}_S$, a target domain $\mathscr{D}_T$ and a corresponding learning task $\mathscr{T}_T$, self-supervised homogeneous ($\mathscr{X}_S = \mathscr{X}_T$) domain adaptation aims to improve the learning of the target predictive function $f_T(.)$ in $\mathscr{D}_T$ using the knowledge in $\mathscr{D}_S$ and $\mathscr{T}_S$ to generate pseudo labels $\boldsymbol{Y}'$ for $\boldsymbol{X}_T$, where $\mathscr{D}_S \neq \mathscr{D}_T$, $P(\boldsymbol{X}_S) \neq P(\boldsymbol{X}_T)$, and $\mathscr{T}_S = \mathscr{T}_D$. In addition, some unlabeled target domain data must be available at training time [24, 81].

Homogeneous domain adaptation is divided into supervised, semi-supervised, and unsupervised cases. The cases range from the supervised case, supply-

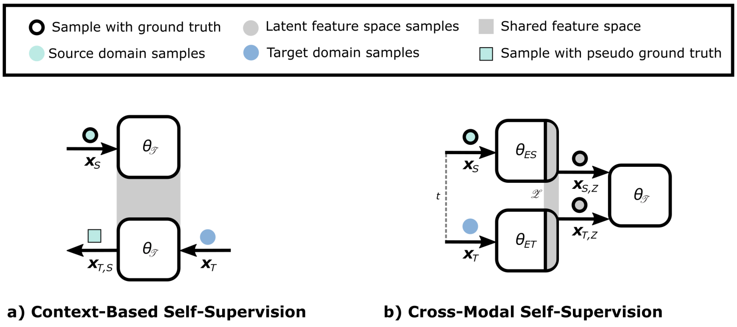


Figure 1.6: Overview of state-of-the-art architectures for self-supervised domain adaptation. Input samples (colored dots) are source domain $\mathbf{x}_S$ (cyan) and respectively target domain samples $\mathbf{x}_T$ (blue). Squared samples depict samples with pseudo ground truth. Dashed lines represent a correspondence between samples, such as an acquisition at the same moment in time t. Shared parameters, and thus a shared feature space, are visualized as gray shapes. Samples within a shared latent feature space are depicted $\mathbf{z} \in \mathscr{Z}$. Encoded or translated samples are indexed with the domains that capture the transition, such as $\mathbf{x}_{T,S}$ representing a translation from the target to the source domain. A dot with a bold stroke represents a sample with available ground truth $\tilde{\mathbf{y}}$. The architectures comprise different neural network architectural elements such as encoder networks θ_E, and the task network $\theta_{\mathscr{T}}$.

ing labeled target data, to the unsupervised case, in which only unlabeled target data is available. Self-supervised domain adaptation is subdivided by the deployed type of self-supervision (see Tab. A.14) and the way this is cast into a neural network's architectural elements. Context-based approaches utilize the context features such as similarity, spatial structure, and temporal structure for self-supervision. Cross-modal-based approaches learn from the correspondence between two samples of different input data characteristics, such as distinct sensor modalities [61, 81]. For an overview of the approaches discussed in the following, refer to Tab. A.14.

Context-based self-supervision (see a) in Fig. 1.6) aims at grouping unlabeled samples [83] based on context [84] within a shared feature space based on the source domain knowledge [82, 85]. A similarity measure, such as spatial distance [86], or temporal [87] context is then utilized as a supervision cue to assign pseudo labels in the target domain [88, 89]. The model can then be trained using the labels in the target domain. Context-based self-supervision can be formulated as an iterative process in which the target domain is assumed to be a sequence of intermediate domains $\mathscr{D}_T = \{\mathscr{D}_{T1}, \mathscr{D}_{T2}, \dots, \mathscr{D}_{Tm}\}$, and the model is continually adapted [90, 91].

Cross-modal-based self-supervision (see b) in Fig. 1.6) makes use of the correspondence of data samples from different input modalities [81]. Sample correspondence of different input modalities, capturing the same state, is used as a supervision signal to learn a feature representation for each of the input data modalities. These feature representations can then be used for recognition and classification tasks in each modality. Further, information available in one modality can be used as a supervision signal (reference) to learn the pattern and deduce the information within the second modality. A prominent example is depth estimation with a single camera, supervised by a LiDAR [92]. Cross-modality is often closely related to sensor modality and can be composed of image data and audio data [93], 3D point cloud data [94], and others.

1.6.5 Preserving Source Domain Knowledge

When fine-tuning a model for a target domain, a fundamental issue is that while the parameters are adjusted to depict the target domain, knowledge of the source domain is lost during domain adaptation. This effect is known as catastrophic forgetting [95] and is usually measured as a drop in a performance measure when evaluated on the target domain as opposed to being validated on the source domain. Often, transferring knowledge to the target domain is not enough. At the same time, preexisting knowledge of the source domain has to be preserved within the neural network. This can be achieved following one of two strategies, an architecture-based or training-based bypassing of catastrophic forgetting (for a detailed overview, see Tab. A.13):

Architecture-based approaches address the capacity and connection design of a neural network. For example, models are extended horizontally, introducing lateral connections to the previously trained layers [96]. This allows the model to preserve source domain knowledge while adapting to further domains in other parts of the neural network architecture.

Training-based approaches address the parameter update. For example, by reducing the plasticity of the essential parameters learned on the source domain [97] or building upon learning shared feature extraction layers and jointly fine-tuning on prediction layers only [60].

1.7 Unresolved Problems and Thesis Outline

Derived from the previous chapters on the state-of-the-art, unresolved problems and open tasks can be determined which aggravate and prevent the efficient extension and unsupervised adaptation of existing deep learning modules and their deployment in perception systems.

Unresolved Problems

- *The common deep learning pipeline is unable to digest unlabeled data:*
  In principle, the development of cognitive perception systems is based on the use of supervised learning. However, the applicability of supervised learning comes to an end together with the availability of labels, limiting the range of an application. In addition, large-scale unlabeled data is squandered and remains untapped for the most part, as unsupervised learning is not enabled to contribute as supervised learning does. Advancement into additional fields of application is dearly paid for by all things concerning the required labeling effort and friction within the conventional deep learning pipeline. Consequently, a holistic concept is missing, which embeds the necessary, novel unsupervised learning approaches that are able to adapt neural networks to emerging domains in a way that allows the neural networks to be consistently deployment-ready and monitorable.

- *Prior epistemic knowledge is not harnessed sufficiently:*
  Although transfer learning is a broadly adopted concept, prior knowledge within models is mostly accessed within feature space representations and purposed as feature extraction. Furthermore, transfer learning is usually followed by a renewed use of supervised learning to impose the implementation of the target task. This renders transfer learning to be an educated initialization approach, which additionally needs a retrofitting of at least some parts of the model. Furthermore, there is no holistic concept to adapt and embed the available prior model knowledge based on unlabeled or continuous data samples. It is, therefore, necessary to restructure the transfer learning approach such that the task model is enabled for direct inference on a transferred shared feature space.

- *A characterization of domain states has been neglected:*
  As training and deployment data is readily assumed to be independent and identically distributed, and purely supervised learning ap-

proaches have been widely adopted, little thought has been given to utilizing domain priors and domain characteristics. However, datasets and domains have been characterized; the knowledge of domain transitions and domain interconnection has not yet seen adequate application within domain adaptation approach selection and domain adaptation itself.

- *Domain adaptation is not implemented for model deployment purposes:*
  Although current research offers an extensive set of domain adaptation approaches, these are mostly developed and tested on domain adaptation itself and remain agnostic to target tasks. Thus, domain adaptation approaches offer limited abilities to actually adapt models and their performance to a target domain. Further, an overall concept is missing, which is able to embed domain adaptation approaches as auxiliaries into the main deployment cycle of neural networks.

- *Supervised learning finds itself within the area of tension between catastrophic forgetting and deployment shift:*
  First, neural networks often find themselves unaware of an occurring deployment shift. Supervised learning in this regard is helpless towards domain shifts, to the degree that the last resort is to reiterate the whole supervised learning pipeline. The neural network is then exposed to catastrophic forgetting. Required are deep learning architectures and training approaches that intrinsically cope with domain shifts while bypassing or making catastrophic forgetting adjustable.

- *Domain adaptation benchmarks are insufficient and lack appropriate process and deployment metrics:*
  Benchmarks for supervised learning-based perception models are available in abundance. However, benchmarks directed at domain adaptation approaches for model deployment are missing. Current supervised learning benchmarks are insufficient in distinguishing inherent domain characteristics within the data. Further, available benchmark datasets come monolithic and fully labeled, providing little incentive and no support for the development and validation of distinct unsupervised domain adaptation approaches along domain shifts. What domain adaptation benchmarks are at hand, they are limited to a set of inadequate domain adaptation metrics, entirely neglecting metrics for the model under deployment.

Objectives and Tasks

Drawing on the unresolved problems mentioned in the previous section, a specific set of tasks is derived. In compliance, the principal objectives of this thesis are:

1. The design of a novel concept for analyzing domain shifts and the subsequent utilization of unsupervised domain adaptation approaches for developing neural networks under deployment. The novel concept and framework are further required to integrate validation capabilities with respect to domain adaptation, as well as deployment metrics. The concept needs to cover the full cycle of continuous development and integration of neural networks under deployment.

2. The development of novel approaches for the extension and domain adaptation of neural networks. The novel approaches need to be label efficient, applicable in a continuous development setting, in control of catastrophic forgetting, and enable the neural network under test to be high performing in the adapted domain or during cross-domain deployment.

3. Extended, novel domain-centered metrics and performance indicators for the quantification of domain adaptation. The re-purposing of performance metrics for neural networks within a domain adaptation setting, with the objective of making domain adaptation approaches quantifiable, comparable, and interpretable.

4. Integration of available supervised learning datasets into a domain adaptation setting by arranging the relevant domain characteristics. The arranged datasets combined with novel metrics establish purpose-designed benchmarks for different types of domain adaptation that allow evaluating the novel concepts and methods.

5. The qualitative and quantitative means for representation of domains, domain shifts, and domain adaptation performances. The descriptive and interpretable representations need to give the developer insight into the underlying domain characteristics of the dataset and metrics to trace the performance development of the domain adaptation process.

6. The implementation of the novel methods and associated tooling in open-source repositories, which are ready to be adopted to a wide range of unsupervised domain adaptation problems within perception.

7. The proof of application and transferability of the novel domain adaptation concept, underlying methods, and metrics, aligned with use-cases of automotive perception and specifically adapting to object detection by night.

Thesis Outline

To improve the unsupervised and continuous deployment of neural networks, the conventional supervised learning pipeline is embedded into a novel concept for domain adaptation, including the classification of domain characteristics in Chapter 2. Consequently, novel approaches for unsupervised domain adaptation are developed and presented in Chapter 3. In order to broaden the evaluation capabilities of domain adaptation approaches under deployment, Chapter 4 introduces extended and specifically developed metrics, culminating into domain adaptation benchmarks. Chapter 5 demonstrates the applicability and feasibility of the novel approaches by evaluation against purpose-designed benchmarks and metrics. Concluding, Chapter 6 summarizes the achieved objectives and work packages. Followed by a discussion of the findings, conclusions, and an outlook for potential research objectives are drawn. Appendix A presents the software and hardware environments, providing the hereupon implemented and open-sourced domain adaptation approaches and further supportive extensions on deep learning fundamentals.

2 Novel Unsupervised Domain Adaptation Concept

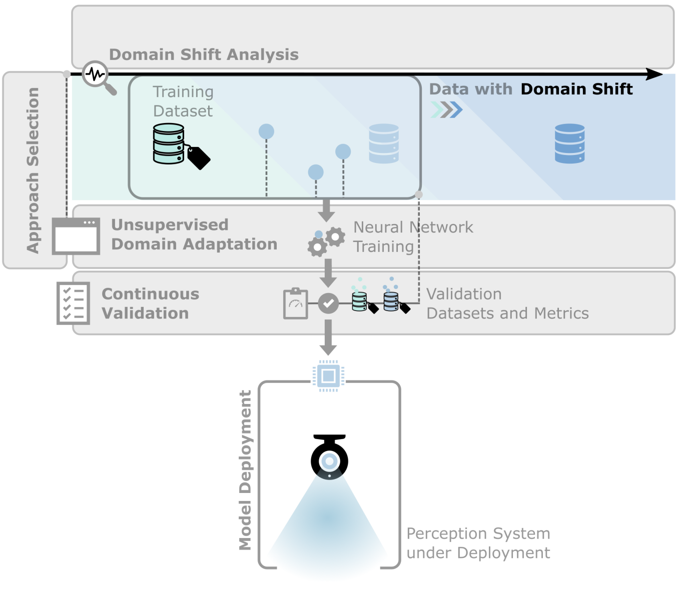
The common supervised deep learning framework (see Fig. 1.3) comprises labeling, data preprocessing, training, validation, and testing, based on a single monolithic dataset. The emergence of domain shifts during the deployment of the neural network is met with reiterating the deep learning framework. Often neglected, systematically addressing domain shifts allows for continuous deployment, validation, and adaptation of the perception system. Hence, the performance and aptitude of the conventional deep learning pipeline are to be complimented. For instance, by involving functional blocks for domain shift analysis, unsupervised domain adaptation, and approach selection, as well as a means for continuous validation, the deep learning pipeline can be enabled towards adaptive model deployment. Accepting domain adaptation as a given and integral part of the deep learning pipeline, this Chapter proposes and introduces a holistic concept and novel framework (see Fig. 2.1) for unsupervised domain adaptation.

2.1 Unsupervised Domain Adaptation Framework

The central and novel functional blocks to extend and adjust the deep learning framework for unsupervised domain adaptation are introduced and itemized in the following:

- **Domain Shift Analysis:**
  The operational design domain has to be analyzed with respect to domain shifts within the entirety of the data - present and potentially forthcoming. The objective is to define the domain shift type and progression (see Sec. 2.2).

- **Approach Selection:**
  The insights of the domain shift analysis are then utilized for selecting an adequate domain adaptation approach (see Sec. 2.3).



Unsupervised Domain Adaptation Framework

Figure 2.1: A deep learning framework, in general, consists of a training dataset, including samples and associated labels, which is then used for training the neural network. The neural network, in turn, is evaluated on the validation dataset with respect to a specified metric set. Reaching a required threshold on this set qualifies the neural network for deployment on the test dataset and, finally, the release as part of the real application (see Fig. 1.3). To address domain shifts within the data (depicted as a change in color from cyan to blue), the novel unsupervised domain adaptation concept introduces functional blocks (depicted as matt gray blocks) to the framework and follows a process accommodating the domain shift (depicted as a black arrow) in general: Domain Shift Analysis followed by the Approach Selection, the Unsupervised Domain Adaptation itself, which is implicitly driven and directed by Continuous Validation.

- **Unsupervised Domain Adaptation:**
  Based on the type of domain shift and deployment requirements, different unsupervised domain adaptation approaches allow for integrating the domain shift into the deep learning framework without additional label efforts (see Ch. 3).

- **Continuous Validation:**
  While adapting to the domain shift, the task performance, as well as the adaptation performance of the neural network, needs to be evaluated and monitored continuously (see Ch. 4).

In conclusion, the unsupervised domain adaptation framework puts the integration of domain shifts at the center of the deep learning framework. Going forward, approaches and methods capable of fulfilling the tasks (domain shift analysis, unsupervised domain adaptation, continuous validation) of the functional blocks introduced by the unsupervised domain adaptation framework are proposed.

The Artificial Petals Taxonomy of Domain Shifts

In the following, novel methods and approaches are introduced concerning their basic functionality and mechanism. To provide a guiding element and support for interpretation by visualization, the synthetic and auxiliary Artificial Petals Dataset is established at this point and remarkably *referenced in italic throughout the text*. The Artificial Petals Dataset makes it possible to depict domain shifts and their characteristics. The dataset is employed throughout all presented methods and approaches and enables cross-referencing and holistic comparisons. The synthetic dataset comprises simplistic flower pictures framed by a black circle (see Fig. 2.2).

The dataset visualizes domain shifts within two dimensions: The major domain-changing characteristic, *the carpel color*, and the minor domain-changing characteristic, *the number of petals*. The flowers with cyan carpels are of the source domain $\mathscr{D}_S$, and those with blue carpels are of the target domain $\mathscr{D}_T$. In favor of simplicity, in the following, the source domain task is defined as a classification regarding the number of flower petals (such as 4). The shift d between domains is approximated as the difference in the number of flower petals.

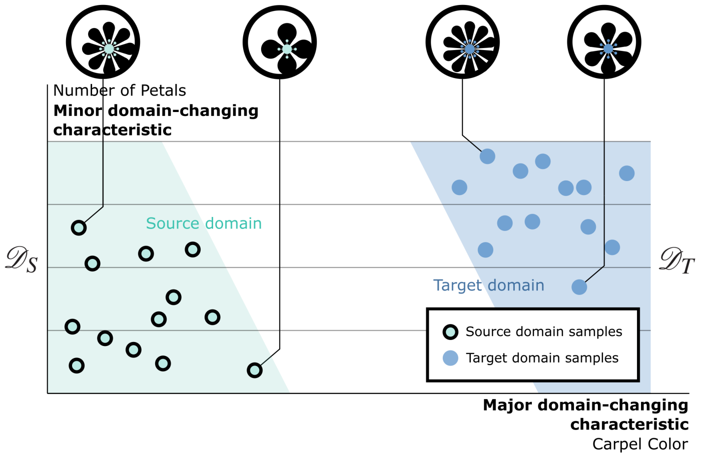


Figure 2.2: Qualitative depiction of two domains within an unsupervised domain adaptation setting (see Sec. 1.6). The source domain $\mathscr{D}_S$ (cyan), and a target domain $\mathscr{D}_T$ (blue). The source domain includes labeled samples enabling supervised training (bold stroke). The target domain samples have no available labels. *Samples of the Artificial Petals Dataset qualitatively depict the relations within the dataset. The domain change emerges along both axes, representing a minor (number of petal transition from* 4 *to* 10*) and major (color of the carpel transition from cyan to blue) domain-changing characteristic*. Adapted from [2].

2.2 Domain Shift Analysis

Domain shifts can emerge from different sources and be of different types. The nuances in their characteristics call for a befitting taxonomy. In the following, a domain shift is considered a covariate shift $P(\mathbf{X}_S) \neq P(\mathbf{X}_T)$ between subsets of a dataset, considered domains $\mathscr{D}_S$ and $\mathscr{D}_T$, with samples $\mathbf{X}_S$ and $\mathbf{X}_T$. The measure of how different the domains are, is denoted $\mathscr{D}_S||\mathscr{D}_T$. The here presented novel taxonomy proposes to distinguish domain shifts based on type and progression: Domain shift type refers to the transition from one domain to the other; it distinguishes between a continuous and a discrete transition of the samples within the dataset. Domain shift progression concerns the sample availability during domain adaptation; a progression is either self-contained when samples are available in each domain; or forthcoming in case a domain is not yet populated when a model is deployed.

Domain Shift Type

The type of domain shift is categorized as either continuous or discrete throughout this work. It is essential to acknowledge that most domain shifts can be represented according to both domain shift types, depending on how the data is acquired, sampled, and processed. For example, the application of object detection at night features a continuous domain shift if the complete transition from day to night is captured within the data or a discrete domain shift in case the night and day conditions are captured without the transitional states. Thus, the domain shift type is either predetermined by the available data or is to be actively shaped and framed by the process of data sampling. Nevertheless, the different types of domain shifts and the different progression characteristics need to be categorized and matched with domain adaptation approaches and assessment approaches.

Discrete Domain Shift:
$\mathscr{D}_S || \mathscr{D}_T \geq m$

A discrete domain shift demonstrates no direct transition of data samples between the source and the target domain. The domains are separated by a threshold margin *m*. The margin is qualitatively depicted as a distance *m* that results in a domain gap. A margin *m* can be a performance measure, a distance within a feature space, or a difference in the label affiliation.

In the Artificial Petals Dataset, an exemplary discrete domain shift emerges between samples with four petals $\boldsymbol{X}_4$ and samples with eight petals $\boldsymbol{X}_8$. The margin is quantitatively depicted as the difference $m = 4$ that results in a domain gap defined by a class shift.

Continuous Domain Shift:
$\mathscr{D}_D || \mathscr{D}_{D+1} \leq m$, with $D \in \{S, \ldots, T\}$

A continuous domain shift consists of data samples, which interconnect the source and target domain with intermediate domains. The adjacent domains are not further apart than a threshold margin *m* and are potentially overlapping.

In the Artificial Petals Dataset, a continuous domain shift emerges between samples with four petals $\boldsymbol{X}_4$ and samples with ten petals $\boldsymbol{X}_{10}$, interconnected by samples with six petals $\boldsymbol{X}_6$ and eight petals $\boldsymbol{X}_8$. The margin is qualitatively depicted as the difference $m = 2$ that allows for a continuous transition between samples with a different number of petals.

Domain Shift Progression

In this work, the progression of a domain shift is categorized as either self-contained or forthcoming. Self-contained domain shifts are characterized by knowledge about the presence of a target domain and available samples in the target domain, while forthcoming domain shifts are by all means undefined. Consequently, applications exposed to a forthcoming domain shift need to rely on methods with an intrinsic mode of continuous adaptation in parallel to the neural network inference.

Self-Contained Domain Shift:
$\mathscr{D}_S \neq \mathscr{D}_T$ with $\mathbf{X}_S, \mathbf{X}_T \in \mathbf{X}$

Common large-scale datasets are static. Any domain shifts within these datasets are self-contained, meaning they can be defined and are, at present, captured and depicted within the dataset.

In the Artificial Petals Dataset, self-contained domain shifts emerge between all available samples in the dataset. For example, this includes samples $\boldsymbol{X}_4$, $\boldsymbol{X}_6$, $\boldsymbol{X}_8$, and $\boldsymbol{X}_{10}$. For each set of samples, for example $\boldsymbol{X}_4$ and $\boldsymbol{X}_{10}$ the domain shift can be defined as a distance $\mathscr{D}_4 || \mathscr{D}_{10}$.

Forthcoming Domain Shift:
$\mathscr{D}_S \neq \mathscr{D}_T$ with $\mathbf{X}_S \in \mathbf{X}$ and $\mathbf{X}_T \notin \mathbf{X}$

A dataset with a forthcoming domain shift, at the time a neural network is trained on a domain $\mathscr{D}_S$, is still in a state of data and, in particular, a domain extension. In a forthcoming domain shift, by definition, not all domains are depicted within the available training data $\mathbf{X}$, and thus only come to exist at a later point in time in the form of additional samples $\mathbf{X}_T$, constituting a further domain $\mathscr{D}_T$.

In the Artificial Petals Dataset, a forthcoming domain shift is the emergence of an additional domain, which is not yet covered within the current dataset. Such a domain, for example, are samples with 12 petals $\boldsymbol{X}_{12}$ *or five petals* $\boldsymbol{X}_5$*. However, it is important to notice that the forthcoming domain remains unknown at the time of acquiring the initial dataset of source domain samples with, for example, four petals* $\boldsymbol{X}_4$*.*

2.3 Approach Selection

When confronted with approach selection for unsupervised domain adaptation, two dimensions clearly stand out during domain analysis: The domain shift type (along the vertical axis in Fig. 2.3) and domain shift progression (along the horizontal axis in Fig. 2.3) within the dataset. It is self-evident that there is an additional dimension of technical requirements that need to be met and thus influence the selection of an appropriate domain adaptation approach. Requirements that form the basis for selection are the realization of the specified task performance, the model's memory requirements for deployment on the target hardware, available computational real-time capability, and manageable annotation efforts or the availability of labeled data. Last but not least, appropriate approaches need to be available which are able to meet the requirements and present domain characteristics.

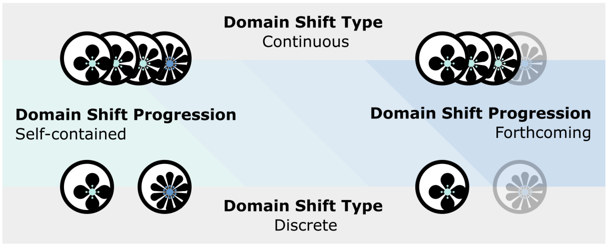


Figure 2.3: The dimensions of approach selection: Domain shift type on the vertical axis and domain shift progression on the horizontal axis. The artificial petals depict the present domain shift as combinations of the two dimensions. The cyan area indicates potential for supervised learning methods, assuming labels are available. The blue area indicates the necessity for implicit or superimposed continuous adaptation, as the domain shift occurs after deployment.

3 Novel Approaches for Unsupervised Domain Adaptation

To complement the unsupervised domain adaptation concept (see Ch. 2), novel unsupervised domain adaptation approaches are developed for discrete self-contained, continuous self-contained, and continuous forthcoming domain shifts (see Sec. 3.1, 3.2, and 3.3). Taken together, the approaches enable the proposed unsupervised domain adaptation concept (see Sec. 2.1) to be applied broadly and highlight its holistic and modular usage potential.

3.1 Sample Supplementation Approach

3.1.1 Overview and Ideation

With the novel unsupervised domain adaptation framework established (see Sec. 2.1), suitable unsupervised domain adaptation approaches need to be developed and expanded upon. Perception, and in particular computer vision, is central to real-world applications and produces an extensive amount of research. Currently, deep learning, in particular, supervised learning, outperforms any other approach to perception and object detection in a wide range of use cases, as will be shown in Sec. 5.3.2. However, supervised learning and its data-driven process introduce unique constraints, such as being dependent on large-scale labeled data and restricting the model's inference performance to samples that are independent and identically distributed (*i.i.d*), and thus within the training data's domain. Further, current state-of-the-art developments on unsupervised domain adaptation address generative models while lacking deployment and validation in application tasks.

The novel sample supplementation approach (SSUP) deploys a first neural network's generative capabilities to supplement the training of a second neural network for object detection. Hereby, the SSUP extends the area of operation of a neural network beyond its source domain without requiring additional label efforts. The novel approach's applicability will be shown in Sec. 5.4.

SSUP's approach to domain adaptation is a generative one. The setting requires samples $\mathbf{x}_i \in \mathbf{X},\ i = 1,\dots,N$ within the same feature space $\mathscr{X}$ with respect to sensor modality, input dimensions and ranges of values, and of different domains $\mathscr{D}$. The domains are the source domain $\mathscr{D}_S = (\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$, including samples and the associated ground truth $\tilde{\mathbf{y}}_i \in \tilde{\mathbf{Y}},\ i = 1,\dots,N$, either an unlabeled target domain $\mathscr{D}_T = (\mathbf{X}_T, \_)$, or an empty domain $\mathscr{D}_E = (\_, \_) \in \mathscr{X}$, which is enclosed by the source domain, however, constitutes of no samples itself (see Fig. 3.1). The available data $\mathbf{X}$ is then used by a generative adversarial neural network (see Sec. 1.6.3) to depict the distribution $P(\mathbf{X})$ and feature space $\mathscr{X}$ of the data $\mathbf{X}$. The generative capability of the generative adversarial neural network is deployed to create additional samples for

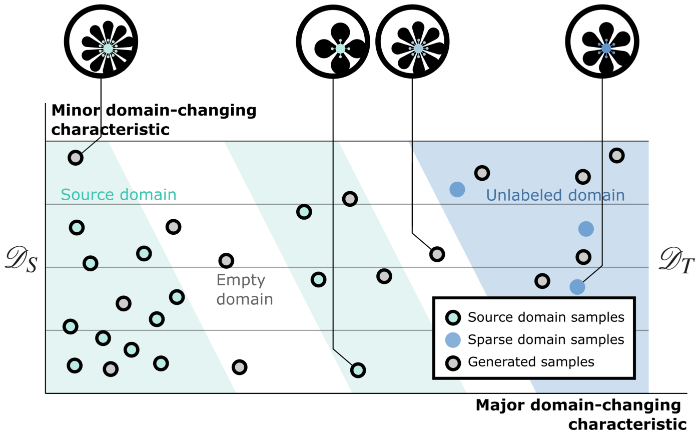
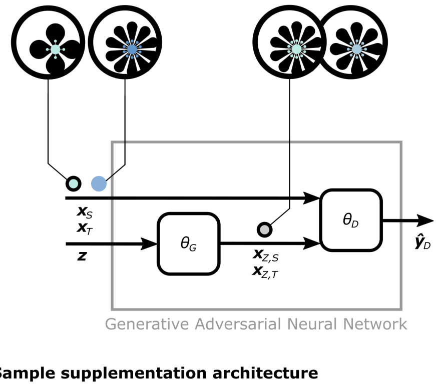


Figure 3.1: Qualitative depiction of two domains within a generative supplementation setting. Given are the source domain $\mathscr{D}_S$ (cyan) and an unlabeled target domain $\mathscr{D}_T$ (blue), with a discrete domain shift, and empty domains (white spaces). The source domain includes labeled samples (indicated by a bold stroke around the circle) and enables supervised training of the task $\mathscr{T}_S$. The domain change emerges along the axes, representing a minor and major domain-changing characteristic in the target domain. Generated samples (gray) allow bridging the empty domains and densifying the domains by adding synthetic samples to preexisting ones.
Samples of the Artificial Petals Dataset qualitatively depict the relations within the dataset. The source domain includes cyan carpels, whereas the target domain includes blue carpels. The generated samples occur across all domains and are depicted with carpel colors ranging from cyan to blue and display a different number of petals. The empty or enclosed domain is defined and expressed by carpel colors between cyan and blue. Adapted from [2].

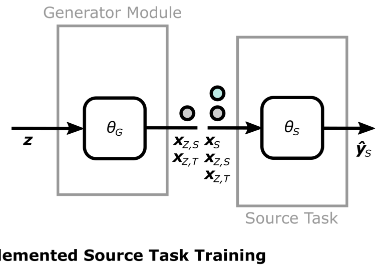
training. In turn, these synthetic samples are added to the initial dataset, used for training a model θ_S performing a source task $\mathscr{T}_S$. Sample supplementation aims for generalization and improving model performance on unseen data in previously empty domains or (sparse) domains with little training samples. Sample generation is delicate, as generative adversarial neural networks are prone to mode collapse or might generate defective samples by introducing label breaks or label switches. These are issues that come with any means of augmentation and need to be guardrailed accordingly with appropriate sanity checks or effective priors. Beyond that, the introduction of supplemental samples, even in the case of mere interpolated samples, allows for trading an increased bias of the model for a reduced variance of the model's predictions and benefits regularization (see Sec. 1.2.4).

3.1.2 Sample Supplementation Architecture

Assuming the availability of an unlabeled target domain $\mathscr{D}_T = (\mathbf{X}_T, \_)$, the SSUP architecture utilizes a generative adversarial neural network to depict the enveloping domain's distribution, enabling the generation of additional data within the domain. By sample supplementation, a model that has been trained on a source task is trained or fine-tuned on the source domain dataset $(\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$again, which has been extended by artificial samples $(\mathbf{X}_Z, \tilde{\mathbf{Y}}_Z)$ created by the generator module θ_G of a generative adversarial neural network (see Fig. 3.2).



(a) Generative adversarial neural network consisting of a generator network θ_G and a discriminator network θ_D. The generator translates from a random noise vector $\mathbf{z} \in \mathscr{Z}$ into artificial samples $\mathbf{x}_{Z,S}, \mathbf{x}_{Z,T} \in \mathscr{X}$. The generator is paired with a discriminator network θ_D which learns to predict $\hat{\mathbf{y}}_D$ an input sample's domain $\mathscr{D}_S, \mathscr{D}_T$ or $\mathscr{D}_Z$.
Samples of the Artificial Petals Dataset qualitatively depict the samples within the approach. The inputs are the source (cyan carpels) and target (blue carpels) samples. And in turn, samples of these same domains are generated. Adapted from [6].



(b) After training the SSUP model, the generator is used standalone to yield additional samples $(\mathbf{X}_Z, \tilde{\mathbf{Y}}_Z)$. Concluding, the source task model is trained on all available domains $\mathscr{D}_S, \mathscr{D}_T$, and $\mathscr{D}_Z$, resulting in improved domain coverage. Adapted from [6].

Figure 3.2

Image-to-Image Translation Neural Network

The image-to-image translation is realized by a generative adversarial neural network, which consists of a generator network θ_G and a discriminator network θ_D (see Fig. 3.2a). The generator network θ_G generates samples $\mathbf{x}_Z \in \mathscr{X}$ of domain $\mathscr{D}_S$ and $\mathscr{D}_T$ by encoding from a random noise vector $\mathbf{z} \in \mathscr{Z}$. The discriminator network θ_D predicts $\hat{\mathbf{y}}_D$ the samples' associations with either being a real sample $\mathbf{x}_S$ or $\mathbf{x}_T$ or being a generated sample $\mathbf{x}_Z$. These predictions are used to determine the loss for training the generator. Generator loss is calculated based on the discriminator's prediction $\hat{\mathbf{y}}_D$, as well as the corresponding domain label $\tilde{\mathbf{y}}_D$ of the input sample. The discriminator is alternately trained on the same loss. The generative adversarial neural network can map and model the representation of the given dataset and its domains $\mathscr{D}_S$, $\mathscr{D}_T$ and then generate additional samples of said domains (see Generative Models in Tab. A.12).

Supplemented Source Task Training

After training the generative adversarial neural network, the generator module can generate additional samples $\mathbf{x}_Z$ (see Generator Module in Fig. 3.2b). The generated samples enable additional training or fine-tuning of a source task model θ_S; this requires that the samples $\mathbf{x}_Z$ implicitly include the respective ground truth $\tilde{\mathbf{y}}_Z$. For example, the ground truth can be available as class affiliations for classification tasks or interrelated temporal information for temporal prediction tasks (such as tracking) or as a ground truth transferred along with an image-to-image translation (such as bounding box coordinates). The neural network θ_S is then trained on the combined dataset $(\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$, $(\mathbf{X}_T, \tilde{\mathbf{Y}}_T)$, $(\mathbf{X}_Z, \tilde{\mathbf{Y}}_Z)$, with respect to a source task $\mathscr{T}_S$ that can be learned based on available labels. After successful supplemented training, the model performs the source task $\mathscr{T}_S$ with increased performance. However, the effects on source domain performance, such as catastrophic forgetting (see Sec. 1.6.5), remain to be analyzed and put in relation to the beneficial effects. Likewise, auxiliary methods are to be developed (see Sec. 4.1.2), which enable to quantify the domain adaptation characteristics of the generated samples.

Approach Configuration

The SSUP approach instantiates hyperparameters, which allow configuring and customizing its deployment. The central hyperparameters for supplemented training revolve around customizing the number and ratio of generated samples for training the task neural network (see Tab. 3.1). Beyond, there are the hyperparameters inherent to any deep learning training process, here, in particular, to train the image-to-image translation neural network (see App. A.1).

Hyperparameter	**Description**
$\#\mathbf{x}_Z$	Number of generated samples
$\#\mathbf{x}_S/\#\mathbf{x}_Z$	Ratio between real and generated samples for training

Table 3.1: Central configuration hyperparameters of the SSUP approach

A detailed study of the hyperparameter setting, their interplay and effects on domain adaptation and catastrophic forgetting during retraining, as well as best practices considering deployment, result from the experiments in Sec. 5.4.4.

3.1.3 Conclusion

Additional training samples can be generated by training and inference of a generative adversarial neural network without requiring any labeling effort. The additional samples and their affiliated labels are then deployed to supplement the training of a source task neural network. Supplemented training is enabled by taking advantage of the ability of generative adversarial neural networks to depict a dataset's distribution, defining the different domains to enable sample generation in a specific target domain. For the first time [6], the approach aims to adapt a neural network to sparse or enclosed target domains.

3.2 Shared Latent Feature Space Approach

3.2.1 Overview and Ideation

SSUP aims to extend the area of operation of a neural network beyond its source domain. The presented extension of a neural network to an additional domain is achieved by combining the supplementation capabilities of generative adversarial neural networks with effective supervised training. As a consequence, the approach inherits the disadvantages of supervised retraining: As of now, any form of fine-tuning is prone to catastrophic forgetting (see Sec. 1.6.5) or necessitates trading-off performance in different domains and further, there is the apparent need for an often nontrivial retraining process. Current state-of-the-art approaches usually tackle these issues by yet another increase in the amount of labeled data, which is by no means a guarantee to bypass catastrophic forgetting nor to ensure domain adaptation and domain coverage. At the same time, image-to-image translation networks are researched as an end in itself, lacking deployment and validation in applications. For the first time, image-to-image translation capabilities aim for cross-domain neural network deployment. The novel shared latent feature space approach's (SHALFS) objective is to learn the translation between two domains and thereby implicitly extend the neural network's field of application by cross-domain adaptation of the target domain without any retraining nor additional labeling.

SHALFS' approach to unsupervised domain adaptation is based on the assumption that images $\mathbf{x}_i \in \mathbf{X}$, $i = 1, \ldots, N$ of different, domains $\mathscr{D}_S = (\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$ and $\mathscr{D}_T = (\mathbf{X}_T, \_)$ with a discrete domain shift, but a common feature space $\mathscr{X}$ (see parameter sharing A.2) hold a common, but initially unknown, shared latent feature space representation in $\mathscr{Z}$ (see Sec. 1.6). The shared latent feature space $\mathscr{Z}$ is then used as a transit zone for image-to-image translation from target domain $\mathscr{D}_T$ to source domain $\mathscr{D}_S$, and vice versa (see Fig. 3.3). Original to our approach, image-to-image translation capabilities are utilized to deploy a model θ_S, that was trained on source domain $\mathscr{D}_S$ to perform the source task $\mathscr{T}_S$, cross-domain on samples $\mathbf{X}_{T,S}$ adapted from the target domain $\mathscr{D}_T$ (see Sec. 3.2.3) [1].

3.2.2 Unsupervised Image-to-Image Translation

The unsupervised image-to-image translation architecture follows an encoder-decoder network architecture while deploying generative neural networks in

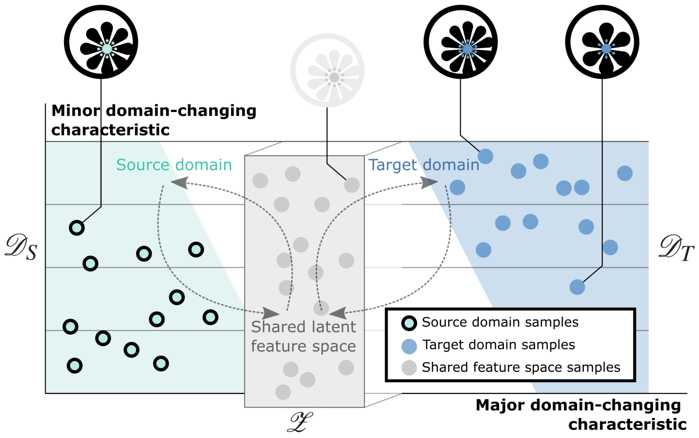

Minor domain-changing characteristic
Source domain
Target domain
$\mathscr{D}_S$
$\mathscr{D}_T$
Shared latent feature space
Source domain samples
Target domain samples
Shared feature space samples
$\mathscr{Z}$
Major domain-changing characteristic

Figure 3.3: Qualitative depiction of an unsupervised image-to-image translation setting. Translation happens between the source domain $\mathscr{D}_S$ (cyan) and the target domain $\mathscr{D}_T$ (blue). Source domain samples (cyan circles) are labeled (bold stroke) and are used for supervised training of the task $\mathscr{T}_S$. The target domain remains unlabeled. The domain change emerges along the axes, representing a minor and major domain-changing characteristic. A common, shared latent feature space $\mathscr{Z}$ is created and enables a common mapping (arrows with dotted lines) of samples within the shared latent feature space (gray) from and to both the source domain and the target domain.
The Artificial Petals Dataset qualitatively depicts the shared latent feature space as a vector with length two, representing the number of petals and the color of the carpel - see gray flower. Adapted from [1].

the decoder paths. The neural network architecture follows a two-stream mechanism evoking a shared latent feature space between the source domain and target domain (see Fig. 3.4).

Shared Latent Feature Space for Stream-Linking

Central to the domain adaptation capability of the neural network architecture is the shared latent feature space implementation (see $\mathbf{z}$ in Fig. 3.4). The shared latent features are induced by introducing a weight constraint in the form of sharing the last layer between encoders θ_{ES} and θ_{ET}, corresponding to sharing the first layer of the generators θ_{GS} and θ_{GT}. Accordingly, the preservation of a shared feature representation $\mathbf{z} \in \mathscr{Z}$ for all input samples $\mathbf{x}_S$ and $\mathbf{x}_T$ is guaranteed.

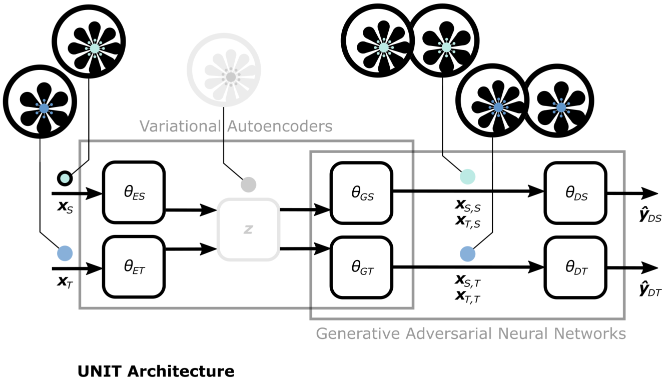


Figure 3.4: Overview of the image-to-image translation architecture (such as UNIT [98]). The two streams take source domain samples $\mathbf{x}_S$ and respectively target domain samples $\mathbf{x}_T$. The encoding of the shared latent feature space $\mathbf{z} \in \mathscr{Z}$ is realized by the source encoder θ_{ES} and target encoder θ_{ET}. The decoding is realized by the source generator θ_{GS} and target generator θ_{GT}. The generators yield translated samples $\mathbf{x}_{S,S}$ (source to source), $\mathbf{x}_{T,S}$ (target to source), $\mathbf{x}_{S,T}$ (source to target), and $\mathbf{x}_{T,T}$ (target to target). For training, the generators are matched by the source discriminator θ_{DS} and target discriminator θ_{DT}.
The Artificial Petals Dataset qualitatively describes the encoding from flowers with blue or cyan carpels into a gray flower in the shared latent feature space. The gray flower representing the number of petals, and the color of the carpel, is then decoded into either a flower with a cyan carpel (source domain) or a flower with a blue carpel (target domain). Adapted from [1].

Encoding and Decoding the Shared Latent Feature Space

The image-to-image translation architecture includes two encoder-decoder networks (see Fig. 3.4). The decoder networks coincide with the generator networks and are thus indexed $_G$.

The source encoder-decoder network consists of θ_{ES}, θ_{GS}. The target encoder-decoder network consists of θ_{ET}, θ_{GT}. The application of encoder-decoder networks is motivated by their capability to reduce the complexity of data distributions, mapping them into lower-dimensional feature spaces: The encoders θ_E map input samples $\mathbf{x} \in \mathbf{X}$ to the shared latent feature space $\mathbf{z} \in \mathscr{Z}$. The decoders θ_G reconstruct the samples $\mathbf{x} \in \mathbf{X}$ from the shared latent feature space $\mathbf{z} \in \mathscr{Z}$ back to their initial feature space and domain $\mathscr{D}_S$, respectively $\mathscr{D}_T$. The encoder-decoder networks are trained with respect to the loss $\mathscr{L}_{EG,S}$ derived from $\mathbf{x}_S$ and $\mathbf{z}_S$ (respectively $\mathscr{L}_{EG,T}$ from $\mathbf{x}_T$ and $\mathbf{z}_T$),

$$\begin{aligned}\mathscr{L}_{EG,S}(\theta_{ES},\theta_{GS}) = \; & KL(q_S(\mathbf{z}_S|\mathbf{x}_S)||p_\eta(\mathbf{z})) \\ & - \mathbb{E}_{\mathbf{z}_S \sim q_S(\mathbf{z}_S|\mathbf{x}_S)}[\log p_{\theta_{GS}}(\mathbf{x}_S|\mathbf{z}_S)],\end{aligned} \tag{3.1}$$

The distribution of the latent feature vector $\mathbf{z}_S$ is denoted $q_S(\mathbf{z}_S|\mathbf{x}_S)$ and randomly sampled from $\mathscr{N}(\mathbf{z}_S|\theta_{ES,\mu}(\mathbf{x}_S),\mathbf{I})$ where $\theta_{ES,\mu}(\mathbf{x}_S)$ is the average output of the encoder θ_{ES}. KL is the Kullback-Leibler divergence [55], a statistical measure of the distance between a first and a second probability distribution. For training the re-parameterization trick [99] is deployed: By adding noise to the reference distribution $p_\eta(\mathbf{z})$ of the shared latent space, which produces outputs $\mathbf{z}_S = \theta_{ES,\mu}(\mathbf{x}_S) + \eta$. The added noise follows a Gaussian distribution, where $\eta \sim \mathscr{N}(\eta|0,\mathbf{I})$ and allows to express a gradient of an expectation as an expectation of a gradient, being differentiable with respect to θ_{ES}.

Learning the Shared Latent Feature Space

For learning the shared latent feature space, the image-to-image translation architecture includes two generative adversarial neural networks (see Fig. 3.4), build upon the two encoder networks θ_{ES} and θ_{ET}. The first, the source GAN, consists of the generator discriminator pair θ_{GS}, θ_{DS}. The second, the target GAN, consists of the generator discriminator pair θ_{GT}, θ_{DT}. The source generator network θ_{GS} generates samples $\mathbf{x}_{S,S}$ and $\mathbf{x}_{T,S} \in \mathscr{X}_S$ of domain $\mathscr{D}_S$ by decoding from the latent vector $\mathbf{z} \in \mathscr{Z}$. The target generator network θ_{GT} generates samples $\mathbf{x}_{T,T}$ and $\mathbf{x}_{S,T} \in \mathscr{X}_T$ of domain $\mathscr{D}_T$ by decoding from the latent vector $\mathbf{z} \in \mathscr{Z}$. Each generator, in turn, outputs two sample streams: The translation stream $\mathbf{x}_{S,T}$ (respectively $\mathbf{x}_{T,S}$), and the reconstruction stream $\mathbf{x}_{S,S}$ (respectively $\mathbf{x}_{T,T}$). For training, only the translation streams are considered. The discriminators θ_D predict the samples' associations with either the source domain or the target domain. These predictions are used to determine the loss $\mathscr{L}_{GAN,S}$ for training the GANs (in Eq. 3.2 exemplarily shown for the source GAN, and respectively holds for the target GAN and the loss $\mathscr{L}_{GAN,T}$),

$$\begin{aligned}\mathscr{L}_{GAN,S}(\theta_{GS,DT}) = \; & \mathbb{E}_{\mathbf{x}_S \sim P(\mathbf{x}_S)}[\log \theta_{DS}(\mathbf{x}_S)] \\ & + \mathbb{E}_{\mathbf{z}_T \sim q_T(\mathbf{z}_T|\mathbf{x}_T)}[\log\ (1-\theta_{DS}(\theta_{GS}(\mathbf{z}_T)))].\end{aligned} \tag{3.2}$$

The loss is calculated based on the discriminator prediction $\hat{\mathbf{y}}_D$ as well as the corresponding domain label $\tilde{\mathbf{y}}_D$ of the input samples.

3.2.3 Cross-Domain Neural Network Deployment

The domain adaptation capability of the unsupervised image-to-image translation enables direct deployment of a model θ_S on the target domain $\mathscr{D}_T$. The model has only been trained to perform a task $\mathscr{T}_S$ on source domain data $\mathbf{X}_S$, and is now enabled to perform the task $\mathscr{T}_S$ on an adapted target domain sample $\mathbf{x}_{T,S}$ as well. This functionality is cast into the unsupervised domain adaptation module:

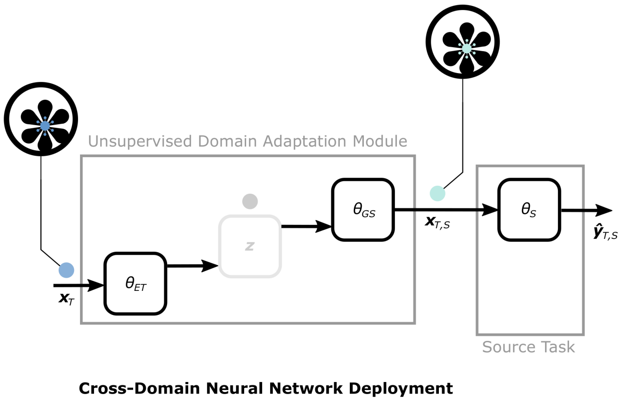


Figure 3.5: Overview of the cross-domain neural network deployment. The unsupervised image-to-image translation network is pruned down, leaving the translation stream from target sample $\mathbf{x}_T$ to its source domain translation $\mathbf{x}_{T,S}$, passing the shared latent feature space vector $\mathbf{z} \in \mathscr{Z}$. The target domain sample $\mathbf{x}_T$ in its translated form $\mathbf{x}_{T,S}$ is then suitable for a model θ_S, being trained on the source domain $\mathscr{D}_S$, to perform a source task $\mathscr{T}_S$, resulting in a prediction $\hat{\mathbf{y}}_{T,S}$.
The Artificial Petals Dataset qualitatively describes the translation from a flower with a blue carpel (target domain) into a cyan flower (source domain), which is then suitable for inference on the source task model. Adapted from [1].

Unsupervised Domain Adaptation Module

The unsupervised domain adaptation module is derived from a pretrained image-to-image translation architecture and is used in inference mode only.

The image-to-image translation capability is pruned, leaving the target to source translation stream,

$$\mathbf{x}_{T,S} = \theta_{GS}(\theta_{ET}(\mathbf{x}_T)). \tag{3.3}$$

The target to source translation stream, in turn, enables cross-domain neural network deployment on the source task $\mathscr{T}_S$.

Source Task

The domain adaptation capability enables the cross-domain deployment of neural networks. The neural network θ_S, required by the definition of the setting of unsupervised domain adaptation (see Sec. 1.6.2), is trained on source domain data $\mathbf{X}_S$, and with respect to a source task $\mathscr{T}_S$. The source task $\mathscr{T}_S$ can be learned based on available labels $\tilde{\mathbf{Y}}_S$ in the source domain $\mathscr{D}_S$. Subsequent to the unsupervised domain adaptation (see Eq. 3.3) the model θ_S is enabled to immediately perform the source task $\mathscr{T}_S$ cross-domain, including the target domain $\mathscr{D}_T$,

$$\hat{\mathbf{y}}_{T,S} = \theta_S(\mathbf{x}_{T,S}). \tag{3.4}$$

An alternative approach to cross-domain adaptation requires the neural network θ_S to be trained on source domain samples $\mathbf{x}_S$ which are mapped to a vector in the shared latent feature space $\mathbf{z} \in \mathscr{Z}$,

$$\hat{\mathbf{y}}_{T,Z} = \theta_Z(\mathbf{x}_{T,Z}). \tag{3.5}$$

During inference, this reduces the number of computational operations by the decoder part θ_{GS}. However, this would again introduce the necessity for retraining on the shared latent feature space.

Inherently, any source task can be integrated into the cross-domain neural network deployment framework. In cognitive perception systems, possible source tasks include, among others: Classification, object detection, object re-identification, and semantic segmentation.

Approach Configuration

The SHALFS approach instantiates hyperparameters, which allow configuring and customizing its deployment. The central hyperparameters for the shared latent feature space revolve around customizing the operation point of the object detector. By configuring the thresholds for the intersection over union and the classification confidence, it is possible to trade off precision and recall of the task neural network under deployment (see Tab. 3.2). A detailed study of the hyperparameter composition, their effects on trading-off recall and precision, and a process for setting a use-case oriented operating point result from the experiments in Section 5.5.5.

Hyperparameter	Description
@IoU	Intersection over union threshold
c	Classification confidence threshold

Table 3.2: Central configuration hyperparameters of SHALFS under deployment.

Beyond, there are the hyperparameters inherent to any deep learning training process, here, in particular, to train the image-to-image translation neural network (see App. A.1).

3.2.4 Conclusion

A shared latent feature space is induced unsupervised by training an image-to-image translation network without requiring any labeling effort in the target domain. Unprecedented, the SHALFS approach to unsupervised domain adaptation follows the intention to provide the means for cross-domain neural network deployment. Original [1] to the approach, the image-to-image translation is interpreted as an unsupervised trained preprocessing module auxiliary to the source task model.

3.3 Continuous Self-Supervision Approach

3.3.1 Overview and Ideation

SSUP and SHALFS aim to extend the performance of a neural network beyond its source domain by including a target domain. However, a discrete domain shift needs to be present, meaning the target domain needs to be distinct from the source domain, and associated samples must be available during training and the domain adaptation process. These approaches do not address scenarios with a continuous domain shift, regardless of the present domain shift progression. State-of-the-art self-supervised learning is deployed to improve and extend a neural network's performance within a given source domain. Continuous domain shifts are neglected in current state-of-the-art self-supervised learning approaches and commonly unattended in practical applications, opening the doors for performance deterioration over time due to deployment shifts.

The subsequent work aims for continuous domain adaptation. The result is the here-developed continuous self-supervision approach (COSS) to domain adaptation. By redesigning the pseudo-label approach (see Sec. 3.3.2) the approach intends to act on data with a continuous domain shift. Making use of the continuous domain shift prior. The secondary objective of this work is the introduction to a cue-based approach (CUEB) to bypass catastrophic forgetting during continuous domain adaptation (see Sec. 3.3.3).

In accordance with Fig. 3.6, the approaches are based on the assumption of a continuous domain shift $\Delta\alpha$ between the source domain $\mathscr{D}_S = (\mathbf{X}_S, \tilde{\mathbf{Y}})$ and target domain $\mathscr{D}_T = (\mathbf{X}_T, \_)$ by intermediate domains $\mathscr{D}_\alpha$ (see Sec. 1.6.4). Samples $\mathbf{X}_S$ and $\mathbf{X}_T$ are within the same feature space $\mathscr{X}_S = \mathscr{X}_T$. Adjacent domains $\mathscr{D}_\alpha$ and $\mathscr{D}_{\alpha+\Delta\alpha}$ are near to superimposition, thus the model's predictions $\theta_\alpha(\mathbf{x}_{\alpha+\Delta\alpha})$ still hold, yet showing minimal deflections characterized by the domain shift. These deflections are subsequently exploited by generating pseudo-labels and, in turn, adapting the model $\theta_{\alpha+\Delta\alpha}$ to the intermediate domain. Starting in the source domain $\mathscr{D}_S$ with an initially supervised trained model θ_S, the self-supervision mechanism is continuously deployed until the model is adapted to the target domain $\mathscr{D}_T$, being enabled to perform the source task on target domain samples $\theta_T(\mathbf{x}_T)$ [2].

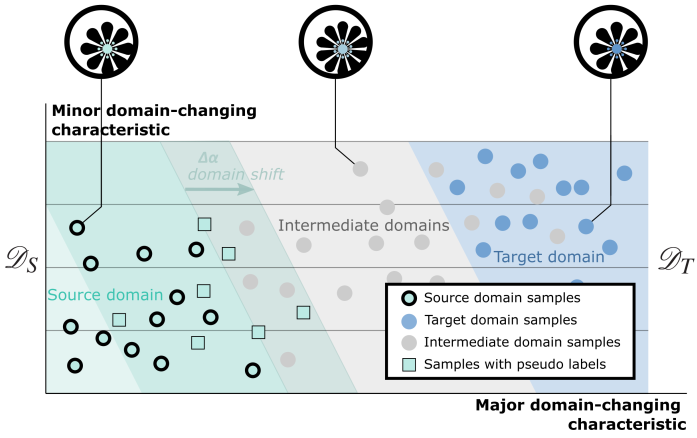


Figure 3.6: Qualitative depiction of a dataset with a continuous domain shift, reaching from the source domain (cyan) to the target domain (blue), passing intermediate domains (gray). Only the source domain includes labeled samples (cyan circles with bold stroke) and enables supervised training on the source domain $\mathscr{D}_S$. The continuous domain change emerges along the axes, representing a minor and major domain-changing characteristic. Assuming that the model's performance is similar in adjacent domains (domain shifted by a small intermediate domain step $\Delta\alpha$) enables generating pseudo-labels on the adjacent domain's samples (cyan boxes with light stroke), in turn, enables self-supervised domain adaptation.
The samples of the Artificial Petals Dataset qualitatively depict the domains and the domain shift within the dataset. The minor (number of petals) and major (color of the carpel) domain change emerges along the axes. Adapted from [2].

3.3.2 Continuous Self-Supervision by Pseudo-Labels

Assuming the availability of a model that has been trained on the source domain, the method utilizes unlabeled data for self-supervised domain adaptation by pseudo-label $\tilde{\mathbf{y}}'_{\alpha+\Delta\alpha}$ generation. COSS is an iterative fine-tuning of a pretrained model on an unlabeled dataset by using the context-knowledge (see Sec. 1.6.4) of a continuous domain shift. The continuous domain shift $\Delta\alpha$ within the dataset is understood in terms of domains $\mathscr{D}_\alpha = (\mathbf{X}_\alpha, _)$and emerges from a context-knowledge which enables to arrange the data in a continuous sequence and permits the assumption that the model's performance is similar for adjacent domains, with sufficiently small domain shifts $\Delta\alpha$, and the associated samples. At last, this also applies for the target domain model $\theta_T(\mathbf{x}_T)$ in $\mathscr{D}_T$.

The advancement of the here presented approach is the direct embedding of the pseudo-label mechanism into the continuous domain adaptation training process of a neural network. In the second step, a design scheme for effectively bypassing catastrophic forgetting is introduced by explicitly integrating a classification-based cue layer into the architecture of the neural network.

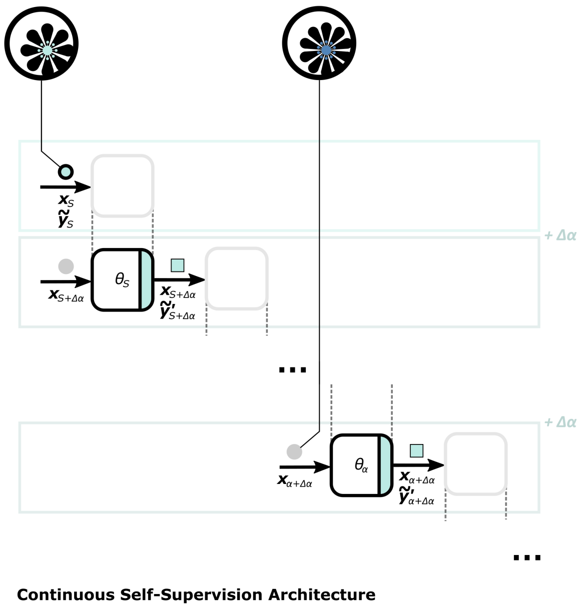

$\mathbf{x}_S$
$\tilde{\mathbf{y}}_S$
$+\Delta\alpha$
θ_S
$\mathbf{x}_{S+\Delta\alpha}$
$\mathbf{x}_{S+\Delta\alpha}$
$\tilde{\mathbf{y}}'_{S+\Delta\alpha}$
...
$+\Delta\alpha$
θ_α
$\mathbf{x}_{\alpha+\Delta\alpha}$
$\mathbf{x}_{\alpha+\Delta\alpha}$
$\tilde{\mathbf{y}}'_{\alpha+\Delta\alpha}$
...
Continuous Self-Supervision Architecture

Figure 3.7: Overview of the COSS architecture. Initially the model is trained supervised on the source domain $\mathscr{D}_S = (\mathbf{x}_S \in \mathbf{X}_S, \tilde{\mathbf{y}}_S \in \tilde{\mathbf{Y}}_S)$ (see first row). The training of a model is depicted as a light gray box. Subsequently, the parameters of the trained model θ_S are then adopted (vertical, dotted lines) to generate pseudo-labels $\tilde{\mathbf{y}}'_{S+\Delta\alpha}$ (see Eq. 3.7 and here depicted as cyan, vertical bar within the black framed box of the model) in the adjacent domain $\mathscr{D}_{S+\Delta\alpha}$. The adjacent or intermediate domain is characterized by the domain shift $\Delta\alpha$. The pseudo-labels and corresponding samples $\mathbf{x}_{S+\Delta\alpha}$ are then, in turn, deployed for training the model, which as a result, is adapted to the intermediate domain. The whole process is iteratively (bold black dots) perpetuated until the target domain $\mathscr{D}_T$ is reached and the model has been conclusively adapted across all domains.
In the first row, the samples of the Artificial Petals Dataset qualitatively depict the continuous domain shift from cyan to blue carpel colors during the progress of the adaptation. Adapted from [2].

Source Task

The new approach assumes a source domain $\mathscr{D}_S = (\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$. Classification and semantic segmentation are examples of possible source tasks $\mathscr{T}_S$. The source task is expected to remain the same during the continuous domain adaptation and is initially learned on available ground truth with labels $\tilde{\mathbf{Y}}_S$ (see Fig. 3.7). The model trained supervised on source domain data $\mathscr{D}_S$ is denoted $\theta_S(\mathbf{x}_S)$.

Pseudo-Label Implementation

The key aspect of continuous self-supervised learning is that a model θ_α is applicable in domains $\mathscr{D}_{\alpha+\Delta\alpha}$ adjacent to the domain $\mathscr{D}_\alpha$ the model θ_α has been trained on. This holds for small domain discrepancies $\Delta\alpha$. Assuming a near superimposition between two adjacent domains $\mathscr{D}_\alpha$ and $\mathscr{D}_{\alpha+\Delta\alpha}$, predictions on the adjacent, intermediate domain samples show valid performance and minimal deflections in the accuracy of the predicted probability distribution,

$$\hat{\mathbf{y}}_{\alpha+\Delta\alpha} = \theta_\alpha(\mathbf{x}_{\alpha+\Delta\alpha}). \tag{3.6}$$

The deflections in accuracy are flattened by the one-hot encoding of the prediction. The control variable j' indexes the prediction $\theta_\alpha(.)_{j'}$, while j indexes the pseudo-label $\tilde{y}'_{\alpha+\Delta\alpha,j}$,

$$\tilde{y}'_{\alpha+\Delta\alpha,j} = \begin{cases} 1, & \text{if } j = \arg\max_{j'} \ \theta_\alpha(\mathbf{x}_{\alpha+\Delta\alpha})_{j'}, \\ 0, & \text{else.} \end{cases} \tag{3.7}$$

The resulting pseudo-label $\tilde{\mathbf{y}}'_{\alpha+\Delta\alpha}$ is an approximate of the ground truth $\tilde{\mathbf{y}}$. The minimal deflections within the prediction are exploited to adapt the model to the shifted domain $\mathscr{D}_{\alpha+\Delta\alpha}$, by generating a pseudo-label for each sample $\mathbf{x}_{\alpha+\Delta\alpha}$ in the dataset, and subsequently retraining the model.

Continuous Self-Supervised Domain Adaptation

The method at its core can be seen as a self-supervised iterative fine-tuning of a neural network pretrained on a classification task in the source domain $\mathscr{D}_S$ (see Alg. 1) across an initially unlabeled dataset, characterized by a continuous domain shift (see Fig. 3.6). An optional extension is to additionally retrain the model, on the source domain samples, subsequent to each training on the intermediate target domain.

Algorithm 1 COSS

```
1: procedure ITERATIVE FINE-TUNING ON PSEUDO-LABELS($\mathscr{D}_S = (\mathbf{X}_S, \tilde{\mathbf{Y}}_S)$ source domain, $\Delta\alpha$ domain shift between two adjacent domains, $\mathscr{D}_\alpha = (\mathbf{X}_\alpha, \_)$ domain-arranged data with $\alpha \in S + \Delta\alpha, \dots, T$ ending in the target domain $\mathscr{D}_T$, $\theta$ untrained model.)

   Initial supervised training of the model:
2:     start in the source domain $S$
3:     train $\theta_S$ on $(\mathbf{X}_S, \tilde{\mathbf{Y}})$

   Iterative fine-tuning along the domain shift $\Delta\alpha$:
4:     while $\alpha < T$ do
5:         inference samples of the adjacent domain $\hat{\mathbf{y}}_{\alpha+\Delta\alpha} = \theta_\alpha(\mathbf{x}_{\alpha+\Delta\alpha})$
6:         generate pseudo-labels $\tilde{\mathbf{Y}}'_{\alpha+\Delta\alpha}$ from predictions $\hat{\mathbf{Y}}_{\alpha+\Delta\alpha}$
7:         train $\theta_{\alpha+\Delta\alpha}$ on $(\mathbf{X}_{\alpha+\Delta\alpha}, \tilde{\mathbf{Y}}'_{\alpha+\Delta\alpha})$
8:         shift into adjacent, intermediate domain $\alpha = \alpha + \Delta\alpha$

9:     return conclusively adapted model $\theta_T$
```

3.3.3 Cue-based Bypassing of Catastrophic Forgetting

Bare continuous self-supervision can not guarantee that source domain performance or any other intermediate performance is preserved (see Catastrophic Forgetting in Sec. 1.6.5) during continuous domain adaptation. Therefore, in the following, a COSS extension is introduced to preserve domain-specific model performance. The novel proposal to bypass catastrophic forgetting can be seen as a cue-accessible horizontal extension of the self-supervised continuous adaptation architecture or a cue-based bypassing of catastrophic forgetting - CUEB.

Cue Implementation

The domain-specific models are accessed via a domain cue $\hat{\mathbf{y}}_{Cue}$. The cue is solely extracted from a given input sample $\mathbf{x}_{(.)}$, inferred by the domain classification model θ_{Cue},

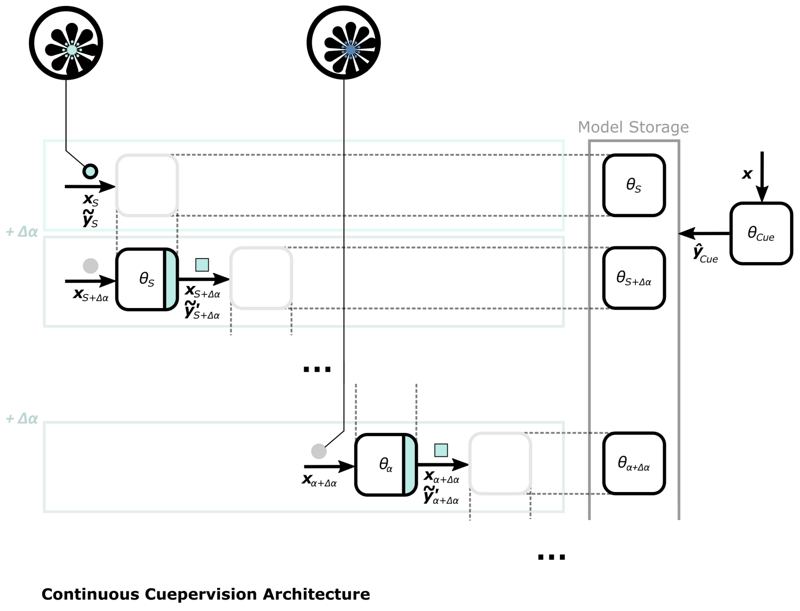
$$\hat{\mathbf{y}}_{Cue} = \theta_{Cue}(\mathbf{x}_{(.)}). \tag{3.8}$$

The auxiliary domain classification model is trained on samples $\mathbf{x}_\alpha$ across all domains and their corresponding domain labels α. The domain labels are derived from the same context-knowledge used to arrange the samples concerning the continuous domain shift (see Alg. 1). Subsequently, the domain-specific model θ_α is selected and deployed for the main task,

$$\hat{\mathbf{y}}_\alpha = \theta_\alpha(\mathbf{x}_{(.)}), \text{with } \alpha = \arg\max \hat{\mathbf{y}}_{Cue}. \tag{3.9}$$

Domain Specific Neural Network Extension

The architectural extension of the neural network means the continuous storing of domain-specific models (see model storage in Fig. 3.8) and their access via a domain cue $\hat{\mathbf{y}}_{Cue}$ that is made available by a domain classification model θ_{Cue}. During inference, the domain cue allows directed access to domain-specific knowledge across all domains: The source domain model θ_S, intermediate domain models θ_α and the target domain model θ_T.



Model Storage
θ_S
$\theta_{S+\Delta\alpha}$
$\theta_{\alpha+\Delta\alpha}$
θ_{Cue}
x
$\hat{y}_{Cue}$
$+\Delta\alpha$
x_S
$\tilde{y}_S$
$x_{S+\Delta\alpha}$
$\tilde{y}'_{S+\Delta\alpha}$
θ_α
$x_{\alpha+\Delta\alpha}$
$\tilde{y}'_{\alpha+\Delta\alpha}$
...
Continuous Cuepervision Architecture

Figure 3.8: Overview of the CUEB architecture. The CUEB architecture is an extension of the COSS architecture (see Fig. 3.7). Both approaches share the pseudo-label generation and subsequent iterative training of the network on pseudo-labels (see Eq. 3.7) in the adjacent domain $+\Delta\alpha$ until the target domain is reached and the model has been conclusively adapted across all domains. The new feature, introduced by CUEB, is that intermediate domain-specific models are stored (see horizontal dashed lines connected to models in the model storage, depicted as a gray, open frame) and assigned to the cue of the current domain. Moreover, initially, an auxiliary domain classification model θ_{Cue} is trained on the samples of all domains $\mathscr{D}_\alpha$ and the corresponding domain labels α. During inference, this allows the model θ_{Cue} to predict $\hat{\mathbf{y}}_{Cue}$ the domain of the sample $\mathbf{x}_{(.)}$ and select the ideally tuned model θ_α for the main task.
In the top row, the samples of the Artificial Petals Dataset qualitatively depict the continuous domain shift from cyan to blue carpel colors during the progress of the adaptation. Adapted from [2].

Low-Level Feature Backbone

Continuous domain adaptation results in high computational efforts, especially due to the repetitive training necessary for adapting to the experienced shifts. To increase computational efficiency during training and inference and to further reduce the required memory of the full model, a low-level feature backbone θ_{low} is introduced for feature extraction.

Due to the common input feature space $\mathscr{X}$ the low-level feature space $\mathscr{Z}$ can be shared, making use of transfer learning (see Sec. 1.6). The pre-trained low-level layers are used for feature extraction, making the low-level feature vector $\mathbf{z}$ shareable between two adjacent domains θ_α and $\theta_{\alpha+\Delta\alpha}$ or two different tasks θ_α and θ_{Cue} (see Sec. 1.6.3).

Sharing (see Sec. A.2) the low-level features by force-setting parameters for two different tasks to equal each other also reduces computational efforts for the inference of CUEB. Both models are deployed on the same input sample $\mathbf{x}_\alpha$. This enables low-level feature vectors $\theta_{low}(\mathbf{x}_{(.)})$ to be calculated only once during inference and then to be used by the domain classification model $\theta_{Cue,high}(\theta_{low}(\mathbf{x}_{(.)}))$, and subsequently by the thereby selected main classification layer $\theta_{\alpha,high}(\theta_{low}(\mathbf{x}_{(.)}))$. This increases computational efficiency, as opposed to computing features from the initial input sample for each task and model.

Approach Configuration

COSS instantiates hyperparameters, which allow configuring and customizing its deployment.

Hyperparameter	Description
o	Opening training epochs
r	Epochs retraining
n	Epochs training on adjacent domain
$\Delta\alpha$	Inter-domain step size
c	Pseudo-label confidence

Table 3.3: Central configuration hyperparameters of the COSS approach under deployment.

The central hyperparameters for COSS and CUEB revolve around customizing the transition and adaptation to adjacent domains (see Tab. 3.3). A detailed analysis of the hyperparameter configurations, their interplay, and effects on transitioning between adjacent domains, as well as best practices under deployment, result from the experiments in Section 5.6.2. Beyond, there are the hyperparameters inherent to any deep learning training process, here in particular for the neural network training process (see App. A.1).

3.3.4 Conclusion

Domain adaptation is implemented in an unprecedented, continuous self-supervised manner by iterative pseudo-label generation and thereon-based fine-tuning of a neural network. Further, a novel cue approach allows for preserving domain-specific knowledge created during continuous domain adaptation and retrieved during inference. Out-of-the-way [2], COSS' approach to domain adaptation addresses continuous neural network deployment, and in the case of CUEB, even without catastrophic forgetting, across continuous domains.

4 Purposed Datasets and Metrics for Domain Adaptation

Deep learning-based perception systems are data-driven, as are the presented novel approaches to unsupervised domain adaptation (see Ch. 2). Hence, benchmarks that fuel the development and validation capabilities of perception systems become increasingly important. Currently available, large-scale datasets (see Sec. 1.4) aim at training and evaluating supervised learning approaches. However, deployed perception systems rely on continuously updated neural network versions and are affected by changing data and annotation accessibility (see Fig. 4.1). Further, initial data is expected to be enriched with labels for supervised training on the datasets; however, when facing growing and expanding datasets, an increasing number of data samples without associated labels need to be handled and trained on. For neural networks in training, or under deployment, within unsupervised domain adaptation settings, only a limited set of metrics are available.

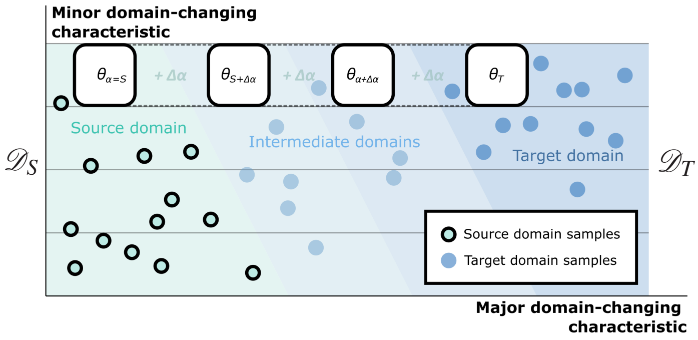


Figure 4.1: Qualitative depiction of a dataset with continuous emerging domain changes, following a domain shift $\Delta\alpha$ along the axes from source domain (cyan) to the target domain (blue) and a neural network's versions θ (displayed within boxes) for continuous domain steps $S,\ S+\Delta\alpha,\ \ldots,\ T$. Adapted from [2].

As unsupervised and continuous domain adaptation approaches (see Ch. 2) gain importance, this chapter offers an extension of current **domain adaptation metrics** (see Sec. 4.1) and introduces two **purpose-designed benchmark datasets** (see Sec. 4.2). In doing so, the designed metrics render domain adaptation interpretable, quantifiable, and comparable. These novel metrics cover unsupervised domain adaptation, as well as metrics for domain adaption regarding continuous domain shifts. Suitable to address the different characteristics (see Fig. 4.1) and needs of unsupervised domain adaptation problems, an advanced assessment concept for domain adaptation needs to be developed. The assessment concept ought to capture the domain adaptation capability as well as the task performance while depicting deployment requirements.

Henceforth, the in this chapter originating domain adaptation metrics, together with the newly tailored datasets, enable in-depth analysis and are indispensable to further drive research and to enable the validation of the entire domain adaptation process.

4.1 Novel Domain Adaptation Assessment Metrics

4.1.1 Overview and Ideation

As large-scale, unlabeled, and continuously expanding datasets emerge, and as they begin to overwhelm supervised learning approaches, there is a need for unsupervised domain adaptation approaches that enable the scaling and continuous deployment of cognitive perception systems. Currently, supervised learning is the general approach when training and deploying deep learning models (see Sec. 1.1.1). Consequently, neural network evaluation, validation, and testing are thus aligned and centered on supervised approaches. Thus, current metrics are expressive when concerning task-specific key performance indicators (see Sec. 1.5.1), yet only a few metrics are available that cover the characteristics of domain adaptation and continuous training processes [1, 2].

This section introduces an extensive, novel domain-centered metric set, with the objective to make domain adaptation approaches quantifiable, comparable, and interpretable. From this point forward, the in this work, developed and defined metrics enable detailed analysis and evaluation of domain adaptation approaches. The guiding principle is that all domain adaptations should be understood in terms of bridging domain shifts (see Fig. 4.1), experienced through continuous and forthcoming domain shifts within a dataset. The starting point of the consideration builds upon an application (see Sec. 4) driven by continuous and unsupervised domain adaptation of a neural network. The novel metric set enables the evaluation of the application by embedding its task performance within the domain adaptation setting (see Sec. 4.1.2). This work further introduces metrics, which allow measuring the degree of applicability of the approaches in practice (see Sec. 4.1.3).

4.1.2 Advanced Task and Adaptation Specific Metrics

Currently, the choice of domain adaptation-specific metrics is limited to catastrophic forgetting (see Sec. 1.6.5) and task-specific performance metrics (see Sec. 1.5.1). The metric set, presented here for the first time, enables the evaluation of a domain adaptation approach by embedding its task performance within a continuously conceived adaptation process (see Fig. 4.1, and Fig. 4.2). In the following, P is the task performance metric (such as the object detection metrics in Sec. 1.5.1). $\mathscr{D}_i$ and $\mathscr{D}_j$ refer to two domains which are separated by a domain shift $\Delta\alpha$, as such that $\mathscr{D}_j = \mathscr{D}_{i+\Delta\alpha}$. Further, a metric $P_{\mathscr{D}_{(.)},\theta_{(.)}}$ is specified by the domain $\mathscr{D}_{(.)}$ the neural network is evaluated on, and the domain $\mathscr{D}_{(.)}$ the neural network $\theta_{(.)}$ has last been trained on, accordingly visualized in Fig. 4.2.

Embedded Task Metrics

The novel embedding of common task performance metrics into the domain adaptation process not only allows to report on the domain adaptation capabilities of a method but, in particular, allows to monitor and analyze the domain adaptation process itself:

In-domain performance P_i defines the identity performance measure, meaning the neural network is evaluated on the domain $\mathscr{D}_i$ it also has been trained on. Domain identity is usually the stage in which performance is best for a specific domain within the domain adaptation process,

$$P_i = P_{\mathscr{D}_i,\theta_i},\ P_i \in [0,1]. \tag{4.1}$$

A special case is the source domain performance P_S.

From there, the *extended domain gap* $DG_{S,i}$ puts the in-domain performance, that has been achieved by adaptation, in relation to the source domain performance,

$$DG_{S,i} = P_S - P_{\mathscr{D}_i,\theta_S},\ DG_{S,i} \in [0,1]. \tag{4.2}$$

The common definition of the domain gap refers to the particular case of the target to source domain gap $DG_{T,S}$.

The *extended domain adaptation* $DA_{j,i}$ depicts the domain adaptation performance by putting an initial or earlier performance of the neural network in perspective to the reached in-domain performance within a reference domain $\mathscr{D}_j$,

$$DA_{j,i} = P_j - P_{\mathscr{D}_j,\theta_i},\ DA_{j,i} \in [0,1]. \tag{4.3}$$

The complement is the *extended catastrophic forgetting* $CF_{i,j}$ that measures the performance deterioration of a neural network θ_j in domain $\mathscr{D}_i$, when adapted to domain $\mathscr{D}_j$,

$$CF_{i,j} = P_i - P_{\mathscr{D}_i,\theta_j},\ CF_{i,j} \in [0,1]. \tag{4.4}$$

Currently, catastrophic forgetting $CF_{S,T}$ refers to the special case of source domain performance, once the domain adaptation reached the target domain. Accordingly, *in-domain catastrophic forgetting* CF_i depicts the potential catastrophic forgetting of a neural network θ_i in domain $\mathscr{D}_i$, once it has been adapted to the target domain $\mathscr{D}_T$,

$$CF_i = P_i - P_{\mathscr{D}_i,\theta_T},\ CF_i \in [0,1]. \tag{4.5}$$

Covering the need to distinguish between the domain the neural network $\theta_{(.)}$ has been trained on, and the domain the neural network is evaluated on $\mathscr{D}_{(.)}$, enables the embedding of the task performance metrics into the domain adaptation process, creating novel embedded task metrics (see Fig. 4.2).

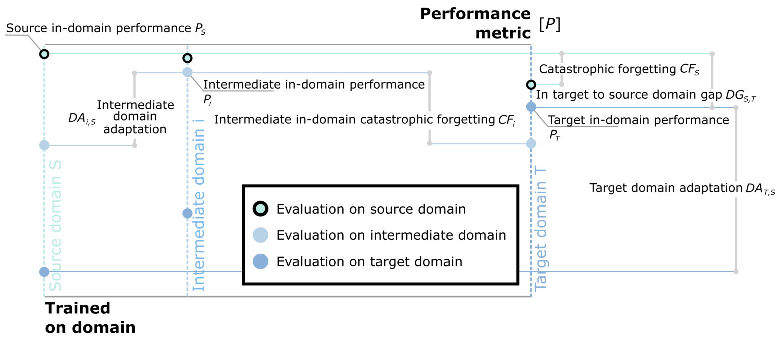

Performance metric [P]
Source in-domain performance P_S
Intermediate domain adaptation $DA_{i,S}$
Intermediate in-domain performance P_i
Intermediate in-domain catastrophic forgetting CF_i
Catastrophic forgetting CF_S
In target to source domain gap $DG_{S,T}$
Target in-domain performance P_T
Target domain adaptation $DA_{T,S}$
Evaluation on source domain
Evaluation on intermediate domain
Evaluation on target domain
Source domain S
Intermediate domain i
Target domain T
Trained on domain

Figure 4.2: Illustration of the novel evaluation system for unsupervised domain adaptation approaches. Aggregation of the regular task performance (performance metric) and the metrics emerging from the nature of domain adaptation (from source to target domain) and continuous domain adaptation (over multiple intermediate domains). The figure depicts the novel unification of performance metrics (y-axis) along the domain adaptation process (x-axis). Adapted from [1, 2].

Further, each metric can also be evaluated across all domains $\mathscr{D}_{\{S,\dots,T\}}$, resulting in the metrics *mean*. In the following, done exemplary for the *mean Performance mP*, and a neural network $\theta_{(i)}$,

$$mP_i = \frac{1}{\#\{S,\dots,T\}} \sum_{n\in\{S,\dots,T\}} P_{\mathscr{D}_n,\theta_i},\; mP_i \in [0,1]. \tag{4.6}$$

The source in-domain performance P_S of the model trained on the source domain is used as a reference point and baseline for the continuous domain adaptation approach. The lower reference point is either the source domain model's performance $P_{\mathscr{D}_T,\theta_S}$ on the target domain or the approximated performance of a random predictor.

Adaptation Process Metrics

Embedding the task metrics into the domain adaptation process enables dedicated adaptation metrics, characterizing the process of domain adaptation itself. For this purpose, this work introduces novel adaptation process metrics such as the fray factor, the continuity factor, and the built knowledge factor:

Fray factor $f_{F,j}$ defines the mean in-domain catastrophic forgetting CF (see Eq. 4.5) of a neural network θ_j, which reached domain $\mathscr{D}_j$ and $CF_{i,j}$ is then evaluated on all $j-1$ domains $\mathscr{D}_S,\ldots,\mathscr{D}_{j-1}$. $f_{F,j}$ quantifies the capability to bypass catastrophic forgetting during domain adaptation and depicts how much it frays from the ideal performance of each domain,

$$f_{F,j} = \frac{1}{j-1}\sum_{i=S}^{j-1} P_i - P_{\mathscr{D}_i,\theta_j},\ f_{F,j} \in [0,1]. \tag{4.7}$$

The *continuity factor* $f_{C,j}$ depicts the mean performance of a neural network θ_j across all j domains $\mathscr{D}_S,\ldots,\mathscr{D}_j$, it has been trained on, up to a domain $\mathscr{D}_j$ it has just reached during the domain adaptation process. When reported along multiple domain adaptation steps, the $f_{C,j}$ allows conclusions regarding the stability and potential reach of the overall domain adaptation process,

$$f_{C,j} = \frac{1}{j}\sum_{i=S}^{j} P_{\mathscr{D}_i,\theta_j},\ f_{C,j} \in [0,1]. \tag{4.8}$$

Built knowledge factor $f_{K,j}$ is a variant of the continuity factor, with $f_{C,T} = f_{K,T}$. Depicting the mean performance of a neural network θ_j when evaluated, across all T domains $\mathscr{D}_S,\ldots,\mathscr{D}_T$. $f_{K,j}$ characterizes domain generalization capabilities of the domain adaptation approach,

$$f_{K,j} = \frac{1}{T}\sum_{i=S}^{T} P_{\mathscr{D}_i,\theta_j},\ f_{K,j} \in [0,1]. \tag{4.9}$$

Adaptation Quantity Metrics

In order to quantify domain shifts, gaps, and adaptation capabilities, it is necessary to link domain characteristics to a measurable quantity. Within a machine learning pipeline, there are different options to access domain characteristics, as expressed by samples. The primary access is to analyze the samples of the domains within a dataset. Secondary access to domain characteristics is by the inference of the model with the samples of a dataset. Domain characteristics can then emerge through the prediction performance or an uncertainty estimation for the model. For the latter, an already trained model is required, and for reliable prediction performance, labeled samples are inevitable. Thus, this work proposes model-agnostic approaches to reach adaptation quantity metrics based on dimensionality reduction of the input samples' features.

The tSNE [59] algorithm maps samples onto a two-dimensional space, concerning their distance in the high-dimensional image feature space (see Sec. 1.5.2). By mapping image samples from different domains such as source and target domain $\mathbf{x}_S \in \mathscr{X}_S$, $\mathbf{x}_T \in \mathscr{X}_T$ and the translated samples $\mathbf{x}_{T,S} \in \mathscr{X}_S$ by the tSNE algorithm, quantitative similarity measures are obtained, between target domain $\mathscr{X}_T$ and source domain $\mathscr{X}_S$. Small distances represent the considerable similarity between any two samples or domains. Between to data cluster center's $(\mathbf{c}_{\mathscr{D}1}, \mathbf{c}_{\mathscr{D}2})$, the Euclidean norm or inter-cluster distance,

$$d_{Eu}(\mathbf{c}_{\mathscr{D}1}, \mathbf{c}_{\mathscr{D}2}) = ||\mathbf{c}_{\mathscr{D}1} - \mathbf{c}_{\mathscr{D}2}||_2, \tag{4.10}$$

or the Mahalanobis distance [100], between a probability distribution $Q_{\mathscr{D}1}$ and a cluster center $\mathbf{c}_{\mathscr{D}2}$, with respect to the inverse covariance matrix $S^{-1}_{\mathscr{D}1}$,

$$d_{Ma}(Q_{\mathscr{D}1}, \mathbf{c}_{\mathscr{D}2}) = \sqrt{\frac{\sum_{i=0}^{N}(\mathbf{c}_{\mathscr{D}2,i} - \mu_{\mathscr{D}1})^T S^{-1}_{\mathscr{D}1}(\mathbf{c}_{\mathscr{D}2} - \mu_{\mathscr{D}1})}{N}}, \tag{4.11}$$

approximate the domain characteristics such as domain shifts $d_{s,Eu}(\mathbf{c}_S, \mathbf{c}_{T,S})$ or $d_{s,Ma}(Q_S, \mathbf{c}_{T,S})$ and domain discrepancies $d_{d,Eu}(\mathbf{c}_S, \mathbf{c}_T)$ or $d_{d,Ma}(Q_S, \mathbf{c}_T)$. The variance $VAR_{\mathscr{D}1}$ or the standard deviation $\sigma_{\mathscr{D}1}$ of the samples $\mathbf{x}$ mapped by tSNE contributes insights into the distribution of specific domain,

$$Var_{\mathscr{D}1} = \sqrt{\frac{\sum_{i=0}^{N}(\mathbf{x}_{\mathscr{D}1,i} - \mu_{\mathscr{D}1})}{N-1}}. \tag{4.12}$$

$$\sigma_{\mathscr{D}1} = \sqrt{\frac{\sum_{i=0}^{N}(\mathbf{x}_{\mathscr{D}1,i} - \mathbf{c}_{\mathscr{D}1})^2}{N}}. \tag{4.13}$$

This is especially of interest for the translated samples $\mathbf{x}_{T,S} \in \mathscr{X}_S$, as it allows to expose defects in the translation model, such as mode collapse. These transferred metrics can be visualized in 2D, are interpretable, and computational efficient.

4.1.3 Applicability and Adaptation Specific Metrics

Currently, domain adaptation approaches are mainly evaluated based on losses that are used during the training of the domain adaptation models. A secondary approach is to qualitatively evaluate the semantic adaptation capabilities by human assessment. However, both evaluation strategies overlook the need to quantitatively depict the domain adaptation performance for actual deployment. When applying domain adaptation, the approaches face deployment requirements, and a particular approach's adaptation characteristic limits the deployment opportunities. These requirements and characteristics are articulated as principal metrics for deployment and are annotation efficiency, sample efficiency, model efficiency, computational efficiency, and run-time efficiency. The set is by no means exhaustive and is intended to be extended with respect to the specific deployment setting at hand.

Applicability and adaptation specific measures

Each applicability and adaptation-specific measure, taken after the domain adaptation process E_T, is referenced against the same measure E_S taken from the initial neural network before domain adaptation. In this way, the initially supervised trained neural network serves as a baseline during deployment. Further, the resulting normalization of the metrics to range from 0, meaning no application of the domain adaptation is possible, to 1, meaning optimal conditions for deployment of the method, allows for comparability across approaches within the limitations E_{lim} of a particular deployment setting.

Annotation efficiency e_A depicts the annotation effort during domain adaptation $E_{A,T} = \#\tilde{\mathbf{y}}_T$ (see Sec. 1.4.2). The annotation effort is put in relation (see Eq. 4.14) to the initial annotation effort in terms of the number of source domain labels $E_{A,S} = \#\tilde{\mathbf{y}}_S$. This gives, e_A is 1 if no further manual labeling is required during domain adaptation, and 0 if the number of manual labels equals or exceeds the number of labels $E_{A,lim} = \#\tilde{\mathbf{y}}_S$ necessary for supervised training. When assuming a constant task $\mathscr{T}_S = \mathscr{T}_T$ during domain adaptation, the annotation effort can be considered constant. As the annotation effort ceases to be constant during domain adaptation, it is necessary to estimate and introduce a factor that maps the change of effort based on the annotation effort of a single source domain sample to the effort for a sample in the current domain.

Sample efficiency e_S measures the number of samples that need to be stored together with their labels $E_{S,T} = \#(\mathbf{x}_T, \tilde{\mathbf{y}}'_T)$ in order to be available during the domain adaptation process for training. The sample efficiency is put in relation (see Eq. 4.14) to the initial sample efforts $E_{S,S} = \#(\mathbf{x}_S, \tilde{\mathbf{y}}_S)$ for supervised training. The limits of sample availability $E_{S,lim}$ usually originate from available memory or legal restrictions such as data protection regulations. e_S is 1 if no samples $E_{S,T} = 0$ need to be stored for domain adaptation, and 0 if the necessary number of samples for domain adaptation can not be stored $E_{S,lim}$, or equal the initial sample efforts for supervised training.

Annotation efficiency e_A and *sample efficiency* e_S are calculated with respect to the efforts during domain adaptation $E_{A/S,T}$, as well as the efforts during initial supervised training $E_{A/S,S}$, limited by a marginal effort $E_{A/S,lim}$,

$$e_{A/S} = \begin{cases} \max\ (0, 1 - \frac{E_{A/S,T}}{E_{A/S,S}}), & \text{if } E_{A/S,T} < E_{A/S,lim} \\ 0, & \text{else.} \end{cases} \tag{4.14}$$

Model efficiency e_M is defined by the increase in model size during the domain adaptation process. Model efficiency e_M puts the model's memory requirements before $E_{M,S}$ and after $E_{M,T}$ domain adaptation into relation (see Eq. 4.15). The limited model size $E_{M,lim}$ arises from hardware constraints such as available memory to store or transfer the model onto an edge device. e_M is 1 if model size remains constant or decreases during domain adaptation and 0 if it surpasses the memory resources for model storage.

Computational efficiency e_C describes the number of operations or weight updates $E_{C,T} = \tau_T$ that are necessary during domain adaptation of the neural network in relation to the operations $E_{C,S} = \tau_S$ necessary for supervised training (see Eq. 4.15). Computational efficiency e_C is 1 if the number of operations equals the number of operations $E_{C,S}$ necessary for supervised training in the source domain. Computational limitations $E_{C,lim}$ occur in case the domain adaptation process needs to keep real-time requirements. Thus, limited by the hardware's computational capacity or the number of samples that need to be trained on, the domain adaptation process can only require a limited number of operations per adaptation step. Computational efficiency e_C is 0 if the number of necessary operations $E_{C,T}$ exceed the actionable operations $E_{C,lim}$.

Run-time efficiency e_R refers to the average time (or the number of operations) needed for inference on a single sample after the domain adaptation process is concluded. Run-time efficiency e_R puts the run-time after domain adaptation $E_{R,T}$ in relation (see Eq. 4.15) to the model's initial run-time $E_{R,S}$. Run-time efficiency e_R is 1 if the adapted model's run-time equals the initial model's run-time. The e_R is 0 for the maximum feasible inference run-time $E_{R,lim}$ as defined by the application's run-time requirements.

Model efficiency e_M, *computational efficiency* e_C, and *run-time efficiency* e_R are calculated with respect to the efforts during domain adaptation $E_{M/C/R,T}$, as well as the efforts during initial supervised training $E_{M/C/R,S}$, limited by a marginal effort $E_{M/C/R,lim}$,

$$e_{M/C/R} = \begin{cases} \min\ (1, \frac{E_{M/C/R,S}}{E_{M/C/R,T}}), & \text{if } E_{M/C/R,T} < E_{M/C/R,lim} \\ 0, & \text{else.} \end{cases} \tag{4.15}$$

Aggregation of applicability and adaptation specific metrics

For the first time, the applicability and adaptation metrics are formulated for domain adaptation processes under deployment.

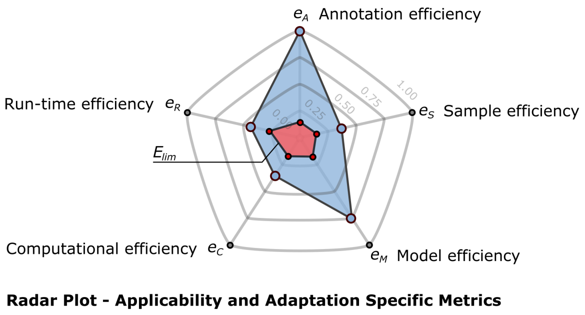


Figure 4.3: For illustration, theoretical applicability and adaptation-specific metrics aggregated into a radar plot. The range of the values is from the center out, $e \in [0,1]$. The efficiency metrics e plotted along the respective axis, span the deployment area (blue). The application-specific limiting boundary conditions $\mathbf{E}_{lim}$ applied to the respective axis result in the no-deployment area (red). Adapted from [1].

Merged, the subsequently presented measures define a set of central approach characteristics (see the red area in Fig. 4.3). Invoked by the problem statement's requirements or resulting from the system and implementation limitations (see Fig. 4.3 and App. A), the applicability and adaptation specific measures are transferred into an aggregated set of metrics, within boundary conditions.

4.2 Purposed Benchmark Datasets

4.2.1 Overview and Ideation

As large-scale, unlabeled, and continuously expanding datasets emerge, and as they begin to overwhelm supervised learning approaches, there is a need for fully controllable and configurable benchmark datasets, designed with the purpose to enable the development and evaluation of unsupervised domain adaptation approaches. This section introduces two novel benchmark datasets. The leitmotif is the explicit specification of domain shifts (see Fig. 4.1) within a dataset.

In the novel **rotatedMNIST dataset** (see Sec. 4.2.2), the domain-changing characteristic emerges by rotations of the MNIST (see Sec. 1.4.1) source domain samples. An innovation and key characteristic of the rotatedMNIST dataset is the fully controllable, continuous domain shift. Based on the configuration, this enables the evaluation of domain adaptation approaches for discrete and continuous domain shifts.

By generation of the **HeartSeg dataset** (see Sec. 4.2.3), this work breaks new ground for domain adaptation within semantic segmentation of biomedical, sequential images. The dataset is purpose-designed to develop domain adaptation approaches in the spatial and temporal domains.

Going by the example, these datasets shall be used as a reference for designing further domain adaptation-purposed datasets or serve as toy-problems for developing domain adaptation methods.

4.2.2 rotatedMNIST Dataset

Overview and Ideation

The rotatedMNIST dataset[1] [2] is an extension of the MNIST, handwritten digits dataset [45]. While the rotatedMNIST dataset's task remains digit classification, the novel dataset is enhanced for domain adaptation by simple, continuous, or discrete, counterclockwise rotation (see Fig. 4.4). Bearing the potential and characteristics of a large-scale dataset with a continuous and forthcoming domain shift, the dataset aims at providing a novel go-to benchmark dataset for continuous domain adaptation (see Sec. 3.3) in a fully adjustable setting. Further advantages of the rotatedMNIST dataset are the low computation and memory requirements per sample.

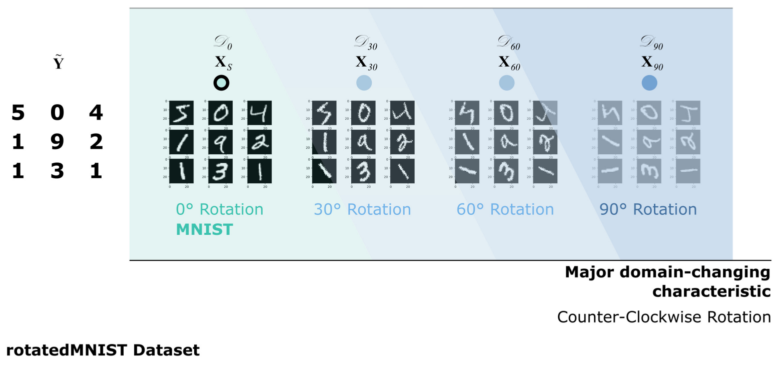


Figure 4.4: Exemplary illustrations of the MNIST dataset samples $\mathbf{X}_S$, and their according digit annotations $\tilde{\mathbf{Y}}$ (0° MNIST source domain $\mathscr{D}_0$). Further samples of the rotatedMNIST dataset are depicted from left to right, according to their major domain-changing characteristic. The dataset arises through rotating the MNIST dataset by small-angle steps, evoking a continuous domain shift relating to the spatial orientation of the digits. The counter-clockwise rotations for 30° ($\mathscr{D}_{30}$), 60° ($\mathscr{D}_{60}$), and 90° ($\mathscr{D}_{90}$) are shown exemplarily, for nine samples of the MNIST dataset. Adapted from [2].

[1] rotatedMNIST dataset: https://osf.io/qgj5d/

Features

The rotatedMNIST dataset is based on the MNIST dataset of handwritten digits, which has a training set of 60000 samples, and a test set of 10000 samples. The MNIST dataset $\mathbf{X}$ consists of $N = 70000$ handwritten digit samples $\mathbf{x}_i \in \mathbf{X},\ i = 1,\dots,N,\ \mathbf{x}_i \in \mathbb{N}^{28\times 28}$ and the respective, one-hot encoded labels $\tilde{\mathbf{y}}_i \in \tilde{\mathbf{Y}},\ i = 1,\dots,N,\ \tilde{\mathbf{y}}_i \in [0;\ 1]^{10}$. A sample $\mathbf{x}_i$ is a gray-scale image (see samples in Fig. 4.4). The depicted digit within an image is centered with respect to its center of mass and respectively cropped, resulting in the feature space $\mathscr{X}_0$ of 28×28 pixels with pixel values in the range of $[0;\ 255]$. The MNIST dataset is extended by simple, continuous, counterclockwise rotation of all samples by a set of angles $\alpha \in [0; \alpha_{max}]$ starting at 0° with α_{max} being the maximal rotational shift and the target domain (for example 90° in Fig. 4.4), resulting in the rotatedMNSIT dataset's controllable, continuous domain shift.

Labeling Policy

The one-hot encoded label $\tilde{\mathbf{y}}_i$ indicates one of the digit classes [0; 9] pictured in the corresponding sample $\mathbf{x}_i$, independent of the state of rotation α. The continuous domains $\mathscr{D}_\alpha = (\mathbf{X}_\alpha, \tilde{\mathbf{Y}})$ consist of $\mathbf{x}_{\alpha,i} \in \mathbf{X}_\alpha$ and $\tilde{\mathbf{y}}_i \in \tilde{\mathbf{Y}}$. The labels remain unaffected by the rotations and are thus adopted unchanged. For the purpose of demonstrating domain adaptation capabilities, the labels $\tilde{\mathbf{Y}}_{\alpha\neq 0}$ are used for validation only and are not accessed during training.

4.2.3 HeartSeg Dataset

Overview and Ideation

The HeartSeg dataset[2,3] [3] (see Fig. 4.5) focuses on semantic understanding of ventricle segments and dimensions in medaka (Oryzias latipes) along the cardiac cycle captured in image sequences. The medaka fish is used as an in-vivo model organism [101, 102] for a variety of subjects in biomedical research [4].

Ventricle segmentation, in particular, contributes to the understanding of cardiovascular diseases and the groundwork for the autonomous perception of ventricular dimensions. The novel dataset is designed for the development

[2] HeartSeg dataset: https://osf.io/snb6p/
[3] Extended HeartSeg dataset: https://osf.io/uyk79/

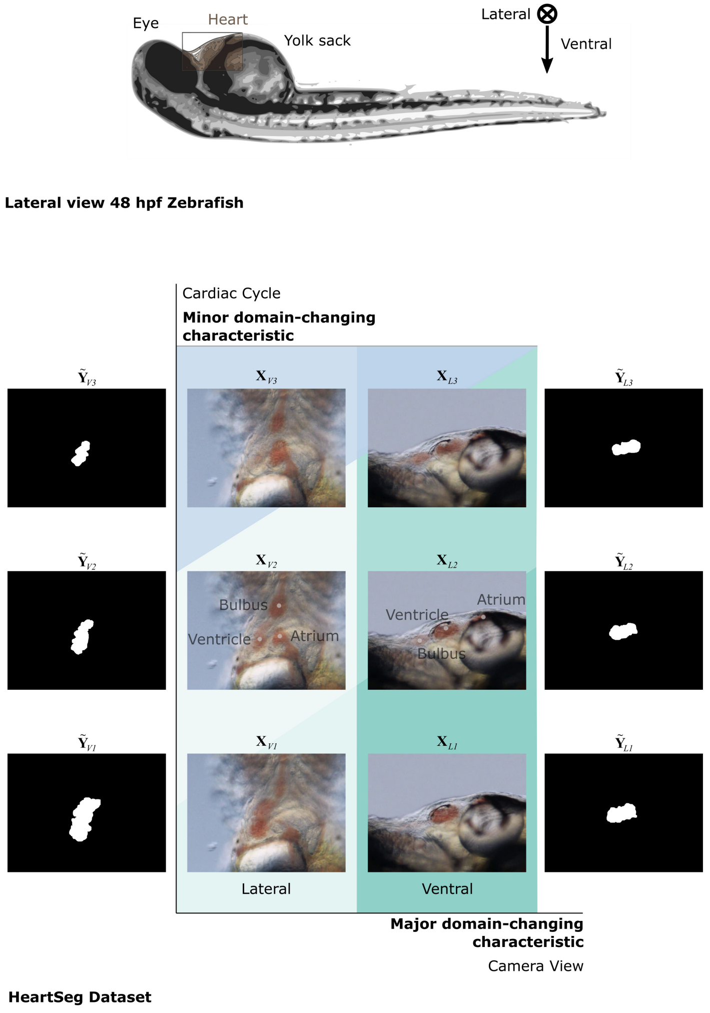

Eye
Heart
Yolk sack
Lateral
Ventral
Lateral view 48 hpf Zebrafish
Cardiac Cycle
Minor domain-changing characteristic
$\tilde{\mathbf{Y}}_{V3}$
$\mathbf{X}_{V3}$
$\mathbf{X}_{L3}$
$\tilde{\mathbf{Y}}_{L3}$
$\tilde{\mathbf{Y}}_{V2}$
$\mathbf{X}_{V2}$
$\mathbf{X}_{L2}$
$\tilde{\mathbf{Y}}_{L2}$
Bulbus
Ventricle
Atrium
Atrium
Ventricle
Bulbus
$\tilde{\mathbf{Y}}_{V1}$
$\mathbf{X}_{V1}$
$\mathbf{X}_{L1}$
$\tilde{\mathbf{Y}}_{L1}$
Lateral
Ventral
Major domain-changing characteristic
Camera View
HeartSeg Dataset

Figure 4.5: On the top, an exemplary illustration of a 48 hpf wildtype zebrafish in lateral view is featured (adapted from [103]). Displayed are the HeartSeg dataset samples $\mathbf{X}$ and their according annotations encoding the ventricle segment $\tilde{\mathbf{Y}}$ as binary label masks. Given, are three ventral samples (V1-V3) of a recording in temporal sequence (bottom to top) and three lateral samples (L1-L3). In the second sample, the descriptions of the ventricle, bulbus arteriosus, and atrium are included in the image. Adapted from [3, 12].

and evaluation of domain adaptation approaches and transfer learning approaches. In this respect, the dataset is tailored to accommodate multiple domain shifts. In the temporal domain, the domain shift of the heart segments is along the cardiac cycle, from diastole to systole. In the spatial domain, the domain shift is between different sensor poses (lateral and ventral). In the semantic domain, the domain shift is between different circulatory vessels (ventricle, atrium, and bulbus arteriosus).

Features

The HeartSeg dataset (see Fig. 4.5) is based on an RGB-image feature space $\mathscr{X}$. The dataset is recorded as image sequences (at 17 fps with 25 or 50 images each) of the medaka's ventricle (1-2 days after hatching of medaka embryos from the fertilized egg). The images are captured from either a ventral or lateral view, which can be differentiated into two domains within the source domain. The images are taken with a 640×480 px resolution. The dataset is recorded by a stationary camera, with fixed focus (SMZ18 Nikon 6x magnification), featuring 59 image sequences. The dataset is split into a training set (725 ventral samples from 29 sequences, 500 lateral samples from 20 sequences) and a test set (250 ventral samples from 5 sequences, 250 lateral samples from 5 sequences). The training set and test set are disjoint.

Labeling Policy

The dataset features binary, semantic, dense pixel-level annotations of a medaka's ventricle segments. Annotations have been carried out by domain-level experts and are made available for all sequences with an overall volume of 1725 samples. Each frame has been annotated individually by manual labeling, supported by the brush-tool of the pixel annotation tool [104]. Annotation of a single frame takes approximately 25 seconds. The labeling policy ensures unperforated label masks. If some foreground is visible (such as pigmentation), it is considered part of the ventricle segment.

5 Applications in Unsupervised Domain Adaptation

5.1 New Mobility Concepts and their Potential

A prediction based on the German In-Depth Accident Study estimates a 15 percent decrease in accidents [105] in longitudinal traffic until 2060 due to progressive automation in Germany [106]. The Bundesanstalt für Straßenwesen [107], and the World Health Organization [108] expect beneficial effects of vehicle automation for traffic safety: Predicting a diminution of the margin for driving errors by the vehicle driver, as well as increased functional safety of the algorithms of autonomous and automated vehicles. The entry of autonomous vehicles into series production is bound to open up new possibilities for individual mobility and public transport. To this day, advanced driver assistance systems were mainly focused on safety and comfort benefits [109]. However, with autonomous vehicles emerging, more eruptive developments in the overall system are to be expected. This growth is already evident in mobility as a service, car-sharing platforms, hybridization of public transport, autonomous parking, and people movers. By 2025, sales of more than 3.5 million vehicles worldwide with level 4 [110] driving functions are expected. The added-value potential associated with the introduction of highly automated driving in Germany has been estimated at up to 8.8 billion Euros. This positive economic starting point is also reflected in the forecasts regarding the sales figures for artificial intelligence-based systems in the automotive sector. Besides, the automotive industry has far greater potential for exploitation: Deep learning-based applications will also penetrate the processes along the entire value chain of the Original Equipment Manufacturers (OEM) and their suppliers, enabling mobility as a service exceeding the mere sale of autonomous vehicles [111].

5.1.1 Economic Potential

The concept of autonomous driving can be understood in terms of the definition of the fourth degree of automation, given by the BASt: "In a defined use case the automated driving platform [system] completely executes the lateral and longitudinal guidance. Here, the driver does not have to monitor the system. Before exiting the use case, the system calls on the driver to assume the driving task with sufficient lead time. If the takeover does not take place, the system returns to the risk-minimal state" [107]. According to the World Health Organization (WHO), more than 1.2 million lives are claimed by traffic injuries every year. In addition to being a health care issue, road traffic accidents are a development issue for low- and middle-income countries, costing those governments approximately 3% of the Gross Domestic Product (GDP) [108].

5.1.2 Safety Potential

In Germany in 2016 alone, 3206 people perished due to traffic injuries, and 2.6 million further traffic accidents occurred according to the Statistisches Bundesamt [112]. However, autonomous vehicles might soon sidestep some of those traffic accidents. Therefore, to estimate the capabilities of autonomous vehicles, regarding their ability to avoid and prevent traffic accidents, a discriminate comparison of the overall performance between humans and autonomous vehicles is inevitable. This comparison becomes possible once the functional specification and the technical limits of autonomous vehicles move into series development. At the same time, past and present traffic accident data of conventional driver-controlled vehicles can form the basis for direct knowledge of the significant accident causes and their changes over time (see Fig. 5.1). A prediction based on the German In-Depth Accident Study (GIDAS) shows an estimated 15 percent decrease of accidents in longitudinal traffic until 2060 due to progressive automation in Germany [106]. In accordance the Bundesanstalt für Straßenwesen (BASt) expects beneficial effects of vehicle automation for traffic safety: They see a diminution of the margin for driving errors by the vehicle driver, as well as increased functional safety of the algorithms of autonomous vehicles [107].

5.2 On Implementation

The implementation details on neural network architecture, the training hyperparameters, and respective tooling are given in detail in the appendix (see App. A.1). The implemented neural network architectures and their previously unseen deployment enable direct application in domains with continuous and discrete domain shifts. Further, the implemented methods enable

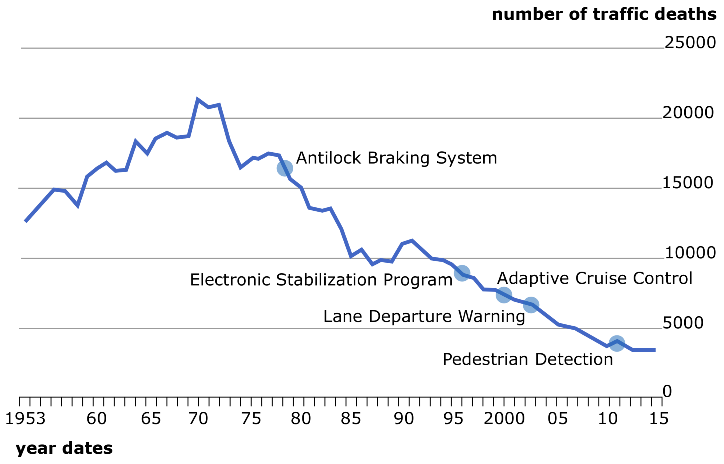


Figure 5.1: Development of traffic deaths in Germany according to [105], displaying the product launch of selected Advanced Driver Assistance Systems (ADAS) from 1953 until 2015.

continuous deployment or cross-domain adaptation. Research and development of novel approaches are enabled by building upon the here presented, publicly available implementation and repositories. The repositories for each novel approach are designed to ensure the accessibility and reproducibility of the developed methods. Besides, by providing the source code[1,2,3] and validation pipeline, the novel methods are open to being extended by further research or transferred to other problem statements, datasets, or neural network architectures. Providing trained neural networks, not least, enables direct inference and deployment of the approaches and models on real-world problem statements, such as night perception for autonomous driving and advanced driver assistance systems. The results and documentation given in the repositories are elaborated in detail and discussed in the following sections on application and results.

[1] SSUP and SHALFS repository: https://osf.io/snmwt/
[2] COSS and CUEB: https://osf.io/qgj5d/
[3] HeartSeg repository: https://osf.io/snb6p/

5.3 On Application and Experiments

Perception under deployment is becoming an increasingly complex task in a continuously changing environment subject to dynamic conditions [5]. Hence, the context of cognitive perception systems moves from large-scale annotated datasets to continuously expanding unlabeled datasets. Based on the newly designed concept for unsupervised domain adaptation and the novel developed approaches SSUP, SHALFS, and COSS (see Ch. 2), algorithms need to be implemented and applied to models and applications. The experimental results need to prove the functionality of the developed methods, and their beneficial characteristics need to be evaluated according to established and newly designed metrics (see Ch. 4). From the presented findings, a general application to a wide range of problems within continuous and unsupervised domain adaptation is deducible. For the first time, the novel unsupervised domain adaptation concept demonstrates all aspects of continuous development, including delivery, validation, and integration of neural networks under deployment.

Notably in automated and autonomous vehicle deployment, the underlying perception systems are subject to different versions, and development stages. Due to the long service life of ground-based vehicles, a significant problem for automobile manufacturers arises when autonomous vehicles are brought into the market, implicitly introducing modules that depend on continuous development and deployment. This renders the development of a perception system that is finalized from the beginning neither economical nor realistic, rather the systems will have to frequently adapt to newly faced requirements outside of the initial operational design domain (ODD).

Driving at night is a safety-critical component within autonomous driving [113]. Even as traffic is reduced during night hours, in 2006, 32.2% of road fatalities in the EU occurred during the night hours (6 pm to 6 am). In 2005, 45% of all pedestrian fatalities in the EU occurred during night hours. These statistics are similarly reflected in the USA and Japan. Normalized to a vehicle mile, this approximately results in a three to four times higher fatality rate during night hours. This imbalance in the fatality rate is traced back to human perception: "Visual acuity, contrast sensitivity, spatial resolution, distance perception and reaction time all deteriorate as overall light levels do" [114]. Consequently, night conditions require autonomous perception systems and advanced driver assistance systems.

5.3.1 Autonomous Driving Night Domain Datasets

In perception applications for autonomous driving, deep learning architectures are enabled by large-scale autonomous driving datasets (see Tab. 5.1) and affiliated benchmarking, such as Cityscapes[4] [115], and KITTI[5] [116].

Dataset	Images	Classes	Labels
Cityscapes [115]	20 k	30	380 k
KITTI [116]	14 k	8	350 k

Table 5.1: Representative extract of large-scale object detection datasets for autonomous driving, lacking large-scale night domain data, providing a context on size and specifications of large-scale datasets for supervised deep learning.

The state-of-the-art datasets and benchmarks for object detectors and other models lack diversity concerning nighttime conditions. For instance, the night domain share within foundation datasets is neglectable for supervised training: COCO (0.23%), ImageNet (0.03%), Pascal VOC (1.24%), KITTI (0.00%). Nevertheless, in autonomous driving, object detection needs to operate under conditions such as low contrast, homogeneity of background, little color information, and low signal-to-noise ratio [117], as experienced in tunnels, due to a heavily overcast sky and, in particular at night. Early on, small-scale datasets such as the iROADs dataset [118] in 2014, and the LISA-Night Dataset [119] in 2012, indicate the need for night domain data for object detection research. Until recently, the lack of publicly available large-scale night domain autonomous driving datasets made it impossible to train or validate data-driven object detection approaches on the night domain. A similar lack is observed for weather conditions, such as rain, fog, and snow [35, 44, 115, 116].

KAIST multi-spectral Dataset[6] [120] aims at all-day perception within autonomous driving, enabling object detection, drivable region detection, depth estimation, and localization. Provided by the Korea Advanced Institute of Science and Technology in 2019, the dataset covers urban, and residential

[4] Cityscapes: https://www.cityscapes-dataset.com/
[5] KITTI: http://www.cvlibs.net/datasets/kitti/
[6] KAIST multi-spectral Dataset: http://multispectral.kaist.ac.kr/

Dataset	Images	Classes	Labels
KAIST multi-spectral [120]	95 k	3	> 305 k
Nighttime Driving [121]	35 k	19	50
BDD 100K [122]	100 k	10	> 1.1 M
Argoverse [123]	85 k	15	> 12.5 k
Boxy [124]	200 k	1	> 1.9 M
Exclusively Dark Image Dataset [125]	7.3 k	12	7.3 k

Table 5.2: Overview of state-of-the-art object detection datasets, which include data in the night domain.

driving environments. It provides co-aligned thermal and RGB cameras. The data has been recorded during sunrise, morning, afternoon, sunset, night, and dawn. The dataset provides around 95k images with around 305k bounding box annotations.

Nighttime Driving Dataset[7] [121] targets semantic image segmentation of nighttime scenes. Introduced in 2018, by the Computer Vision Lab of ETH, it provides 50 pixel-wise semantic segmentation annotations for testing on the night domain, and 35k unlabeled images ranging from daytime to twilight to nighttime.

BDD100K Berkeley Deep Drive Dataset[8] [122] focuses on real-world autonomous driving. Provided by the Berkeley Deep Drive Consortium, the large-scale dataset of 100k videos comprises 100k annotated key-frames. The annotations range from object bounding boxes with occlusion and truncation information and class labels to lane markings, drivable areas, and full-frame semantic segmentation. The images have been recorded in New York and the Bay Area, including Berkeley and San Francisco.

Argoverse Dataset[9] [123] strives to facilitate autonomous driving through highly detailed maps. Made available by ARGO AI in 2019, Argoverse is a 3D object tracking and motion forecasting dataset, including two high-

[7] Nighttime Driving Dataset: http://people.ee.ethz.ch/ daid/NightDriving/
[8] BDD100K: http://bdd-data.berkeley.edu/
[9] Argoverse Dataset: https://www.argoverse.org/

definition maps. The sensor set includes LiDAR and camera modalities. Most important in this context is the share of nighttime recordings within the dataset of 85k images.

Boxy Dataset[10] [124] pursues the progress within vehicle detection on freeways. Contributed by BOSCH in 2019, the large-scale dataset includes a diverse range of domains such as sunny, rainy, and nighttime driving. With more than 1.9 million 3D bounding box annotations with pixel-level accuracies on 200k images, it allows to benchmark object detection in depth.

Exclusively Dark Image Dataset[11] [125] supports research on object detection and image enhancement within low-light environments. Generated in 2019 at the University of Malaya, ExDark is a collection of 7363 low-light images from very low-light environments with ten gradations and twelve object classes annotated on both image class level and local object bounding boxes.

5.3.2 Object Detection by Night

For object detection at night, research has been pursued with rule-based approaches such as histogram equalization [126], and image contrast enhancement [117, 127, 128]. These approaches can be understood as preprocessing, translation, or mapping directed from the target domain to the source domain. Approaches targeting autonomous driving focus on detecting the road [129] or the vehicles' rear lights [130] under night conditions, based on explicit rules or classifiers. Object detection has been addressed on the sensory level by relying on different sensor modalities such as LiDAR, Radar, or thermal cameras. However, RGB cameras represent the cheapest and most commonly used type of sensor for object detection [43].

For a large part, rule-based approaches have been developed before deep learning became a synonym for computer vision. Recently, data-driven object detectors (see Sec. 1.3) enable accurate object detection in the day domain, where large-scale datasets are available (see Sec. 1.4). Yet, data-driven approaches raise another set of issues, such as the effort of data acquisition and annotation within a potentially infinite set of domains.

[10] Boxy Dataset: https://boxy-dataset.com/boxy/
[11] ExDark: https://github.com/cs-chan/Exclusively-Dark-Image-Dataset

The evaluation of object detection (Faster-RCNN, see Sec. 1.3.2) on the night data of the BDD100K dataset [122] shows an average precision drop of 1.0% and, more concerning, a drop in recall by 12.6% and in the F_1 score by 13.4%. The BDD100K data is diverse, including various weather conditions, scene types, cities, and different test vehicles for data recording. Even more so, the findings reflect the need to extend the state-of-the-art object detectors to the night domain in order to achieve reliable performance, in particular for autonomous driving applications.

The drop clearly reflects the shortcomings of supervised learning on small-scale datasets and, as a neural network needs to adapt to new domains and needs to perform under an extended range of conditions. In this context, performance with respect to the time of day is especially interesting, as the perception task becomes more challenging with the adaptation to nighttime conditions. The lack of large-scale, nighttime datasets prevents supervised learning approaches. While at the same time, the continuous shift from the day to the night domain, due to the circadian rhythm, predestines perception by night for, yet to be developed, unsupervised and continuous domain adaptation approaches.

5.4 Sample Supplementation Approach

5.4.1 Motivation

A significant domain gap (see Sec. 5.3.2) is observed for object detection by night with a neural network trained only in the day domain. Neural network training involves the acquisition efforts and preparation of an adequate amount and diverse set of labeled data. Data annotation, especially in the night domain, is often an ambiguous, expensive, time-consuming, and challenging task.

SSUP, the novel generative domain extension approach, creates target domain samples in the feature space of the night domain. Training on these artificial samples, being tied to the ground truth of their related source domain samples, is proven to be an effective way to increase the target domain performance of the neural network, for the first time, on the BDD100K dataset. Further, the neural network's (object detector's) domain of operation (night) is indirectly extended by training on generated data.

Subsequent experiments with SSUP aim at demonstrating the potentials and limitations of artificial sample generation. As SSUP is closely associated with retraining in a supervised manner, the results also cover the analysis of induced catastrophic forgetting. Results are to be demonstrated for the use-case of perception for autonomous driving at night, based on the BDD100K dataset. The results ought to cover qualitative findings and quantitative analysis of how to close an object detector's domain gap when retraining on artificially generated data.

5.4.2 Night Training-Sample Generation

The night domain training-sample generation (see Fig. 5.2) is realized by the generator part of an unsupervised image-to-image translation architecture (see Fig. 3.4 and Sec. 3.1) on BDD100K [122] image samples at a resolution of 1280×720 px (for further details see Sec. 5.3.1). The implementation details on neural network architecture, the training hyperparameters, and respective tooling are given in detail in the appendix (see App. A.1).

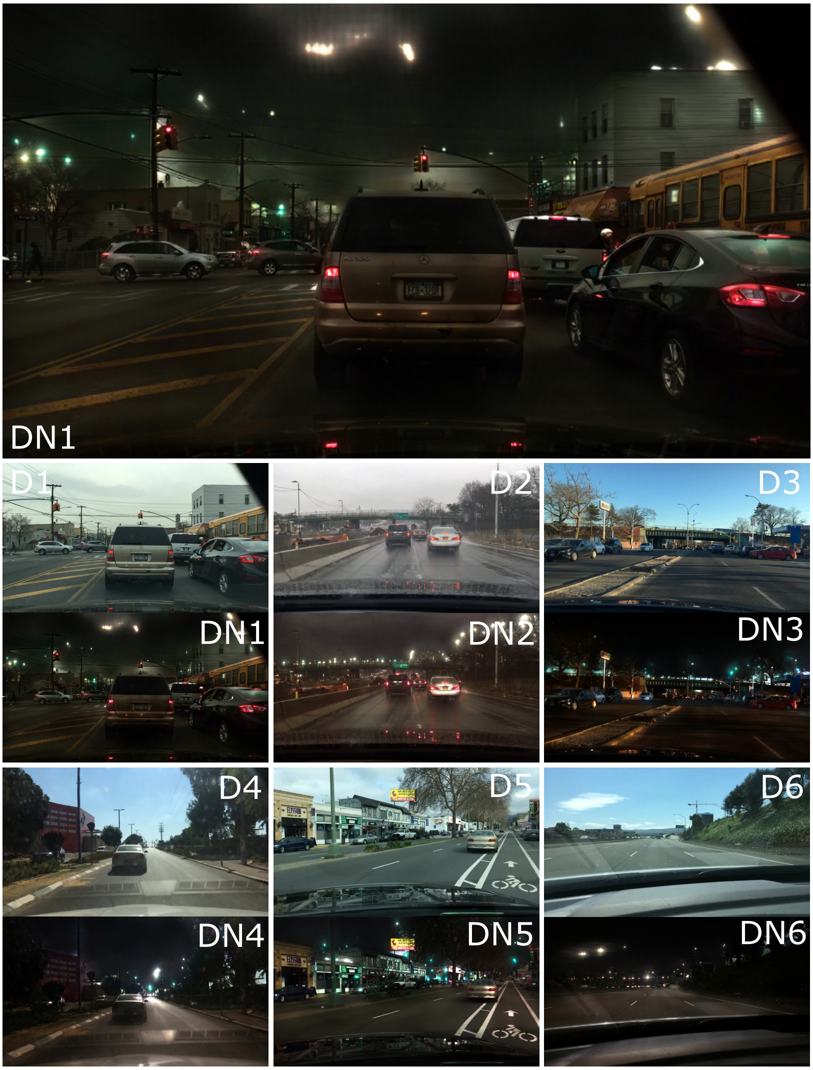


Figure 5.2: Night training-sample generation from day to night. Original image samples x_S of the day domain ($\mathscr{X}_S$: D1 - D6) are randomly drawn from the BDD100K validation data. The image samples are translated to the night domain $x_{S,T}$: Day to night (DN1 - DN6) translations are shown below the corresponding images. On top, sample DN1 is displayed enlarged to enable a detailed, qualitative analysis of the translated image. Adapted from [1].

Semantic Adaptation

From a qualitative perspective, the generated night samples compared to their source day samples show the characteristics that one expects from night domain samples; a decline of contrast (see D4 and DN5), reduced color, reduced expressiveness of semantic and structural details (see DN2 and DN3). For background segments generated, night samples collapse into regions of homogeneity in parts, leading to a complete loss of semantic information (the pedestrian on the left edge of DN1 is hardly visible). These changes precisely reflect the characteristics expected to be experienced in the night domain.

Semantic Artifacts

The generated night samples' artifacts emerge, especially from former bright areas and large pixel gradients translated into town lighting and traffic lights, as seen in DN1 and DN5. Former sky and plant foliage segments default into homogeneous dark areas, as seen in DN4 and DN6. Clouds that stand out from the background translate into town light source artifacts (see the top center in DN1 and the top left in DN6). Shading within the source domain experiences an amplification (see the hard shoulders in DN3 and DN4), while mirages and reflection remain comparatively constant (see the wet road in DN2 and the cowling in DN5). For the most part, the artifacts do not impair the generated samples' semantic quality, particularly when compared to the by far more potent overall change in the semantic state of the translation from day to night, imposed by the generation process itself. Furthermore, emerging artifacts are meaningful in the context of translation (see emerging lamps and lighting in DN1 to DN6).

5.4.3 Visualizing Generative Shifts

From an application perspective discussing semantic artifacts is not expedient, as human analysis of semantic artifacts does not allow for conclusions regarding the model performance or potential misclassifications attributed to malign semantic artifacts. In the following, the newly developed adaptation metrics (see Sec. 4.1.2) for quantifying translation quality are applied, and their results are discussed in detail.By mapping samples from the (day) source domain $\mathbf{x}_S \in \mathscr{X}_S$ and (night) target domain $\mathbf{x}_T \in \mathscr{X}_T$ and the samples $\mathbf{x}_{S,T}$ generated in the night domain, it is possible to visualize emerging generative shifts. Further, quantitative measures are obtained based on the Mahalanobis distance for the domain discrepancy ($d_{d,Ma} = 15.40$ see the left plot in Fig. 5.3), between target domain $\mathscr{X}_T$ (night) and source domain $\mathscr{X}_S$ (day) and the domain shift ($d_{s,Ma} = 11.89$ see the right plot in Fig. 5.3), between source domain $\mathscr{X}_S$ (day) and the generated samples' domain $\mathscr{X}_{S,T}$ (generated night).

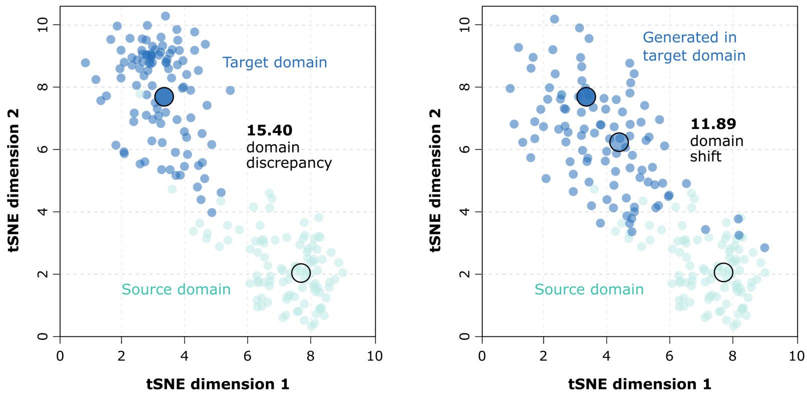


Figure 5.3: The left plot shows the night target domain (dark blue) and day source domain (cyan). The right plot shows the centroid of the night target domain (dark blue), the generated night target domain (transparent dark blue) in relation to the day source domain (cyan). Both plots are based on the same 2D tSNE mapping and visualize 100 random samples. The large dots with a black border depict the centroids for each of the domains. Domain discrepancy and domain shifts are calculated according to the Mahalanobis distance. The axes depict the two dimensions of the tSNE mapping. Adapted from [1].

The distance between the target domain night samples and the generated night samples is reduced to a domain discrepancy of $d_{d,Ma} = 1.10$. Further, it is to point out that the target domain's underlying distribution stays intact when generating night samples. The integrity of a domain's distribution is quantified by the covariance matrices that stay at the same magnitude of $Cov(\mathbf{X}_T) = [[1.13, -0.54], [-0.54, 2.23]]$ for the night and $Cov(\mathbf{X}_{S,T}) = [[3.23, -1.65], [-1.65, 2.61]]$ for the generated night samples.

5.4.4 Generative Domain Extension by Retraining

Usually, a neural network experiences a performance drop due to a domain gap when inferred on a domain on which it has not been trained. For example, object detection performance on the BDD100K validation data drops when shifting operations from the day domain into the night domain (see *retrained on day* in Tab. 5.3). In safety-critical cognitive perception systems such as autonomous driving, recall is essential. Consequently, reducing the number of false negatives and mitigating the significant decrease in recall for object detection by night is a core objective.

Setting	**Day Domain**	**Night Domain**
Retrained on day (baseline)		
Recall	64.7 %	43.1 %
Precision	87.5 %	88.9 %
F_1 score	74.4 %	**58.1** %
Retrained on night		
Recall	20.4 % (↓)	74.8 % (↑)
Precision	52.0 % (↓)	85.3 % (↓)
F_1 score	**29.3** % (↓)	75.0 % (↑)
Generative Domain Extension		
Recall	25.0 % (↓)	**50.3** % (↑)
Precision	50.0 % (↓)	71.1 % (↓)
F_1 score	33.3 % (↓)	**58.9** % (↑)

Table 5.3: Evaluation of the RFCN object detector on the BDD100K validation dataset with respect to the recall, precision, and F_1 score metrics. The IoU threshold is @0.5, and the classification confidence threshold is @0.5. All models are initialized with the COCO dataset. The last set of rows presents the results of the generative domain extension approach by retraining the model for one epoch on generated night samples. The arrows reflect the change in performance with respect to the baseline.

The starting point is a neural network that has been pretrained on the day domain. Based on the ability to translate day samples into night samples (see Sec. 5.4.2, and Sec. 5.4.3), the neural network is retrained on the generated night domain samples, using the ground truth adopted from the former day domain samples. Generated samples enable domain extension (see Sec. 3.1), adapting to domains in which no annotated real-world training data is available, as for the night domain in the here presented experiments.

Retrained on day functions as baseline, the RFCN (see Sec. 1.3) object detector is evaluated on the day domain and night domain after being trained with early stopping on the BDD100K day training data. Especially on unseen samples within the unseen night domain, the model's predictive performance is problematic - establishing a Recall baseline of 43.1%. The RFCN object detector is chosen as it is a well-balanced object detector centered within the spectrum of performance and inference pace (details shown in Tab. 1.1).

Generative domain extension in perspective compares favorable, especially for the safety-critical recall in the night domain: Improving to 50.3% compared to the baseline's 43.1%. Simultaneously, the F_1 score in the day domain drops significantly due to the experienced catastrophic forgetting by retraining, from 74.4% to 33.3% motivating the shift from targeted extension into a transfer approach. Generative domain extension within SSUP provides an instrument for trading-off precision for recall.

Retrained on night experiments further sharpen the retraining capabilities on generated data. Retraining with early stopping on the BDD100K night domain data, the object detector achieves an F_1 score of 75.0% in the night domain, clearly outperforming the reference object detector's performance (58.1%) and the *generative domain extension* performance (58.9%). However, this approach also endures the deterioration of the performance in the day domain due to catastrophic forgetting during retraining, from 74.4% to 29.3%.

The placement of the generative domain extension approach SSUP into the context of deploying domain adaptation approaches is given and concluded in Sec. 5.7.1, together with recommendations for application.

5.5 Shared Latent Feature Space Approach

5.5.1 Motivation

SHALFS introduces the adaptation of a neural network to a target domain by implicit image-to-image translation. In contrast to the generative domain extension (see Sec. 5.4), this method does not need any paired samples between source and target domain, nor does it need labels to be available in either domain (find a detailed comparison in Sec. 5.7.2). Nevertheless, the novel module is able to deploy object detectors in a target domain (night) it has not been trained on by implicitly expanding the neural networks domain of operation. SHALFS mitigates domain gaps. Even more so, the domain adaptation capabilities significantly improve in the target domain and are verified by qualitative analysis of the image-to-image translation capabilities (see Sec. 5.5.2 and Sec. 5.5.3), by quantitative analysis of the domain and shift distances within a shared feature space embedding (see Sec. 5.5.4) and explicit object detector performance on the target domain (see Sec. 5.5.5).

Besides breaking ground for decoupling domain adaptation approaches from retraining the neural network under deployment, the experiments on the SHALFS intend to demonstrate the new-developed concept of cross-domain adaptation. The results are to be shown for the use case of perception for autonomous driving at night. The results mean to cover qualitative findings and quantitative performance achievements within object detection, supported by the novel domain adaptation metrics and visualization methods.

5.5.2 Night-to-Day Image Translation

The plain image-to-image translation (see Fig. 5.4) from night to day by the domain adaptation module (see Fig. 3.4) is carried out on BDD100K [122] image samples at a resolution of 1280×720 px. The implementation details on neural network architecture, the training hyperparameters, and respective tooling are given in detail in the appendix (see App. A.1).

Semantic Adaptation

The samples translated into the day domain demonstrate a qualitative increase in contrast (see samples ND2 and ND3), improved color, and feature expressiveness (especially see samples ND1 and ND4) in comparison to their original counterparts in the night domain. The day-translated samples show

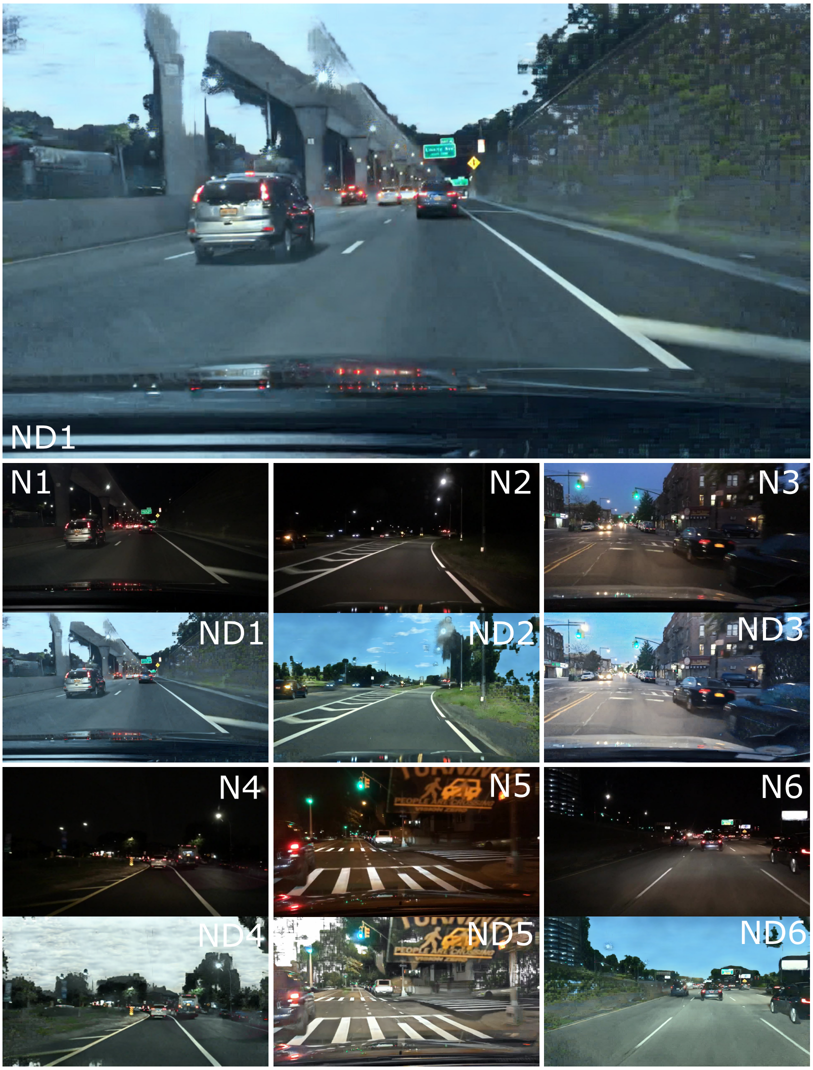


Figure 5.4: Image-to-image translation from night to day. Original image samples $\mathbf{x}_T$ of the night domain ($\mathscr{X}_T$: N1 - N6) are randomly drawn from the BDD100K validation data. The image samples are translated to the day domain $\mathbf{x}_{T,S}$: Night-to-day (ND1 - ND6) translations are shown below the corresponding images. On top, sample ND1 is displayed enlarged to enable a detailed, qualitative analysis of the translated image. Later, the here presented qualitative results are quantified. Adapted from [1].

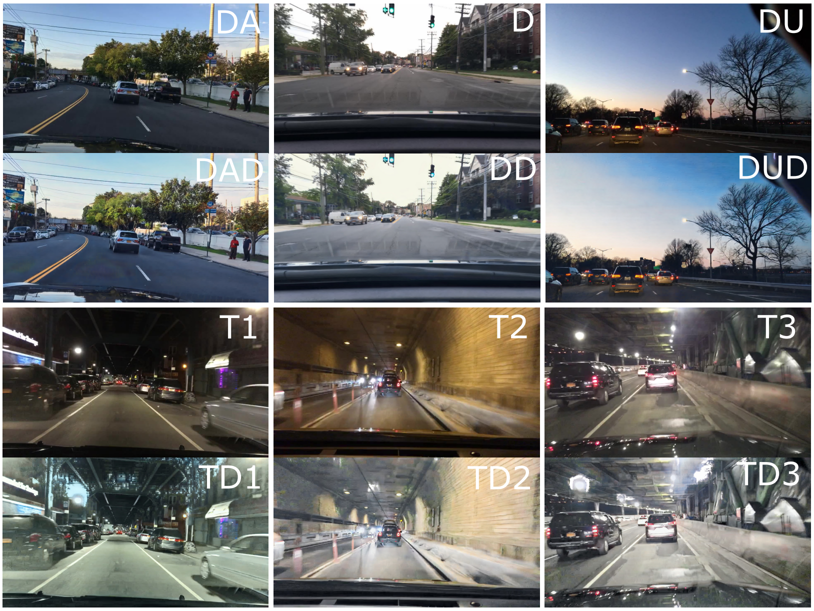


Figure 5.5: Image-to-image translation from dawn, day, dusk, or tunnel to day. Original image samples $\mathbf{x}_S$ of the dawn domain (DA), the day domain (D), the dusk domain (DU), and tunnel samples (T) are randomly drawn from the BDD100K validation data. The image samples are translated to the day domain $\mathbf{x}_{S,T}$: Dawn to day (DAD), day to day (DD), dusk to day (DUD), and tunnel to day (TD) translations are shown below the corresponding images. Adapted from [1].

an enhancement in human perception capabilities (such as the perception of other traffic participants, see ND2, ND4, and ND5).

Semantic Artifacts

The translated samples show artifacts, particularly in regions of former homogeneities, such as background segments consisting of sky segments and plant foliage (see ND1 and ND4). In the here presented context, automated detection and marking of artifacts can be assumed to be feasible. To quantify, the robustness and general occurrence of artifacts in the image's upper third (mainly consisting of sky segments) has been validated explicitly for false positives of the object detector (here RFCN in Sec. 1.3). In the night domain, the object detector experiences 13 false positives on the validation set. The number of false positives determines a false positive rate of 0.27%

in the translated domain, which by no means annul the quantitative beneficial effects of the night-to-day translation, which are an order of magnitude higher, as will be subsequently demonstrated. The results show that the object detector performance is sufficiently robust for occurring artifacts in the image-to-image translation, making artifacts in the upper third of the image neglectable. Even though the experienced artifacts do not impair the object detector performance to justify further filter operations, restricting the region of interest of the image space prevents false positives implicitly and should be considered in general.

Domain-Specific Influence

The translation achieves consistent results (see Fig. 5.5) when applied to the source and intermediate target domains: Dawn (see DAD), day (see DD), and dusk (see DUD) samples from a range of scenes included in the dataset (such as highway, parking lot, city street, partly clouded, rainy, and others). Simultaneously, semantic features crucial to the detection task are preserved (as quantified in detail by the beneficial effect of the domain adaptation, which is subsequently analyzed), for example, in cars and persons.

5.5.3 Extended Side Mirror Deployment

The semantic adaptation achieved by the image-to-image translation is capable of enhancing human perception capabilities when it comes to night perception. The in this work developed cross-domain adaptation module (see Sec. 3.2.3) is deployed on a test vehicle, realizing the extended side mirror application[12] for maneuvering and reverse parking under night-time conditions. The image input stream of both side cameras is translated from the night domain into the day domain by deploying the novel cross-domain adaptation approach. Subsequently, the translated images are visualized with displays, which enhance the conventional side mirrors' functionality.

The test vehicle for the E Side Mirror application is based on a Hyundai platform (see Fig. 5.6). The platform has been ZF-customized and equipped with a four-camera sensor setup. The deployed cameras are identical to Sekonix SF3325 - one front-view camera, one rear-view camera, and the two backward-facing side mirror cameras [131]. The image input stream,

[12] This work contributes the algorithmic share of the winning project of the ZF Excellence Award 2019 in the category products and manufacturing processes - "The ZF Excellence Award is the most important innovation competition within the ZF Group."

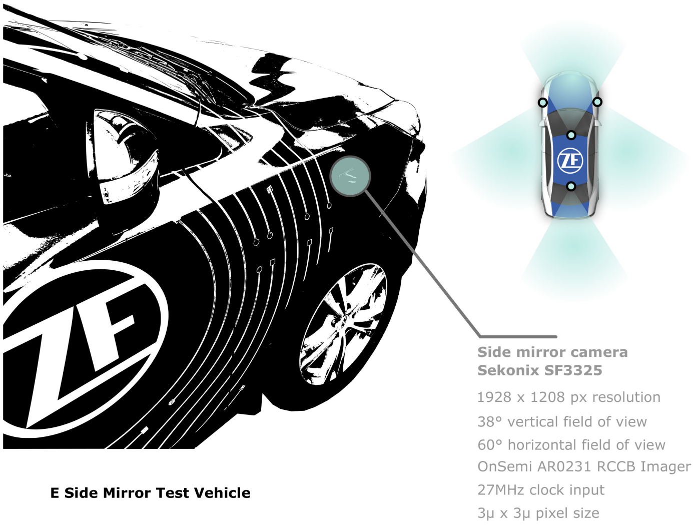

E Side Mirror Test Vehicle
Side mirror camera
Sekonix SF3325
1928 x 1208 px resolution
38° vertical field of view
60° horizontal field of view
OnSemi AR0231 RCCB Imager
27MHz clock input
3μ x 3μ pixel size

Figure 5.6: Test vehicle for the E Side Mirror application. Displayed on the left is the Hyundai platform, showing the backward-facing side mirror camera, highlighted by the cyan circle (for a more detailed specification of the Sekonix SF3325 [131]. The sensor setup consists of four Sekonix SF3325 cameras. The field of vision of the test vehicle is depicted in the upper right corner as a bird's eye view. The circles represent the camera's mounting position.

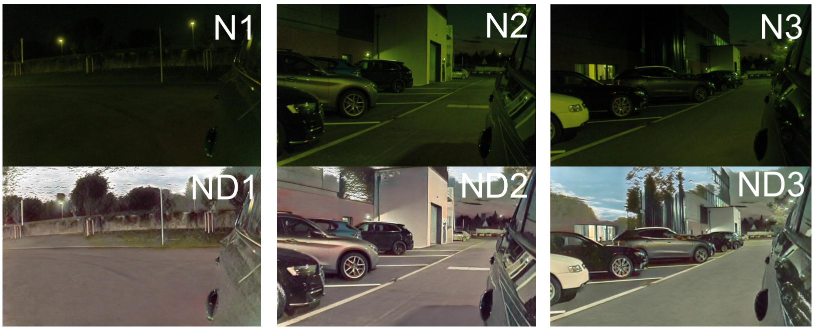


Figure 5.7: Image-to-image translation from night to day for images of the right backward-facing side mirror camera. Original image samples $\mathbf{x}_T$ of the night domain ($\mathscr{X}_T$: N1 - N3) are recorded with the E Side Mirror platform for validation. The image samples are translated to the day domain $\mathbf{x}_{T,S}$: Night-to-day (ND1 - ND3) translations are shown below the corresponding images.

at 30 frames per second, is processed by the NVIDIA Drive PX2, based on NVIDIA's DriveOS [132], inferencing the novel image-to-image translation model (see Sec. 3.2.3 and Sec. A.1). The translated images (see the bottom row in Fig. 5.7) are visualized on SmallHD 702 LCD field monitors with a resolution of 1920 $\times$ 1080, a color depth of 24 bit, 7-inch monitor with 1000 Nits brightness [133]. The E Side Mirror project is an Advanced Driver Assistance System (ADAS) that has reached concept and prototype level for night perception. The objective is to make maneuvering at night safer and more comfortable for human drivers[13,14]. In the following, SHALFS is being evaluated qualitatively and quantitatively.

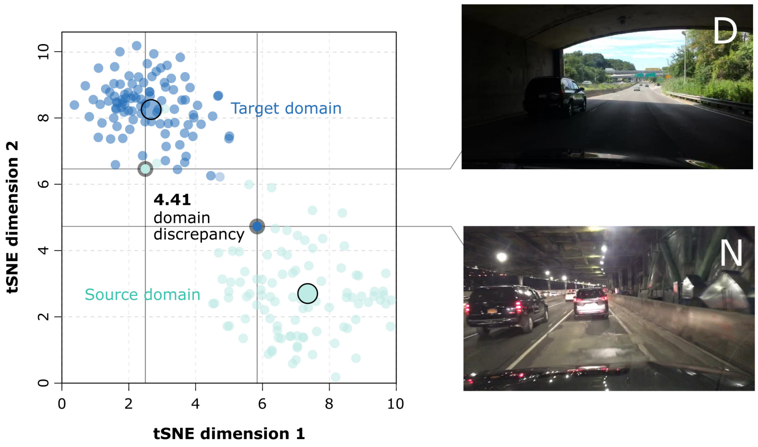
5.5.4 Visualizing Domain Discrepancies and Shifts

At this point, the evaluation of the image translation capabilities has been qualitative only, displaying the translated samples for human inspection (see Fig. 5.4). As this might be insightful for individual samples, this is not feasible to state reliable performance metrics. In order to quantify domain adaptation measures such as domain discrepancies and domain shifts of the image-to-image translation, the tSNE algorithm is deployed (see Sec. 1.5.2). This approach is crucial as it addresses the current research issue of how to quantitatively evaluate GAN architectures, proposing feature embedding algorithms. The domain discrepancy (see Fig. 5.8a) based on the Mahalanobis distance in the two-dimensional tSNE feature space is $d_{d,Ma} = 4.41$ for the night and the day cluster. Further, a quantitative measure is obtained for the domain shift $d_{s,Ma}$ (see Fig. 5.8b), realized by the image-to-image translation from target domain $\mathscr{X}_T$ (night) into the source domain $\mathscr{X}_{T,S}$ (night-to-day). During image-to-image translation, data samples $\mathbf{x}_T$ drawn from a single data distribution or domain (night) are being shifted into samples $\mathbf{x}_{T,S}$ from another data distribution or domain $\mathscr{X}_S$ (day domain). The experienced domain shift in the two-dimensional tSNE feature space is $d_{s,Ma} = 0.95$ for a translation from night to day.

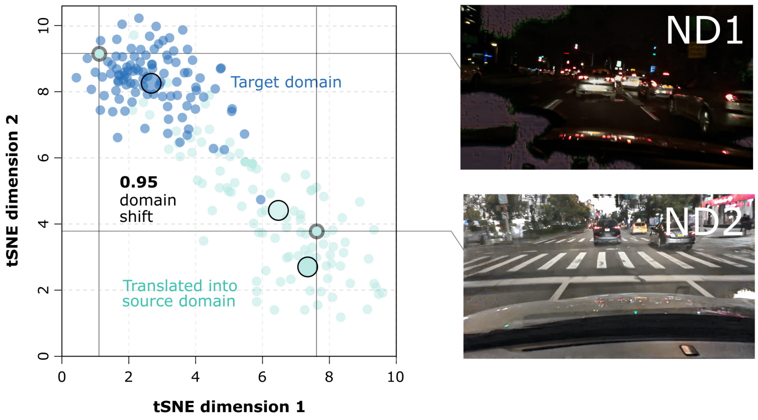
By image-to-image translation of the target domain, the domain discrepancy to the source domain is reduced to $d_{d,Ma} = 3.52$ between the centroids of the source domain and the translated source domain; this explicates a reduction in domain discrepancy of 20.18%. Quantifying the result of the unsupervised domain adaptation by the tSNE algorithm shows the successful shift within the feature space - see the qualitative result of the domain shift highlighted by

[13] Patent on object detection for night perception systems [g]

[14] Patent on lane detection and segmentation for night perception systems [n]



(a) tSNE Mapping Night and Day Samples. The large dots with a black border depict the centroids for each of the two domains depicting domain discrepancy $d_{d,Ma}$ between the night domain and day domain. The samples D and N (indicated by a bold stroke around the circle) exemplary demonstrate the overlap between the day and night domain and vice versa. Overlaps arise for day samples with night domain characteristics, such as in tunnels (D), or for night samples with day domain characteristics, such as in beneficial lightning conditions (N). Adapted from [1].



(b) tSNE Mapping Night and Night-to-Day Translated Samples. The large dot with a black border in dark cyan represents the source/day domain centroid, depicting the effective domain shift $d_{s,Ma}$ from the night domain to the night-to-day domain. The sample ND1 presents an example where the domain adaptation failed, and a domain shift did not occur. ND2 shows a sample where the domain adaptation into the day-to-night domain has been successful. Adapted from [1].

Figure 5.8: 2D tSNE mapping and visualization of 100 random samples from the target/night domain (dark blue) and the corresponding source domain translations (cyan). The axes depict the dimensions of the tSNE mapping.

the centroids and the shifted point cloud in Fig. 5.8b. The tSNE algorithm, combined with the novel adaptation quantity metrics (see Sec. 4.1.2), point out failed domain adaptations, which manifest in minor or non-existent shifts during domain adaptation, as perceivable in ND1 in Fig. 5.8b).

5.5.5 Domain Adaptation for Object Detection by Night

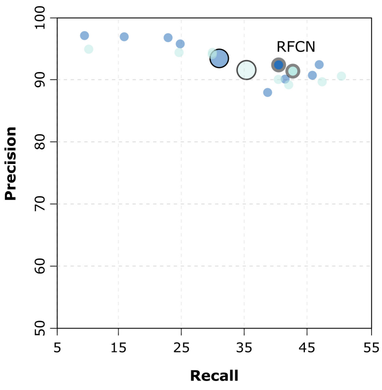
The results so far have shown that domains can be shifted from one to another by unsupervised image-to-image translation. Further, SHALFS enables cross-domain deployment for object detectors based on the presented translation capabilities.

To demonstrate the quantitative performance effects of the cross-domain deployment of object detectors, a night perception benchmark for domain gap reduction based on a set of night perception datasets is set up in the following, based on the BDD100K dataset, the Argoverse dataset, and the Boxy dataset, as well as, newly defined and established domain adaptation metrics (see Sec. 4.1), and an extensive set of publicly available, pretrained object detectors. The object detectors are pretrained on the COCO dataset and subsequently inferred on either the original data of a night perception dataset or the night perception data translated into the day domain by the cross-domain adaptation module. The object detectors are evaluated at an operating point with a confidence threshold @0.5 and an IoU threshold @0.5 to ensure comparability. The cross-domain adaptation module has only been trained on the training data of the BDD100K dataset.

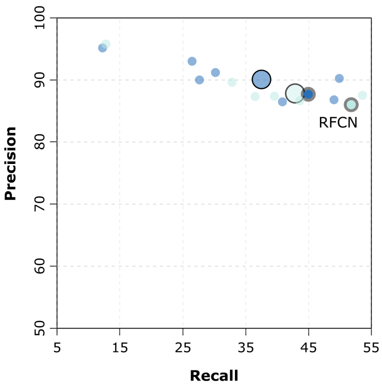
Precision and Recall

In safety-critical cognitive perception systems, recall is prioritized over precision. Simultaneously, in object detection by night, a significant decrease in the recall is observed due to the neural network's domain gap when it is pretrained on the day domain only - for comparison, on the day domain, the average object detector performs with an F_1 score of 59.0%, precision of 94.5%, and recall of 44.4%. In the night domain, the average object detector performs lower with an F_1 score of 45.1%, precision of 93.5%, and recall of 31.8%.

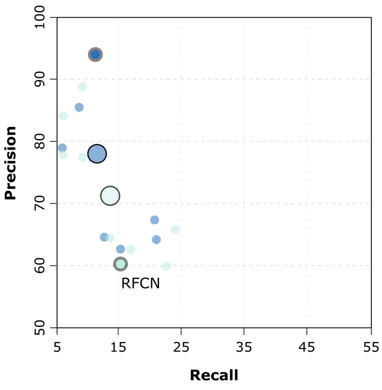
The validation of the object detector set confirms a general performance increase (see Fig. 5.9a) when deploying the novel cross-domain adaptation module. On the BDD100K dataset the F_1 score (std) increases on average



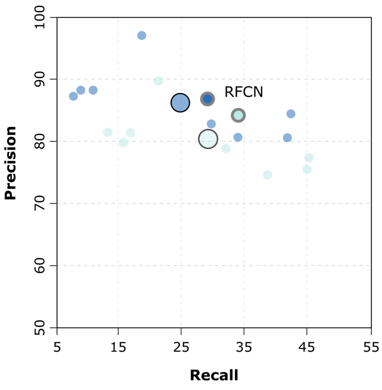
(a) Night-to-day object detector precision over recall on the vehicle class of the BDD100K.



(b) Night-to-day object detector precision over recall on the vehicle class of the Argoverse dataset (see Sec. 5.3.1).



(c) Night-to-day object detector precision over recall on the vehicle class of the Boxy dataset.



(d) Night-to-day object detector precision over recall on the pedestrian class of the BDD100K.

Figure 5.9: Precision over recall for a set of nine pretrained object detectors (see Sec. 1.3). The performance is reported on the night domain (blue) and the translated night-to-day domain (cyan). The large dots with a black border depict the average performance over all object detectors for each of the domains. The RFCN object detector, which in the following is subject to further detailed analysis, is highlighted by a broad, gray border. The figure indicates an increase in recall, trading off with a decrease in precision. Adapted from [1].

by 3.9% (±3.5%), the recall by 3.4% (±2.6%), the precision decreases on average by 1.3% (±1.3%).

Similar effects can be observed for the object detector set on other night perception datasets as well, solidifying the findings. On the Argoverse dataset (see Fig. 5.9b), the F_1 score (std) increases on average by 4.9% (±2.6%), the recall by 5.6% (±2.6%), and the precision decreases on average by 1.9% (±1.3%).

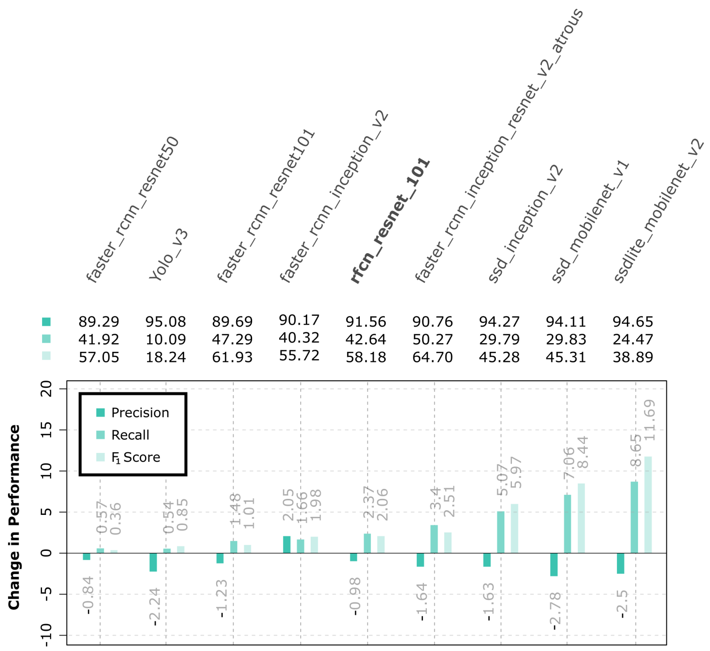
On the Boxy dataset (see Fig. 5.9c), the F_1 score (std) increases on average by 2.9% (±2.3%) with respect to the object detector set, the recall by 2.1% (±1.5%), the precision decreases on average by 6.9% (±10.6%).

Pedestrian Detection at Night

The demonstrated overall performance increase is reflected by the F_1 score, and the recall, in particular, is reproducible for the pedestrian class (see Fig. 5.9d). The F_1 score (std) increases on average by 4.7% (±3.2%), the recall by 4.3% (±1.5%), the precision decreases on average by 5.8% (±1.7%). The targeted improvement of the cognitive perception systems in recall for driving at night has been realized. The approach demonstrates a general performance increase in the F_1 score, independent of the deployed object detector, across multiple night perception datasets and the object classes of vehicles and pedestrians.

Performance on Individual Detectors

When analyzing the object detectors on an individual level, the findings (see Fig. 5.10) confirm the in this work presented averaged trends. With a decrease in model size and faster inference pace, fewer parameters, and model capacity, the beneficial effects of domain adaptation on neural network performance are amplified. For example on the BDD100K dataset, the *ssdlite_mobilenet_v2_coco* (as introduced in Tab. 1.1) experiences an F_1 score improvement of 11.7%, a recall improvement of 8.7% and a precision drop of 2.5%, as opposed to the larger *rfcn_ resnet101_coco* (RFCN) which experiences an F_1 score improvement of 2.1%, a recall improvement of 2.4% and a precision drop of 1.0%.



	faster_rcnn_resnet50	Yolo_v3	faster_rcnn_resnet101	faster_rcnn_inception_v2	**rfcn_resnet_101**	faster_rcnn_inception_resnet_v2_atrous	ssd_inception_v2	ssd_mobilenet_v1	ssdlite_mobilenet_v2
■	89.29	95.08	89.69	90.17	91.56	90.76	94.27	94.11	94.65
■	41.92	10.09	47.29	40.32	42.64	50.27	29.79	29.83	24.47
■	57.05	18.24	61.93	55.72	58.18	64.70	45.28	45.31	38.89

Figure 5.10: The object detectors' performances in percent on the translated night-to-day domain. The figure shows the change in performance for a set of nine individual object detectors when translating the samples from night to day. The evaluation has been carried out on the BDD100K validation dataset for vehicles. The evaluation occurs at the operating point with an IoU threshold @0.5 and a classification confidence threshold @0.5.

Receiver Operating Characteristic

Up to now, the effects of SHALFS' cross-domain adaptation module have to be interpreted as a desirable, beneficial trade-off centering on recall over precision. The impression of being a trade-off is caused by the constriction to a single operating point (see dots with gray border in Fig. 5.11a) with IoU threshold @0.5 and a classification confidence threshold @0.5 during evaluation.

A comprehensive analysis of the receiver operating characteristic (ROC) over the whole operation space of an object detector reveals a general, beneficial effect of the cross-domain adaptation module beyond a single operating point. The analysis follows a two-step approach:

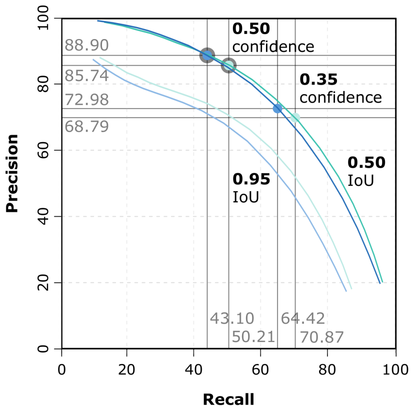
1. Introduction of a representative object detector and its characteristics within SHALFS cross-domain adaptation.
2. Broaden the analysis from a single operating point to the full operation space.

The focus is put on one specific object detector to engage with a more detailed analysis. The RFCN object detector is a well-balanced object detector centered within the spectrum of performance and inference pace (see Fig. 5.10). The RFCN object detector is retrained on the BDD100K day training data (see Tab. 5.4) and evaluated on the BDD100K validation dataset. Validation on the day domain yields an F_1 score of 74.4%. In the night domain, the object detector achieves an F_1 score of 58.1%, or an absolute F_1 score object detection domain gap of 16.3% between the day domain and night domain. After the SHALFS cross-domain adaptation, the F_1 score domain gap is reduced to 11.1%.

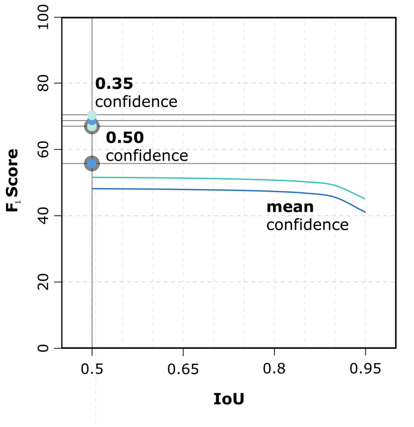
Contrasting the precision over recall curve of the night domain (blue) with the night-to-day domain (cyan), a positive performance shift (to the upper right) is apparent (see curves in Fig. 5.11a). With an increase in the IoU threshold (from @0.5 to @0.95), the increase in performance shift becomes more distinctive and beneficial for the novel SHALFS cross-domain adaptation approach. The same positive performance shift effect is perceptible in areas that favor high recall over high precision. Concluding, the perceived trade-off is a pseudo decrease in precision due to the fixed, single operation point. In closing, SHALFS cross-domain adaptation provides an overall improved performance for object detection in night perception (see the F_1 Score increase between the blue and the cyan line in Fig. 5.11b).

Alternative Training Schemes

A range of different training schemes is available for a model to learn object detection by night, such as supervised training and retraining. For comparison, the SHALFS is contrasted with various alternative training schemes, and baseline approaches (see Tab. 5.4). The RFCN object detector is selected as a model, as it constitutes a representative object detector according to Fig. 5.10. The evaluation is carried out on the BDD100K dataset. In order to evaluate the effects of the domain adaptation approach on the target and



(a) Precision over recall for the night domain (blue color) and the night-to-day domain (cyan color), each for an IoU threshold @0.5 (upper curves), and @0.95 (lower curves) respectively. Adapted from [1].



(b) F_1 Score over IoU for the night domain (blue color) and the night-to-day domain (cyan color), each averaged over confidence threshold @[0.05 : 0.9] with steps of 0.05. The superior performance of cross-deployment in the night-to-day domain is apparent over different IoU values. The expected F_1 Score drop for high IoU thresholds is observable for deployment on either domain.

Figure 5.11: The performance characteristics are reported on the night domain (blue) and the translated night-to-day domain (cyan). Notice the continuously present, positive shift of the ROC curves for the night-to-day domain in relation to the ROC curves for the night domain. The solid lines centered on dots represent the operating points at an IoU @0.5 and confidence threshold @0.35, or @0.5 (dots with gray border).

source domain performances, the training schemes are evaluated on the day domain $\mathscr{D}_S$ and night domain $\mathscr{D}_T$ of the BDD100K validation data.

Setting	**Day Domain**	**Night Domain**
Retrained on day (baseline)		
Recall	64.7 %	43.1 %
Precision	87.5 %	88.9 %
F_1 score	74.4 %	**58.1** %
Trained on COCO		
Recall	54.3 % (↓)	40.3 % (↓)
Precision	92.7 % (↑)	92.5 % (↑)
F_1 score	68.5 % (↓)	56.1 % (↓)
Retrained on night		
Recall	20.4 % (↓)	74.8 % (↑)
Precision	52.0 % (↓)	85.3 % (↓)
F_1 score	29.3 % (↓)	75.0 % (↑)
Retrained on day and night		
Recall	76.9 % (↑)	75.6 % (↑)
Precision	74.2 % (↓)	66.5 % (↓)
F_1 score	75.5 % (↑)	70.8 % (↑)
Cross-domain adaptation (SHALFS)		
Recall	64.9 % (↑)	50.2 % (↑)
Precision	87.3 % (↓)	85.7 % (↓)
F_1 score	74.5 % (↑)	**63.3** % (↑)

Table 5.4: Evaluation of the RFCN object detector on the BDD100K validation dataset is for an IoU threshold @0.5 and a classification confidence threshold @0.5. All models are initialized with the COCO dataset.

Trained on COCO directly deploys the publicly available RFCN object detector that has been pretrained on the COCO dataset. An in-depth comparison with other object detectors is available in Fig. 5.10). The pretrained RFCN object detector demonstrates the beneficial effect of retraining on designated data by reporting an overall lower performance than the retraining scheme. The F_1 score is 68.5% in the day domain and 56.1% in the night domain.

Retrained on night is the supervised retraining scheme targeting the night domain and has the object detector retrained on the BDD100K night domain data. In the night domain, the retrained object detector achieves an F_1 score of 75.0%, outperforming the reference and domain adaptation approach's object detection capabilities. However, the catastrophic forgetting during retraining and, consequently, deterioration of performance in the day domain must not be overlooked. The retrained object detector drops to an F_1 score of 29.3%, thus underperforming all other approaches evaluated in the day domain.

Retrained on day and night is the third retraining scheme and allows retraining of the object detector with early stopping on the BDD100K day domain and night domain data. The retrained object detector achieves an F_1 score of 70.8% and a recall of 75.5% in the day domain, outperforming the reference and domain adaptation approach's object detection capabilities, evaluated in the night domain. However, the required annotation effort ($e_A = 0$) and the compilation and the memory requirements ($e_S = 0$) of a combined dataset that needs to be kept in storage during training are not negligible in a holistic evaluation as laid out in the newly introduced applicability and adaptation specific metrics (see Sec. 4.1.3).

SHALFS cross-domain adaptation deploys the reference object detector *retrained on day*, which has been retrained on the BDD100K day domain data within the cross-domain adaptation approach: The day domain is translated into the day-to-day domain, and the night domain is translated into the night-to-day domain for evaluation (see Sec. 3.2.3). As expected, based on previous findings (see Sec. 5.5.5), the novel cross-domain adaptation approach yields improved object detection performance both in the day domain (an F_1 score of 74.5%) and night domain (an F_1 score of 63.3%). SHALFS is capable of closing the domain gap in object detection at night. The outcome is an average recall improvement by 7.1% while decreasing the average precision by 3.16%. This amounts to a reduction in the F_1 score domain gap of an absolute 5.3% in the night domain (see Tab. 5.4).

5.6 Continuous Self-Supervision Approach

5.6.1 Motivation

Domain adaptation often causes a performance drop in the source domain - catastrophic forgetting. On top of that, emerging domains are often heavily shifted and vary widely from the source domain, making current context-based approaches ambiguous by disregarding available knowledge of the data distribution and, here, a continuous shift. Thus the continuous domain adaptation approach is assessed by means of the domain adaptation performance metric *DA* (see Eq. 4.3) and catastrophic forgetting metric *CF* (see Eq. 4.4).

COSS demonstrates the efficient extension and domain adaptation of existing neural networks. The approach deploys a continuous data distribution shift prior, together with preexisting knowledge of the source domain.The continuous domain adaptation approach is sufficient to render target domain labels unnecessary by pseudo-label deployment while preserving source domain performance utilizing a cue.
Experiments with COSS for the first time target domain shifts, which only become apparent during deployment. Further, the results enable to analyze the robustness and limits of applicability and especially the potential of bypassing catastrophic forgetting in a continuous unsupervised setting with CUEB.

5.6.2 The Mechanism of Continuous Domain Adaptation

COSS is implemented for classification, and the evaluation is carried out on the validation set of the MNIST dataset and rotatedMNIST dataset. The implementation details on neural network architecture, the training hyperparameters, and respective tooling are given in detail in the appendix (see App. A.1).

The approach-specific hyperparameters (see Tab. 5.5) are analyzed, and the hyperparameters' influence on domain adaptation and catastrophic forgetting is investigated to discuss the mechanism of the novel continuous domain adaptation approach.

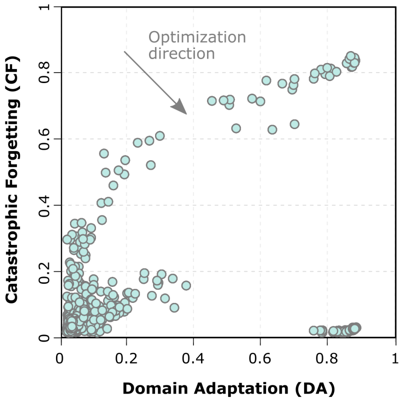
Hyperparameter Description	Abbreviation	Range of Values
Opening training epochs on $\mathscr{D}_0$	o	$\{1, 10, 50\}$
Epochs retraining on $\mathscr{D}_0$	r	$\{0, 1, 2, 3, 5\}$
Epochs training on $\mathscr{D}_{\alpha+\Delta\alpha}$	n	$\{1, 2, 3, 5\}$
Inter-domain step size	$\Delta\alpha$	$\{0.1, 1, 2, 10\}$
Pseudo-label confidence threshold	c	$\{0.0, 0.5, 0.9\}$

Table 5.5: Configuration hyperparameter space of COSS, with the abbreviations of the hyperparameters and the corresponding range of values in which the hyperparameters have been considered.

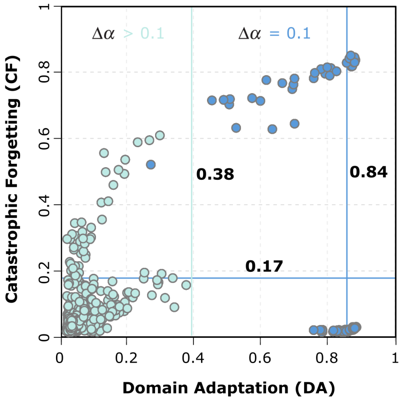
For the analysis, two guiding metrics have been defined (see Sec. 4.1.2), maximizing domain adaptation performance *DA* (see Eq. 4.3) and minimizing catastrophic forgetting *CF* (see Eq. 4.4). Initially, each configuration of the range of values of the hyperparameter configuration space has been considered and processed within a grid search scheme. It is necessary to remark that for searches that have a single optimal hyperparameter as objective, random search [134] is recommended due to the extensive training expenses inherent to neural networks on large-scale datasets. However, in the following, the full configuration space is searched and analyzed to provide a methodology for hyperparameter selection within COSS. Noticeable cluster formations and areas of high performance are highlighted in the following and linked to the according hyperparameter decision, enabling the transfer of the found pattern to other datasets and tasks. The general optimization direction is towards increasing domain adaptation capabilities while reducing catastrophic forgetting (see Fig. 5.12a).

Inter-Domain Step Size ($\Delta\alpha$)

Evaluating the influence of the hyperparameters, $\Delta\alpha$ the inter-domain step size stands out as the most influential hyperparameter with respect to domain adaptation capabilities. The inter-domain step size $\Delta\alpha$ correlates inversely *DA* (see Fig. 5.12b) and there is no configuration $DA \geq 0.38$ without $\Delta\alpha = 0.1$



(a) **Visualization of the complete hyperparameter configuration space** (see Tab. 5.5) of the continuous domain adaptation approach. The configuration space is displayed within the dimensions of catastrophic forgetting *CF* and domain adaptation *DA*. The gray arrow illustrates the general optimization direction.



(b) **Impact of the inter-domain step size** $\Delta\alpha$ hyperparameter on the absolute domain adaptation performance and absolute catastrophic forgetting. The inter-domain step size $\Delta\alpha$, here the angular change due to the rotation, defines the domain gap's magnitude that is bridged during a single adaptation step. The blue dots represent the configurations for an inter-domain step size of $\Delta\alpha = 0.1$. The cyan line shows the maximal domain adaptation performance for configurations with larger inter-domain step sizes.

Figure 5.12

The inter-domain step size $\Delta\alpha$ defines the prior's utilization of the continuous domain shift within the dataset. The continuous domain adaptation approach is based on a step-wise domain adaptation on intermediate domains. The inter-domain step size $\Delta\alpha$ specifies how far the model moves along the domain shift during a single adaptation step (see Fig. 3.7).

Appointing $\Delta\alpha = 0.1$ improves DA to 0.84 ($\pm$0.03) and CF reaches an average of 0.17 ($\pm$0.33). The standard deviation of CF and the plotted configurations with $\Delta\alpha = 0.1$ (see blue dots in Fig. 5.12b) make the formation of a cluster due to one of the other hyperparameters apparent.

Epochs Retraining (r)

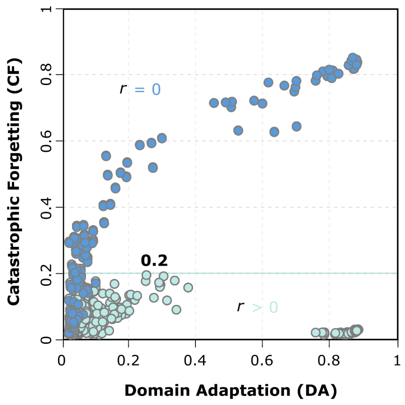
The retraining r on the source domain data, which is done subsequent to each of the training sequences on the intermediate target domains (see Alg. 1 and Sec. 3.3.2) has a major impact on capabilities to limit catastrophic forgetting.

When it comes to catastrophic forgetting, any number of epochs greater than zero of retraining r on the source domain lessens the deteriorative effect on the source domain performance. Due to repeated supervised retraining on the source domain, source domain knowledge and performance are preserved within the model. This finding emerges particularly clearly when studying the catastrophic forgetting for zero epochs retraining (see blue dots in Fig. 5.13a). When retraining for one or more epochs $r \geq 1$, the catastrophic forgetting drops significantly, meaning below a CF of 0.2 for each of the configurations. It is to be kept in mind that bypassing catastrophic forgetting is not to be treated as an end in itself, and the requirement needs to be specified in alignment with the use-case at hand.

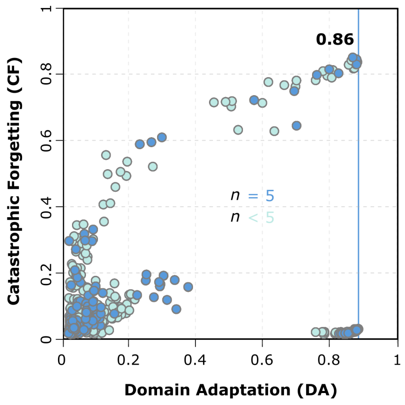
Training on Intermediate Target Domain (n)

The number of epochs n the model is recurrently trained on the intermediate target domain, enabled by self-supervised generated pseudo-labels, is the central component of the domain adaptation approach (see Alg. 1 and Fig. 3.7), and thus is invariably enabled ($n \geq 1$).

The number of epochs trained n on an intermediate target domain decides the model's level of convergence on the current, intermediate target domain (see blue dots in Fig. 5.13b). A larger number of epochs means a further progressed convergence and adaptation across past domains. Moreover, the higher the intermediate target domain performance, the higher the quality



(a) **Impact of the epochs retraining** *r* hyperparameter over combinations of the hyperparameter configuration space (see Tab. 5.5), on the domain adaptation performance and the catastrophic forgetting. Blue dots show configurations with no retraining ($r = 0$). The cyan line shows the maximal catastrophic forgetting for configurations with retraining ($r \geq 1$). Retraining on the source domain mitigates catastrophic forgetting by safeguarding the source domain knowledge during repetitive retraining.



(b) **Impact of the number of epochs trained on the intermediate target domain** *n* on the absolute domain adaptation performance and absolute catastrophic forgetting over combinations of the hyperparameter configuration space (see Tab. 5.5). Training on the intermediate target domain *n* is the means to adapt the neural network. Thus, this hyperparameter controls the extent to which adaptation takes place. The blue dots show configurations with five epochs training on the intermediate domain ($n = 5$). These configurations include the best domain adaptation performances (best *DA* performance is marked by the blue line).

Figure 5.13

of the self-supervised generated pseudo-labels become for the second next intermediate target domain, and greatly determine the successful progression towards the target domain.

Opening Training Epochs (*o*)

The opening training epochs *o* define the number of epochs the model is initially pretrained on the source domain data in a supervised manner (see Sec. 3.3.2 and illustrated in the first row of Fig. 3.7). The opening training epoch-hyperparameter *o* determines how strong the source domain knowledge is encaptured within the source domain model. A large number of opening training epochs corresponds to an increase in source domain performance and an already proceeded convergence of the model on the source domain.

The opening training epochs hyperparameter does not correlate with catastrophic forgetting (see Tab. 5.6). Whether the model forfeits its source domain performance is not determined by the initial source domain performance or knowledge it initially acquired. Nevertheless, the opening training epochs hyperparameter strongly correlates with the domain adaptation and self-supervised adaptation capabilities to intermediate target domains, especially the first adaptation step starting out from the source domain. At the same time, the effect of the opening training epochs on domain adaptation is superimposed by the effects of other hyperparameters to the end that a visualization comparable to Figure 5.13 is of no avail.

However, the beneficial effects of increasing the number of opening training epochs are tied to the source domain performance converging to 97.8% for 50 opening training epochs (as depicted in the upper left corner of Fig. 5.14a). The performance convergence makes experiments for larger numbers of opening training epochs unnecessary. The initial source domain performance has a considerable influence on the pseudo-label quality and thus positively influences the domain adaptation capabilities.

Pseudo-Label Confidence Threshold (*c*)

The pseudo-label confidence threshold *c* correlates little with the domain adaptation and less with catastrophic forgetting (see Tab. 5.6). For small intermediate domain step sizes such as 0.1, the neural network remains confident, as expected, following the assumption of superimposition of two adjacent domains. In contrast to expectation, even larger intermediate-domain step sizes, such as 2 or 10, which, judged by the domain adaptation performance, break the assumption of superimposition, do not show increased uncertainty within the model. This alludes to the circumstance that the confidence of a neural network's predictions loses its expressiveness when leaving its source domain. The predictive confidence of a neural network is often misinterpreted as an estimate of model uncertainty. Especially in the context of domain adaptation, the here presented findings highlight this bias and urge future research in the direction of uncertainty estimation under domain shifts.

5.6.3 Capabilities of Continuous Domain Adaptation

Having studied the fundamental hyperparameters (see Tab. 5.5) and having understood their influence (see Sec. 5.6.2 and Tab. 5.6) within the continuous domain adaptation approach, allows to analyze the capabilities and limitations of the novel continuous domain adaptation approach.

Guiding Metrics	**Hyperparameters**				
	o	r	n	$\Delta\alpha = 0.1$	c
Guidance	$(\uparrow)$	$(\sim)$	$(\uparrow)$	$(\downarrow)$	$(-)$
DA	0.39	0.33	0.15	–	0.00
CF	0.00	−0.64	0.02	–	0.00

Table 5.6: Pearson correlation of the configuration hyperparameters with respect to the continuous domain adaptation metrics *DA* and *CF*. The correlations are calculated for $\Delta\alpha$ set to 0.1, removing the impact of configurations that collapsed during continuous self-supervised domain adaptation due to a large intermediate domain step. The first row provides generalized guidance to set the hyperparameter configuration: Arrows indicating the direction in which the value of the hyperparameter should be optimized, $(-)$ indicating that the approach is agnostic to the parameter, and $(\sim)$ indicating a use-case dependency of the hyperparameter.

This subsection will focus on the top of the class continuous domain adaptation configurations within the configuration space (as defined in Tab. 5.5) on the rotatedMNIST dataset. Taking a closer look at configurations with optimal performance on the key performance indicators, domain adaptation (*DA*) and catastrophic forgetting (*CF*), enables analyzing the capabilities of

Configuration	**Metrics**		
	DA	*CF*	$f_K(std)$
(I) $o50\_r0\_n5\_\Delta\alpha 0.1\_c0.0$	**0.868**	0.840	0.567 (0.324)
(II) $o50\_r1\_n5\_\Delta\alpha 0.1\_c0.9$	0.862	0.016	0.882 (0.087)
(III) $o50\_r2\_n1\_\Delta\alpha 0.1\_c0.0$	0.847	**0.004**	**0.912 (0.053)**

Table 5.7: Optimal (highlighted in bold) performing self-supervised continuous domain adaptation configurations concerning domain adaptation *DA*, catastrophic forgetting *CF*, and mean accuracy $f_K(std)$ after the last adaptation step. Configuration (I) achieves the optimal *DA*, demonstrating the capability to reach source domain performance in the target domain continuously. Configuration (II) achieves the optimal trade-off between *CF* and *DA*, when weighted equally. Configuration (III) achieves the leading built knowledge factor f_K, when all domains are taken into account, as well as the optimal *CF*, demonstrating the ability to limit catastrophic forgetting during the continuous self-supervised domain adaptation process.

continuous domain adaptation, which are: Achieving source domain performance in the target domain, limiting catastrophic forgetting in the source domain, coping with the domain gap width, and coping with constant, continuous adaptation repetitions (see Tab. 5.7).

Achieving Source Domain Performance

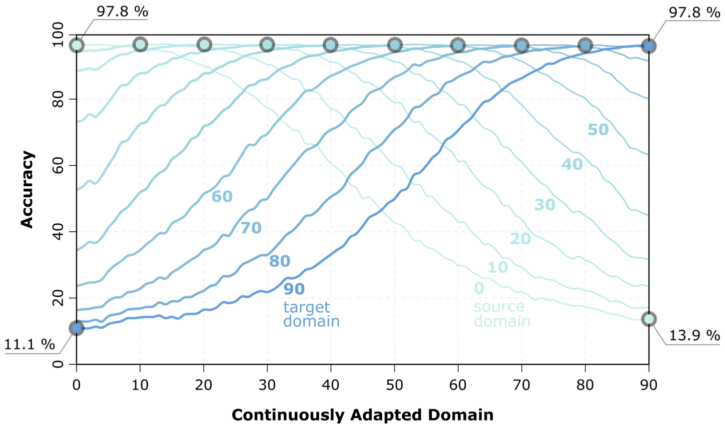
Central to any domain adaptation approach is its capability to adapt to a target domain, aiming for source domain performance. Based on a configuration with exclusively training self-supervised (see configuration (I) in Tab. 5.7), the capability for domain adaptation into the target domain is optimal. The characteristics of the domain adaptation approaches are depicted (see Fig. 5.14) within the novel validation system (introduced in Fig. 4.2).

The source domain accuracy of the model trained on the source domain is 97.8% (see cyan dot with gray border in the upper left corner of Fig. 5.14a) and used as a reference point and baseline for the domain adaptation approach. The initial model has a target domain accuracy of 11.1%, approximately equivalent to a random classifier's performance ($\approx$ 10%) for a task with ten classes (digits from 0 to 9). Once the domain adaptation process is completed, the target domain accuracy achieves 97.8% (see blue dot with gray border in Fig. 5.14a), reaching source domain performance in the target domain. A closer inspection (see dots with a gray border at the top in Fig. 5.14a) makes evident that this finding holds for each intermediate domain, continuously reaching source domain performance during adaptation: 97.8%, 98.1%, 98.0%, 98.0%, 97.9%, 97.9%, 97.8%, 97.7%, 97.8%, and closing 97.8%.

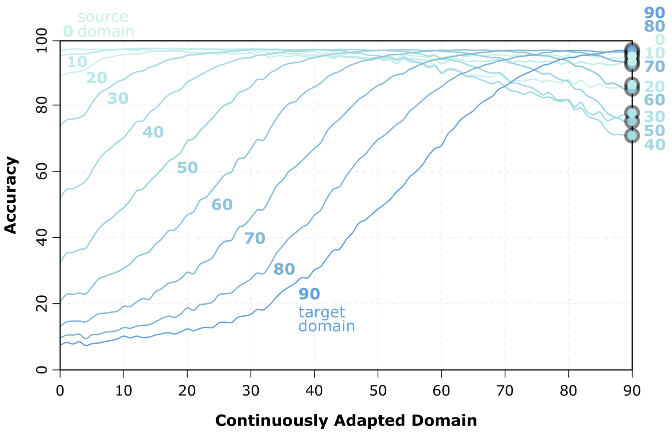
Further, a drop in domain performance is observable once the model adjusts to the adjacent intermediate target domain. Both the performance increase and deterioration follow a wave-shaped pattern. The more distant the continuously adapted model gets from a specific domain, the more the performance deteriorates. For example, the source domain performance drops to an accuracy of 13.9% when the model just reaches the target domain (see cyan dot with gray border in the lower right corner of Fig. 5.14a). This effect is best known as catastrophic forgetting. Limiting catastrophic forgetting is another central aspect of domain adaptation approaches.

Limiting Catastrophic Forgetting

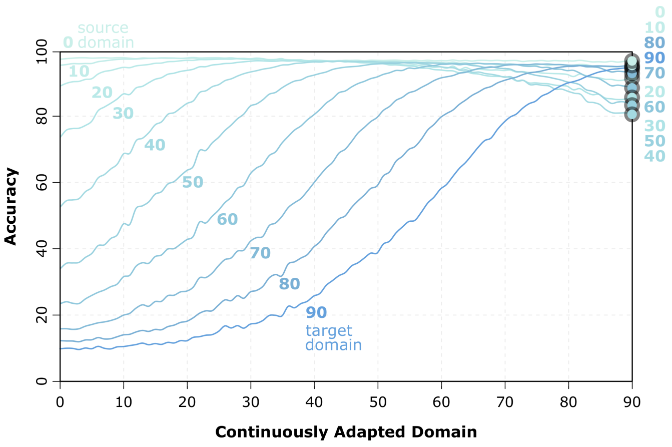
The most salient difference when comparing configurations (see Tab. 5.7) with retraining on the source domain (see Fig. 5.14b and Fig. 5.14c), with



(a) Configuration (I): Zero epochs retraining, five epochs adaptation training.



(b) Configuration (II): One epoch retraining, five epochs adaptation training.



(c) Configuration (III): Two epochs retraining, one epoch adaptation training.

Figure 5.14: COSS showing different levels of catastrophic forgetting. Development of the classification accuracy over the continuously adapted domain, the model has been trained on in the process of continuous self-supervised domain adaptation. The colored lines (from the cyan source domain to the blue target domain) represent the evaluation on a set of domains (for a domain indication, see the colored captions), ranging from the source to the target domain. The dark blue line, for example, represents the evaluation on the target domain over the domains that have been continuously reached by the model at a given step within the adaptation process. The accuracy of a model evaluated on a domain, once the model continuously reaches the target domain, is depicted as a colored dot with a gray border. Adapted from [2].

those that do not retrain (see Fig. 5.14a), is the absent drop in accuracy after the domain has been adapted to. Especially once reaching the target domain, this is reflected by the built knowledge factor f_K (std) over all domains (see definition in Eq. 4.6). For configuration (I), without retraining, f_K is 56.7% (±32.4%). For configuration (II), with a single epoch of retraining, f_K rises to 88.2% (±8.7%). Catastrophic forgetting in the source domain almost vanished (CF of 1.6%) while keeping the domain adaptation DA at a comparable level of 86.2%. Retraining also has a major effect on the standard deviation of the accuracies, reducing the standard deviation by 73.15%.

The order in which the model performs on different domains when reaching the target domain allows further observations (see Fig. 5.14b). Domains that only have been recently adapted to, as expected, rank highest for accuracy: 90 (97.6%) and 80 (97.1%); the same holds for the source domain that has been used for retraining: 0 (96.3%). To be highlighted is the subsequent order of the domains' performances, which get in line with respect to a domain's distance to the domains that have been trained on the latest ($\mathscr{D}_0$, and $\mathscr{D}_{90}$,), meanwhile dropping in accuracy: 10, 70, 20, 60, 30, 50, and finally 40 (72.6%). The domains located closer to the target domain and thus to the side that experiences five times the number of training epochs have an accuracy advantage.

Once the training ratio is inverted and moderated, resulting in a single adaptation training epoch and two epochs retraining (Fig. 5.14c), the order in which the model performs on different domains conforms to the expectation. Domains close to the source domain, used for training, rank highest for accuracy: domain 0 (97.6%) and domain 10 (96.1%); the same holds for domains that only have been recently adapted to, domain 80 (95.7%) and domain 90 (95.2%). The case of the target domain running up behind domains 0, 10, and 80 is due to incomplete convergence, which is a result of the low number of training epochs the model has experienced for adaptation in the target domain. Similar effects can be found throughout the adaptation process. For example, when reaching domain 80, accuracy on domain 70 (96.0%) outperforms accuracy on domain 80 (95.6%). Respectively continued, this pattern is consistently observable.

When overlaying configuration (II) and (III), the mitigation of the convergence, by reducing the number of adaptation training, becomes even more apparent (see solid lines in Fig. 5.15). First, the optimal in-domain accuracy, as evaluated on the respective domains, of configuration (III) falls short

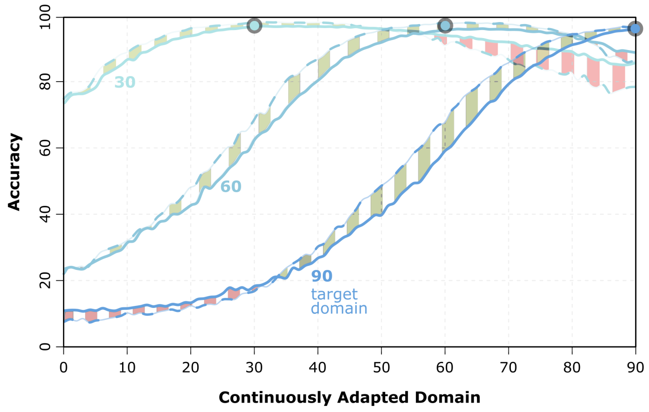


Figure 5.15: Development of the classification accuracy over the continuously shifting domains the model has consecutively been trained on in the process COSS. The dashed lines represent configuration (II) (one epoch retraining, five epochs adaptation training), and the solid lines represent configuration (III) (two epochs retraining, one epoch adaptation training). The evaluation of the two configurations is compared with each other on the exemplary domains 30, 60, and 90. The bar pattern between two lines evaluated on the same domain highlight the shift of their accuracy development during continuous domain adaptation. When continuously reaching a domain, a model's accuracy evaluated on that domain is depicted as a colored dot with a gray border.

compared to configuration (II). Besides, the accuracy gained per adaptation step (see the green area between dotted and solid lines in Fig. 5.15) is higher for configuration (II) when approaching the in-domain performance. At the same time, configuration (II) shows a higher loss in accuracy per adaptation step when moving away from the in-domain (see the red area between dotted and solid lines in Fig. 5.15). These two effects can be traced back to the different ratios of retraining and adaptation training results and become visible in two intersections of the configurations' curves.

Nevertheless, configuration (III) shows a long-term sustainable and more stable development of the overall accuracies with slight accuracy deterioration, little catastrophic forgetting of 0.4%, and in comparison, an improved mean accuracy of 91.2% and a further reduction in the standard deviation of the accuracies to 5.3%. (see the entire performance in Tab. 5.7).

Coping with Adaptation Step Size

Domain gap width or the continuous domain shift's adaptation step size is central to the self-supervised continuous domain adaptation approach. This has been analyzed and established in Fig. 5.12b. In consequence, the general applicability of the continuous domain adaptation approach depends on how far the domain adaptation can reach in a single adaptation iteration. Data preprocessing, here, controlled rotation is able to influence the domain width between two adjacent domains. However, this controllability and potential for influence can not be taken for granted and is highly dependent on the configurability of the dataset and task. Based on configuration (III) (see Tab. 5.7), the capability and reach of bridging the domain gap by the self-supervised continuous domain adaptation approach is elaborated and visualized (see Fig. 5.16).

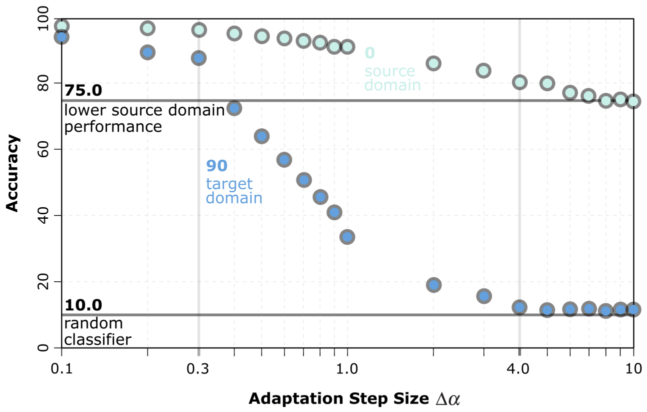


Figure 5.16: Development of the classification accuracy over a log-scale of the adaptation step size (inter-domain step size $\Delta\alpha$) during the process of the continuous self-supervised domain adaptation (see configuration (III) in Tab. 5.7). The classification accuracy is visualized for an evaluation on the source domain (cyan dots with gray border) and the target domain (blue dots with gray border). The lower solid gray line depicts the performance of a random classifier, and the upper solid line represents the lower end of the source domain performance. Adapted from [2].

From the previous configuration analysis (see Fig. 5.12b), it is evident that the smaller the gap or shift between two adjacent domains, the more successful the continuous domain adaptation becomes. Up to now, smaller step sizes have been equivalent to an increased number of overall training epochs. Compensating for the number of epochs, they have been scaled concerning a configuration with an intermediate step size of $\Delta\alpha = 0.1$. This results in $r * \Delta\alpha/0.1$ epochs retraining and $n * \Delta\alpha/0.1$ adaptation training epochs. The observations remained comparable, whereas smaller domain steps are highly beneficial to the approach.

Increasing the domain step between two adjacent domains, at some point, breaks the approach's initial assumption of superimposition (see Sec. 3.3), which is necessary to ensure the validity of the model's prediction and accuracy of the pseudo-label generation. Thus, the exponential decline in domain adaptation performance does not come as a surprise, as errors in the pseudo-label generation propagate and build up the prediction error. Defining the range in which an approach is functional is ambiguous without referring to a specific use case. The missing-context impediment is resolved by the deployment of heuristic baselines to enable the assessment of approach performance into two stages. The first baseline is at $\approx 75\%$, at the lower end of source domain performance, up to the point of $\Delta\alpha \leq 0.3^\circ$ (88.4%) where domain adaptation and source domain performance are in an equal position. The second baseline is at $\approx 10\%$, the performance of a random classifier, starting with $\Delta\alpha \geq 4^\circ$ (12.8%), the effects of domain adaptation are no longer measurable. As the retraining is performed in a supervised manner, there is only a slight increase in catastrophic forgetting $\Delta\alpha \leq 0.3^\circ$ (2.5%), which becomes more visible when the adaptation process falls apart in general $\Delta\alpha \geq 4^\circ$ (20.0%).

Coping with Adaptation Repetitions

Self-supervised domain adaptation in practice targets the deployment within applications that inherit a constant, continuous shift. This Section concentrates on the convertible performance guarantees when coping with a large number of adaptation repetitions is crucial for continuous adaptation while experiencing deployment shift. Further deployment-specific and use-case-oriented considerations are postponed (see Sec. 5.7.1).

In the last consequence, the general applicability of continuous domain adaptation depends on coping with a large number of iterative adaptation repetitions. Based on a configuration without retraining (see (I) in Tab. 5.7),

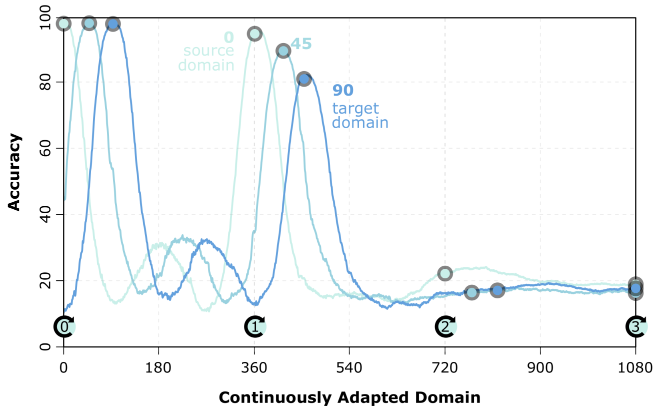


Figure 5.17: Development of the classification accuracy over multiple continuous domain shift repetitions in which the model is being adapted through COSS. The plot depicts the accuracy development over three 360° rotations (revolutions are reflected by a spinning arrow on the cyan ground, with the respective count). The lines represent the evaluation of configuration (I) (no retraining, five epochs adaptation training) on a set of domains for improved clarity, exemplary for domains 0, 45, and 90. When continuously reaching a domain, the model's accuracy evaluated on that domain is depicted as a colored dot with a gray border.

the limits of adaptation repetitions by the self-supervised continuous domain adaptation approach are elaborated and visualized in Fig. 5.17.

The previously presented experiments (see Fig. 5.14 ff.) only demonstrate COSS' ability for continuous domain adaptation on the rotatedMNIST in the limited range from 0° to 90°. In theory, continuous domain adaptation can be repeated infinitely. Due to the ideally suited characteristics of the rotatedMNIST dataset, which allows for an infinite domain shift by rotation, the empiric capabilities concerning adaptation repetition are stress tested in the following experiment. The recurrent zero degree crossings at multiples of 360° enable the comparison of performances concerning the number of revolutions on a specific domain (see Fig. 5.17).

The first finding is that the approach is capable of adapting a model to the full range of domains during the first revolution of the sample: From domain 0 (97.9%) to domain 360 (94.7%), with only a slight accuracy deterioration.

Subsequently, the approach experiences a drop in accuracy raised by error propagation within the pseudo-label generation. This friction loss motivates further work on bypassing forgetting during self-supervised continuous domain adaptation and the need for auxiliary and supportive methods to stabilize continuous domain adaptation.

Cue-based Bypassing of Catastrophic Forgetting

Until this point, the findings reflect that catastrophic forgetting is to be limited by retraining supervised on the source domain (see Fig. 5.14c and Fig. 5.15). As a second strategy, CUEB (see approach details in Sec. 3.3.3) successfully bypasses catastrophic forgetting during continuous domain adaptation.

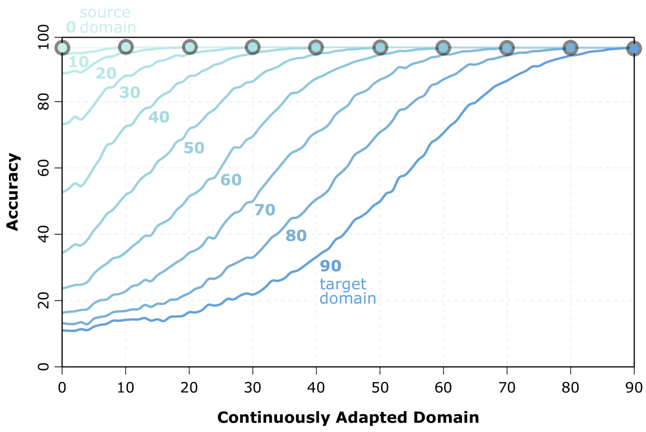


Figure 5.18: CUEB Bypassing Catastrophic Forgetting. Classification accuracy over the continuously shifted domains the model has been trained on in the process of CUEB. The colored lines (from the cyan source domain to the blue target domain) represent the evaluation on a set of domains (for details, see colored captions), ranging from the source to the target domain. The dark blue line, for example, represents the evaluation on the target domain over the domains that have been continuously reached by the model at a given moment. When continuously reaching a domain, a model's accuracy evaluated on that domain is depicted as a colored dot with a gray border. Notice how the performance after the in-domain step remains constant due to the cue-based classification and continuous model extension. The effects on performance due to model miss-selections due to the cue prediction are included; nevertheless negligible. Adapted from [2].

This bypass is achieved by keeping domain-specific parameters available and accessible by employing a cue. The effect is that in-domain performance is maintained throughout the continuous domain adaptation. Figure 5.18 illustrates how the model's accuracy, for a given domain, remains constant after it has been reached (colored dots with gray border) during continuous domain adaptation. When reaching a domain and thereby reaching the optimal performance of a model on that domain, the model's classification layer is stored and linked with a domain cue for subsequent retrieval. The domain classification model θ^{DC}, has a mean accuracy (std) of 98% ($\pm$0.7%) for cue-prediction, averaged over all domains and classes. However, the expected 2% cue-driven model miss-selection has no impairing influence on the adaptation performance (see Fig. 5.18).

The approach is an extension to COSS, the previously presented continuous domain adaptation approach, merging domain adaptation capabilities over continuous domains, with minimal catastrophic forgetting and maximal domain adaptation with respect to the transitioned domains (see f_K CUEB in Tab. 5.8).

Configuration	**Metrics**		
	DA	CF	f_K
(I) $o50\_r0\_n5\_\Delta\alpha0.1\_c0.0$	0.868	**0.840**	0.567 (0.324)
(III) $o50\_r2\_n1\_\Delta\alpha0.1\_c0.0$	0.847	**0.004**	0.912 (0.053)
(IV) CUEB on (I)	0.863	**0.005**	**0.974 (0.005)**

Table 5.8: Self-supervised continuous domain adaptation configurations with respect to domain adaptation DA, catastrophic forgetting CF, and the built knowledge factor f_K after the last adaptation step. Configuration (I) achieves the optimal DA, demonstrating the capability to continuously reach source domain performance in the target domain. CUEB on configuration (I) achieves the optimal f_K, which indicates a constant decrease of catastrophic forgetting along all domains, up to and beyond reaching the target domain. While ranking a narrow second in DA and CF for comparison, see configuration (III). CUEB renders retraining unnecessary while at the same time reducing catastrophic forgetting (in bold).

Sample size efficiency is ideal; neither samples nor labels need to be kept available after training; this suits imminent and continuous training scenarios, with effects comparable to COSS configurations without retraining. The approach allows integrating newly recorded data into an already learned intermediate domain $\mathscr{D}_\alpha$. For this purpose, only the layer θ^C_α for the corresponding domain has to be trained on the intermediate domains. In turn, the store-keeping of domain-specific parameters throughout the continuous domain adaptation process results in a decline in model size efficiency when it comes to memory requirements to store the model and its cue-linked layers and backbones.

5.6.4 Continuous Learning for Semantic Domain Shifts

Deep learning increasingly advances biomedical research, deploying neural networks in a wide range of different computer vision tasks, such as classification, object detection, and not least, semantic segmentation. Neural networks, though are commonly trained supervised on large-scale, labeled datasets. These preconditions present a challenge in biomedical pattern recognition, as datasets are in most cases small-scale, challenging to obtain, expensive to label, and frequently heterogeneously labeled. Furthermore, heterogeneous labels are a challenge for supervised methods, due to their inherent semantic domain shift. The semantic domain shift arises if for an individual sample not all classes are labeled.

By framing heterogeneous labels as a semantic domain shift, and making use of COSS, it is possible to continuously train a neural network while adapting to semantic domain shifts. The target domain $\mathscr{D}_T$ is characterized by samples missing source domain ground truth but introducing target domain labels. The training on the target domain is enabled by a class asymmetric loss tailored[15] to the needs of COSS - allowing for incomplete or heterogeneous ground truth information (see Fig. 5.19). For development and validation purposes, a biomedical small-scale, multi-class semantic segmentation dataset is needed. The in this work introduced heartSeg dataset (see Sec. 4.2.3) is based on the biological model system of the Medaka fish's cardiac system and provides labels for the atrium as well as ventricle.

Continuous semantic domain adaptation

Initially, the dataset is fully labeled $\tilde{\mathbf{Y}}_S$, holding ground truth information of a single class across all samples $\mathbf{X}$. The semantic domain shift unfolds within the ground truth information, expanding by labels of a second class $\tilde{\mathbf{Y}}_T$.

[15] Frugal labeling repository: https://osf.io/uyk79/

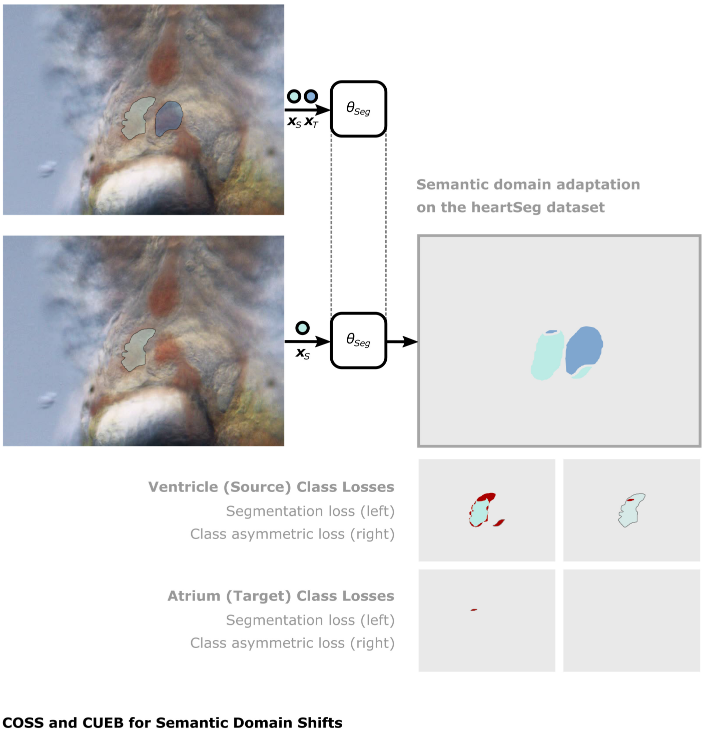

x_S x_T
θ_{Seg}
x_S
θ_{Seg}
Semantic domain adaptation on the heartSeg dataset
Ventricle (Source) Class Losses
Segmentation loss (left)
Class asymmetric loss (right)
Atrium (Target) Class Losses
Segmentation loss (left)
Class asymmetric loss (right)
COSS and CUEB for Semantic Domain Shifts

Figure 5.19: Objective functions for semantic domain adaptation. The images in sequence from the top, depict forward passes of a training step during the domain adaptation. In the bottom image, only the source domain ground truth is available, while in the top image the semantic shift already provides additional atrium labels. A class asymmetric objective function ensures that training in the target domain is carried over multiple training steps without reflecting a loss (see the empty tile on the bottom right) based on a missing target domain label mask within source domain samples. False predictions for target domain instances, that overlay source domain labels, however, are reflected in the loss (see the small patch to the left on the bottom left tile). This approach allows to build up knowledge in the target domain while being able to retrain on the source domain at the same time.

Thereby, arises the need for training on the target domain. This is non-trivial, due to the necessity to bypass catastrophic forgetting on the source domain while ensuring the preservation of domain knowledge in the newly emerging target domain.

Thus, validation revolves around the relation between domain adaptation (measured as mean intersection over union $mIoU$) and the magnitude of label availability (measured as the percentage of dropped labels in percentage %) in the target domain (see Fig. 5.20). Specifically, this means that the dataset is modified such that the ventricle class is always fully labeled, and the atrium class contains a decreasing percentage of labels to simulate varying magnitudes of semantic domain shifts in the ground truth of the data, reflected by the availability of atrium labels.

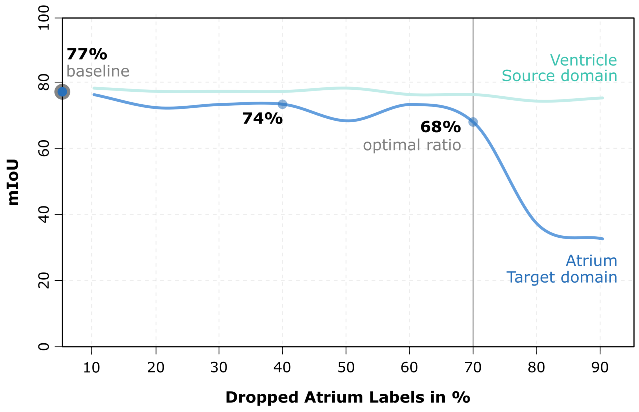


Figure 5.20: According to the $mIoU$, the results are categorized over the percentage of dropped atrium labels. The performance of the ventricle class stays relatively constant. This is to be expected as the constant availability of ventricle labels is synonymous with the continuous retraining of the COSS method. The atrium class shows a slowly decreasing performance when increasing the number of dropped labels. Dropping 40% atrium class labels from our dataset, results in a marginal performance deterioration of $\approx 2\%$ in IoU, compared to the benchmark in which all labels are available. The baseline is depicted as a blue dot with a gray border, while the optimal performance-label ratio, is highlighted by a gray line. All values are averaged over ten consecutive runs. Adapted from [12].

The general ablation study, forms the baseline for the semantic domain adaptation experiment. Starting from training on the fully labeled atrium (73% *mIoU*) and ventricle class (78% *mIoU*), labels are continuously dropped from both classes. When 70% of the labels are dropped, the performance of the atrium is at 27% and the ventricle 38% of the *mIoU* metric over all classes, averaged over 10 consecutive runs.

The continuous semantic domain adaptation (see Fig. 5.20), depicts a constant performance on the fully labeled ventricle class at around 80% *mIoU*. The constant performance in the source domain illustrates the capability of bypassing catastrophic forgetting during semantic domain adaptation. Further, the performance on the atrium class representing our target domain remains stable up to 70% dropped atrium labels, where the performance is at 68% *mIoU*. These results highlight the potential to train on sparsely labeled target domain samples.

Towards cue-based self-supervision for spatial semantic domain shifts

The continuous semantic domain adaptation approach demonstrates a frugal character with respect to the required labels in the target domain. The semantic domain shift enables the extension of the ground truth by an additional class.

In cases in which, the emerging class has a large semantic overlap with the source domain class, the domain shift opens the opportunity for framing it as a spatial domain shift. With this framing, a spatial cue-based approach could in turn make a pseudo-label approach feasible. Coming back full circle to the COSS and CUEB approaches.

The continuous semantic domain adaptation further allows for domain adaptations along two unfolding semantic shifts. For example, two datasets with different source domain ground truth (heterogeneously labeled), continuously converge into each other, forming a single dataset when they both reached their common target domain.

Semantic segmentation enables high-throughput experiments and is of high interest to biomedical research. Our approach and analysis show competitive results in supervised training regimes and encourage frugal labeling for continuous semantic domain adaptation within biomedical image recognition.

5.7 Discussion and Conclusion of the Novel Methods

Concluding, the novel approaches are taken apart, set out, and subsequently confronted with each other. In this way, the novel approaches are contextualized within the novel unsupervised domain adaptation concept. The advantages and adversities of the methods are highlighted and discussed. The Section is complemented by application recommendations and best practices for the here presented approaches.

5.7.1 Concluding the Domain Adaptation Evaluation

Further research, development, and benchmarking of unsupervised domain adaptation approaches are enabled by providing two novel datasets. Both the rotatedMNIST dataset, as well as the HeartSeg dataset, are purpose-designed with respect to domain adaptation approaches for cognitive perception systems. In addition, introducing a novel evaluation concept (see Sec. 4.1) for domain adaptation approaches renders the datasets into benchmarks. The presented evaluation concept and metric set enable the assessment of unsupervised, discrete domain adaptation approaches and continuous domain adaptation approaches. In this way, the evaluation results become comparable across the board of different domain adaptation methods.

Concluding the Sample Supplementation Approach

The capabilities of the newly introduced generative domain extension approach SSUP for object detection at night[16] have been presented. This work highlights the potential of training on generated samples, including their respective ground truth $e_A = 1$, for domain extension[17] [6]. Training on the generated samples is compute efficient $e_C = 1$, as the expected training efforts for fine-tuning are commonly less than the initial source domain training. However, the retraining is weak when it comes to sample efficiency $e_S = 0$, as the generated samples need to be kept in memory for supplementation during the domain adaptation process. Characteristic for fine-tuning approaches such as SSUP, the adapted model remains constant in model efficiency $e_M = 1$ and compute efficiency $e_R = 1$ during inference.

[16] Patent on the generation of validation data with generative adversarial networks [a]
[17] Patent on training approach for recurrent neural networks based on generated data [b]

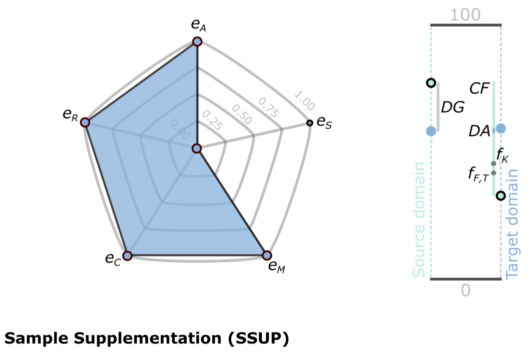


Figure 5.21: Aggregated evaluation of the SSUP method based on application metrics within the radar plots (left) and task metrics within the novel evaluation plots (right). The radar plots depict efficiency metrics for annotation, sample, model, computational, and run-time efficiency (for a detailed specification, see Sec. 4.1.3 and Fig. 4.3). The task metric plot depicts the initial domain gap DG (gray bar), the domain adaptation DA (light blue bar), and catastrophic forgetting CF (cyan bar). For a detailed definition of the task metrics see Section 4.1.2 and Figure 4.2. The cyan dots represent evaluation on the source domain, and the blue dots represent evaluation on the target domain.

While the sample generation is proven to be successful when approaching it from a feature-level perspective, the gains are not trivial to harvest. The difficulties are highlighted by the experienced catastrophic forgetting and fray factor $CF = f_F = 41.1\%$ and are underlined by the poor knowledge built up $f_K = 46.1\%$ along with the arguably negligible domain adaptation of $DA = 0.8\ \%$. Especially when generating samples in a challenging domain, such as the night domain, in which low information areas are a characteristic of the domain itself. Further, the research helped broaden an understanding and the process of quantifying domain gaps $DG = 16.3\%$ (here between day and night object detection performance) and the mechanism of domain extension. It immensely helped lay the foundation for approaching the task from another angle, substantiating through the subsequent approaches.

Concluding the Shared Latent Feature Space Approach

This section has shown the capabilities of the novel cross-domain adaptation approach SHALFS in object detection at night[18] [7]. The beneficial characteristics[19] of unsupervised image-to-image translation have been deployed in an advanced driver assistance use-case.

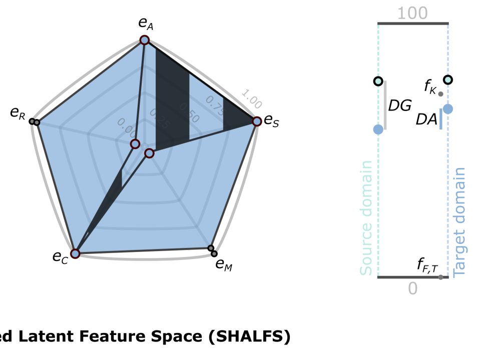


Figure 5.22: Aggregated evaluation of the SHALFS method based on application metrics within the radar plots (left) and task metrics within the novel evaluation plots (right). The radar plots depict efficiency metrics for annotation, sample, model, computational, and run-time efficiency (for a detailed specification, see Sec. 4.1.3 and Fig. 4.3). The task metric plot depicts the initial domain gap *DG* (gray bar) and the domain adaptation *DA* (light blue bar). Catastrophic forgetting *CF* is circumvented and can thus not be illustrated. For a detailed definition of the task metrics see Section 4.1.2 and Figure 4.2. The cyan dots represent evaluation on the source domain, and the blue dots represent evaluation on the target domain.

The shared latent feature space approach does not require keeping samples or labels in memory $e_S = 1$ during domain adaptation. SHALFS deployment as a cross-domain adaptation for object detection by night enables decreasing the domain gap ($DG = 16.3\%$ between day and night before domain adaptation) by $DA = 5.3\%$ and a knowledge built up at $f_K = 60.75\%$, without experiencing catastrophic forgetting $CF = f_F = 0\%$. Domain adaptation is achieved without additional labeling effort $e_A = 1$ and enables the direct deployment of pretrained object detectors[20] without having to retrain $e_C = 1$

[18] Patent on object detection for night perception systems [g]
[19] Patent on parametrization for artificial neural networks [d]
[20] Patent on unauthorized passenger recognition [h]

while unfolding into the target domain. At the expense of the cross-domain adaptation module, slightly decreasing overall model efficiency $e_M < 1$ and run-time efficiency $e_R < 1$, the modular framework and the demonstrated interchangeability of the object detector within the cross-domain adaptation increase flexibility and functional adaptation through the object detector and potential for functional extensions[21].

Concluding the Continuous Domain Adaptation Approaches

The results of self-supervised continuous domain adaptation[22] demonstrate the configuration space analysis and capabilities of this novel approach [2]. The extension through CUEB[23] utilizes the context-knowledge about a continuous domain shift, as well as a continuous distribution characteristic within a dataset. CUEB optimizes sample efficiency for CUEB to $e_S = 1$ as no repetitive source domain training is necessary, as opposed to COSS where source domain retraining routines $e_S = 0$ are central to mitigate catastrophic forgetting.

The self-supervised continuous domain adaptation approach iteratively overcomes the performance gap between source domain, intermediate domains, and target domain $DG = 86.7\%$ by utilizing unlabeled and, in the process, pseudo-labeled $e_A = 1$ large-scale datasets. In use cases, such as autonomous driving, these unlabeled large-scale datasets are readily available. Further, the entire dataset does not have to be available during the initial source domain training, enabling continuous training and seamless knowledge built up at $f_K = 0.912$ for COSS and $f_K = 0.974$ for CUEB during continuous domain adaptation. COSS comes at the expense of computational efforts $e_C = 0$, while model efficiency $e_M = 1$ and run-time efficiency $e_R = 1$ is optimal as the architecture is not extended as the model's weights are merely updated. With this research, methods that enable bypassing catastrophic forgetting have been designed and analyzed. It is shown how catastrophic forgetting can be mitigated by retraining routines (COSS) $CF = 0.004\%$ or entirely bypassed $CF = f_F = 0\%$ by extending the neural network architecture by adding a cue-mechanism (CUEB). Bypassing catastrophic forgetting enables the safe, continuous deployment of models.

[21] Patent on input data compression for anonymization during acquisition [p]
[22] Patent on continuous domain adaptation for neural networks based on pseudo-labels [i]
[23] Patent on cue-based inference for domain adaptation by neural networks [k]

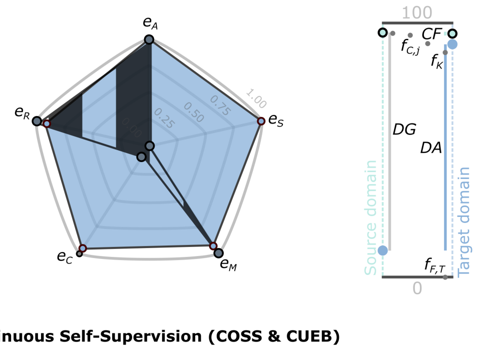


Figure 5.23: Aggregated evaluation of the CUEB (light blue fill) and COSS (gray-transparent striped fill) method based on application metrics within the radar plots (left) and task metrics within the novel evaluation plots (right). The radar plots depict efficiency metrics for annotation, sample, model, computational, and run-time efficiency (for a detailed specification, see Sec. 4.1.3 and Fig. 4.3). The task metric plot for COSS depicts the initial domain gap *DG* (gray bar), the domain adaptation *DA* (light blue bar), and catastrophic forgetting *CF* (cyan bar). The cyan dots represent evaluation on the source domain, and the blue dots represent evaluation on the target domain. For detailed information on the adaptation process metrics (f_C, f_F, and f_K) for specific configurations, it is suggested to revisit Section 5.6. For a detailed definition of the task metrics see Section 4.1.2 and Figure 4.2.

Potentially a large set of computer vision tasks and applications are able to be interpreted and stated as continuous domain adaptation problems. Applications are expected to apply the continuous domain adaptation to additional object classes, differing sensor modalities, and sensor positions or deployment of the novel approach to different tasks, such as semantic segmentation or object detection. This work will serve as an adaptation approach to help solve the challenge of measuring and reducing the domain gaps inherent to data-driven applications while bypassing catastrophic forgetting and annotation efforts in continuously changing domains. Not least, the domain-specific neural network extension method opens the possibilities of computational efficient continuous training $e_C \approx 1$, as merely domain-specific high-level layers need to be adapted and stored $e_M < 1$. Computational efficiency for training improves by approximately 98.7% compared to COSS since the low-level feature representation layer (401920 parameters) only has to be trained only once and remains constant for subsequent inter-domain steps and thus parameter updates. Making the core feature extraction reliant on a pretrained backbone, the adverse effects of adding a cue-mechanism on model efficiency and run-time efficiency[24] $e_R \approx 1$ are mitigated. Run-time efficiency remains near optimal, as the number of operations only slightly increases from 407050 operations by 1.2%, due to the additional 5130 operations needed to predict the cue.

5.7.2 Cross-Method Selection and Comparison

In the following, the novel approaches to unsupervised domain adaptation are assessed within a general deployment context. The limitations and advantages of the methods for applications are further identified. First, the crucial characteristics and capabilities of the methods are detailed. To conclude, the different methods are put in relation to each other, and application recommendations are given.

Cross-Method Comparison

The methods proposed and developed within this thesis are compared using application metrics introduced in the concluding evaluation of the domain adaptation methods in Section 5.7.1 and presented in the aggregated form of an overlay in Figure 5.24.
In general, unsupervised domain adaptation implicates the need to adapt a neural network to an intermediate domain or target domain that has not been part of the initial source domain and does not obtain explicit ground truth information, such as labels. Being independent of ground truth in the target domain, unsupervised domain adaptation methods in all shapes are efficient with respect to additional annotation effort. Sample efficiency diminishes for generative approaches such as SSUP, requiring access to stored labels, and for continuous approaches such as COSS that are stabilized by supervised continuous retraining, which requires access to the source domain or intermediate domain samples and labels. In SSUP's pure retraining-driven domain adaptation, model efficiency is optimal, as the number of parameters within the model remains the same. If the domain adaptation is realized by an architectural extension, such as the cue-mechanism of CUEB, or a modular extension, such as the cross-domain adaptation module of SHALFS, model efficiency inevitably drops accordingly. Depending on the size of these extensions and their role during inference, this directly correlates with

[24] Patent on cue-based run-time reduction for neural networks [f]

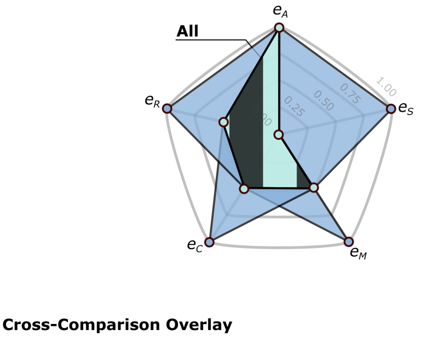


Figure 5.24: Overlay of the radar plots of SSUP, SHALFS, COSS, and CUEB. The cyan-striped area is covered by all methods and approaches. When requirements exceed this area, a particular approach needs to be selected to satisfy them. Potentially reachable areas are depicted in light blue.

the run-time efficiency of the model. To conclude, the here presented continuous domain adaptation methods, such as COSS and CUEB, as well as the retraining-driven generative approach of SSUP, require a large amount of compute.

Best Practices

Discrete unsupervised domain adaptation, such as in SSUP or SHALFS, enables larger domain gaps and does not require the two domains to be interconnected. This also brings the advantage that detailed knowledge of the shift is not required to train the domain adaptation module. The modular characteristic further allows deployment of the neural network on the source task, which can thus be capsuled from the domain adaptation itself.

However, it must not be forgotten that unsupervised domain adaptation either engages with newly forthcoming domains or, until then, neglected domains. Thus, real-world deployment and open-world scenarios provide enough pretext for unsupervised domain adaptation to be understood as a continuous process tackling the emerging deployment domain gap over time.

A retraining framework or a means for continuous adaptation thus always needs to be an integral part of deploying a neural network.

As framed in this thesis, continuous unsupervised domain adaptation, as promoted in COSS and CUEB, heavily relies on the source domain performance that has been achieved by supervised learning. Consequently, high source domain performance is beneficial to the domain adaptation outcome. Moreover, it has been shown that retraining on the source domain, as well as opting for smaller domain shift increments, acts in a stabilizing way for continuous domain adaptation. Otherwise, there is a risk of the adaptation capability breaking down gradually. Further, continuous domain adaptation depends on the prior of the continuous, infinitesimal shift. Thus, in continuous self-supervised domain adaptation, the domain shift needs to be well understood and structured to be helpful. On the other hand, it becomes evident that forgetting is by no means inevitably catastrophic and, at times, even beneficial. Consequently, this research comes to question whether the notion of forgetting in deep learning is purely catastrophic. This is especially the case when a task experiences semantic shifts over time. At last, when deploying unsupervised domain adaptation, method selection is a matter of intended and required model behavior and the domain context of the deployment.

6 Conclusion

Unsupervised, continuous development and deployment of neural networks across multiple domains pose high demands on the employed methods and models. Even minor domain adaptations are complex and often require a renewed neural network deployment cycle from square one. However, it is neither realistic nor economical to develop conclusively generalized neural networks. Moreover, in high-performing neural network applications, the domain adaptation is commonly reliant on a supervised learning strategy based on time-consuming, expensive, manually annotated data. As unlabeled large-scale datasets become predominant, they originate the need for research and development regarding previously neglected unsupervised domain adaptation approaches and their application's prerequisites.

This work proposes deploying unsupervised domain adaptation approaches to extend the unsupervised and continuous development of neural networks. The thesis categorizes types of domain shift and capitalizes on the source domain's already available knowledge representations to enable further unsupervised domain adaptation on unlabeled data. Methods for the efficient extension and transformation of existing deep learning modules are developed to resolve the problem statement of neural network adaptation across multiple domains. These methods get cognitive perception systems ready for the challenges of the repetitive emergence of new domains. Therefore, this work proposes novel self-supervision and unsupervised domain adaptation approaches for continuous learning, expanding the neural networks' domain of operation. The core contributions are summarized subsequently.

To begin, the characterization of domain states and analysis of their implications on neural network training in the context of domain adaptation (see Ch. 2). Research and development of novel unsupervised domain adaptation approaches concerning different domain states: Sparse domains (see Sec. 3.1), discrete domains (see Sec. 3.2), and continuous domains (see Sec. 3.3). The novel approaches for unsupervised domain adaptation are followed by the modification and extension of metrics concerning the different variants of domain adaptation (see Sec. 4.1). Further, methods to interpret and visualize domain states and the effects of domain adaptation approaches are presented. Closing, the design, and generation of datasets (see Sec. 4.2) to support domain adaptation research and development are discussed and put into practice. Building on the, in this work, developed and implemented domain adaptation approaches, these are evaluated on large-scale datasets for autonomous driving by night (see Ch. 5). Concluding, the different domain adaptation approaches are compared, thereof method selection criteria are identified, and application recommendations are given. The publicly available implementation of the domain adaptation approaches provides full-stack repositories (see App. A.1), which enables to connect and tie future research to this work.

The key message of this work is that unsupervised domain adaptation for cognitive perception systems is feasible. In combination, the outcomes take a stand for a paradigm shift from supervised learning stratagems to unsupervised learning stratagems. Considering the future development of cognitive perception systems, unsupervised learning stratagems live up to the emergence of unlabeled large-scale datasets. The most significant efforts of this work have been channeled into scientific publications and patents (see A.4), the latter are in the following compiled to a list:

1. Development of a concept for domain state characterization enabling the design of domain adaptation methods based on deep learning approaches, including network architectures, hyperparameter determination, and training process design.
2. Systematic design of a concept for unsupervised domain adaptation, covering the full cycle of continuous development and integration of neural networks under deployment.
3. Integration of novel domain-centered metrics and performance indicators for the quantification of domain adaptation. This work contributes a strategy to base the quantitative evaluation of unsupervised and continuous domain adaptation methods on task performance, adaptation performance, and deployment requirements. Hereby reaching the objective of making domain adaptation approaches quantifiable, comparable, and interpretable.
4. Providing a qualitative and quantitative means for representation of domains, domain shifts, and domain adaptation performances. The descriptive and interpretable representations give insights into the underlying domain characteristics relevant to the adaptation process.

5. Making datasets available for domain adaptation-specific training and validation by generating novel datasets and customizing existing datasets in accordance with their domain characteristics. The results are purpose-designed benchmarks for different types of domain adaptation.
6. Development and implementation of SSUP for generative domain extension in sparse domains or discrete domain shifts, based on a semi-supervised retraining approach.
7. Development and implementation of SHALFS for unsupervised domain adaptation for discrete domain shifts based on a shared latent feature space. SHALFS provides a novel way for the implicit cross-domain deployment of pretrained neural networks, opening the possibility of bypassing catastrophic forgetting and seamless and modular extensions of the operational design domain.
8. Development and implementation of COSS for self-supervised domain adaptation in continuously shifting domains based on pseudo-label generation.
9. Development and implementation of CUEB, enabling an architectural approach to bypass catastrophic forgetting in continuous self-supervised learning scenarios.
10. Proof of concept and functionality through applications in autonomous driving prototypes: E-Side Mirror and validation on large-scale datasets for the use case of domain adaptation for driving by night and other purpose-designed benchmarks.

Potentials for further Research

The newly presented methods are conclusively implemented, their domain adaptation capabilities and functionalities have been proven, and their results have been evaluated on novel unsupervised domain adaptation-specific metrics. At the same time, catastrophic forgetting is not yet entirely prevented by each of the presented methods. Unsupervised domain adaptation approaches still need to reach source domain performance in adapted domains. Across the deep learning board, supervised problem statements need to be rethought and reinterpreted from an unsupervised learning perspective, meaning the transition from labeled datasets to large-scale unlabeled and continuously expanding datasets for training neural networks. These problem statements potentially range from domain adaptations by a change in sensor position, introducing an advanced sensor modality, or pure deployment shift.

Potentials for methodological developments are within self-supervised approaches and shared feature space approaches. It has to be investigated which further supervision cues and knowledge priors, such as reference sensors, or passive human supervision, are available and suitable for semi-unsupervised and unsupervised domain adaptation. Work has also been initiated that deals with real-time inference and training capabilities on edge devices. Extending large-scale unlabeled datasets can also open up other application areas for present and future methods. Building upon methods to visualize and quantify domain shifts and epistemic uncertainties under domain shift might contribute to the safety envelope for autonomous perception systems, supporting the safety argument and highlighting the need for domain adaptation. The underlying mechanisms and effects of catastrophic forgetting need further be researched in the context of continuous learning and deployment.

These open problems are extensive, yet from a factual perspective, they merely reflect the contributions and progress within unsupervised domain adaptation from the other end of the scale. Thus, this work encourages and closes with an imperative for further development and research within unsupervised and continuous domain adaptation and domain representations.

A Appendix

1. Implementation and Repositories
2. Deep Learning Fundamentals
3. Related Work Tabular Overview
4. Lists of Related Publications, Patents, and Supervised Work

A.1 Implementation and Repositories

The objective of implementing the proposed, novel unsupervised domain adaptation approaches (see Ch. 2) is to provide, for the first time, an executable version and foundation for further development, research, and deployment of the very same. In the following, each approach's implementation to domain adaptation is assembled and organized in the form of a repository.

Hardware and Software Specifications (Sec. A.1) include the requirements, software packages, and their versions in order to set up the environment on your machine.

Neural Network Training reports the settings of the hyperparameters and data necessary for the training process. The associated neural network architectures depict the interplay of general architectural elements such as layers and units and the associated functions.

Codebase Overview gives an overview of the source code that comes with the repository as well as its overarching purpose within the repository.

Hardware Specifications

Graphical Processing Unit

Deep Learning research is driven by and highly correlated to the development of computing hardware. A High Performance Computer (HPC) is employed throughout the research to carry out the computationally expensive tasks, such as data preprocessing, training, and evaluation of neural networks and novel methods. As of this writing, the HPC comprises multiple NVIDIA® TESLA® P100 Graphical Processing Units (GPU) based on the NVIDIA® Pascal™ GPU architecture (for detailed specifications, see Tab. A.1). The GPUs are accessed by use of MobaXTerm [135] and driven by Cuda compilation tools (8.0 V8.0.61) and CuDNN (5.1.10).

GPU Specifications	
GPU Architecture	NVIDIA® Pascal®
NVIDIA® CUDA® Cores	3584
Double-Precision Performance	5.3 TeraFLOPS
Single-Precision Performance	10.6 TeraFLOPS
Half-Precision Performance	21.2 TeraFLOPS
GPU Memory	16 GB CoWoS HBM2
Memory Bandwidth	732 GB/s
Interconnect	NVIDIA® NVLink™
Max Power Consumption	300 W
ECC	Native support with no capacity or performance overhead
Thermal Solution	Passive
Form Factor	SXM2
Compute APIs	NVIDIA® CUDA® , DirectCompute, OpenCL™, OpenACC

Table A.1: Specifications of the NVIDIA® TESLA® P100 GPU accelerator according to the NVIDIA® datasheet [136].

Image Translation Repository

The general concept of the shared latent feature space approach is based on an unsupervised two-stream mechanic to learn a shared representation (see Sec. 3.2). For cross-domain neural network deployment, the common representation enables an unsupervised domain adaptation capability which enables direct deployment of an object detector on translated night data.

Software Specifications

The primary programming language for this repository is Python in version (2.7.13), with additionally utilized packages (see Tab. A.2). For visualization, the statistical programming language R [137] was deployed.

Python Packages	**Version**
TensorFlow [34]	(1.2+)
Torch	(0.4.1)
Torchvision	(0.2.1)
TensorboardX	(1.4)
Numpy	(1.17.1)

Table A.2: Version specifications of the utilized python software packages of the image translation repository.

Neural Network Training

The unsupervised image-to-image translation [98] is based on the proposed unsupervised domain adaptation module (see Sec. 3.2). Find the training hyperparameter configuration in Tab. A.3.

The approach does not require paired or labeled night domain samples. For training, the BDD100K dataset is deployed. The training utilizes 70000 samples from the training set and 10000 samples from the validation set. The original dimensions of 1280×720 px are kept for the labeled keyframe images (see detailed information in Sec. 5.3.1). The samples can be separated into 39986 annotated night images and 52511 annotated day images.

Training Hyperparameters	Value
Weight initialization	Kaiming He
Loss GAN	Binary Cross-entropy Loss
Loss VAE	Cycle Consistency Loss
Optimizer	Adam
Epochs	15
Learning rate ϵ	0.0001
First-momentum term ρ_1	0.5
First-momentum term ρ_2	0.999
Learning rate decay $d,\ \tau$	$0.5,\ 100000$
Batch size	1

Table A.3: **UNIT Neural network training hyperparameters** of the image translation repository. The hyperparameters are explained in the fundamentals part of this thesis on neural network training (see Sec. 1.2.2). Hyperparameters, when not stated explicitly, follow the default setting of the deployed deep learning framework (referring to software specifications A.2).

Training Hyperparameters	Value
Weight initialization	Pretrained model
Regularization	L^2 parameter norm penalty
Weight updates	10k (Early stopping A.2)
Learning rate ϵ	0.00003
First-momentum term ρ_1	0.9
Batch size	1

Table A.4: **RFCN Neural network training hyperparameters** of the image translation repository. The hyperparameters are explained in the fundamentals part of this thesis on neural network training (see Sec. 1.2.2). Hyperparameters, when not stated explicitly, follow the default settings of the hyperparameter configuration proposed initially for training the RFCN model [41].

The 7285 dusk and dawn images are not considered during training. To increase the coverage of high-frequent semantics in the sample, the discriminator is evaluated on randomly cropped areas (patches) instead of the complete generated samples [138]. Besides, the discriminator takes three generated inputs for a single sample, two of them down-sampled with factors two and four, improving multi-scale domain adaptation capabilities [37].

The source task is object detection (see Fig. 3.5), and is cross-domain deployed, subsequent to the unsupervised image-to-image translation. For a full list of deployed object detectors, see Tab. 1.1 and Sec. 1.3. In the cases that the object detector has been retrained, see the hyperparameter configuration for retraining in Tab. A.4.

For validation and testing of the cross-domain deployment on the source task, the BDD100K bounding box coordinates of persons (129262 object instances) and cars (1021857 object instances) have been used, and the data is distinguished concerning the night and day domain. Whenever the RFCN object detector was retrained, BDD100K bounding box coordinates were used.

File Name	**Description**
Neural Network Training:	
train.py	UNIT training module
Evaluation and Analysis:	
test.py	Inferencing UNIT model on single frame
test_batch.py	Inferencing UNIT model on folder
ObjectDetector.py	Object detector module
hungarian.py	KPI calculation module
Models:	
unit_n2d.hdf5	Trained UNIT Night-to-day cross-domain adaptation model

Table A.5: Overview codebase of the image translation repository.

The original validation set is used as the test set to conclusively evaluate the domain adaptation performance and the object detectors' performances.

Codebase Overview

The shared latent feature space repository[1] is publicly available and consists of the complete code base (see Tab. A.5), with detailed information on the hyperparameter settings, neural network architecture, deployed data, data pipeline, validation scripts, results, evaluation, and documentation [1].

[1] Shared latent feature space repository: https://osf.io/snmwt/

Continuous Self-Supervision Repository

The concept of the continuous self-supervision approach is assuming a continuous domain shift and exploits this knowledge by a self-supervised pseudo label approach to domain adaptation (see Sec. 3.3).

Software Specifications

The primary programming language used throughout this work is Python in version (3.6.9), with additionally utilized packages (see Tab. A.6).

Python Packages	**Version**
TensorFlow [34]	(1.14.0)
Keras	(2.2.5)
Numpy	(1.17.1)
sklearn	(0.0)
Pillow	(6.1.0)

Table A.6: Version specifications of the utilized python software packages.

Training Hyperparameters	**Value**
Weight initialization	Glorot
Loss	Binary Cross-entropy Loss
Optimizer	Adam
Epochs	50
Learning rate ϵ	0.01
Batch size	32

Table A.7: **Classification neural network training hyperparameters** of the self-supervised continuous domain adaptation repository. The hyperparameters are explained in the fundamentals part of this thesis on neural network training (see Sec. 1.2.2). Hyperparameters, when not stated explicitly, follow the default setting of the deployed deep learning framework (referring to software specifications A.6).

Neural Network Training

The continuous self-supervised domain adaptation is based on a fully connected neural network θ_α for classification.Find the training hyperparameter configuration in Tab. A.7.

The neural network is extended into the target domain by self-supervised iterative fine-tuning (see Fig. 3.3.2). The configuration for iterative fine-tuning introduces further method-specific hyperparameters, which are discussed and analyzed (see Sec. 5.6.2 and Tab. 5.5).

Self-supervised continuous domain adaptation requires a dataset with a controllable, continuous domain shift. For the research, therefore, the rotatedMNIST dataset (see Sec. 4.2.2) is introduced. The domain shift within the data emerges based on context knowledge, such as the spatial transition from rotating the original MNIST samples by a specific angle step size. The domain $\mathscr{D}_{90}$ is termed the target domain and marks the end of a set of intermediate domains (see Eq. A.1).

$$\mathscr{D}_\alpha \text{ with } \alpha(s) = \Delta\alpha\ s, s \in \mathbb{N}, \alpha \leq 90. \tag{A.1}$$

The maximal angle α_{max} is bound above at 90° to obviate rotation variant ground truth, such as would be the case for digits 6 and 9 and rotations of $\alpha = 180°$.

The Cuepervision approach (see Sec. 3.3.3) is trained with the same training hyperparameters and data as outlined by the continuous self-supervised domain adaptation approach (see Tab. A.7). The novel contribution is to be found in the architectural subtlety of introducing a feature extractor linked to a domain classification layer and the main classification layer of the source task.

Codebase Overview

The continuous self-supervision repository[2] is publicly available and consists of the complete code base (see Tab. A.8, with detailed information on the hyperparameter settings, neural network architecture, deployed data, data pipeline, validation scripts, results, evaluation, and documentation [2].

[2] Continuous self-supervision repository: https://osf.io/qgj5d/

File Name	**Description**
Neural Network Training:	
mnist_train.py	Training implementation
mnist_data_handler.py	Data loader module
mnist_adaptation.py	Main neural network module
Evaluation and Analysis:	
mnist_visualize.py	Visualization module
analysis.py	Continuous adaptation analysis module
Models:	
sparse_model_1.hdf5	One epoch trained classification model
sparse_model_50.hdf5	50 epochs trained classification model

Table A.8: Overview codebase of the continuous self-supervision repository.

HeartSeg Repository

At the core of the HeartSeg repository is providing a controllable, adjustable biomedical image benchmark for spatial and temporal, continuous, and discrete domain adaptation (see Sec. 4).

Codebase Overview

The HeartSeg repository[3] is publicly available and consists of the entire dataset with samples and labels, the complete code base (see Tab. A.9, with detailed information on the hyperparameter settings, neural network architecture, deployed data, data pipeline, validation scripts, results, evaluation, and documentation [3].

File Name	**Description**
Neural Network Training:	
unet_model.py	U-net implementation
data.py	Training data pipeline
Evaluation and Analysis:	
data_loader.py	Load data from stream
data_writer.py	Write logging and prediction data
inference.py	Inferencing segmentation model
segmentation_analysis.py	Analyzing segmentation frame-wise
timeseries_analysis.py	Analyzing segmentation over time
ventricular_dimensions.py	Determine ventricular dimensions
Models:	
unet_heart.hdf5	Trained U-net segmentation model

Table A.9: Overview codebase of the HeartSeg repository.

[3] HeartSeg repository: https://osf.io/snb6p/

Software Specifications

The primary programming language used throughout this work is Python in version (3.6.7), with additionally utilized packages (see Tab. A.10).

Python Packages	**Version**
TensorFlow [34]	(1.12.0)
Keras	(2.2.4)
Numpy	(1.15.4)
scipy	(1.2.0)
openCV 2	(3.4.4.19)

Table A.10: Version specifications of the utilized python software packages.

Neural Network Training

Training Hyperparameters	**Value**
Weight initialization	Glorot
Loss	Sørensen-Dice Loss
Optimizer	Adam
Learning rate ϵ	0.00001
First-momentum term ρ_1	0.9
First-momentum term ρ_2	0.999
Epochs	60
Batch size	8
Dropout	0.5
Augmentation	flip, zoom, rotate, shift

Table A.11: **U-net neural network training hyperparameters**. Hyperparameters are explained in the fundamentals part of this thesis on neural network training (see Sec. 1.2.2). Hyperparameters, when not stated explicitly, follow the default setting of the deployed deep learning framework (specified in A.10).

A.2 Deep Learning Fundamentals

Backpropagation and Stochastic Gradient Descent

A neural network receives an input **x** and generates an output **y**; this is called inference or forward propagation. During training, the output is compared to the ground truth data by a cost function (see Chapter A.2). In order to increase performance, the cost information is then backpropagated through the network by means of calculating a gradient and updating the weights accordingly [139]. The gradient $g(\theta) = \nabla_\theta J(\theta)$ represents the derivative of a multi-dimensional function, in this case, the cost function J regarding the parameter set or, in other words, the weights $\mathbf{W}$ and $\mathbf{b}$. The gradient is defined for any point p on an n-dimensional differential function J. The norm of the gradient $|g|$ expresses the magnitude of the ascent at point p [26].

The derivatives for each of the n parameters θ_i whereas $i \in \{1, 2, \dots, n_i\}$ in the network, are computed by propagating the gradient, and deploying the chain rule of calculus (see Equations A.2 and A.3).

$$\begin{aligned} \nabla_x y &= \left(\frac{\partial h}{\partial x}\right)^\top \nabla_h y, \\ \frac{\partial y}{\partial x} &= \frac{\partial y}{\partial h}\frac{\partial h}{\partial x}. \end{aligned} \tag{A.2}$$

More generally, each node $u^{(i)}$ with parameter $\theta^{(i)}$ is linked to an operation for forward propagation, namely an activation function (see Chapter A.2) $f(\mathbb{A}^{(i)})$ while $\mathbb{A}^{(i)}$ is the set of all predecessor nodes, also parents $\mathscr{P}$, $u^{(j)}$ of node $u^{(i)}$ on the path to the root, for $j < i$ with $j \in \mathscr{P}(u^{(i)})$. Starting from the output $u^{(n)}$, the gradient is propagated, and consequently the gradient for each layer and unit emanates.

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i \in \{k; j \in \{\mathscr{P}(u^{(k)})\}\}} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}. \tag{A.3}$$

The gradient on each unit is subsequently used to update the parameter set θ (see Equation A.4). This process is repeated by means of iterative gradient descent in order to minimize the cost $J(\theta)$, which is determined during forward propagation and calculated by a loss function $L(\hat{\mathbf{y}}, \mathbf{y})$ [18].

$$\begin{aligned} \theta' &= \theta - \epsilon\, g(\theta), \\ \theta' &= \theta - \epsilon\, \nabla_\theta J(\theta). \end{aligned} \tag{A.4}$$

The learning rate ϵ represents the step size for updating the model's parameters during training (see Equation A.4). Intuitively larger step sizes correspond with faster descent, while risking to overstep the optimal loss. In turn, a low step size will have linear improvements taking more time to descend into an optimum [26]. In order to trade off those characteristics, one strategy is to decay the learning rate by d during the training process every τ weight updates (see Equation A.5) [34, 140].

$$\epsilon_n = \epsilon_0\, d^{\lfloor \frac{1+n}{\tau} \rfloor}. \tag{A.5}$$

Mutually, the learning rate is stated to be among the most important hyperparameters for neural network training [18, 26, 140].

Activation Function

A unit u of a neural network computes an affine transformation $\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$ on a given input $\mathbf{x}$, which is coined activation state $\mathbf{z}$. Subsequently, a non-linear activation is put into use; namely, the activation function $f(\mathbf{z};\theta)$, resulting in an output $\mathbf{y}$ [26]. The activation function theoretically needs to be differentiable at each point $\mathbf{x}$ on f, due to the optimization strategy of gradient descent (see Chapter A.2). In practice, activation functions with only a small amount of non-differentiable points are still valid, as it is tolerable for the minima of the cost function to be non-differentiable, as it is not expected to actually reach a minimum during gradient descent. Moreover, the superimposed numerical errors legitimize utilizing the derivative of either side of the point [18]. Different types of activation functions used in a neural network come with prevalent characteristics. Some activation functions are sensitive in an interval $[-0.5; 0.5]$, which motivates data preprocessing such as zero-centering and normalizing the input values [140].

Rectified linear unit

The linear unit, or rectified linear unit (ReLU), uses the activation function of Equation A.6. The ReLU has constant first-order derivatives and zero second-order derivatives, which make optimization and gradient descent more effective since the gradient direction is consistent and the gradient does not vanish [46]. ReLUs have the disadvantage of vanishing weights when updated too far into the negative domain, and this effect is reduced by leaky ReLU units [141] that introduce a slight negative slope in the negative domain.

$$\begin{aligned} f(z) &= \max\{0, z\}, \\ f'(z) &= \begin{cases} 0, & \text{if } z < 0, \\ 1, & \text{if } z > 0. \end{cases} \end{aligned} \tag{A.6}$$

Sigmoid

The sigmoid unit $\sigma(z)$ (see Equation A.7) saturates over most of its range of values: To one, for positive values of z, or to zero, for negative values of z. The activation function is sensitive for z around zero.

$$\begin{aligned} f(z) &= \frac{1}{1 + \mathrm{e}^{-x}}, \\ f'(z) &= (1 - f(z)) f(z). \end{aligned} \tag{A.7}$$

Objective Function

Optimization is based on the objective function J, with $\theta^* = \arg\min\ J(\theta)$. During the optimization process, J is minimized by means of determining θ. In machine learning, J is mutually named as cost function, loss function, or error function. Obviously, the definition of J is of great influence for the training process of neural networks, as it is for conventional machine learning methods, in a way that it determines the gradient (see Chapter A.2) by comparing the models output $\mathbf{y} = f(\mathbf{x};\theta)$ with the empirical distribution of the ground truth data $\tilde{\mathbf{y}}$.
The neural network model $f(\mathbf{x};\theta)$ can be interpreted in terms of a probability density $p(\mathbf{y} \mid \mathbf{x};\theta)$, with random variables $\mathbf{x}$ and $\mathbf{y}$. This enables a cost function to be determined by calculating the negative log-likelihood or cross-entropy between the model's distribution and the probability density of the ground truth data $\tilde{p}$, whereas $\mathbb{E}$ is the expectation value (see Equation A.8). As such, the training process attempts to match the model's distribution to the empirical one of the ground truth.

$$J(\theta) = -\mathbb{E}_{\mathbf{x},\tilde{\mathbf{y}}\sim\tilde{p}}\ \log\ p(\tilde{\mathbf{y}}|\mathbf{x},\theta). \tag{A.8}$$

By assuming the model's distribution to be Gaussian, $\mathcal{N}$ (see Equation A.9) and a parametrization of $\sigma = 1$, it is evident that the mean squared error is equivalent to a maximum likelihood scheme [18].

$$\begin{aligned}
p(\tilde{\mathbf{y}} \mid \mathbf{x},\theta) &= \mathcal{N}(\tilde{\mathbf{y}}; f(\mathbf{x};\theta),\sigma^2),\\
p(\tilde{\mathbf{y}} \mid \mathbf{x},\theta) &= -m\log\sigma - \frac{m}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{m}||\tilde{\mathbf{y}}_i - f(\mathbf{x}_i;\theta)||^2,\\
p(\tilde{\mathbf{y}} \mid \mathbf{x},\theta) &= \mathcal{N}(\tilde{\mathbf{y}}; f(\mathbf{x};\theta),\mathbf{I}),\\
p(\tilde{\mathbf{y}} \mid \mathbf{x},\theta) &= -\frac{m}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{m}||\tilde{\mathbf{y}}_i - f(\mathbf{x}_i,\theta)||^2,\\
J(\theta) &= \frac{1}{2}\ \mathbb{E}_{\mathbf{x},\tilde{\mathbf{y}}\sim\hat{p}}\ ||\tilde{\mathbf{y}} - f(\mathbf{x};\theta)||^2 + \text{const.}
\end{aligned} \tag{A.9}$$

The mean squared error, thus, follows the properties of maximum likelihood: Assuming an infinite amount of ground truth samples $\tilde{\mathbf{y}}_m$, with $m \to \infty$, which are considered to be generated by $f(\mathbf{x};\theta_0)$, the estimated parameter set θ approaches the true values θ_0 (consistency) [142].

Nonconvex Gradient-Based Optimization

In general, optimization revolves around determining θ^* by minimizing an objective function's error $J(\theta)$. When deploying gradient-based optimization on deep neural networks, additional challenges arise due to the general nonconvex case: Infinite amount of local minima, nonidentifiability, batch-based inexact gradients, vanishing and exploding gradients, and the poor correspondence between local and global structures [18]. Restating the problem statement of optimization from finding the true and exact values θ_0 for the underlying model to finding a sufficiently optimal set of values θ^* (see Equation A.10).

$$\theta^* \neq \theta_0. \tag{A.10}$$

In the following, optimization algorithms are introduced along with strategies and tactics, overcoming optimization problems for the nonconvex case of neural networks.

Stochastic Gradient Descent

Backpropagation and gradient descent (see Sec. A.2) have already been introduced. An extension of this method is the stochastic gradient descent (SGD) (see Alg. 2): Following the gradient of a randomly selected set of m training samples, batches. The central idea is to observe an unbiased estimate of the gradient by taking the average gradient of a batch only (stochastic) instead of the whole training set (deterministic). Equation A.11) shows that the standard error of the mean decreases only by $\sqrt{m}$ when increasing m the number of samples in a batch.

$$\frac{\sigma}{\sqrt{m}}. \tag{A.11}$$

At the same time, a reduction in m reduces and sets an upper bound to the computational effort for each update step, which is a necessity when dealing with large or even growing datasets, as is the case for machine learning and data-driven approaches in general (see Fig. 1.1a). Further assuming *i.i.d*, it is to be assumed that samples to a degree show similarities and thus have comparable leverage on the gradient [18, 143, 144].

Momentum Methods

Stochastic gradient descent (see A.2) has drawbacks when dealing with small gradients or variance in subsequent gradients since the method relies on the

Algorithm 2 Stochastic gradient descent - $O(n)$

```
1: procedure WEIGHT UPDATE ($\theta$ initial weights, $\epsilon_i$ learning rate in iteration $k$, $m$ batch size)
2:     $k \leftarrow 1$
3:     while stopping criterion not met do
4:         Estimate average batch gradient: $\hat{\mathbf{g}} = \frac{1}{m}\nabla_\theta \sum_i^m L(f(\mathbf{x}_i;\theta);\mathbf{y}_i)$
5:         Update the weights: $\theta' = \theta - \epsilon_k\, \hat{\mathbf{g}}(\theta)$
6:         $k \leftarrow k+1$
```

gradient at each iteration only (see Alg. 2). The momentum method [145] averages gradients of past iterations, yielding velocity $\mathbf{v}$ (see Equation A.12) and considers them for the current weight update. $\alpha \in [0,1)$ specifies the magnitude of the exponential decay of the averaged gradients. Assuming unit weight, $\mathbf{v}$ can be seen analogously to the Newtonian momentum, hence the naming. The equations for stochastic gradient descent with momentum are thus Eq. A.12.

$$\begin{aligned} \mathbf{v} &= \alpha\mathbf{v} - \epsilon\mathbf{g}, \\ \theta' &= \theta + \mathbf{v}. \end{aligned} \tag{A.12}$$

Besides the standard momentum method, other momentum methods differ on whether the gradient is evaluated before or after applying the velocity update [146, 147].

Adaptive Methods

The notion of adaptive methods [148, 149, 150] is that the learning rate should be set locally, being sensitive for each parameter and iteration, instead of globally. The most commonly applied of these methods is the Adam (Adaptive momentum) optimizer [150]: An algorithm for first-order gradient-based $\mathbf{s}$ optimization of stochastic objective functions, based on adaptive estimates of second-order moments $\mathbf{r}$ (see Equation A.13). ρ_1 and ρ_2 are the exponential decay rates for the accumulated moment estimates, and δ is a small constant utilized for numerical stability. t marks the current iteration count. $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$ are the bias-corrected terms of the moment estimates during initialization.

$$
\begin{aligned}
\mathbf{s} &= \rho_1 \mathbf{s} + (1-\rho_1)\mathbf{g}, \\
\mathbf{r} &= \rho_2 \mathbf{r} + (1-\rho_2)\mathbf{g} \odot \mathbf{g}, \\
\hat{\mathbf{s}} &= \frac{\mathbf{s}}{1-\rho_1^t}, \\
\hat{\mathbf{r}} &= \frac{\mathbf{r}}{1-\rho_2^t}, \\
\theta' &= \theta - \epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}.
\end{aligned}
\tag{A.13}
$$

Adam implicitly determines the magnitude of the learning rate for each parameter adaptively. ϵ is further an upper bound for the learning rate only.

Weight Initialization

Optimization of neural networks is iterative and heavily depends on the initial model weights or model parameters θ: Deciding whether and how fast the algorithm converges. Concerning generalization, weight initialization aims to break symmetries [20] between different units connected to the same inputs by assigning them different initial values and thus making them depict different functions and capture different patterns. Beyond, there is little scientifically confirmed knowledge on the influence and mechanics of weight initialization, resulting in the predominance of well-performing heuristics, such as Kaiming [151], Xavier [152], and others [153, 154]. In practice, randomly drawing weights from a uniform distribution is a computationally efficient way to achieve this (see Equation A.14), with m being the number of inputs to guarantee consistent activation variance over all layers [18]. To further ensure all layers to have the same gradient variance, n the number of outputs is considered, resulting in a normalized weight initialization (see Equation A.15) [152].

$$w_{jk} \sim U[-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}], \tag{A.14}$$

$$w_{jk} \sim U[-\sqrt{\frac{6}{n+m}}, \sqrt{\frac{6}{n+m}}]. \tag{A.15}$$

Other strategies strive to initialize the parameters by pretraining the model on a similar, often simpler task. Other strategies are immediately utilizing an already trained model, with the same architecture, as an initial point. In many cases, these approaches profit from faster convergence due to the directed initialization and improved generalization due to the parameters containing information about the preceding reference model or task. These approaches touch the field of transfer learning [24].

Batch Normalization

When updating the weights of a neural network, it is done under the assumption that the other weights stay the same, yet in practice, all weights are updated and have a strong influence on each other. As a consequence, the optimization process suffers from a covariate shift that is induced by higher-order effects in deep networks. Batch normalization [155] mitigates this effect, by normalizing $\mathbf{H}_{i,j}$ a minibatch of activations element-wise by the mean of each unit $\boldsymbol{\mu}_j$ and the standard deviation of each unit $\boldsymbol{\sigma}_j$ (see Equation A.16).

$$\mathbf{H}'_{i,j} = \frac{\mathbf{H}_{i,j} - \boldsymbol{\mu}_j}{\boldsymbol{\sigma}_j}. \tag{A.16}$$

In that way, the input distribution will be propagated throughout all layers of the network. During inference, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are set to running averages determined during training.

Skip Connection

In neural networks, and mainly in CNNs [42], skip connections, also referred to as shortcut connections, are used to bypass features with lower abstraction into deeper layers. This has the effect of regaining spatial information which otherwise would have been lost in layers with features of higher abstraction. The introduced skip connections $\mathbf{x}$ execute an identity mapping, plainly element-wise adding the output of the initial layer $f(\mathbf{x})$ to the output of the skipped layers $\mathbf{y}$, assuming that both outputs have the same dimensions (see Equation A.17).

$$\mathbf{y} = f(\mathbf{x}, \{\mathbf{W}\}) + \mathbf{x}. \tag{A.17}$$

The gradient flow exploration shows that the introduction of skip connections entails a favorable precondition of the weight matrices.

Parameter Norm Penalties

Adding a parameter norm penalty $\Omega(\theta)$ to the objective function $J(\theta;\mathbf{X},\mathbf{y})$, is a soft constraint on the capacity of the model. $\alpha \in [0, \inf)$ scales the share in the loss during optimization (see Equation A.18).

$$\tilde{J}(\theta;\mathbf{X},\mathbf{y}) = J(\theta;\mathbf{X},\mathbf{y}) + \alpha\Omega(\theta). \tag{A.18}$$

As biases $\mathbf{b}$ induce less variance, only weights $\mathbf{w}$ are penalized. The most common parameter norm penalty is the L^2 norm [156] (see Equations A.19).

$$\begin{aligned} \Omega(\theta) &= \frac{1}{2}||\mathbf{w}||_2^2, \\ \mathbf{w} &= (1-\alpha\epsilon)\mathbf{w} - \epsilon\nabla_{\mathbf{w}}J(\mathbf{w};\mathbf{X},\mathbf{y}). \end{aligned} \tag{A.19}$$

Parameter norm penalties are also known as weight decay, shifting the parameters closer to the origin of the parameter space while contracting dimensions of $\mathbf{w}$, to which the objective function is insensitive. The dimensions can be interpreted as eigenvectors, whereas the sensitivity is mirrored by the eigenvalues of the Hessian $\mathbf{H}$.

Parameter Sharing

Prior knowledge of parameter dependencies is explicitly entered into the network architecture by parameter sharing through force setting parameters $\mathbf{w}^A$ to equal $\mathbf{w}^B$. Prior knowledge of parameter dependencies is, for example, translation invariance in images (such as in CNNs), similar inputs recurring over time (such as in RNNs), or the mapping to a shared feature space (such as Encoder-Decoder architecture). As a consequence of parameter sharing, the memory requirements are reduced, and the model's capacity is decreased, which leads to a regularization effect (see Section 1.2.3).

Minibatch Method

In Section A.2, the SGD algorithm was introduced, motivating the central idea of minibatch methods, observing an unbiased estimate of the gradient from a batch of randomly drawn samples. From this point, it is apparent that batch-design is a crucial factor for optimization and regularization. The batch design is twofold. Batch size is a trade-off between a more unbiased estimate of the gradient (larger batch size) against regularization effects due to the noisiness of the biased estimate of the gradient (smaller batch size)

[157]. While small batch sizes require small learning rates due to the noise-induced instability, the iterations to reach convergence might increase heavily. Large batch sizes are prone to exceed hardware limitations (RAM) during computation. As GPUs follow the single instruction multiple data (SIMD) paradigm [158], data parallelism improves runtime. Parallelism capabilities are achieved by matching the physical processors of the GPU by sampling power-of-two batch sizes. Sample selection is made by random sampling to get an unbiased estimate of the gradient over the whole dataset, assuming a pattern in the dataset, especially in its consecutive frames. As dataset size (see Fig. 1.1a) increases, generalization is improved by decreasing the number of iterations in which an individual sample is utilized for training.

Augmentation

Increasing the number of samples n for training reduces the generalization gap and improves the neural network's performance (for more on this, see 1.2.3). In practice, making additional data available can demand a great effort for data collection, data preparation, and ground truth annotations. Especially in object recognition, data augmentation presents a useful solution to generate additional data by transforming a sample $\mathbf{x}$ in the training set and keeping the affiliated ground truth label $\mathbf{y}$ [159].

Dropout

Dropout [160] can be understood as training an ensemble of second networks within a single neural network. A second network within the ensemble is designed by removing input and hidden units, meaning zero multiplication of the units' output, from the encompassing network during training. A different second network is chosen during each training step by randomly removing or dropping units defined by the binary mask vector μ and the sample probability $p(\mu)$. The number of dropped units, also the dropout rate is commonly between 0 and 0.5. The second networks share parameters (see Section A.2) of the encompassing network θ, implicitly training multiple second networks at once and consequently keeping down memory requirements and computational effort.

$$\sum_{\mu} p(\mu) p(y|\mathbf{x};\theta_{\mu}) = p(y|\mathbf{x};\theta). \tag{A.20}$$

The parameter sharing causes each hidden unit to function in different ensembles, putting a regularization pressure on every hidden unit [161], improving generalization. The output of all ensembles is approximately the output of the encompassing network (see Equation A.20), making the approach computationally feasible during inference.

Early Stopping

When training a model with a large capacity, the training error steadily decreases over the training epochs. At some point, the model overfits the training samples, leading to an increased validation error. Early stopping determines the point of least validation error in hindsight, returning the model parameters at that training step θ^*. This is to be understood as a regularization method that prevents the model from exploiting its full capacity by a hard parameter constraint [26, 28, 162].

Hyperparameter Search

ANN design comes with a set of hyperparameters λ defining the architecture and training process. Hence it is necessary to define a process to approximate the optimal set of hyperparameters λ^* to reach the optimal parameter set θ^* that minimizes the overall validation loss $\mathscr{L}$ of the ANN. The optimal hyperparameter set is defined as Eq. A.21.

$$\lambda^* = \operatorname*{argmin}_{\lambda} \mathscr{L}(\theta_\lambda). \tag{A.21}$$

Due to the computational expense of training neural networks, hyperparameters are often handcrafted by human experts. When it comes to shallow neural networks, the hyperparameter search can profit from and be performed in a more methodized manner. Bergstra et al. [134] showed that random search is more efficient than grid search when it comes to dealing with high dimensional spaces: Sampling in a grid of points gives a uniform coverage in the original parameter space, though an inefficient coverage of the subspaces. On the other hand, random sampling gives uniform coverage in the parameter subspaces. Each hyperparameter, within the set λ, is independently and randomly drawn from the uniform distribution $U[\text{lower bound}, \text{upper bound}]$, conforming to the hyperparameter's interval and scale type (such as Boolean, integer or float). Random search is insusceptible to overfitting due to the nature of randomness during hyperparameter configuration.

Sample and Feature Visualization with tSNE

The t-distributed stochastic neighbor embedding (tSNE) is a non-linear dimensionality reduction algorithm. tSNE is used for visualizing high-dimensional data in low-dimensional spaces by mapping. tSNE casts the pattern of high-dimensional data based on the pairwise similarities of data points with multiple features by transferring the similarity measurement into a low-dimensional mapping.

Algorithm 3 t-distributed stochastic neighbor embedding

```
 1: procedure TSNE($\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$ data set with samples from different domains, $Perp$ cost function parameter perplexity, $T$ number of iterations, $\epsilon$ learning rate, $\alpha(t)$ momentum:)
 2:     $p_{j|i}$, calculate pairwise similarities through $Perp$
 3:     $p_{i,j} \leftarrow \frac{p_{j|i}+p_{i,j}}{2n}$
 4:     $\mathscr{Y}^{(0)} = y_1, y_2, \ldots, y_n$ from $\mathscr{N}\ (0, 10^{-4}\boldsymbol{I})$, sample initial mapping
 5:     for $t = 1$ to $T$ do
 6:         $q_{j|i}$, calculate pairwise low-dimensional similarities
 7:         $\frac{\delta C(p_{j|i}, q_{j|i}, y_i, y_{j\in\{1;n\}})}{\delta y_i}$
 8:     $q_{j|i}$, calculate conditional probability for low-dimensional points $y_j$ and $y_i$
 9:     $\overline{\sum}_t = G_t \sum_{t-1} G_t^T + R_t$

    State correction:
10:     $K_t = \overline{\sum}_t H_t^T (H_t \overline{\sum}_t H_t^T + Q_t)^{-1}$
11:     $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
12:     $\sum_t = (I - K_t H_t)\overline{\sum}_t$

13:     return $\mu_t$, $\sum_t$
```

A.3 Related Work Overview

The related work on unsupervised homogeneous domain adaptation, self-supervised domain adaptation, and catastrophic forgetting is organized with respect to the type and accompanied by a short description (details and placing in context see Ch. 1).

Unsupervised Homogeneous Domain Adaptation

Reference	**Approach**	**Type**
Coupled Generative GANs [68] *2016*	Parameter-sharing constraint enforcing a multi-domain joint distribution	Generative Model
Pixel-Level Domain Transfer [65] *2016*	GAN architecture for pixel-level image generation in the target domain	Generative Model
Unsupervised pixel-level domain adaptation with GANs [163] *2017*	GAN architecture for unsupervised transformation from source to target image	Generative Model
Image-to-Image translation with conditional adversarial networks [66] *2017*	Input-transformation tuples to learn on transformation loss	Generative Model
Unsupervised domain adaptation by backprop [76] *2015*	Domain-confusion loss to learn a shared feature space	Non Generative Model
Adversarial Discriminative Domain Adaptation [77] *2017*	Untied parameter sharing to learn a shared feature space	Non Generative Model
DLID: Deep Learning for Domain Adaptation by Interpolating domains [74] *2013*	Learning an interpolating path of features to transition from source to target domain	Encoder Decoder Reconst.
Domain separation networks [70] *2016*	Training a shared and separated encoders for representation learning	Encoder Decoder Reconst.

Domain generalization for object recognition with multi-task autoencoders [71] *2015*	Learning a domain-invariant representation by multiple reconstruction networks	Encoder Decoder Reconst.
Deep reconstruction-classification networks for unsupervised domain adaptation [75] *2016*	Learning a representation for reconstruction and classification	Encoder Decoder Reconst.
Unpaired image-to-image translation using cycle consistent adversarial networks [78] *2017*	Adversarial loss for unpaired image-to-image translation	Adversarial Reconst.
Dualgan: Unsupervised dual learning for image-to-image translation [80] *2017*	Dual-GAN architecture to train a translator from two sets of unlabeled images	Adversarial Reconst.
Learning to discover cross-domain relations with GANs [79] *2017*	Reconstruction is done into both domains based on an adversarial loss	Adversarial Reconst.
Unsupervised image-to-image translation networks [98] *2017*	Combining reconstruction cycle consistency, and adversarial loss	Adversarial Reconst.
CyCADA: Cycle consistent adversarial domain adaptation [164] *2018*	Adapt representations at pixel and feature-level, enforces cycle consistency, and task loss	Adversarial Reconst.

Table A.12: Overview of state-of-the-art unsupervised domain adaptation, featuring references, year of publication, a short description, and a type categorization of the domain adaptation approach.

Catastrophic Forgetting

Reference	Approach	Type
Overcoming catastrophic forgetting in neural networks [97] *2017*	Decreasing plasticity of the model during retraining by protecting the most important parameters.	Training based
Learning without Forgetting [60] *2018*	Transferring to an additional task by shared feature extraction and joint training on old and new task.	Training based
Progressive Neural Forgetting [96] *2018*	Extending neural networks by additional layers and lateral connections to previous layers.	Architecture based

Table A.13: Overview state-of-the-art approaches to avoid catastrophic forgetting, featuring reference, year of publication, a short description, and a type categorization of the domain adaptation approach.

Self-Supervised Domain Adaptation

Reference	Approach	Type
Semi-Supervised Self-Training of Object Detection Models [86] *2005*	Self-supervised training of an object detector model by iteratively assigning labels by the model itself	Similarity Context based
Pseudo-Label: The Simple And Efficient Semi-Supervised Learning Method for DNNs [82] *2013*	Training on unlabeled data by assigning the class affiliation with the maximum predicted probability (entropy regularization)	Similarity Context based
Unsupervised Visual Representation Learning by Graph-based Consistent Constraints [83]	Graph-based pairing of unlabeled samples with cycle consistency. Pairs are used as supervision	Similarity Context based

2016	signal	
Deep Clustering for Unsupervised Learning of Visual Features [88] *2018*	Jointly learning of neural network parameters and cluster assignments as supervision signal	Similarity Context based
Boosting self-supervised learning via knowledge transfer [85] *2018*	Clustering-based assignment of pseudo-labels based on a prelearned feature space	Similarity Context based
Find your own way: Weakly -supervised segmentation of path proposals for urban autonomy [87] *2017*	Self-supervised generation of pseudo-labels based on temporal projection	Temporal Context based
Look, listen and learn [93] *2017*	Use correspondence of modalities as supervision signal to learn a representation	Cross Modal based
Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera [94] *2019*	Self-supervised generation of depth pseudo-labels for images from depth images from depth measurements	Cross Modal based

Table A.14: Overview of state-of-the-art self-supervised domain adaptation, featuring reference, year of publication, a short description, and a type categorization of the domain adaptation approach.

A.4 Lists

List of Related and Disclosed Patents

[a] DE102018206108A1, EP000003557487A1 - 23.10.2019
Mark Schutera - ZF Friedrichshafen AG. The invention relates to an evaluation device for generating validation data for an intelligent algorithm.

[b] DE102018206110A1, EP000003557490A1 - 23.10.2019
Mark Schutera - ZF Friedrichshafen AG. The invention relates to a method for training artificial neural networks based on generated spatio-temporal data.

[c] DE102018207977A1 - 28.11.2019
Mark Schutera, Tim Härle, Devi Alagarswamy - ZF Friedrichshafen AG. The invention relates to the field of driver assistance systems and, more particularly, to a method and apparatus for securing a vehicle occupant of a vehicle with a seat belt device.

[d] DE102018217943A1 - 23.04.2020
Mostafa Hussein, **Mark Schutera** - ZF Friedrichshafen AG. The present invention relates to an evaluation device for parameterizing a recurrent neural network for generating training data, a parameterization method for a recurrent neural network, and a method for generating training data.

[e] DE102018219760A1 - 20.05.2020
Mark Schutera, Prof. Dr. Stefan Elser - ZF Friedrichshafen AG. The present invention relates to a method for training a recurrent neural network, a method for providing driver assistance, and a collision prediction system for protection against and/or during collisions of a vehicle.

[f] DE102018219996A1 - 28.05.2020
Mark Schutera - ZF Friedrichshafen AG. The present invention relates to a training method for a neural network providing a neural network with a reduced runtime, in particular with results of at least constant quality.

[g] DE102018221313A1 - 10.06.2020
Mark Schutera, Mostafa Hussein - ZF Friedrichshafen AG. The invention relates to a method for detecting objects in night shots by means of a de-

tector that is trained to detect objects in day shots of an environment of an automatically operated vehicle.

[h] DE102019206991A1 - 19.11.2020
Tim Härle, **Mark Schutera**, Christian Herzog - ZF Friedrichshafen AG. The invention relates to a system for a passenger transport vehicle for recognizing unauthorized persons. The invention also relates to a passenger transport vehicle comprising such a system. The invention also relates to a method for recognizing unauthorized persons.

[i] DE102019211672A1 - 04.02.2021
Mark Schutera - ZF Friedrichshafen AG. The present invention relates to a training method for a neural network based on self-supervised and continuous learning.

[j] DE102019212408A1 - 26.02.2021
Devi Alagarswamy, **Mark Schutera**, Tim Härle, Martin Seyffert, Vincent Choquet - ZF Friedrichshafen AG. The invention relates to a method for determining the body weight and/or the seat position of a vehicle occupant of a motor vehicle, a control device for a motor vehicle, and a motor vehicle.

[k] DE102020200998A1 - 29.07.2021
Mark Schutera - ZF Friedrichshafen AG. The present invention relates to a method for processing sensor data by means of an artificial neural network and a file with a plurality of topologies for a large number of second artificial neural networks based on environment-related characteristics.

[l] DE102020201742A1 - 12.08.2021
Mark Schutera, Frank Hafner - ZF Friedrichshafen AG. Method for selecting training data related to the sensor environment with the following steps: recognition of an inadequately trained situation due to human interaction; identification of sensor data relating to the inadequately trained situation.

[m] DE102020201743A1 - 12.08.2021
Frank Hafner, **Mark Schutera** - ZF Friedrichshafen AG. Training method for an artificial neural network which is trained to process sensor data from a first sensor, with the following steps: Providing first sensor data from the first sensor and from second sensor data from a second sensor, the first sensor data having structures for improved machine interpretability of the first sensor data. Projecting the structures of the first sensor data onto the second

sensor data into projected second sensor data; Training the artificial neural network with the projected second sensor data.

[n] DE102020205346A1 - 28.10.2021
Mostafa Hussein, **Mark Schutera**, Tobias Mindel - ZF Friedrichshafen AG. The invention relates to a computer-implemented method for recognizing a lane in night shots by means of a detector which is designed to recognize lanes in day shots of the surroundings of an automatically operated vehicle. The invention also relates to a computer program product for recognizing a lane in night shots, a control device for a vehicle for regulating and/or controlling the automated operation of the vehicle, and a vehicle that is configured for automated operation.

[o] DE102020205470A1 - 04.11.2021
Mark Schutera, Frank Hafner - ZF Friedrichshafen AG. A method for processing optical sensor data, comprising the following steps: acquiring optical sensor data; Generating a copy of the optical sensor data, the copy having a reduced resolution compared to the original sensor data; Examining the copy of the optical sensor data to determine an area of increased relevance; Projecting the area of increased relevance of the copy onto the original sensor data; Cutting out the projected area in the original sensor data in order to further process the area of increased relevance based on the original sensor data.

[p] DE102021202072A1 - 08.09.2022
Mark Schutera, Frank Hafner - ZF Friedrichshafen AG. A computer implemented method for processing raw data from environment sensing sensors of a driving system or of a traffic infrastructure element comprising the steps of training a first artificial neural network (backbone), after which input data are compressed to latent features not interpretable by humans, providing extracted first layers, inputting the raw data recorded during operation of the driving system or the traffic infrastructure element, and storing the latent features obtained as output from the raw data features.

[q] DE102021207699A1 - 26.01.2023
Mark Schutera, Frank Hafner - ZF Friedrichshafen AG. A method for partially unsupervised training of a deep neural network comprising the steps of: Predefining a class hierarchy of an auxiliary class and a main class; where the objects of the main class are completely enclosed by objects of the auxiliary class (such as unexpected objects present on the driving path); providing a model pre-trained on the segmentation of objects of the auxiliary class (such

as free space); Fusion of the annotations to the main and auxiliary classes in the training dataset and subsequent training.

[r] DE102021208371A1 - 09.02.2023
Frank Hafner, **Mark Schutera** - ZF Friedrichshafen AG. Computer-implemented method and computer program for decoupling execution times of data processing procedures of a first environment perception system and a second environment perception system of a driving system.

[s] DE102021208423A1 - 09.02.2023
Mark Schutera, Hendrik Vogt - ZF Friedrichshafen AG. Method and computer program for safeguarding a perception system against contamination of an perception system imaging sensor and perception system of an automated driving system.

List of Supervised Related Work

Simon Seitz (Karlsruhe Institute of Technology) - WS17/18
Vehicle cut-In detection for adaptive cruise control with neural networks
Master Thesis

Mostafa Hussein (Freiburg University) - SS18
Image-to-image translation for domain adaptation from CARLA to KITTI
Master Thesis

Hendrik Vogt (Karlsruhe Insitute of Technology) - WS18/19
Object reidentification in multi-camera settings
Internship

Hendrik Vogt (Karlsruhe Insitute of Technology) - SS19
Continuous domain adaptation - object detection from day to night
Master Thesis

Kshama Ramesh (Technical University Chemnitz) - WS19/20
Inference runtime optimization for object detection by night
Master Thesis

Luca Rettenberger (Karlsruhe Institute of Technology) - WS20/21
Methods for the frugal labeler: Multi-class semantic segmentation on heterogeneous labels
Research Project

Stefan Bühler (Karlsruhe Institute of Technology) - WS20/21
Ventricle meets Myocardium: In vivo semantic segmentation analysis and high-throughput tooling
Bachelor Thesis

Sahil Arora (University of Applied Sciences Ravensburg-Weingarten) - SS21
Domain adaptation in camera mounting position shifts
Bachelor Thesis

Stefan Bühler (Karlsruhe Institute of Technology) - WS21/22
Unsupervised deployment shift approximation through epistemic uncertainty
Internship

Marcus Bentele (University of Applied Sciences Vorarlberg) - SS22
Tackling performance biases in object detection for autonomous driving
Master Thesis

Isabel Janez (Ravensburg University of Cooperative Education) - WS22/23
Validation strategies for trustworthy autonomous driving functions
Internship

Stefan Bühler (Karlsruhe Institute of Technology) - SS23
Anonymization at the borders of human and machine pattern recognition
Master Thesis

Related Publications

[1] M. Schutera, M. Hussein, J. Abhau *et al.*, “Night-to-day: Online image-to-image translation for object detection within autonomous driving by night,” *IEEE Transactions on Intelligent Vehicles*, pp. 480–489, 2020. doi: 10.1109/TIV.2020.3039456

[2] M. Schutera, F. M. Hafner, J. Abhau *et al.*, “Cuepervision: Self-supervised learning for continuous domain adaptation without catastrophic forgetting,” *Image and Vision Computing*, vol. 106, p. 104079, 2021. doi: 10.1016/j.imavis.2020.104079

[3] M. Schutera, S. Just, J. Gierten *et al.*, “Machine learning methods for automated quantification of ventricular dimensions,” *Zebrafish*, vol. 16, no. 6, pp. 542–545, 2019. doi: 10.1089/zeb.2019.1754

[4] M. Schutera, T. Dickmeis, M. Mione *et al.*, “Automated phenotype pattern recognition of zebrafish for high-throughput screening,” *Bioengineered*, vol. 7, no. 4, pp. 261–265, 2016. doi: 10.1080/21655979.2016.1197710

[5] M. Schutera, F. M. Hafner, H. Vogt *et al.*, “Domain is of the essence: Data deployment for city-scale multi-camera vehicle re-identification,” *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 1–6, 2019. doi: 10.1109/AVSS.2019.8909858

[6] M. Schutera, S. Elser, J. Abhau *et al.*, “Strategies for supplementing recurrent neural network training for spatio-temporal prediction,” *at-Automatisierungstechnik*, vol. 67, no. 7, pp. 545–556, 2019. doi: 10.1515/auto-2018-0124

[7] M. Schutera, N. Goby, D. Neumann *et al.*, “Transfer learning versus multi-agent learning regarding distributed decision-making in highway traffic,” *International Conference on Autonomous Agents and Multiagent Systems*, vol. 2129, no. 10, pp. 57–62, 2018.

[8] M. Schutera and F. Hafner, *Why does my neural network not learn?: Coffee table solutions*. Seattle, United States of America: Kindle Direct Publishing, 2022.

[9] J. Bernhard, M. Schutera, and E. Sax, “Optimizing test-set diversity: Trajectory clustering for scenario-based testing of automated driving systems,” *IEEE International Intelligent Transportation Systems Conference*, pp. 1371–1378, 2021. doi: 10.1109/ITSC48978.2021.9564771

[10] J. Bernhard, T. Schulik, M. Schutera *et al.*, “Adaptive test case selection for DNN-based perception functions,” *IEEE International Symposium on Systems Engineering*, pp. 1–7, 2021. doi: 10.1109/ISSE51541.2021.9582499

[11] M. Schutera, N. Goby, S. Smolarek *et al.*, “Distributed traffic light control at uncoupled intersections with real-world topology by deep reinforcement learning,” *Conference on Neural Information Processing Systems*, vol. 32, 2018.

[12] M. Schutera, L. Rettenberger, C. Pylatiuk *et al.*, “Methods for the frugal labeler: Multi-class semantic segmentation on heterogeneous labels,” *PLoS ONE*, vol. 17, no. 2, pp. 1–14, 2022. doi: 10.1371/journal.pone.0263656

[13] M. Schutera, L. Rettenberger, and M. Reischl, “Automated zebrafish phenotype pattern recognition: 6 years ago, and now,” *Zebrafish*, 2022. doi: 10.1089/zeb.2022.0027

[14] M. P. Schilling, T. Scherr, F. R. Münke *et al.*, “Automated annotator variability inspection for biomedical image segmentation,” *IEEE Access*, vol. 10, pp. 2753–2765, 2022. doi: 10.1109/ACCESS.2022.3140378

[15] F. M. Hafner, M. Zeller, M. Schutera *et al.*, “Backbone analysis: Structured insights into compute platforms from CNN inference latency,” *IEEE Intelligent Vehicles Symposium*, pp. 1801–1809, 2022. doi: 10.1109/IV51971.2022.9827260

[16] J. Bernhard, J. Schmidt, and M. Schutera, “Density based anomaly detection for wind turbine condition monitoring,” *1st International Joint Conference on Energy and Environmental Engineering*, pp. 87–93, 2022. doi: 10.5220/0011358600003355

Bibliography

[17] W. McCelloch and W. Pitts, “A logical calculus of the idea immanent in neural nets,” *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943. doi: 10.1007/BF02478259

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, United States of America: MIT Press, 2016.

[19] S. Kieffer, J. Avigad, and H. Friedman, “A language for mathematical knowledge management,” *Studies in Logic, Grammar and Rhetoric*, vol. 18, pp. 51–66, 2009. doi: 10.48550/arXiv.0805.1386

[20] S. Bem and H. de Jong, *Theoretical issues in psychology: An introduction*. United Kingdom: SAGE Publications, 2013.

[21] G. Shepherd, *The significance of real neuron architectures for real network simulations*. Cambridge, United Kingdom: MIT Press, 1993. doi: 10.5555/174471.174479

[22] J. M. Benítez, J. L. Castro, and I. Requena, “Are artificial neural networks black boxes?” *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 1156–1164, 1997. doi: 10.1109/72.623216

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, United States of America: MIT Press, 2005.

[24] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. doi: 10.1109/TKDE.2009.191

[25] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991. doi: 10.1016/0893-6080(91)90009-T

[26] D. Kriesel, “A brief introduction on neural networks,” University Bonn, Lecture Notes, 2015. [Online]. Available: https://www.dkriesel.com/science/neural_networks

[27] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. New York, United States of America: Psychology Press, 1949.

[28] C. M. Bishop, *Pattern Recognition and Machine Learning*. Heidelberg, Germany: Springer-Verlag, 2006.

[29] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Heidelberg, Germany: Springer-Verlag, 1995. doi: 10.1007/978-1-4757-3264-1

[30] M. A. Boden, "Machine perception," *The Philosophical Quarterly (1950)*, vol. 19, no. 74, pp. 33–45, 1969.

[31] X. Feng, Y. Jiang, X. Yang *et al.*, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309–320, 2019. doi: 10.1016/j.vlsi.2019.07.005

[32] G. Trivino and A. Sobrino, "Human perceptions versus computational perceptions in computational theory of perceptions," *Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology*, pp. 327–332, 2009.

[33] Z. Zhao, P. Zheng, S. tao Xu *et al.*, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865

[34] M. Abadi, A. Agarwal, P. Barham *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," *tensorflow.org*, 2015. doi: 10.5281/zenodo.4724125

[35] T. Lin, M. Maire, S. J. Belongie *et al.*, "Microsoft COCO: Common objects in context," *European Conference on Computer Vision*, pp. 740–755, 2014. doi: 10.1007/978-3-319-10602-1_48

[36] W. Liu, D. Anguelov, D. Erhan *et al.*, "SSD: Single shot multibox detector," *European Conference on Computer Vision*, pp. 21–37, 2016. doi: 10.1007/978-3-319-46448-0_2

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv.org*, 2014. doi: 10.48550/arXiv.1409.1556

[38] A. G. Howard, M. Zhu, B. Chen *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv.org*, 2017. doi: 10.48550/arXiv.1704.04861

[39] C. Szegedy, W. Liu, Y. Jia *et al.*, "Going deeper with convolutions," *Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2014. doi: 10.1109/CVPR.2015.7298594

[40] R. B. Girshick, “Fast R-CNN,” *IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015. doi: 10.1109/ICCV.2015.169

[41] J. Dai, Y. Li, K. He *et al.*, “R-FCN: Object detection via region-based fully convolutional networks,” *Advances in Neural Information Processing Systems*, pp. 379–387, 2016. doi: 10.5555/3157096.3157139

[42] K. He, X. Zhang, S. Ren *et al.*, “Deep residual learning for image recognition,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90

[43] J. Janai, F. Güney, A. Behl *et al.*, “Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art,” *Foundations and Trends in Computer Graphics and Vision*, vol. 12, pp. 1–308, 2020. doi: 10.1561/9781680836899

[44] M. Everingham, L. Van Gool, C. K. I. Williams *et al.*, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. doi: 10.1007/s11263-009-0275-4

[45] Y. LeCun, C. Cortes, and C. Burges, “MNIST handwritten digit database,” *AT&T Labs*, vol. 2, 2010.

[46] J. Deng, W. Dong, R. Socher *et al.*, “ImageNet: A large-scale hierarchical image database,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Conference on Neural Information Processing Systems*, vol. 1, pp. 1097–1105, 2012. doi: 10.1145/3065386

[48] S. Krig, *Computer vision metrics: Survey, taxonomy, and analysis*. California, United States of America: Apress, 2014. doi: 10.1007/978-1-4302-5930-5

[49] M. de Berg, O. Cheong, M. van Kreveld *et al.*, *Computational geometry*, 3rd ed. Germany: Springer Vieweg, 2008. doi: 10.1007/978-3-540-77974-2

[50] J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. doi: 10.1137/0105003

[51] Y. Bar-Shalom, F. Daum, and J. Huang, “The probabilistic data association filter,” *IEEE Control Systems*, vol. 29, no. 6, pp. 82–100, 2009. doi: 10.1109/MCS.2009.934469

[52] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” *International Conference on Machine Learning*, vol. 23, pp. 233–240, 2006. doi: 10.1145/1143844.1143874

[53] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966. doi: 10.1515/9781400835386

[54] R. Guidotti, A. Monreale, S. Ruggieri *et al.*, “A survey of methods for explaining black box models,” *ACM Computing Surveys*, vol. 51, no. 5, 2018. doi: 10.1145/3236009

[55] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951. doi: 10.1214/aoms/1177729694

[56] X. Huang, L. Wu, and Y. Ye, “A review on dimensionality reduction techniques,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 10, 2019. doi: 10.5120/ijca2017915260

[57] K. F. Pearson, “On lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. doi: 10.1080/14786440109462720

[58] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *Journal of Open Source Software*, vol. 29, p. 861, 2018. doi: 10.21105/joss.00861

[59] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[60] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081

[61] G. Csurka, *Domain adaptation for visual applications: A comprehensive survey*. Switzerland: Springer, Cham, 2017. doi: 10.1007/978-3-319-58347-1_1

[62] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” *Neurocomputing*, vol. 312, pp. 135–153, 2018. doi: 10.1016/j.neucom.2018.05.083

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014. doi: 10.1145/3422622

[64] Q. Gan, Q. Guo, Z. Zhang *et al.*, "First step toward model-free, anonymous object tracking with recurrent neural networks," *International Conference on Learning Representations*, 2015. doi: 10.48550/arXiv.1511.06425

[65] D. Yoo, N. Kim, S. Park *et al.*, "Pixel-level domain transfer," *European Conference on Computer Vision*, pp. 517–532, 2016. doi: 10.1007/978-3-319-46484-8_31

[66] P. Isola, J.-Y. Zhu, T. Zhou *et al.*, "Image-to-image translation with conditional adversarial networks," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976, 2017. doi: 10.1109/CVPR.2017.632

[67] A. Anoosheh, T. Sattler, R. Timofte *et al.*, "Night-to-day image translation for retrieval-based localization," *International Conference on Robotics and Automation*, pp. 5958–5964, 2019. doi: 10.1109/ICRA.2019.8794387

[68] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *Advances in Neural Information Processing Systems*, pp. 469–477, 2016. doi: 10.5555/3157096.3157149

[69] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013. doi: 10.1109/TPAMI.2013.50

[70] K. Bousmalis, G. Trigeorgis, N. Silberman *et al.*, "Domain separation networks," *Advances in Neural Information Processing Systems*, pp. 343–351, 2016.

[71] M. Ghifary, W. Bastiaan Kleijn, M. Zhang *et al.*, "Domain generalization for object recognition with multi-task autoencoders," *IEEE International Conference on Computer Vision*, pp. 2551–2559, 2015. doi: 10.1109/ICCV.2015.293

[72] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017. doi: 10.1080/01621459.2017.1285773

[73] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *2nd International Conference on Learning Representations*, 2014. doi: 10.48550/arXiv.1312.6114

[74] S. Chopra, S. Balakrishnan, and R. Gopalan, “DLID: Deep learning for domain adaptation by interpolating between domains,” *International Conference on Machine Learning*, 2013.

[75] M. Ghifary, W. B. Kleijn, M. Zhang *et al.*, “Deep reconstruction-classification networks for unsupervised domain adaptation,” *European Conference on Computer Vision*, pp. 597–613, 2016. doi: 10.1007/978-3-319-46493-0_36

[76] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” *International Conference on Machine Learning*, vol. 37, pp. 1180–1189, 2015.

[77] E. Tzeng, J. Hoffman, K. Saenko *et al.*, “Adversarial discriminative domain adaptation,” *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 2962–2971, 2017. doi: 10.1109/CVPR.2017.316

[78] J.-Y. Zhu, T. Park, P. Isola *et al.*, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017. doi: 10.1109/ICCV.2017.244

[79] T. Kim, M. Cha, H. Kim *et al.*, “Learning to discover cross-domain relations with generative adversarial networks,” *International Conference on Machine Learning*, vol. 70, pp. 1857–1865, 2017.

[80] Z. Yi, H. Zhang, P. Tan *et al.*, “Dualgan: Unsupervised dual learning for image-to-image translation,” *IEEE International Conference on Computer Vision*, pp. 2849–2857, 2017. doi: 10.1109/ICCV.2017.310

[81] L. Jing and Y. Tian, “Self-supervised visual feature learning with deep neural networks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 4037–4058, 2020. doi: 10.1109/TPAMI.2020.2992393

[82] D.-H. Lee, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” *International Conference on Machine Learning*, vol. 3, p. 2, 2013.

[83] D. Li, W.-C. Hung, J.-B. Huang *et al.*, “Unsupervised visual representation learning by graph-based consistent constraints,” *European Conference on Computer Vision*, pp. 678–694, 2016. doi: 10.1007/978-3-319-46493-0_41

[84] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999. doi: 10.1145/331499.331504

[85] M. Noroozi, A. Vinjimoor, P. Favaro *et al.*, “Boosting self-supervised learning via knowledge transfer,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9359–9367, 2018. doi: 10.1109/CVPR.2018.00975

[86] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-supervised self-training of object detection models,” *IEEE Applications of Computer Vision*, vol. 1, pp. 29–36, 2005. doi: 10.1109/ACV-MOT.2005.107

[87] D. Barnes, W. Maddern, and I. Posner, “Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy,” *IEEE International Conference on Robotics and Automation*, pp. 203–210, 2017. doi: 10.1109/ICRA.2017.7989025

[88] M. Caron, P. Bojanowski, A. Joulin *et al.*, “Deep clustering for unsupervised learning of visual features,” *European Conference on Computer Vision*, pp. 132–149, 2018. doi: 10.1007/978-3-030-01264-9_9

[89] S. Hochreiter and J. Schmidhuber, *Long short-term memory*. Cambridge, United States of America: MIT press journals, 1997, vol. 9.

[90] M. Wulfmeier, A. Bewley, and I. Posner, “Incremental adversarial domain adaptation for continually changing environments,” *IEEE International Conference on Robotics and Automation*, pp. 1–9, 2018. doi: 10.1109/ICRA.2018.8460982

[91] G. I. Parisi, R. Kemker, J. L. Part *et al.*, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, pp. 54–71, 2019. doi: 10.1016/j.neunet.2019.01.012

[92] I. Alhashim and P. Wonka, “High quality monocular depth estimation via transfer learning,” *arXiv.org*, 2018. doi: 10.48550/arXiv.1812.11941

[93] R. Arandjelovic and A. Zisserman, “Look, listen and learn,” *IEEE International Conference on Computer Vision*, pp. 609–617, 2017. doi: 10.1109/ICCV.2017.73

[94] F. Ma, G. V. Cavalheiro, and S. Karaman, “Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera,” *International Conference on Robotics and Automation*, pp. 3288–3295, 2019. doi: 10.1109/ICRA.2019.8793637

[95] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in *Psychology of Learning and Motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[96] A. A. Rusu, N. C. Rabinowitz, G. Desjardins *et al.*, “Progressive neural networks,” *arXiv.org*, 2016. doi: 10.48550/arXiv.1606.04671

[97] J. Kirkpatrick, R. Pascanu, N. Rabinowitz *et al.*, “Overcoming catastrophic forgetting in neural networks,” *National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. doi: 10.1073/pnas.1611835114

[98] M.-Y. Liu, T. Breuel, and J. Kautz, “Unsupervised image-to-image translation networks,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 700–708, 2017. doi: 10.5555/3294771.3294838

[99] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *International Conference on Learning Representations*, vol. abs/1312.6114, 2014. doi: 10.48550/arXiv.1312.6114

[100] P. C. Mahalanobis, “On the generalised distance in statistics,” *Proceedings of the National Institute of Sciences of India*, vol. 2, pp. 49–55, 1936. doi: 10.1007/s13171-019-00164-5

[101] L. Yang, N. Y. Ho, R. Alshut *et al.*, “Zebrafish embryos as models for embryotoxic and teratological effects of chemicals,” *Reproductive Toxicology*, vol. 28, no. 2, pp. 245–253, 2009. doi: 10.1016/j.reprotox.2009.04.013

[102] J. Bakkers, “Zebrafish as a model to study cardiac development and human cardiac disease,” *Cardiovascular Research*, vol. 91, no. 2, pp. 279–288, 2011. doi: 10.1093/cvr/cvr098

[103] R. L. Lamason, M.-A. P. Mohideen, J. R. Mest *et al.*, “SLC24A5, a putative cation exchanger, affects pigmentation in zebrafish and humans,” *Science*, vol. 310, no. 5755, pp. 1782–1786, 2005. doi: 10.1126/science.1116238

[104] A. Bréhéret, *Pixel annotation tool*, 2017. [Online]. Available: https://github.com/abreheret/PixelAnnotationTool

[105] Deutsches Statistisches Bundesamt, “Unfallentwicklung auf Deutschen Straßen,” DESTATIS, Wiesbaden, Germany, Tech. Rep., 2016.

[106] T. Unselt, R. Schöneburg, and J. Bakker, “Insassen und Partnerschutz unter den Rahmenbedingungen der Einführung autonomer

Fahrzeugsysteme," VDI/VW-Gemeinschaftstagung Automotive Security, Wolfsburg, Germany, Technical Report 29, 2013.

[107] T. M. Gasser, C. Arzt, M. Ayoubi *et al.*, "Rechtsfolgen zunehmender Fahrzeugautomatisierung," Bundesanstalt für Straßenwesen, Bergisch Gladbach, Germany, Research Report 83, 2012.

[108] World Health Organization, "Global status report on road safety 2020," *Research Report*, 2020.

[109] H. Winner, F. Lotz, S. Hakuli *et al.*, "Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort," *Handbuch Fahrerassistenzsysteme*, 2015. doi: 10.1007/978-3-658-05734-3

[110] SAE International, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," *On-Road Automated Driving committee*, no. J3016_202104, pp. 1–41, 2014. doi: 10.4271/J3016_201104

[111] A. Cacilo, S. Schmidt, P. Wittlinger *et al.*, "Hochautomatisiertes Fahren auf Autobahnen - industriepolitische Schlussfolgerungen," *Veröffentlichungsdatenbank Fraunhofer-Publica*, vol. Studie im Auftrag des Bundesministeriums für Wirtschaft und Energie, 2015.

[112] Deutsches Statistisches Bundesamt, "Verkehr - Verkehrsunfälle," DESTATIS, Wiesbaden, Germany, Tech. Rep., 2017.

[113] C. Hughes, R. O'Malley, D. O'Cualain *et al.*, *Trends towards automotive electronic vision systems for mitigation of accidents in safety critical situations.* United Kingdom: IntechOpen, 2011. doi: 10.5772/12907

[114] C. Fors and S.-O. Lundkvist, *Night-time traffic in urban areas: A literature review on road user aspects.* Sweden: Statens väg-och transportforskningsinstitut, 2009.

[115] M. Cordts, M. Omran, S. Ramos *et al.*, "The Cityscapes dataset for semantic urban scene understanding," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016. doi: 10.1109/CVPR.2016.350

[116] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012. doi: 10.1109/CVPR.2012.6248074

[117] A. Nazib, C. Oh, and C. W. Lee, “Object detection and tracking in night time video surveillance,” *International Conference on Ubiquitous Robots and Ambient Intelligence*, vol. 10, pp. 629–632, 2013. doi: 10.1109/URAI.2013.6677410

[118] M. Rezaei and M. Terauchi, “iROADS dataset intercity roads and adverse driving scenarios,” *Enpeda Image Sequence Analysis Test Site*, vol. 8333, 2013. doi: 10.1007/978-3-642-53842-1_6

[119] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, “Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012. doi: 10.1109/TITS.2012.2209421

[120] Y. Choi, N. Kim, S. Hwang *et al.*, “KAIST multi-spectral day/night data set for autonomous and assisted driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 934–948, 2018. doi: 10.1109/TITS.2018.2791533

[121] D. Dai and L. van Gool, “Dark model adaptation: Semantic image segmentation from daytime to nighttime,” *IEEE International Conference on Intelligent Transportation Systems*, pp. 3819–3824, 2018. doi: 10.1109/ITSC.2018.8569387

[122] F. Yu, W. Xian, Y. Chen *et al.*, “BDD100K: A diverse driving video database with scalable annotation tooling,” *Conference on Computer Vision and Pattern Recognition*, pp. 2633–2642, 2018. doi: 10.48550/arXiv.1805.04687

[123] M.-F. Chang, J. W. Lambert, P. Sangkloy *et al.*, “Argoverse: 3D tracking and forecasting with rich maps,” *Conference on Computer Vision and Pattern Recognition*, pp. 8740–8749, 2019. doi: 10.1109/CVPR.2019.00895

[124] K. Behrendt, “Boxy vehicle detection in large images,” *IEEE International Conference on Computer Vision*, pp. 840–846, 2019. doi: 10.1109/ICCVW.2019.00112

[125] Y. P. Loh and C. S. Chan, “Getting to know low-light images with the Exclusively Dark dataset,” *Computer Vision and Image Understanding*, vol. 178, pp. 30–42, 2019. doi: 10.1016/j.cviu.2018.10.010

[126] J. Kim, H. Hong, and K. Park, “Convolutional neural network-based human detection in nighttime images using visible light camera sensors,” *Sensors*, p. 1065, 2017. doi: 10.3390/s17051065

[127] J. Tang, J. Kim, and E. Peli, “Image enhancement in the JPEG domain for people with vision impairment,” *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 11, pp. 2013–2023, 2004. doi: 10.1109/TBME.2004.834264

[128] C. Chen, Q. Chen, J. Xu *et al.*, “Learning to see in the dark,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3291–3300, 2018. doi: 10.1109/CVPR.2018.00347

[129] J. M. A. Alvarez and A. M. Lopez, “Road detection based on illuminant invariance,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 184–193, 2011. doi: 10.1109/TITS.2010.2076349

[130] R. K. Satzoda and M. M. Trivedi, “Looking at vehicles in the night: Detection and dynamics of rear lights,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4297–4307, 2019. doi: 10.1109/TITS.2016.2614545

[131] I. S. Team, “SF332X-10X family preliminary datasheet - 2M pixel automotive camera,” SEKONIX Corp., Korea, Manual, 2016.

[132] NVIDIA Corporation, *Vibrante Linux SDK 4.1 for DRIVE PX 2 - Development guide 4.1.6.1 Beta 2.0 Release*, California, United States of America, 2017.

[133] SmallHD, *MON-702 bright full HD field monitor*, North Carolina, United States of America, 2019.

[134] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[135] *MobaXTerm: Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more*, Mobatek, 2017. [Online]. Available: https://mobaxterm.mobatek.net/

[136] NVIDIA Corporation, *NVIDIA TESLA P100 GPU accelerator*, California, United States of America, 2016.

[137] R Core Team, “R: A language and environment for statistical computing,” R Foundation for Statistical Computing, Vienna, Austria, Manual, 2013. [Online]. Available: http://www.R-project.org/

[138] P. Isola, J. Zhu, T. Zhou *et al.*, “Image-to-image translation with conditional adversarial networks,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976, 2017. doi: 10.1109/CVPR.2017.632

[139] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, ser. Neurocomputing: Foundations of Research. Cambridge, United Kingdom: MIT Press, 1988.

[140] A. Karpathy, "CS231n: Convolutional neural networks for visual recognition," Stanford University, Lecture Notes, 2018. [Online]. Available: http://cs231n.github.io/neural-networks-3/

[141] B. Xu, N. Wang, T. Chen *et al.*, "Empirical evaluation of rectified activations in convolutional network," *arXiv.org*, 2015. doi: 10.48550/arXiv.1505.00853

[142] W. K. Newey and D. McFadden, *Large sample estimation and hypothesis testing*, ser. Handbook of Econometrics. Elsevier, 1994, vol. 4, p. 2131.

[143] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. doi: 10.1214/aoms/1177729586

[144] L. Bottou, "Online learning and stochastic approximations," *Online Learning in Neural Networks*, vol. 17, no. 9, p. 142, 1998.

[145] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964. doi: 10.1016/0041-5553(64)90137-5

[146] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(\frac{1}{k^2})$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[147] I. Sutskever, J. Martens, G. Dahl *et al.*, "On the importance of initialization and momentum in deep learning," *International Conference on Machine Learning*, pp. 1139–1147, 2013. doi: 10.5555/3042817.3043064

[148] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011. doi: 10.5555/1953048.2021068

[149] G. Hinton, N. Srivastava, and K. Swersky, "Overview of mini-batch gradient descent," *Neural Networks for Machine Learning Lecture*, vol. 6a, 2012.

[150] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2014. doi: 10.48550/arXiv.1412.6980

[151] K. He, X. Zhang, S. Ren *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123

[152] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[153] J. Martens, "Deep learning via Hessian-free optimization," *International Conference on Machine Learning*, pp. 735–742, 2010. doi: 10.5555/3104322.3104416

[154] D. Mishkin and J. Matas, "All you need is a good init," *arXiv.org*, 2015. doi: 10.48550/arXiv.1511.06422

[155] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine Learning*, pp. 448–456, 2015. doi: 10.5555/3045118.3045167

[156] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, pp. 950–957, 1992. doi: 10.5555/2986916.2987033

[157] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003. doi: 10.1016/S0893-6080(03)00138-2

[158] J. L. Hennessy and D. A. Patterson, *Computer Organization and Design: The Hardware/Software Interface*. San Francisco, United States of America: Morgan Kaufmann Publishers Inc., 1998, vol. 2.

[159] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Visual Recognition*, pp. 1–8, 2017. doi: 10.48550/arXiv.1712.04621

[160] N. Srivastava, G. Hinton, A. Krizhevsky *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[161] G. E. Hinton, N. Srivastava, A. Krizhevsky *et al.*, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv.org*, 2012. doi: 10.48550/arXiv.1207.0580

[162] L. Prechelt, *Early stopping - but when?* Heidelberg, Germany: Springer-Verlag, 1998. doi: 10.1007/3-540-49430-8_3

[163] K. Bousmalis, N. Silberman, D. Dohan *et al.*, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 95–104, 2017. doi: 10.1109/CVPR.2017.18

[164] J. Hoffman, E. Tzeng, T. Park *et al.*, “CyCADA: Cycle-consistent adversarial domain adaptation,” *International Conference on Machine Learning*, vol. 35, 2018.

List of Equations

List of Figures

List of Tables

Index