UNIVERSITÄT ZU LÜBECK

From the Institute of Signal Processing
of the University of Lübeck
Director: Prof. Dr.-Ing. Alfred Mertins

Incorporating Prior Knowledge of Invariances into Deep Models for Computer Vision

Dissertation
for Fullfilment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by
M. Sc. Benjamin Coors
from Leonberg

Lübeck, 2022

First referee: PD Dr. A. Condurache
Second referee: Prof. Dr. E. Barth
Date of oral examination: 28.04.2022
Approved for printing. Lübeck, 02.05.2022

Abstract

In the past decade, deep learning has revolutionized the field of computer vision. Nowadays, most challenging computer vision tasks are commonly solved with deep neural networks. Yet, while neural networks can offer state-of-the-art performance, they generally require large labeled training datasets to fulfill their full potential. As the collection and annotation of a large-scale computer vision dataset is work intensive and expensive, it is therefore worth asking how deep learning's hunger for data can be reduced.

One approach for improving the data efficiency of machine learning models such as neural networks is the use of prior knowledge. A particularly powerful form of prior knowledge are invariances, which define the visual distractors to which a model should not pay attention to. In this thesis, three ideas for incorporating invariances into the learning process, architecture or training data of deep neural networks are investigated.

First, an unsupervised similarity loss is proposed, which utilizes the fact that labels do not change under a known set of geometric transformations of the input. This invariance property acts as a regularizer when training an image classification model with little labeled training data and enables the use of unlabeled data for semi-supervised learning.

Second, the SphereNet framework for learning spherical representations in omnidirectional images is introduced. In contrast to regular perspective images, only few omnidirectional datasets exist which are generally of smaller size than their perspective counterparts. Here, SphereNet facilitates a transfer from the perspective to the omnidirectional domain by encoding invariance to the distortions in the equirectangular representation of omnidirectional images into the architecture of deep convolutional neural networks.

Third, the Novel Viewpoint Adaptation (NoVA) model is presented which enables an adaptation from a viewpoint in which a large labeled dataset is available to a novel viewpoint, potentially within a novel domain, in which few or no labeled examples exist. For this, NoVA utilizes the prior knowledge about the transformation between the two viewpoints as well as the fact that the semantic labels in 3D space should be invariant to the change in viewpoint to translate the input images and labels to the novel viewpoint.

Extensive experiments validate the effectiveness of all three proposed methods in comparison to current state-of-the-art baseline approaches for a variety of challenging computer vision tasks including image classification, object detection, semantic segmentation and optical flow.

Kurzfassung

Deep Learning hat im vergangenen Jahrzehnt das Gebiet der Bildverarbeitung revolutioniert. Heutzutage stellen tiefe neuronale Netze für eine Vielzahl an herausfordernden Bildverarbeitungsaufgaben den Stand der Technik dar. Jedoch benötigen diese große annotierte Trainings-Datensätze um ihr volles Potential auszuschöpfen. Da die Aufnahme und Annotation solcher Datensätze allerdings sehr arbeits- und kostenintensiv ist, stellt sich die Frage, wie sich der Hunger tiefer neuronaler Netze nach Daten reduzieren lässt.

Ein Ansatz um die Dateneffizienz maschineller Lernverfahren wie neuronaler Netze zu verbessern ist die Nutzung von Vorwissen, insbesondere in Form von Invarianzen. Diese definieren welchen visuellen Änderungen ein Bildverarbeitungsmodell keine Bedeutung schenken sollte. In dieser Arbeit werden drei Ideen zum Einbau von Invarianzen in den Lernprozess, die Architektur und die Trainingsdaten tiefer neuronaler Netze untersucht.

Erstens wird eine unüberwachte Ähnlichkeits-Fehlerfunktion vorgeschlagen, welche darauf basiert, dass sich die Zielwerte nicht unter bestimmten geometrischen Transformationen des zugehörigen Eingabebildes verändern. Diese Invarianz-Eigenschaft agiert als Regularisierer wenn ein Bildklassifikationsmodell mit wenigen annotierten Daten trainiert wird und erlaubt die Nutzung von Daten ohne Zielwerte für semi-überwachtes Lernen.

Zweitens wird das SphereNet Framework zum Lernen sphärischer Repräsentationen aus omnidirektionalen Bildern eingeführt. Im Gegensatz zu regulären perspektivischen Bildern existieren nur wenige omnidirektionale Datensätze, welche allgemein von kleinerer Größe als ihre perspektivischen Gegenstücke sind. Hier ermöglicht SphereNet durch die Einbindung von Invarianz zu Verzerrungen, welche durch die äquirektanguläre Repräsentation in omnidirektionalen Bildern hervorgerufen wird, in die Architektur tiefer Faltungsnetze einen Transfer aus der perspektivischen zur omnidirektionalen Domäne.

Drittens wird das Novel Viewpoint Adaptation (NoVA) Modell präsentiert, welches eine Adaptation von einem Blickwinkel, in dem ein großer annotierter Datensatz verfügbar ist, zu einem neuen Blickwinkel ermöglicht, welcher sich möglicherweise in einer anderen Domäne befindet und in welchem wenige oder keine annotierten Daten vorhanden sind. Hierbei nutzt NoVA das Vorwissen über die Transformation zwischen den zwei Blickwinkeln sowie die Tatsache, dass die semantischen Zielwerte im 3D-Raum invariant zu einer Änderung des Blickwinkels sein sollten, um die Eingabebilder und Zielwerte in den neuen Blickwinkel zu übersetzen.

Umfangreiche Experimente bestätigen die Effektivität aller drei Ansätze im Vergleich zu existierenden Ansätzen aus dem aktuellen Stand der Technik für eine Vielzahl von herausfordernden Bildverarbeitungsaufgaben wie der Bildklassifikation, Objektdetektion, semantischen Segmentierung und des optischen Flusses.

Acknowledgments

I would like to thank my supervisors, PD Dr. Alexandru Paul Condurache and Prof. Dr. Andreas Geiger, my colleagues at Bosch, in particular fellow Bosch PhD students Felix, Julia and Matthias, my colleagues in the Autonomous Vision Group at the Max Planck Institute for Intelligent Systems, Aditya, Aseem, Carolin, Cusuh, David, Despoina, Eshed, Fatma, Joël, Kashyap, Katja, Lars, Mengqi, Michael^2, Seungjun, Simon, Yiyi and Zhaoyang, as well as my parents, Mechthild and Andreas, and my partner, Dongmyeong, for their encouragement, guidance, patience and never-ending support throughout my PhD journey.

Contents

Chapter 1

Introduction

Deep convolutional neural networks have revolutionized the field of computer vision and currently offer state-of-the-art performance for many computer vision tasks. However, their success comes with the need for large annotated training datasets, which take considerable time and effort to create. While in some cases large annotated datasets readily exist, novel application scenarios as well as different camera sensors or setups often require the recording and labeling of new a dataset. In order to reduce the labeling effort, it is thus worthwhile to investigate how performant computer vision models can be trained when only limited or potentially no labeled data is available.

A well-studied approach to improve the data efficiency of a machine learning system is the utilization of prior knowledge, particularly in the form of *invariance* or *equivariance* with respect to a set of known transformations of the input data. While invariance signals that a specific transformation should not affect the output of a model, equivariance signals that the output of a model should transform in accordance with the input transformation. For example, an image classification model should be invariant to translations or scale changes of an object, whereas an object detection model should be equivariant to these transformations as it needs to estimate an object's bounding box. Thus, it is important that the right invariance or equivariance property is chosen depending on the task at hand.

Recently, the benefits of incorporating invariance or equivariance have also been demonstrated for deep neural networks. Yet, existing approaches are often limited to simple geometric transformations such as rotations or to the single task of image classification. In this thesis, three novel ideas for incorporating prior knowledge of invariances into deep models are presented. The proposed methods cover a variety of transformations and challenging computer visions tasks which are of high relevance for real-life applications such as autonomous driving or advanced driver-assistance systems.

A first proposed method is concerned with *learning invariance* with respect to a specific set of transformations in a deep image classification model. For the task of image classification, transformations such as translations, rotations, scale changes or deformations of an object in the scene do not change the identity of the object and therefore should not influence the model's classification. While variations in an object's scale or position occur naturally within a dataset, they can also be synthetically created with the help of data augmentation (e.g. by rotating, cropping or warping the image). The proposed method utilizes the simple insight that an image's class should not change under data augmentation to introduce a novel unsupervised similarity loss which acts as an effective regularizer in the presence of little annotated training data and facilitates the learning of transformation invariant representations in a semi-supervised manner.

A second proposed approach *encodes invariance* to distortions which are present in omnidirectional images into the architecture of deep convolutional neural networks. While large labeled datasets of perspective images are commonly available, few large-scale omnidirectional datasets exist. At the same time, omnidirectional data comes with the challenge of containing an additional factor of variation that is introduced by the distortions in the equirectangular representation of omnidirectional images. Here, an object's appearance will vary greatly depending on its longitudinal position in the image. This thesis presents the novel SphereNet framework which adapts the sampling locations of the convolutional filters to effectively reverse the distortions and wraps the convolutional filters around the image sphere, thereby enabling an improved transfer of existing convolutional neural network models to the omnidirectional case for a variety of tasks including object detection, semantic segmentation and optical flow estimation.

Finally, a third proposed model deals with *adapting the training data for invariance* to a change in viewpoint. When applying a model that was trained with labeled data from a given source viewpoint in a novel target viewpoint, it can be observed that the model's performance will drop significantly. Yet, while it is easy to record new data points in the target viewpoint, it is a labor-intensive task to label them. Thus, the proposed Novel Viewpoint Adaptation (NoVA) model utilizes the prior knowledge about the change in viewpoint in order to translate the source view images and labels to the target viewpoint. This translation is performed in an unsupervised way by utilizing an explicit representation of the 3D scene geometry and does not require source-target view pairs. Based on the translated source view data, a semantic segmentation model is then trained which performs well in the target view despite having been trained without any true target view labels.

1.1 Contributions

In summary, this thesis makes the following contributions:

- A novel *similarity loss* which acts as an effective regularizer when little labeled training data is available by utilizing labeled as well as additional unlabeled training for learning a transformation invariant image classification model in a supervised or semi-supervised manner [22].
- The *SphereNet* framework for learning spherical image representations from omnidirectional images by encoding distortion invariance into the filters of deep convolutional neural networks. SphereNet retains the original spherical image connectivity and, by building on regular convolutions, enables the transfer of perspective models to omnidirectional inputs for a variety of computer vision tasks [20].
- The *Novel Viewpoint Adaptation (NoVA)* model for the unsupervised adaptation of a labeled source view dataset to a novel target viewpoint in which no labeled data is available. By utilizing the known transformation between the two viewpoints, the viewpoint change itself no longer has to be learned by the model and NoVA instead reduces the task to the well-studied problems of depth estimation, image inpainting and stylization [21].

1.2 Thesis Outline

The structure of this thesis is as follows: Chapter 2 presents the foundations and current state-of-the-art in the field on incorporating prior knowledge of invariances/equivariances into deep neural network models for computer vision. In the following Chapter 3, the proposed method on learning transformation invariance with weak supervision is presented. In Chapter 4, the SphereNet framework for learning spherical representations from omnidirectional images is described. Chapter 5 then gives details on the Novel Viewpoint Adaptation (NoVA) model that supports an unsupervised adaptation to a novel viewpoint in which no labeled data is available. Finally, Chapter 6 draws conclusions about the obtained results and outlines possible lines of future work.

Chapter 2

Theoretical Foundations

“ *Recognition is the science and engineering of models that have effective invariance and equivariance properties.* ”

Ross Girshick, *ICVSS*, 2017

In this chapter, we will provide an introduction to the incorporation of prior knowledge of invariances into machine learning and computer vision models. While the first part of this chapter will focus on an overview on the foundations of invariance in machine learning models, the second part of the chapter will go into depth on recent approaches for incorporating invariance or equivariance into deep neural network models.

Prior knowledge, as defined by Schölkopf et al. [123], refers to all the information about the task at hand which is available in addition to the training data itself. An example of prior knowledge that is fundamental to machine learning is the general *smoothness assumption* which states that a test example which is similar to a training example should also be assigned to the same class or have a corresponding output. This assumption enables a machine learning model to generalize from its training data to novel unseen test examples. Another fundamental form of prior knowledge that is commonly utilized in the field of image recognition is the prior knowledge about *transformation invariances*.

For many tasks, it is known that the features or output of an image recognition model should not change under certain transformations of its input data. For example, the output of a handwritten digit classifier should not change when a digit is slightly deformed, rotated or scaled. In this case, the model should be *invariant* to these transformations. More formally, we can say that the output of a classifier f for a given input x should be

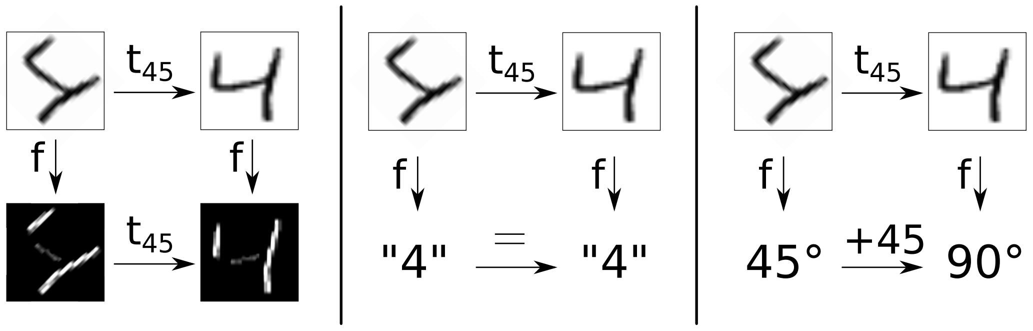


Figure 2.1: **Equivariance** (left) vs. **Invariance** (middle) vs. **Covariance** (right). While the output of an equivariant function f preserves the input rotation t_{45} in its output, the output of an invariant classification function f remains the same despite the transformation of the input. On the other hand, the output of a covariant orientation estimation function f changes as a function of the rotation of the input. Adapted from [98].

equal to its output for a transformed version of x with respect to a transformation t from a set of transformations $\mathcal{T}$:

$$f(x) = f(t(x)) \tag{2.1}$$

However, not all tasks require a model to be invariant but may instead require the output to be *covariant* or *equivariant* with respect to a transformation of the input. For example, a model which is tasked with estimating the orientation of a handwritten digit would struggle to do so based on rotation invariant features, which discard the information about changes in digit orientation. Instead, it would benefit from covariant or equivariant features that would not discard but contain the information about the change in digit orientation. While an equivariant model would represent the transformation of the input with the same transformation of its output (see Eq. (2.2)), a covariant model would associate the transformation t of the input with a second transformation t' of the output, that would itself be a function of t (see Eq. (2.3)). Fig. 2.1 visualizes this difference in order to make the distinction between equivariance, invariance and covariance more clear.

$$t(f(x)) = f(t(x)) \tag{2.2}$$

$$t'(f(x)) = f(t(x)) \tag{2.3}$$

Thereby, both invariance and equivariance can be seen as special cases of covariance where $t' = \mathbb{I}$, the identity, for invariance and $t' = t$ for equivariance.

While historically the topic of transformation invariance has received more attention in the research community (see the survey paper by Tuytelaars & Mikolajczyk [145]), equivariance has recently come into more focus [14, 17, 19, 72]. One reason for this change is that the attention of the computer vision community has somewhat shifted from the simpler task of image classification, which mostly benefits from transformation invariance, to more challenging tasks such as object detection or pose estimation, in which equivariant features are beneficial in order to predict an object's position or pose. However, the choice to which specific transformation a model should be invariant or equivariant is highly task-specific and must be taken with care as invariance or equivariance properties which are not adapted to the task at hand may have a negative impact on model performance [145].

Nowadays, many different approaches exist for incorporating invariance or equivariance properties into a machine learning or image recognition model. In general, one can distinguish approaches which aim to *learn* invariance from approaches that directly *encode* invariance into the architecture of the model. Furthermore, one can distinguish between *global* invariance approaches, that aim for invariance wrt. a global transformation of the input, and *local* invariance approaches, which target invariance wrt. a transformation of a local neighborhood of the input such as an image segment or an image patch.

Below, Chapter 2.1 will review the foundations of incorporating invariance and equivariance into image recognition models. As the importance of local invariant feature detectors and descriptors such as HOG [26] or SIFT [95] has diminished in recent years, we will focus our discussion on invariant and equivariant learning-based image recognition models. Afterwards, Chapter 2.2 will give a more in-depth overview of approaches for encoding or learning invariant or equivariant *deep* neural network models, which have recently gained more interest in the computer vision research community. This part will also present the related works that we will later utilize as baseline approaches in our experiments.

2.1 Invariance in Machine Learning Models

In this part, we will present the foundations for our later deep dive into invariant deep models by looking at past ideas for incorporating invariance into machine learning algorithms. Here, we will also introduce convolutional neural networks and highlight how their invariance and equivariance properties played a key role in their recent success story which has made them the go-to solution for nearly any computer vision problem.

One approach for incorporating invariance into a machine learning model such as a neural network is to learn it from the training data. Given a suitable large enough labeled training dataset, a large enough model capacity and long or unlimited time for training a model can learn the desired transformation invariances from the data during training. However, in order for the training data to be suitable for learning a desired invariance, the training data must contain enough variation with respect to the transformation to which the model should learn to be invariant. As this is difficult to measure and ensure, **data augmentation** is commonly used to enlarge the training set with identity-preserving transformed copies of training samples [2, 3, 103, 128].

The use of data augmentation for learning invariance has the advantage of being very simple to implement, in particular for the task of image classification, where it usually only requires an augmentation of the input data and not of the corresponding labels, as well as being compatible with a wide variety of transformations. Yet, as reported by Simard et al. [127], the use of data augmentation has the disadvantage of extending the training dataset with samples that are heavily correlated with the original training samples. This in turn slows down the training of the model significantly as it makes learning algorithms such as backpropagation very inefficient. Furthermore, data augmentation cannot guarantee that the desired invariances are actually learned as a lack of model capacity or too short training time might result in the model not learning full invariance.

In order to place an additional soft constraint on the model to learn the desired invariance from the training data, **regularization** can be used. An example for an invariance inducing regularizer is Tangent Prop [127], which penalizes the derivatives of the classification function in the direction of the transformation to which the model should learn to be invariant. This is implemented by modifying the weight-update rule to include an additional regularization objective. Similar invariance inducing regularizers have also been proposed for Support Vector Machine (SVM) models [9, 124].

An advantage of the use regularization over data augmentation is that it enables a model to learn invariance from *fewer* rather than from more training samples, thereby effectively improving the data efficiency of the training. However, unlike data augmentation, it is not as simple to derive and implement as the influence of the regularization objective has to be carefully tuned with respect to the original learning objective. Besides, as with data augmentation, regularization cannot guarantee that the model learns the desired invariance. Therefore, the invariance properties of a model trained with data augmentation or regularization should ideally be measured and verified after training.

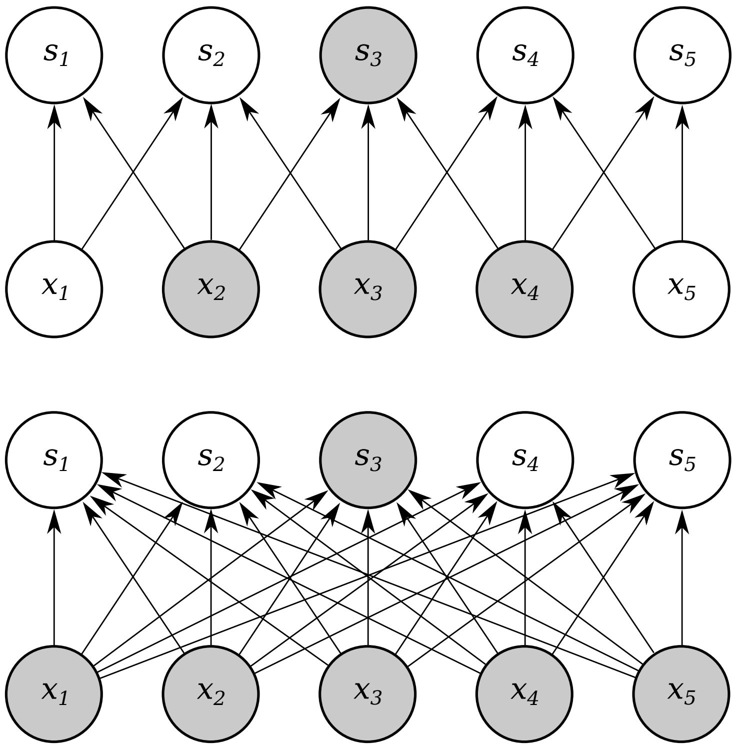


Figure 2.2: **Local Connectivity** of Convolutional Neural Networks. While a neuron s_3 in a convolutional layer is only connected to a local neighborhood of width 3 *(top)*, it is connected to all spatial locations of the input in a fully-connected layer *(bottom)*. Adapted from [48].

A third approach for incorporating invariance into machine learning systems are **model constraints**, that effectively *hard-code* the desired invariance into the architecture of the model. Thus, unlike data augmentation or regularization, which both aim to learn an (approximate) invariance from the training data, the use of model constraints can offer *invariance guarantees*. Yet, it may not always be trivial to develop an invariant model architecture, in particular if the model should not just be invariant to a single geometric transformation such as translations or rotations but should be invariant to a larger set of different and possibly more complex transformations of the input.

With regards to neural networks, a prominent example of encoding invariance and equivariance constraints into the architecture of the model are *Convolutional Neural Networks* (CNNs) [81], whose recent success in the field of computer vision can be partly explained by their built in invariance and equivariance properties.

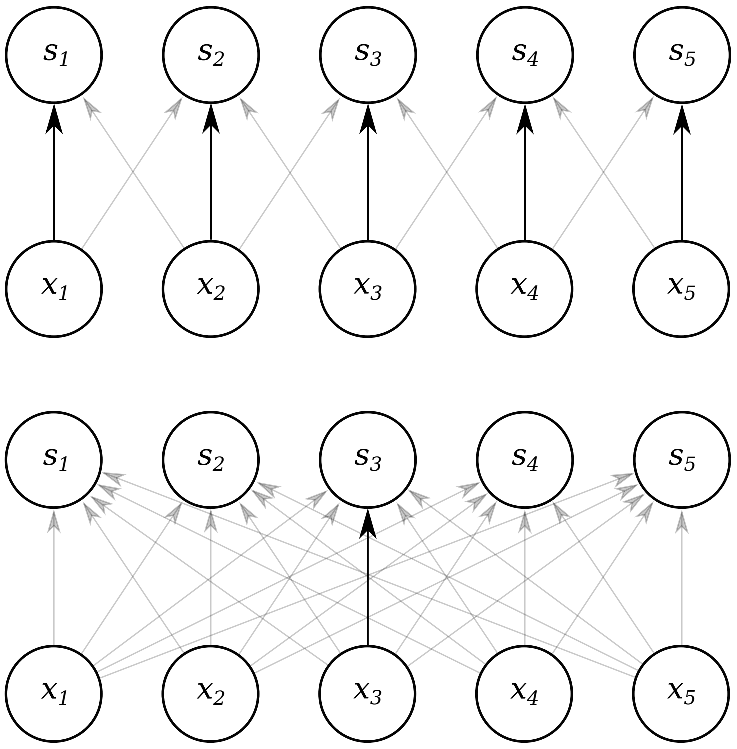


Figure 2.3: **Parameter Sharing** of Convolutional Neural Networks. Shared parameters are indicated by black arrows. While the convolutional model shares parameters between adjacent neurons *(top)*, a fully-connected model has no parameter sharing *(bottom)*. Adapted from [48].

Convolutional Neural Networks and their predecessor the NeoCognitron [38] are inspired by Hubel and Wiesel's discovery of a hierarchical model of simple and complex cells in the primary visual cortex of cats [60]. While simple cells respond to a stimulus within a localized *receptive field*, a small region of the visual field that the cell responds to, complex cells have a larger receptive field and exhibit spatial invariance in their output as they pool over the responses of a number of simple cells.

This structure is replicated in a Convolutional Neural Network: Similar to the model of Hubel and Wiesel, Convolutional Neural Networks also employ the concept of a local receptive field. Unlike in a fully-connected layer, in which each neuron is connected to all spatial locations of its input, each neuron in a convolutional layer is connected only to a small local region of its input (see Fig. 2.2). The extent of a neuron's *local connectivity* is a hyperparameter of the network. It is shared between all neurons in a given layer.

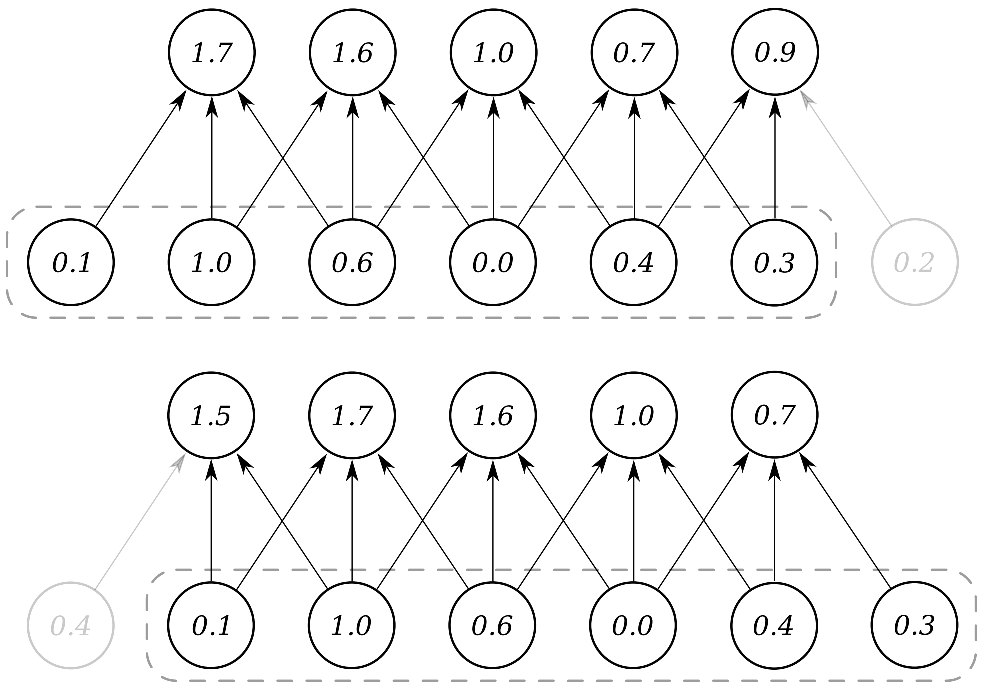


Figure 2.4: **Translation Equivariance** of Convolutional Neural Networks. The output of a convolutional layer, which is here visualized for a fixed convolutional kernel of width 3 with weights of 1 at every location, is equivariant to a translation of its input: As the input is shifted to the right by one pixel from *top* to *bottom*, the output of the convolutional layer is similarly shifted by one pixel to the right. Adapted from [48].

Neurons with the same local receptive field are stacked along the depth dimension. Spatially adjacent neurons in a convolutional layer can have overlapping local receptive fields. Here, the overlap is controlled by a *stride* hyperparameter which defines the fixed spatial shift between local receptive fields of adjacent neurons. The stride along with a layer's receptive field size and its spatial *padding* of the input along the input borders controls the spatial dimension of the output of a convolutional layer.

A given neuron has weights to every spatial location in its receptive field. Here, the set of a neuron's weights is referred to as a filter (or kernel). In order to keep the number of weights of the network low, these weights are shared between neurons at the same depth level of a convolutional layer (see Fig. 2.3). Thereby, these neurons detect the same features at all spatial locations of the input. As a consequence, the output of a convolutional layer is *translation equivariant*: When the input to a convolutional layer is translated, its output is also translated. This property is visualized in Fig. 2.4.

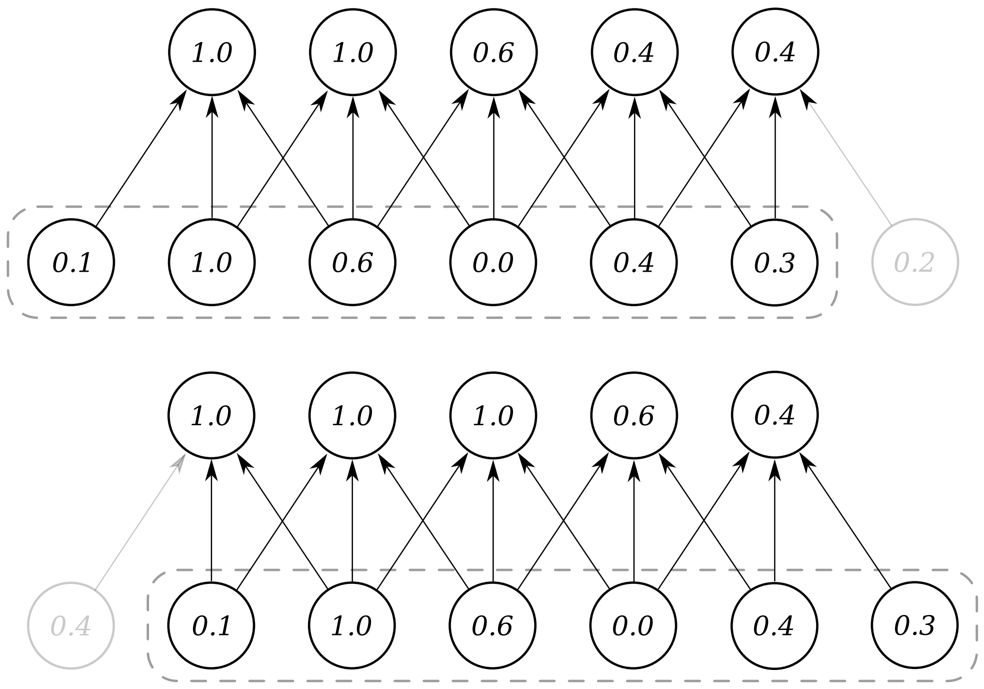


Figure 2.5: **Translation Invariance** of Convolutional Neural Networks. The output of a max-pooling layer is sensitive only to the maximum value in its neighborhood but not to its exact location. Therefore, shifting an input by one pixel to the right from *top* to *bottom* yields an unchanged output for more than half of the neurons. Adapted from [48].

Besides convolutional layers, CNNs commonly also utilize *pooling* layers, which are inserted between successive convolutional layers and which can additionally perform a downsampling operation. Commonly, max- or average-pooling is employed. The insertion of a pooling layer results in *translation invariance*: A small shift of the input to a pooling layer may not result in a changed output (see Fig. 2.5).

Similar concepts of encoding invariance properties have also been proposed by other machine learning models such as in scattering networks [6], which can be seen as a special case of convolutional neural networks. Similar to CNNs, scattering networks cascade convolutions with non-linearities. However, unlike CNNs, they do not learn the convolutional filters but instead utilize a fixed filter bank of scaled and rotated wavelet filters that are obtained from a mother wavelet (e.g. a complex Morlet wavelet). In each layer of the scattering network, this wavelet transform is preceded by an averaging and followed by a modulus non-linearity, which yields a representation that is invariant to small translations and stable to small deformations of the input.

Scattering networks have since been extended to invariance to rotations and scalings [108, 126]. Yet, as scattering networks involve no learning, they are typically combined with a trainable classifier such a Support Vector Machine (SVM). While they show promise in a very low data regime, in which very little labeled data is available for training, they are generally outperformed by modern convolutional neural networks which highlights the benefit of using learnable instead of fixed filters. This is also one reason why we focus our work in this thesis on incorporating or learning additional invariances in (deep) convolutional networks that offer state-of-the-art performance. Indeed, scattering networks themselves have recently been combined with deep convolutional neural networks in order to boost their performance, as proposed by Oyallon et al. [107].

Finally, it is important to note that unlike data augmentation and regularization, which usually aim for (approximate) invariance with respect to *global* transformations of the input data, for example by training with examples that have been globally transformed, the pooling layers in scattering and convolutional networks encode *local* transformation invariance. However, successive convolutional and pooling layers will increase the size of the local invariance region and a deep model of convolutional and pooling layers can thereby exhibit global translation invariance at the later network layers. Alternatively, a global pooling operation can be used to transform a locally equivariant or invariant representation into a globally invariant representation. Yet, as with the choice between invariance and equivariance, the choice between local or global invariance / equivariance is highly task-specific as tasks which do not require invariance or equivariance on a global level may still benefit from it on a local level.

2.2 Invariance and Equivariance in Deep Neural Networks

In this section, we will review recent works on extending the invariance and equivariance properties of deep convolutional neural networks from translations to a larger group of transformations. We start by first looking at approaches which aim at *learning* invariance or equivariance in Section 2.2.1. In a second part in Section 2.2.2 we then outline the most relevant works on *encoding* additional invariance or equivariance properties into deep neural networks. A final part in Section 2.2.3 focuses solely on deep *domain* invariance, in which approaches utilize regularization as well as an adaptation of the input data to learn invariance wrt. a change of the domain of the input data.

2.2.1 Learning Invariance / Equivariance in Deep Models.

Learning Invariance. While convolutional neural networks are by design equivariant to translations, Lenc & Vedaldi show that CNNs also learn approximate equivariance to other transformations in earlier layers of the network and approximate invariance in later layers of the network [83]. Similarly, Goodfellow et al. find that deep convolutional networks learn increasingly invariant features in each network layer, which suggests that depth is generally helpful for learning invariance [49]. However, the invariances which are learned by the network tend to be restricted to transformations which commonly occur in the training data (e.g. horizontal flips or rescaling), whereas there are few invariant features for unexpected or uncommon transformations such as vertical flips or 90° rotations [83].

Therefore, it is critical that the training dataset covers all of the transformations for which invariance is desired. Yet, as it is often not feasible to record and label a dataset that covers all desired transformations, data augmentation, which was introduced in the previous section, is commonly used in deep learning to artificially inflate the training data with transformed copies of the input images [75, 128]. Data augmentation is a simple but effective approach approach to incorporate high-level *a priori* knowledge about the desired invariances into the learning process of a deep network and is applicable to almost any identity-preserving transformation. However, as the transformed copies carry only little incremental information, data augmentation makes backpropagation inefficient and the training slow [127]. For this reason, several works have investigated how invariance can be more explicitly learned in deep neural networks [24, 55, 63, 76, 77, 102, 119, 156].

A well-studied method for learning spatial invariance in deep neural networks are *spatial transformer networks* (STNs) [63]. In STNs a spatial transformer module, effectively a small sub-network with its own learnable parameters, enables the model to actively spatially transform its input data before it is fed into the main network. Here, the spatial manipulation is conditioned on the input data itself and is trained end-to-end through backpropgation of the main network's loss to the spatial transformer. In order to apply the spatial transformation of the input in a differentiable way, bilinear sampling is used. Fig. 2.6 visualizes the structure of a spatial transformer network, in which a localization network outputs the spatial transformation parameters ω. These parameters are then used by a grid generator to create a sampling grid t_ω, which is applied to the input by a sampler component. Incorporating a spatial transformer thereby enables a model to select a region of interest in the input image and transform it to a canonical orientation or pose.

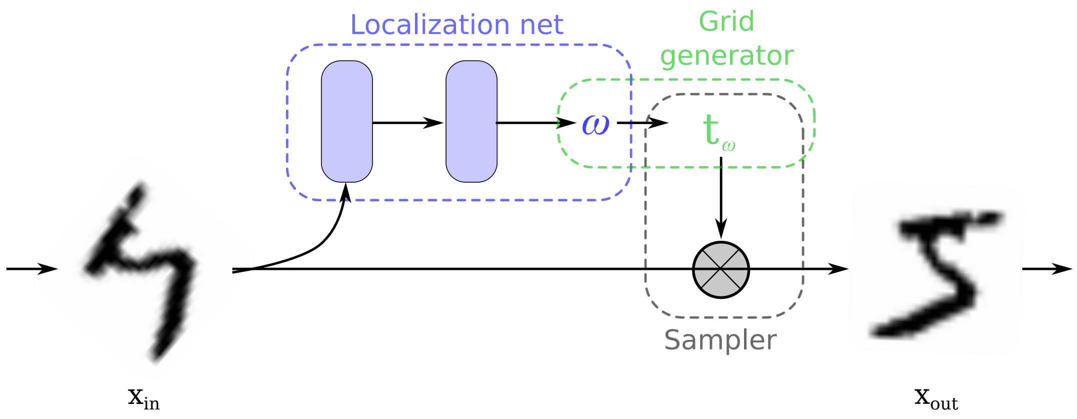


Figure 2.6: **Spatial Transformer**. A localization network predicts the transformation parameters ω conditioned on an input image x_{in}, based on which a grid generator creates a transformation t_ω that is then applied by a warper to output x_{out}. Adapted from [63].

Thereby, a spatial transformer can remove the variance wrt. spatial transformations from the input data so that the following task network is no longer required to learn invariance itself and the model as a whole becomes invariant. While spatial transformer networks have influenced a number of follow-up works [35, 84, 85, 138], they cannot offer invariance guarantees as they rely on the localization network to predict the correct transformation for a given input image. Yet, the localization network may not generalize to all input image transformation. Furthermore, STNs are limited to applying a single global transformation to the input, which makes them inapplicable for multi-object vision tasks such as object detection in which different objects may be transformed differently.

For this reason, Dai et al. present the idea of *deformable convolutional networks* (DCNs). Instead of performing a global transformation of the input data as in spatial transformer networks, DCNs locally transform intermediate feature maps of the network by adapting the sampling grid locations of the convolutional and pooling kernels [24]. The offsets to the regular sampling grid locations are predicted by an additional convolutional or fully-connected layer which output an offset field (see Fig. 2.7). By not predicting a global transformation but transforming the convolutional or pooling kernels locally, DCNs are thus able to apply different transformations to different feature map regions, which makes them suitable for complex, dense multi-object tasks. In fact, the authors show that integrating deformable convolutions into state-of-the-art semantic segmentation and object detection models significantly improves their performance.

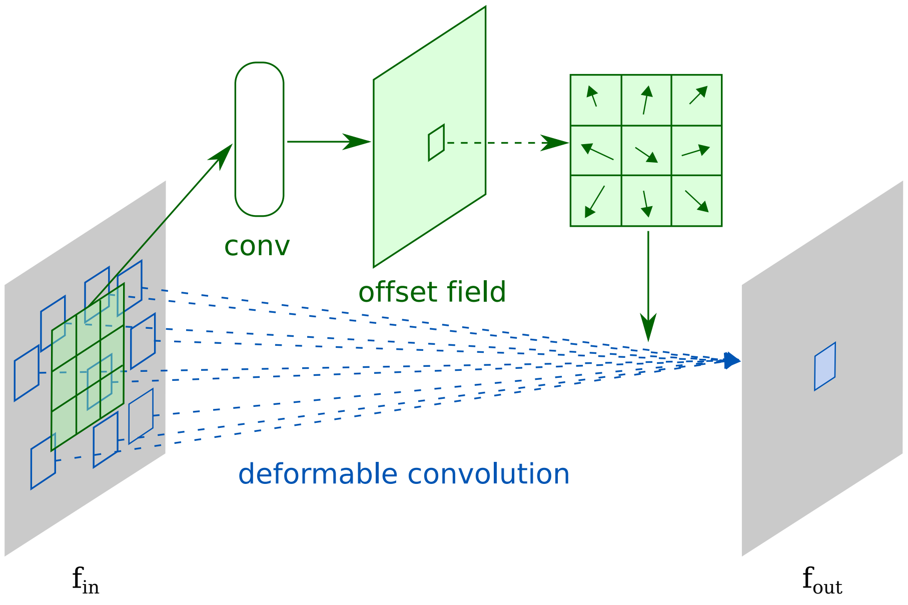


Figure 2.7: **Deformable Convolution**. A convolution layer predicts an offset field conditioned on an input feature map f_{in}. These offsets are added to the sampling grid locations of a deformable convolution layer, which produces the output f_{out}. Adapted from [24].

Yet, as in spatial transformer networks, deformable convolutional networks do not utilize prior knowledge about the transformations to which the network should be invariant. This means that the network itself has to learn from the training data which transformation invariance would be useful for the task at hand. However, this may differ from the desired invariance properties of the model. And as with other learning-based approaches, DCNs cannot guarantee that invariance is successfully learned during training.

Another approach which enables a network to learn invariance to unknown transformations are *tiled* convolutional neural networks (Tiled CNNs) [102]. Here, weight sharing in CNNs is adapted so that instead of adjacent units sharing their weights, only hidden units which are k steps away from each other share their weights. By pooling across the output of neighboring untied filters, this approach enables the network to learn additional local invariances to transformations such as scaling or rotation. The authors show that despite the reduction of weight sharing this idea improves model performance, in particular when only limited labeled training data is available. Yet, in contrast to spatial transformer and deformable convolutional networks, Tiled CNNs do not utilize an explicit representation of the transformation to which the network learns to be invariant. This makes an analysis into which invariances are actually learned by the network more complex.

Learning Equivariance. The use of pooling to achieve invariance in (tiled) convolutional neural networks however has the drawback that it loses information about the precise spatial relationships between high-level object parts in an image. This loss of local information can be problematic even for tasks which exhibit global translation invariance such image classification. For this reason, Hinton et al. [55] propose to replace the scalar output of a regular CNN layer with a more informative vector output of a *capsule*, which contains the probability of the presence of a visual entity as well as its instantiation parameters such as its pose. Importantly, these instantiation parameters are trained to be approximately equivariant wrt. so that a transformation of the object or object part is represented by a corresponding change of the instantiation parameters.

The concept of capsules is combined with a dynamic routing scheme by Sabour et al. [119], in which the pooling-based routing between layers of a CNN is replaced with a routing-by-agreement approach, in which a capsule sends its output to an appropriate parent capsule in the next layer. Intuitively, this dynamic routing approach aims at assigning object parts, which have been detected by a lower-level capsule, to the correct object whole in a higher-level level capsule. The authors show that a capsule network which utilizes dynamic routing achieves state-of-the-art results on MNIST and significantly improves upon a CNN baseline on an overlapping digit classification task.

What capsule networks are effectively solving is the *inverse graphics* task. While computer graphics renders an image from a hierarchical geometric representation of objects in the scene, inverse graphics or inverse rendering aims to deconstruct an image into its hierarchical parts and their instantiation parameters. This goal is shared with other recent works in the field of computer vision. However, unlike capsule networks, which adapt an equivariant representation in all layers of the network, these often only learn a single equivariant latent code in an encoder-decoder architecture, in which the encoder performs inverse rendering and the decoder performs re-rendering [76, 156].

An example for this line of research are the *Deep Convolution Inverse Graphics Networks* (DC-IGNs) by Kulkarni et al. [76]. Here, a variational autoencoder is trained to learn a disentangled equivariant latent code by presenting the network with mini-batches that only contain samples that vary wrt. a single variable during training. For example, a mini-batch might contain samples in which only the angle of a face changes but the face's identity remains unchanged. By keeping all but one component of the latent code constant, the encoder is then forced to represent the variation in the data with a single equivariant neuron while all other neurons in the latent code are trained for invariance.

In contrast to DC-IGNs, which learn an equivariant representation without supervision for the active transformation, Worrall et al. [156] impose explicit transformations on the latent code by training with transformed image pairs and their relative transformation vectors. Yet, this level of supervision is not commonly available in real-world datasets. In fact, even the creation of mini-batches which only vary wrt. a single transformation can be infeasible in real-world applications. Furthermore, both DC-IGNs as well as the approach by Worrall et al. focus on learning equivariance wrt. the transformations of a single object in an image, which is insufficient for complex, multi-object scenes.

An inverse-graphics approach for multi-object scenes was recently introduced by Kundu et al. [77]. The proposed *3D-RCNN* extends the popular R-CNN object detection framework [46, 52, 116] to not only predict an object's bounding box and class but also its 3D shape and pose. 3D-RCNN represents an object's shape with the help of shape priors based on a collection of CAD models. As ground truth shape and pose information are rarely available in real-world datasets, it also proposes a novel differentiable render-and-compare loss. Here, the predicted 3D shape and pose are first utilized to render segmentation masks and depth maps for each object in the scene, which are then compared with ground truth instance segmentation and depth map annotations. Thereby, more readily available 2D annotations can be utilized as supervision for the 3D shape and pose parameters of the model. The render-and-compare loss is used to fine-tune the network's pose and shape prediction after first training the network on a synthetic dataset that contains ground truth shape and pose information. In this way, 3D-RCNN achieves state-of-the-art results on challenging benchmarks such as Pascal3D+ and KITTI.

As mentioned at several points in this section, the presented learning-based approaches suffer from the drawback that they cannot offer any invariance or equivariance guarantees. Therefore, another direction of research is interested in explicitly encoding invariance or equivariance properties into the architecture of deep convolutional networks and thereby guaranteeing these properties by design. We review the most relevant recent works below.

2.2.2 Deep Invariant and Equivariant Architectures

Whereas the above presented learning-based approaches are often compatible with a set of different input transformations, methods that encode invariance or equivariance into the network architecture are usually restricted to a single specific transformation. We therefore structure this section wrt. the main transformation considered by each method.

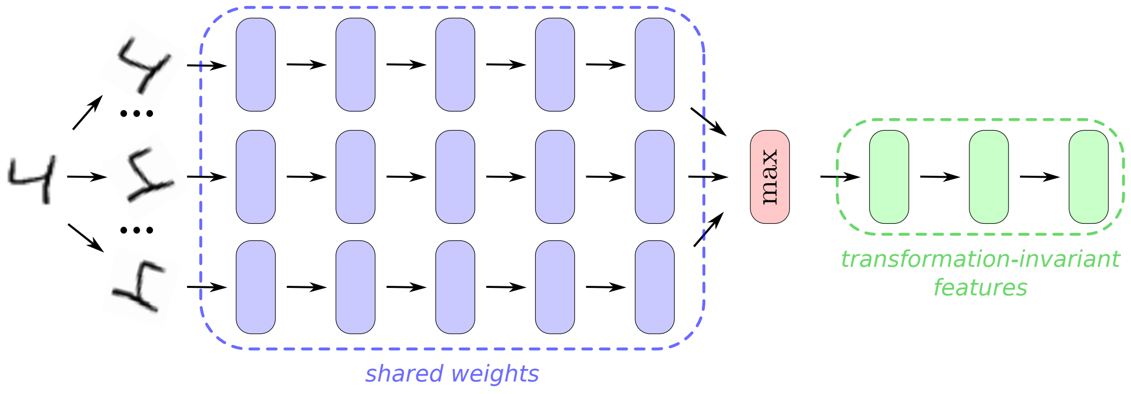


Figure 2.8: **TI-POOLING** feeds multiple transformed (e.g. rotated) copies of an input image into parallel Siamese network streams which share their weights. An element-wise max-pooling operation then obtains transformation-invariant features. Adapted from [79].

2D Rotation Invariance. For encoding invariance to 2D rotations into deep convolutional neural networks, Laptev et al. propose a network architecture which feeds multiple rotated copies of an input into a parallel Siamese network [79]. A novel *transformation-invariant pooling* (TI-POOLING) operator then pools over the feature maps of the transformed input copies and selects an optimal canonical instance of the input to be passed on to the subsequent fully-connected network layers (see Fig. 2.8). Similarly, Fasel & Gatica-Perez propose an architecture in which a virtual input image generator creates multiple transformed copies of an input which are fed into a CNN with shared feature maps [37]. By finding a canonical orientation of the input image, these approaches share similarities with spatial transformer networks (STNs). Yet, while they have a computational overhead at inference time compared to STNs due to multiple transformed copies being passed through the network, their restriction to a specific set of transformations, which do not have to be learned, results in a simpler model with shorter training time. However, as with spatial transformers, a limitation of approaches which are based on augmenting the input data is that they can only model invariance wrt. a single global transformation.

An alternative to creating rotated copies of an input, is to create a filter bank which consists of rotated copies of one canonical filter which all share their weights [99]. Rotation invariance is then again achieved by using an orientation pooling layer which pools over the responses of the multiple rotated versions of a single filter. In contrast to approaches such as TI-POOLING, which are based on a rotation of the input data, methods that rotate the filters of the network instead model invariance wrt. local rotations. The authors show that incorporating local invariance can improve the performance of a CNN compared to training with data augmentation on a texture classification task. Yet, in many applications besides classification it is beneficial not to model rotation invariance but equivariance, in particular for tasks that require an estimation of pose or orientation.

2D Rotation Equivariance. One approach for achieving rotation equivariance in convolutional neural networks is to transform their translation equivariance into equivariance wrt. rotations by converting rotations into translation via a log-polar transform [35, 54]. While warped convolutions by Henriques & Vedaldi [54] achieve this by warping the inputs to a fixed grid with a fixed polar origin, Esteves et al. first identify the object center using a polar origin predictor before transforming the input into log-polar coordinates [35]. The convolutional classification networks in both models remain unchanged but now operate on polar images. While these approaches achieve global rotational equivariance, they break the translation equivariance property of CNNs which is problematic for tasks that require a regression of object locations such as object detection.

Therefore, other approaches aim to achieve rotation equivariance in convolutional neural networks in addition to translation equivariance. An example are the *Group Equivariant Convolutional Neural Networks* (G-CNNs) proposed by Cohen & Welling [18]. Here, regular convolutions, which shift a set of filters over the input, are replaced with *group convolutions* (G-convolutions), that replace the shift transformation with a more general transformation of the filters from a discrete symmetry group G. Specifically, they consider the symmetry group $G = p4$, which consists of all translations and rotations by 90°, and the group $G = p4m$, which in addition also contains mirror reflections.

Their G-convolutions are implemented by first creating an augmented filter bank. For this, the convolutional filters are transformed with rotations, in the case of $G = p4$, or, in the case of $G = p4m$, with rotations and flips. The augmented filter bank is then applied by using regular planar convolutions. While the output of a single G-convolutional layer is equivariant wrt. the transformations of the considered symmetry group G, a subsequent max-pooling operation over the rotations or rotation-flips results in a locally invariant and globally equivariant representation. This property is visualized in Fig. 2.9 for a G-CNN consisting of a single G-convolution followed by max-pooling. The authors show that by extending weight sharing to rotated filter copies, they can not only improve performance and data efficiency on a task which is explicitly rotation invariant (rotated-MNIST) but also on a dataset of natural images that does not feature explicit image rotations (CIFAR10).

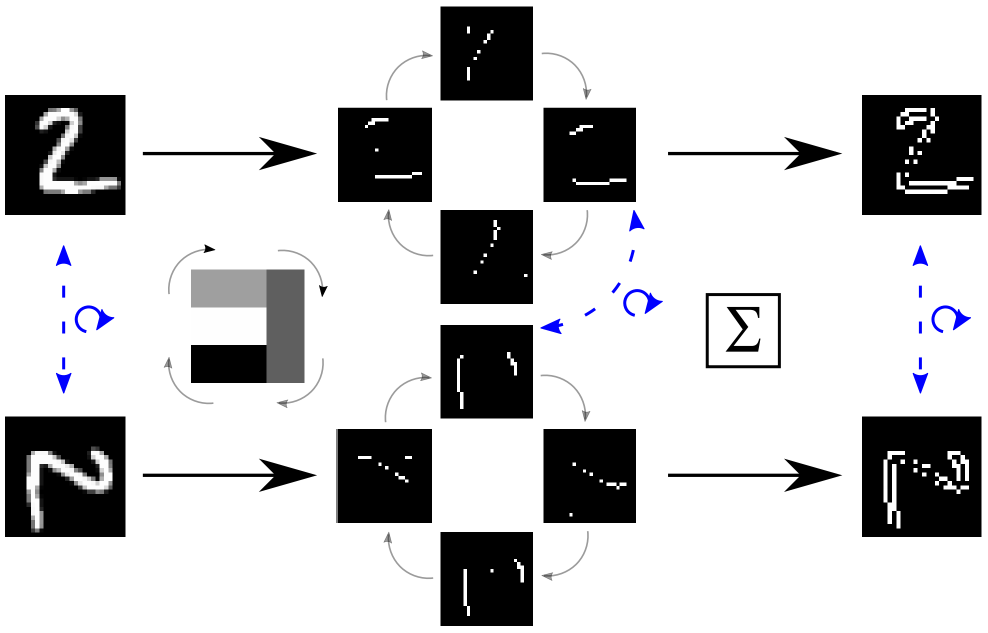


Figure 2.9: **Group Equivariant CNNs (G-CNNs)**. G-CNNs extend the equivariance properties of CNNs to additional symmetry groups such as $G = p4$, which consists of rotations by 90°. For this, a G-CNN applies its convolution with 4 rotated versions of the same kernel. Max-pooling over the orientations then produces a locally rotation invariant and globally rotation equivariant representation. Adapted from [150].

A similar idea for extending CNNs to cyclic symmetry was also proposed by Dieleman et al. [29]. However, instead of rotating the convolutional filters, they rotate the feature maps, which adds the requirement that the input to the network has to be square. As in G-CNNs, they also restrict themselves to rotations by multiples of 90° to avoid interpolation.

With regards to equivariance to non-90° rotations, *Oriented Response Networks* (ORNs) by Zhou et al. [168] and *Rotation Equivariant VectorField Networks* (RotEqNets) by Marcos et al. [98] use bilinear interpolation to rotate the filters by non-multiples of 90°. However, with an increase in considered rotation angles, the dimension of the resulting feature maps increases accordingly. In order to keep a compact representation, RotEqNets therefore directly pool over rotations after every convolutional layer and only pass forward the maximum feature response (*max*) and its orientation (*argmax*). This is in contrast to ORNs, which perform pooling over orientations only after the final convolutional layer.

Yet, the use of bilinear interpolation can be problematic as it introduces interpolation artifacts which can harm model performance. A solution to perform non-90° rotations without bilinear interpolation is presented by *Steerable Filter CNNs* (SFCNNs) [152].

SFCNNs build on G-CNNs but extend them to arbitrary filter orientations by using steerable filters. These steerable filters are defined as linear combinations of elementary circular harmonics filters, where the complex-valued weight of each elementary filter is learned by the network during training. By manipulating the phase of the learned weights, an exact rotation of the learned filters by a chosen number of orientations is performed. The authors find that performance saturates when around 12 orientations are used.

A generalization to patch-wise equivariance wrt. continuous 360° rotations is presented by *Harmonic Networks* (H-Nets) [155]. Similar to SFCNNs, H-Nets also utilize circular harmonics but instead of learning a linear combination of circular harmonics, they directly restrict the filters to be from the circular harmonics family. During training, H-Nets then learn the radial profile and phase offset terms of the circular harmonics filters, which control their shape and orientation. As the phase of a circular harmonics filter rotates with a rotation of the input, its filter response is continuously rotationally equivariant. Yet, despite their improved equivariance properties, H-Nets do not reach the performance of RotEqNets or SFCNNs as their circular harmonics filters appear to lack discriminativity.

3D Rotation Equivariance. Several of the above presented ideas for encoding 2D rotation equivariance have recently been extended to equivariance wrt. rotations in 3D.

For voxel-based representations, *CubeNet* by Worrall et al. [154] uses Klein's four group (*Klein's Vierergruppe*) to extend G-CNNs to 3D roto-translation equivariance. While the cube group S_4 contains all 24 right-angle rotations of a cube, Klein's four group K_4 is limited to an identity element plus three 180° rotations around each coordinate axes, which CubeNet applies in order to create an augmented filter bank of transformed convolutional filters. Similarly, Winkels & Cohen [153] propose *3D G-CNNs* which also build on group-convolutions but utilize different symmetry groups (D_4, D_{4h}, O or O_h). Yet, while both CubeNet and 3D G-CNNs demonstrate improved data efficiency and performance, they are limited to equivariance wrt. a discrete set of rotations.

In contrast, *3D Steerable CNNs* [151], *Tensor Field Networks* (TFNs) [141] and *N-body Networks* (NBNs) [71] achieve full rotation equivariance to the continuous SO(3) group. However, the way in which they achieve this goal differs somewhat. 3D Steerable CNNs adapt H-Nets for continuous equivariance to 3D rotations in voxel grids by replacing the circular harmonics filters, which are used in H-Nets, with convolutional filters that are restricted to the spherical harmonics family. Similarly, TFNs and NBNs also build on H-Nets and restrict the convolutional filters to spherical harmonics but operate on point clouds instead of voxel grids.

Spherical harmonics are also utilized by *Spherical CNNs* in order to learn SO(3) equivariant representations from spherical images [16, 34]. An alternative to Spherical CNNs for processing spherical images are the recently proposed *Gauge Equivariant CNNs* and *Icosahedral CNNs* [15] which approximate spherical signals with signals on the surface of an icosahedron. Unlike Spherical CNNs, they utilize the regular *conv2d* convolution operator and thereby benefit from its highly optimized implementation in modern deep learning frameworks. As a result, the computational cost and memory requirement of running an Icosahedral CNN is significantly lower than the cost and memory usage of running a Spherical CNN, especially at higher input resolutions.

Scale Invariance & Equivariance. Beside the multitude of works on encoding 2D and 3D rotation invariance or equivariance into deep neural networks, some works have recently also considered incorporating invariance or equivariance to *scale* variations.

As with rotation invariance, local scale invariance can be achieved by applying filters at multiple scales and then max-pooling over the scale channel of the feature responses [66]. However, this only retains the maximum magnitude but discards the information about the scale at which it occurred. Thus, Marcos et al. [97] utilize the vector field representation of RotEqNets [98] to additionally encode the scale of the maximum feature response. They thereby obtain a locally scale-equivariant network and demonstrate performance improvements on scale-invariant classification and a scale factor regression task.

A further improvement was proposed by Ghosh & Gupta [44] who replace the regular convolutional kernels with scale-steered kernels that use a log-radial harmonics filter basis. In contrast to [66] and [97], which reshape an input before and after applying a convolution, scale-steerable filters require no reshaping and thereby avoid interpolation artifacts. Consequently, they outperform the models by Kanazawa et al. [66] and Marcos et al. [97] on a scale-invariant classification task. However, the work by Ghosh & Gupta only focuses on scale invariance but does not consider the case of scale equivariance.

In contrast, *deep scale-spaces* (DSS) introduced by Worrall & Welling [157] implement discrete scale equivariance with the help of dilated convolutions. In order to avoid the need for interpolation, they restrict themselves to integer dilations. Their results demonstrate a large improvement in semantic segmentation performance on the challenging CityScapes benchmark [23] compared to a regular CNN model.

Other Invariances. Other invariances which have been considered in deep neural networks include permutation invariance, e.g. in the seminal work of PointNet [113], as well as domain invariance, which we will discuss in the following section.

2.2.3 Deep Domain Invariance

Domain invariance is the focus of works in the field of *domain adaptation* (DA). Due to the high relevance of domain adaptation techniques for later parts of this thesis, in particular Chapters 3 and 5, we introduce the main DA concepts in detail in this section.

Domain adaptation deals with the problem of adapting a model from a *source domain*, in which a large number of labeled training samples is available, to a *target domain*, which is visually different from the source domain and in which little or no labeled training data is available. While the former is referred to as the *supervised* scenario, the latter is referred to as the *unsupervised* domain adaptation scenario, which will be the focus of this section.

The need for performing domain adaptation arises as a model which has been trained on source domain data will not perform well on target domain data if the source and target domain data are not drawn from the same distribution and the two domains are too visually different. This is for example the case for a model that has been trained on synthetic data from simulation which will not perform well on real-world samples [57, 94, 112, 136]. While it is expensive and time-consuming to enable the training of a target domain model by creating a new large labeled target domain dataset, it often requires little effort to record some unlabeled target domain samples and, potentially, label a small set of them.

For supervised domain adaptation, the simplest approach is to fine-tune a model that was trained on source domain data on the limited target domain data which is available [45, 106]. However, this may result in overfitting when only few labeled target samples are available and is not applicable in the unsupervised case in which no labeled samples are available in the target domain. Thus, it is essential to utilize available unlabeled samples in the target domain in addition to the labeled samples from the source domain.

In the next sections we will review two classes of approaches for utilizing labeled source domain data in addition to unlabeled target domain data to perform an unsupervised domain adaptation in deep neural networks. The first class of approaches performs a *feature-level* adaptation [39, 40, 91, 92, 147–149]. Feature-level methods aim at learning features which are invariant to the domain shift from the source to the target domain data. The second class of approaches are *image-level* approaches [4, 56, 125] which do not focus on an adaptation of the features of a model but rather on an adaptation of the training data by translating the source domain images to the style of the target domain.

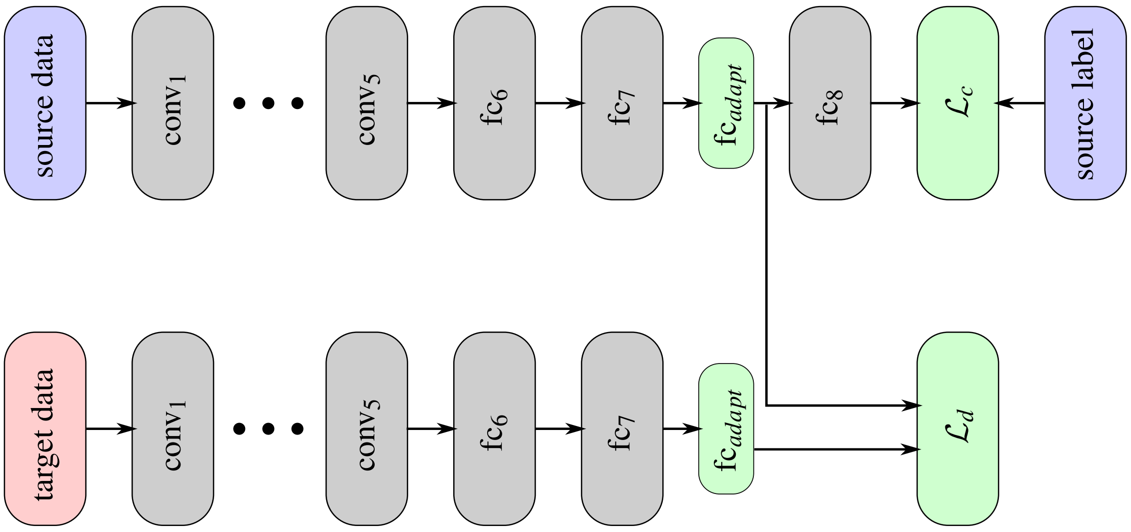


Figure 2.10: **Feature-level Adaptation with Domain-Confusion Loss.** Feature-level adaptation is performed by inserting a domain confusion loss $\mathcal{L}_{domain}$ into a two-stream classification network architecture. Adapted from [149].

2.2.3.1 Feature-Level Adaptation

Feature-level domain adaptation approaches [39, 40, 91, 92, 147–149] aim at aligning the feature distributions of the source and target domain via an additional loss term.

One class of feature-level approaches performs feature alignment by inserting an additional *domain confusion* loss into a two-stream network architecture in which the source and target network streams share their weights (see Fig. 2.10). The domain confusion loss aims at minimizing the distance between source and target distributions and is most widely based on Maximum Mean Discrepancy (MMD) [91, 92, 149]. For a source dataset X_S, a target dataset X_T and a feature representation $\phi(\cdot)$, which is the feature map of the chosen adaptation layer fc_{adapt}, the MMD metric is empirically estimated as follows:

$$MMD(X_S, X_T) = \left|\left| \frac{1}{|X_S|} \sum_{x_S \in X_S} \phi(x_S) - \frac{1}{|X_T|} \sum_{x_T \in X_T} \phi(x_T) \right|\right| \tag{2.4}$$

The network is then trained with a combination of a classification loss $\mathcal{L}_c(X_S, Y_S)$, which is applied on source domain data X_S with corresponding labels Y_S, and the domain confusion loss $\mathcal{L}_d(X_S, X_T) = \lambda MMD^2(X_S, X_T)$ with weight parameter λ. Other works also utilize a domain confusion loss but instead minimize the difference in higher order feature statistics [73, 135, 137] or replace MMD with a Joint Distribution Discrepancy [93].

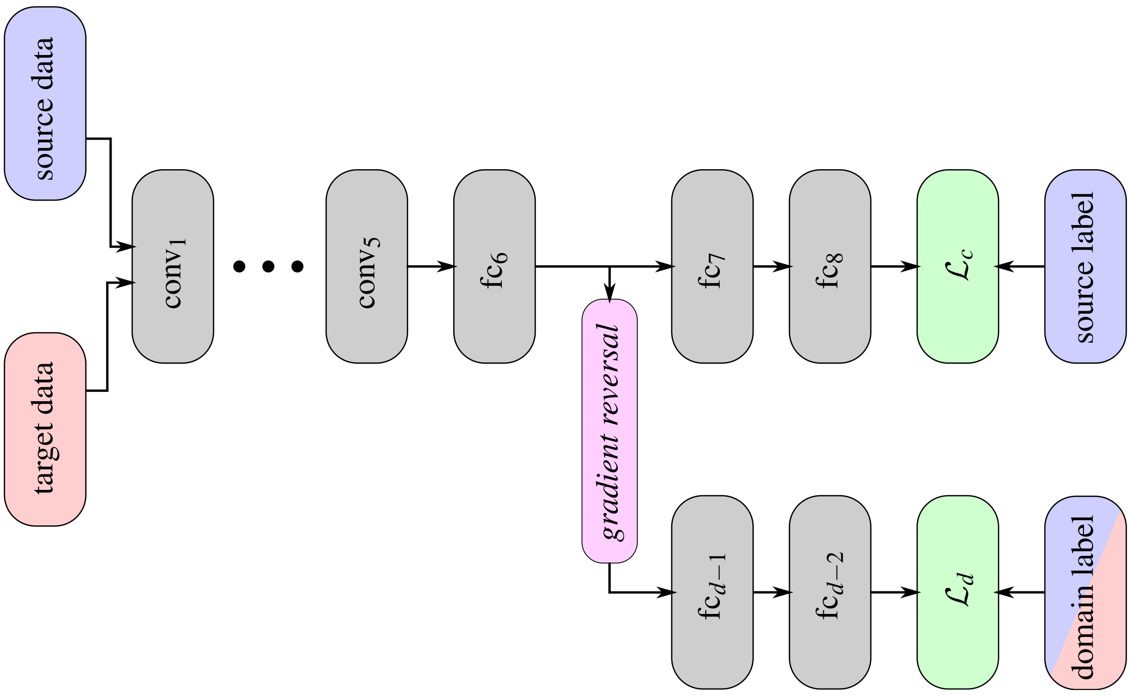


Figure 2.11: **Feature-level Adaptation with Domain-Adversarial Training.** Feature-level adaptation is performed by adding a domain classifier which is connected to the feature extraction network by a gradient reversal layer. Adapted from [39].

Alternatively, the domain invariant features can be learned by *domain-adversarial* training (see Fig. 2.11) [39,40,147,148]. Here, an additional domain classifier is attached to the shared feature extraction part of the classification network. This domain classifier which is parameterized by a sub-network with weights W_d is tasked with predicting the domain of each input sample given a representation vector $\phi(\cdot)$ from the classifier's feature extraction network. It is trained with a regular cross-entropy classification loss:

$$\mathcal{L}_d = \sum_{i=0}^{N_S+N_T} \Big\{ d_i \log \hat{d_i} + (1-d_i)\log(1-\hat{d_i}) \Big\} \tag{2.5}$$

where $\hat{d_i} \in \{0,1\}$ is the predicted and d_i the ground truth domain label for a sample $x_i \in X_S \cup X_T$ and N_S, N_T are the number of source and target samples, respectively.

Importantly, the domain classifier is attached to the classifier's feature extractor via a gradient reversal layer (GRL). While the GRL acts as an identity transform during forward propagation, it reverses the domain classifier's gradients by multiplying them with a negative value -λ before passing them on to the classifier's feature extractor during backpropagation. As a consequence, the binomial cross-entropy for the domain classification task is thus maximized wrt. the weights of the domain classifier W_d while it is minimized wrt. the weights of the classifier's feature extractor W_c.

While the domain classifier thus over the course of training learns to better discriminate between source and target domain samples, the feature extraction network is adapted in a way so that its features become more and more domain invariant, which in turn makes it more and more difficult for the domain classifier to predict the domain of the input. When training the full model in combination with a classification loss, the domain adversarial training setup is therefore an effective way to learn features which are discriminative with respect to the class of the input but also invariant with respect to the input's domain.

However, aiming for domain invariance is not always optimal but can in fact be detrimental to the discriminative power of a model [117]. The reason for this is that a domain invariant model may ignore important individual characteristics of each domain, in particular wrt. a pixel-level appearance shift between source and target domain data. Thus, some works [5, 117] propose to also allow a model to learn the differences between the source and target domain. Yet, these approaches are still based on an alignment on feature level and do not enforce any semantic consistency, which means that the target features of one class may be mapped to the source features of a different class. Therefore, an alternative class of approaches considers a semantically-consistent *image-level* adaptation.

2.2.3.2 Image-Level Adaptation

Image-level domain adaptation methods [4, 56, 125] perform a distribution alignment between the source and target domain in image space by translating source domain images to the target domain. In contrast to feature-level approaches, they thereby offer a more humanly interpretable adaptation, as they enable a visualization of the individual steps as well as the quality of the adaptation from the source to the target domain.

Image-level adaptation is commonly performed with a variant of *Generative Adversarial Networks* (GANs) [50]. In GANs, an *adversarial* loss enables the generation of images by a generator network G which are indistinguishable from real images of a target domain for a discriminator network D. For performing domain adaptation, we typically want to condition this image generation on a source domain image. *Conditional GANs* [100] have shown promising performance for image-to-image translation tasks [62] but require source-target image pairs which are often not available in domain adaptation tasks.

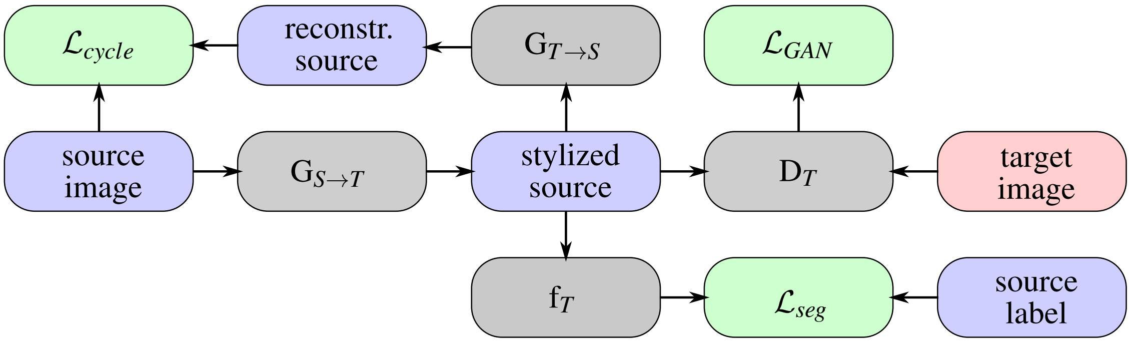


Figure 2.12: **Cycle-Consistent Image-level Adaptation.** Image-level adaptation is performed by training a CycleGAN with generators $G_{S\to T}$ and $G_{T\to S}$ and a discriminator D_T using a cycle-consistency loss $\mathcal{L}_{cycle}$ and an adversarial loss $\mathcal{L}_{GAN}$. The stylized source images and original source labels are then used to train a task segmentation network f_T with a segmentation loss $\mathcal{L}_{seg}$. Adapted and simplified from [56].

In the unpaired setting, Liu et al. [88] propose an unsupervised image-to-image translation framework based on *Coupled GANs* (CoGANs) [87] where consistency between the two domains is enforced with weight sharing without a need for corresponding images from the source and target domain. Other works [4, 125, 139] ensure a consistent unpaired translation with an additional loss term that encourages similarity in a predefined metric space, such as a content space [4], image space [125] or feature embedding space [139].

In contrast, *Cycle-Consistent Adversarial Networks* (CycleGANs) [169] offer a solution for performing unpaired image-to-image translation without weight sharing or a predefined similarity function. Instead, CycleGANs utilize a cycle-consistency loss which penalizes differences between an original source image and a reconstruction of the source image after it has been mapped to the target domain by a generator $G_{S\to T}$ and subsequently mapped back to the source domain by a second generator $G_{T\to S}$.

CycleGANs are successfully utilized for performing domain adaptation in the recently proposed *Cycle-Consistent Adversarial Domain Adaptation* (CyCADA) model [56], which is visualized in Fig. 2.12. CyCADA trains a CycleGAN in order to translate the source domain images to the target domain and then utilizes the translated source domain images in combination with the corresponding original source domain labels to train a task semantic segmentation network f_T, which performs well on real target domain images. However, the use of a CycleGAN model by CyCADA means that two generator networks have to be trained, of which only one is utilized to translate the source domain data to the target domain during the actual adaptation process.

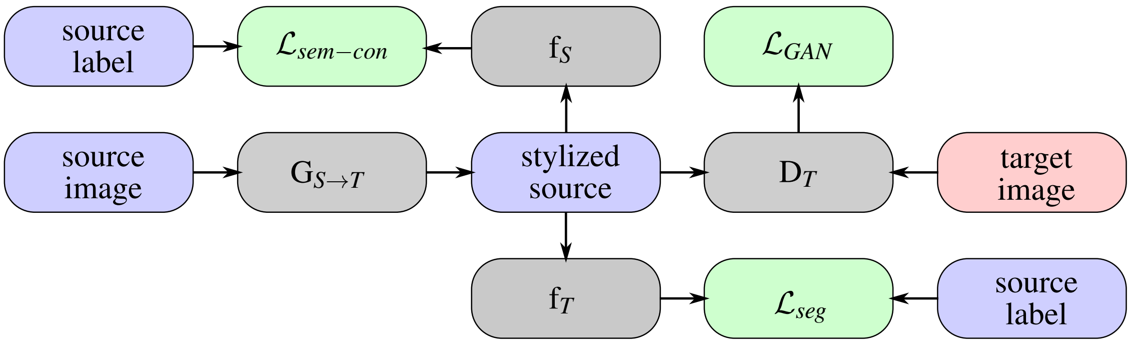


Figure 2.13: **Cycle-Free Semantically-Consistent Image-level Adaptation.** As an alternative to cycle-consistency, a semantic consistency loss $\mathcal{L}_{sem-con}$ can be applied which is computed between the output of a source segmentation model f_S, which is trained on source data data, and the source segmentation label. This encourages the generator $G_{S\to T}$ to output images which are not only visually close to the target domain, as ensured by the adversarial loss $\mathcal{L}_{GAN}$ but also consistent with the semantics of the original source image. Adapted and simplified from [146].

An alternative to the use of a CycleGAN model has been proposed by Tzeng et al. with their Semantic Pixel-Level Adaptation Transform (SPLAT) approach [146]. Instead of the use of cycle-consistency, SPLAT exploits a semantic consistency loss $\mathcal{L}_{sem-con}$ to ensure the consistency between the semantics of a source image and a corresponding stylized source image (see Fig. 2.13). . For this, the semantic consistency loss penalizes differences between the source segmentation label and the predictions of a semantic segmentation network f_S, that has been trained on labeled source data, for a stylized source image. As in CyCADA, SPLAT then utilizes the trained generator $G_{S\to T}$ to translate the source images to the target domain in order to train a task segmentation model f_T.

While current state-of-the-art image-level domain adaptation such as CyCADA or SPLAT work well for changes in image style such as adapting from synthetic to real-world images [56, 146], they struggle with changes that require an understanding of the scene geometry, which is the case when adapting to a different viewpoint. However, before we will look into the viewpoint adaptation problem in Section 3, we will first present an approach for learning transformation invariant representations with weak supervision.

Chapter 3

Learning Transformation Invariant Representations with Weak Supervision

Training image classifiers which are robust to a specific set of geometric transformations of the input is a central problem in computer vision. In this chapter, we investigate how a novel unsupervised similarity loss can be used to *learn* invariance with respect to a variety of geometric transformations of the input.

The most commonly used approach to ensure the robustness of a deep classifier model to geometric transformations is data augmentation [75, 79, 128], in which the size of the training set is artificially increased by augmentation (e.g. rotation, cropping or scaling) of the training images. Data augmentation is very simple to implement and is compatible with a large set of different input transformations. Furthermore, it also acts as a regularizer, which prevents a classifier from overfitting to the training set. Yet, data augmentation comes with the drawback that it does not explicitly enforce transformation invariance and that it slows down training significantly as it makes backpropagation inefficient due to the transformed samples being highly correlated with the original input samples.

An alternative to learning invariance by data augmentation is to utilize network architectures which aim at transforming the input or intermediary feature maps of the model in order to remove variance from the data. Examples for such resampling-based approaches include spatial transformer networks (STNs) [63], which perform a global spatial transformation of the input, and deformable convolutional networks (DCNs) [24], which learn offsets to the regular sampling grid of the convolutional filters. However, while these approaches increase the flexibility of a network in handling geometric transformations, they assume that a canonical representation can be easily deduced from the input. Furthermore, unlike our approach, they do not support the utilization of available prior knowledge on which invariances would be helpful for a given task.

In contrast to learning-based approaches, several recent works propose to encode invariances into the architecture of deep convolutional neural networks (CNNs) [18,70,130, 155, 168]. However, these methods are currently restricted to specific and comparatively simple transformations such as rotations or flips and are non-trivial to extend. What is more, they do not always build on regular convolutions and thus cannot necessarily utilize the highly optimized implementation of the regular convolution operator in modern deep learning frameworks. This also means that, unlike our proposed method, they cannot necessarily build on the latest CNN architecture or training innovations.

In this chapter, we propose to leverage a novel unsupervised similarity loss for training deep neural networks to learn invariance wrt. arbitrary geometric transformations. The similarity loss is computed with respect to transformed copies of an input and presents an effective regularizer to enforce the desired transformation invariances. As it builds on data augmentation, the similarly loss is simple to implement for a large set of different input transformation. Yet, in contrast to naïve data augmentation, it explicitly enforces transformation invariance and encourages smooth decision boundaries with respect to transformations of the input. This makes the training more efficient and leads to improved classification performance, in particular in the presence of little annotated examples.

Furthermore, unlike prior ideas for learning invariance, it also enables the easy incorporation of additional unlabeled examples, as the similarity loss does not require label information and is thus suitable for enabling semi-supervised learning.

Our similarity loss shares some similarities with ideas from transformation-invariant pooling (TI-POOLING) [79], which also utilizes multiple transformed copies of an input but pools over the responses of the augmented input copies. This means that inference time grows exponentially with the dimension of the transformation, thereby potentially rendering this approach infeasible for real-time applications. In contrast, our loss presents a soft constraint for learning invariance which does not affect test time performance.

The idea of using an input sample more than once in a forward pass has previously also been proposed for protecting neural networks against adversarial perturbations [101, 163]. We similarly aim to improve model generalization but focus on transformation invariance by enforcing similarity between feature representations of transformed input images.

The proposed approach is also related to prior work on self-supervised [1, 30, 104] and semi-supervised learning [51, 121], in which similar auxiliary losses are utilized. However, in contrast to these works, we directly optimize the desired loss metric and utilize a similarity loss for learning *transformation invariant* representations.

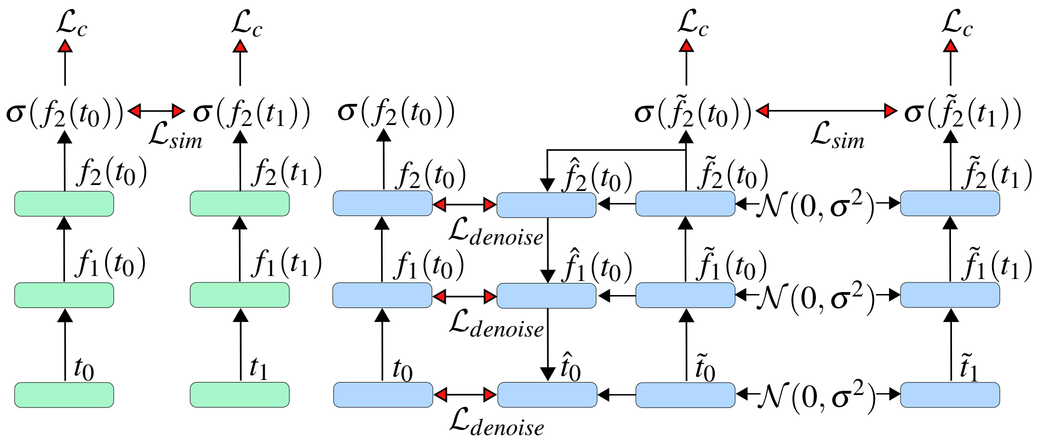


Figure 3.1: Abstract network architectures considered in this work. A similarity loss $\mathcal{L}_{sim}$ is placed on the the final layer of a Siamese (Fig. 3.1a) or ladder network (Fig. 3.1b) in order to enforce similarity between the outputs of transformed copies of an input.

In detail, the **contributions** of this chapter are:

- We propose a similarity loss which acts as an additional regularizer and utilizes unlabeled training data for learning transformation invariance.
- We present a detailed investigation on the weighting and placement of the loss.
- We show improved performance in supervised and semi-supervised learning on rotated MNIST and GTSRB when little labeled data is available.

3.1 Method

While the proposed similarity loss is applicable to a variety of tasks, we use image classification as a test bench in this work. Let $x \in \mathbb{R}^{w\times h\times c}$ denote an image of dimensions $w \times h$ with c channels and let $f : \mathbb{R}^{w\times h\times c} \to \mathbb{R}^C$ be a non-linear mapping represented by a neural network which takes an input image x and produces a score for each of the C classes. Let further $t_0, t_1 \in \mathcal{T}$ denote two transformations from a set of transformations $\mathcal{T}$ (*e.g*, rotation, affine, perspective) which take the input image x and produce transformed versions $t_0(x) \in \mathbb{R}^{w\times h\times c}$ and $t_1(x) \in \mathbb{R}^{w\times h\times c}$ of it. Finally, let $f_l(t(x))$ denote the feature maps of the neural network in layer l when passing $t(x)$ as input. For clarity, we will drop the dependency on the input image x in the following.

In order to encourage a deep neural network to learn transformation invariant representations, we propose the use of a similarity loss $\mathcal{L}_{sim}$ which penalizes large distances between the predictions or feature embeddings of transformed copies of the input. In its simplest form, the similarity loss is computed using a Siamese network architecture, where the transformed copies of the input are simultaneously fed into separate streams of the network that share their weights. By transforming both inputs, convergence of the model is accelerated and overfitting to small label sets is avoided. An abstract network architecture, where $\mathcal{L}_{sim}$ is applied at the final network layer L, is illustrated in Fig. 3.1a.

At inference time only a single stream of the network is used, keeping runtime constant with respect to the size of the transformation space. For training, the similarity loss $\mathcal{L}_{sim}$ is added to the supervised classification loss $\mathcal{L}_c$, which is applied on the output of both network streams, to form the total loss $\mathcal{L}_{total}$ where a weight parameter λ controls the influence of $\mathcal{L}_{sim}$:

$$\mathcal{L}_{total} = \mathcal{L}_c + \lambda \mathcal{L}_{sim} \tag{3.1}$$

Here, $\mathcal{L}_c$ is the usual cross-entropy loss which is applied to the softmax outputs $\sigma(f_L(t_0))$ and $\sigma(f_L(t_1))$ of the final network layer L for the transformed input sample:

$$\mathcal{L}_c = -\sum_{i=1}^{C} y_i \log \sigma_i(f_L(t_0)) - \sum_{i=1}^{C} y_i \log \sigma_i(f_L(t_1)) \tag{3.2}$$

where $\sigma_i(x) = \exp(x_i) / \sum_{j=1}^{C} \exp(x_j)$ denotes the softmax function and where $y_i = 1$ if i is the ground truth class and $y_i = 0$ otherwise.

The similarity loss encourages the output of the neural network at layer l, f_l, to be similar for both streams. It is defined as the distance between the outputs of the transformed input pair at layer l for an appropriate distance metric $\mathcal{D}(\cdot,\cdot)$:

$$\mathcal{L}_{sim} = \mathcal{D}(f_l(t_0), f_l(t_1)) \tag{3.3}$$

We propose to use a distance metric which measures the correspondence between the likelihood of the transformed input copies. More specifically, $\mathcal{D}(\cdot,\cdot)$ is calculated by flattening the network output f_l at a given layer l and applying the softmax activation:

$$\mathcal{D}(f_l(t_0), f_l(t_1)) = -\sum_{i=1}^{C} \sigma_i(f_l(t_0)) \log \sigma_i(f_l(t_1)) \tag{3.4}$$

A similar distance metric has previously been proposed by [163] in order to stabilize models against small input perturbations such as the addition of uncorrelated Gaussian noise. It is inspired by the work of [101] on virtual adversarial training, which showed that a distance function based on the Kullback-Leibler (KL) divergence smoothens the model distribution with respect to the input around each data point. Our work extends this approach to training models for invariance to geometric transformations of the input and is the first to perform a detailed investigation on the optimal weighing and placement of the loss.

Since the similarity loss $\mathcal{L}_{sim}$ does not require label information, it enables semi-supervised learning with partially labeled data. Until recently, ladder networks were the state-of-the-art architecture for semi-supervised learning [115]. Ladder networks are denoising autoencoders with lateral connections, into which the similarity loss $\mathcal{L}_{sim}$ can be easily integrated by duplicating the corrupted or uncorrupted encoder path of the ladder network. The duplicated encoder path again shares its weight with the other encoder paths of the ladder network. A simple ladder network architecture where the corrupted encoder path of the ladder network is duplicated and $\mathcal{L}_{sim}$ is applied on the final output layer L is illustrated in Figure 3.1b.

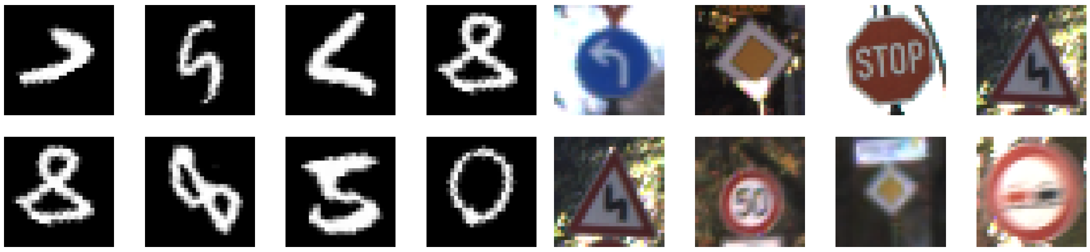
As before, $\mathcal{L}_{sim}$ is added to the total loss $\mathcal{L}_{total}$ where it serves as a second unsupervised loss next to the denoising loss $\mathcal{L}_{denoise}$, which aims to minimize the difference between a clean layer output f_l and the output of a denoising function $\hat{f}_l$ given a corrupted output $\tilde{f}_l$ on all L layers of the network. The classification loss $\mathcal{L}_c$ is then computed on the outputs of the noisy encoder paths $\tilde{f}_L$ where $\lambda_{denoise}$ is a weight parameter of the denoising loss $\mathcal{L}_{denoise}$ and $f_0(t_0) = t_0$.

$$\mathcal{L}_{total} = \mathcal{L}_c + \lambda_{denoise}\mathcal{L}_{denoise} + \lambda\mathcal{L}_{sim} \tag{3.5}$$

$$\mathcal{L}_c = -\sum_{i=1}^{C} y_i \log\sigma_i(\tilde{f}_L(t_0)) - \sum_{i=1}^{C} y_i \log\sigma_i(\tilde{f}_L(t_1)) \tag{3.6}$$

$$\mathcal{L}_{denoise} = \sum_{l=0}^{L} ||f_l(t_0) - \hat{f}_l(t_0)||^2 \tag{3.7}$$

In all models, the classification loss $\mathcal{L}_c$ is only applied on the labeled training samples. On the other hand, the similarity loss $\mathcal{L}_{sim}$ and, in case of a ladder network, the denoising loss $\mathcal{L}_{denoise}$ can be applied on both labeled and unlabeled samples of the training data in order to perform semi-supervised learning.



(a) Rotated MNIST [80]   (b) GTSRB [132]

Figure 3.2: **Dataset Examples.** Example images from the two datasets used in the experimental evaluation of the proposed similarity loss for learning transformation invariance.

3.2 Experimental Evaluation

We first validate our approach in terms of learning *rotation invariant* representations on the commonly used rotated MNIST task [80]. Second, we demonstrate the effectiveness of our technique on the more challenging German Traffic Sign Recognition Benchmark (GTSRB) [132]. Incorporating *perspective invariances* using the proposed similarity loss, our method leads to significant improvements over the baselines for this task.

3.2.1 Experimental Setup

The rotated MNIST classification task [80] is the standard benchmark for evaluating transformation invariance in neural networks [18, 79, 130, 155], despite possible ambiguities between rotated digits such as a rotated 6 and 9. The rotated MNIST dataset (see Figure 3.2a) was created by rotating MNIST digits with uniformly sampled angles between 0 and 2π radians and consists of 12,000 training and 50,000 test samples. As in the original MNIST dataset [82], the images are greyscale and of size 28×28 pixels. We split the dataset into 10,000 training and 2,000 validation samples for determining the hyperparameter λ.

The German Traffic Sign Recognition Benchmark (GTSRB) [132] consists of 39,209 training and 12,630 test images with 43 classes in total. We rescale the original images (see Figure 3.2b) of varying size to 32×32 pixels and normalize them.

In order to perform a fair comparison between data augmentation and the use of a similarity loss, we make sure that every model is being shown the same amount of data in each training epoch. As a similarity loss model utilizes each input sample x twice in every training step under the transformations t_0 and t_1, we also present t_0 and t_1 to the data augmentation baseline in each training step. During training, data augmentation is performed online in a randomized manner. For rotated MNIST, t_0 and t_1 rotate the input x in every training step with an angle which is uniformly sampled between 0 and 2π radians.

In the case of GTSRB, we train for invariance to projective transformations, as traffic signs need to be correctly classified from different angles and distances. The augmentation with a projective transformation is performed by estimating an essential matrix using the eight-point algorithm from a set of point correspondences between the image corners and a randomized set of points. These points are randomly sampled from a uniform distribution within a distance of ± 6 pixels in both dimensions of the image corners.

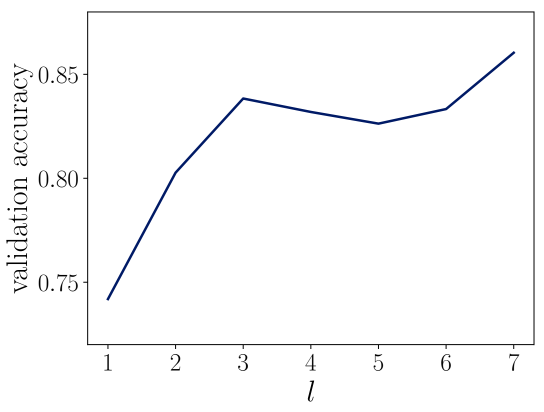
For all experiments we use the same randomization seeds for model comparisons but vary the seed across runs and for all experiments report the average numerical results over five independent runs.

3.2.2 Supervised Learning on Rotated MNIST Subset

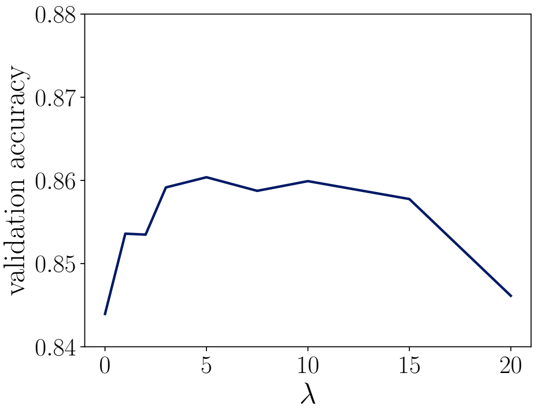
For supervised learning, we integrate the similarity loss $\mathcal{L}_{sim}$ into an all convolutional network architecture [131] and use a subset of $N_s = 100$ labeled samples of the rotated MNIST dataset for training, where each class is represented equally often (*i.e.*, 10 times).

Our network closely resembles the CNN reference architecture for the rotated MNIST task in [18]. This network is constructed from seven convolutional layers, where each but the last layer uses filters of size 3×3 while the last layer uses filters of size 4×4. The convolutional filters are applied with a stride of 1×1. A max-pooling layer of stride and size 2×2 is inserted after the second convolutional layer. All but the last layer use batch normalization [61] before ReLU nonlinearities, followed by dropout with a keep probability of $p = 0.7$. On the last layer the softmax activation is applied. We use the Adam optimizer [69] with a base learning rate of 0.001 and train with 100 samples per mini-batch.

In a first experiment, we evaluate the effect of applying the similarity loss on different layers l of the network (see Figure 3.3a). We find that applying the similarity loss on the last layer results in the highest validation accuracy. This result is in line with findings by [18], which showed that enforcing premature invariance in early layers of the network is undesirable. For all future experiments, we therefore only apply the similarity loss on the final output layer L of a network.



(a) Similarity loss placement layer l



(b) Similarity loss weight parameter λ

Figure 3.3: **Hyperparameter Study** on the placement and weight of the similarity loss on the supervised rotated MNIST task.

In addition, we perform a coarse hyperparameter search with a selected set of weight parameters $\lambda \in [1.0, 2.0, 3.0, 5.0, 7.5, 10.0, 15.0, 20.0]$. The results are plotted in Figure 3.3b and show improved validation accuracies for a wide range of λ values compared to using only data augmentation ($\lambda = 0.0$). While the performance is very robust to the choice of the weight parameter λ, we can observe a drop in validation accuracy when λ is large ($\lambda = 20.0$).

Table 3.1a confirms performance improvements on the test set for the proposed similarity loss with $\lambda = 5.0$ when training for 100 epochs with $N_s = 100$ labeled samples. Compared to a test error of 14.8% when training with data augmentation, we obtain an improved test error of 13.4% when training with an additional similarity loss. Additionally, we also obtain a better test error than our reimplementations of a harmonic network [155] and a group-equivariant P4CNN [18], which replace the regular convolutions in the network architecture with harmonic or group-equivariant convolutions, respectively.

As a baseline we also evaluate the performance on the full dataset of $12,000$ labeled examples. Here, no significant improvement is obtained by the similarity loss compared to a data augmentation model (see Table 3.1b). Both data-driven methods are outperformed by harmonic networks [155] and a group-equivariant P4CNN [18].

Our results suggest that applying a similarity loss improves generalization and outperforms data augmentation as well as encoded transformation invariances when the number of labeled samples is small.

Table 3.1: **Results for the Supervised Rotated MNIST task** using N_s labeled samples.

Method	Test error (%)
Harmonic Networks [155]	21.5
Data augmentation	14.8
P4CNN [18]	14.2
Similarity loss	**13.4**

(a) $N_s = 100$.

Method	Test error (%)
Data augmentation	3.7
Similarity loss	**3.6**
P4CNN [18]	2.28
Harmonic Networks [155]	1.69

(b) $N_s = 12,000$.

3.2.3 Semi-Supervised Learning

The unsupervised nature of the similarity loss $\mathcal{L}_{sim}$ makes it suitable as an additional guidance for semi-supervised learning problems in order to utilize unlabeled data.

3.2.3.1 Rotated MNIST

As a first architecture for semi-supervised learning on rotated MNIST, we use the convolutional architecture from Section 3.2.2 and a subset of $N_s = 100$ labeled samples. Additionally, we use the remaining training samples as unlabeled data. Each mini-batch is constructed from 100 labeled and 100 unlabeled samples, where $\mathcal{L}_c$ is only applied on the labeled samples while $\mathcal{L}_{sim}$ is applied on the full mini-batch. As before, we perform a hyperparameter study of the λ weight parameter from a set $\lambda \in [1.0, 2.0, 3.0, 5.0, 7.5, 10.0, 15.0, 20.0]$ (see Figure 3.4). When comparing to the λ-study for supervised learning (see Figure 3.3b), we can now observe higher validation accuracies and again find the performance to be very robust.

Additionally, we perform a data ablation study where we vary the size of the labeled training set N_s. The results of the data ablation study are visualized in Figure 3.5. The figure demonstrates that the similarity loss is especially helpful when only very little labeled data is available. The benefit of a similarity loss (for a weight parameter of $\lambda = 10.0$ and 100 training epochs) is confirmed on the test set where the final test error is lowered by more than 2% compared to training only with data augmentation (see Table 3.2a). Furthermore, the final test error is more than 1% lower compared to using the similarity loss on only the labeled images, which confirms the ability of the similarity loss to exploit additional unlabeled data.

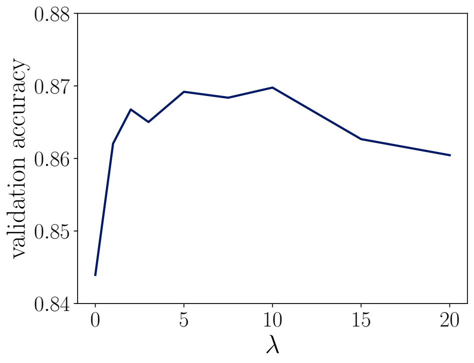


Figure 3.4: **Hyperparameter Study** for weight parameter λ on semi-supervised rotated MNIST.

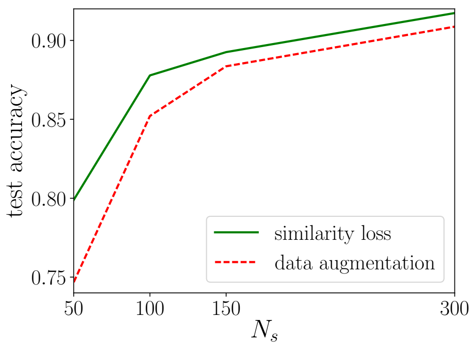


Figure 3.5: **Accuracy vs. number of labeled samples** N_s on semi-supervised rotated MNIST.

Table 3.2: **Results for the Semi-Supervised Rotated MNIST Task** using $N_s = 100$ labeled samples.

Method	Test error (%)
Data augmentation	14.8
Similarity loss	**12.2**

(a) Siamese Network.

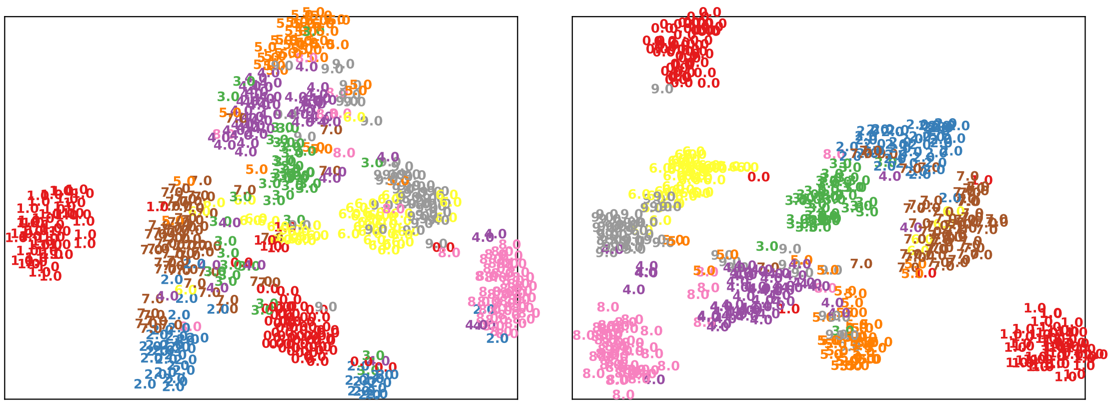
Method	Test error (%)
Data augmentation	8.0
Similarity loss (clean path)	**7.6**
Similarity loss (noisy path)	**6.8**

(b) Ladder Network.

We also observe improved class separability when visualizing the learned feature representations in the last layer of the model (see Figure 3.6).

For a second set of semi-supervised learning experiments, we incorporate the similarity loss into the fully connected ladder network architecture proposed by [115] for the permutation invariant MNIST task. It features layers of size 784-1000-500-250-250-250-10 with respective denoising weight parameters $\lambda_{denoise} = [1000.0, 10.0, 0.10, 0.1, 0.1, 0.1, 0.1]$. The noisy encoder path uses Gaussian corruption noise with standard deviation 0.3. We train the network with mini-batches of 100 labeled and 256 unlabeled samples using the Adam optimizer and a base learning rate of 0.02 for 300 training epochs.

Table 3.2b displays the final test accuracies for utilizing our similarity loss in a ladder network with a weight parameter of $\lambda = 20.0$, which was determined in a separate hyperparameter search. We again find the addition of a similarity loss to be beneficial.



(a) Data Augmentation (b) Similarity Loss

Figure 3.6: **Improved Class Separability.** t-SNE visualizations of the learned feature representations on the final network layer.

Incorporating it in the noisy encoder path results in a better performance of 6.8% compared to a final test error of 7.6% in the clean encoder path. This can be explained by the Gaussian noise of the noisy encoder path providing additional regularization. Our results demonstrate that the use of a similarity loss also enables improving the performance of a previous state-of-the-art model architecture, specially designed for the semi-supervised learning task.

3.2.3.2 German Traffic Sign Recognition Benchmark

As a final experiment, we consider the German Traffic Sign Recognition Benchmark (GTSRB) [132]. The network architecture for this task is an all-convolutional model, which resembles the All-CNN-C architecture proposed by [131] for the CIFAR-10 task [74]. It consists of nine convolution layers. The first four layers have 96, the, following layers 192 filters, all of size 3×3. They are applied with stride 1 except in the third and sixth layer where a stride of 2 is used. After the final convolutional layer average pooling is performed. Dropout is applied on the input with a probability of 0.2 and on the convolutional feature maps with 0.5. All layers use ReLU nonlinearities, batch normalization and weight decay of 0.001. A softmax activation is applied after the final layer. The network is trained with stochastic gradient descent and Momentum with mini-batches of 100 labeled and 100 unlabeled samples. A learning rate of 0.05 is decayed over the course of 100 training epochs.

Table 3.3: Results for the semi-supervised GTSRB task with $N_s = 2150$.

Method	Test error (%)
Data augmentation	14.8
Similarity loss	**9.6**

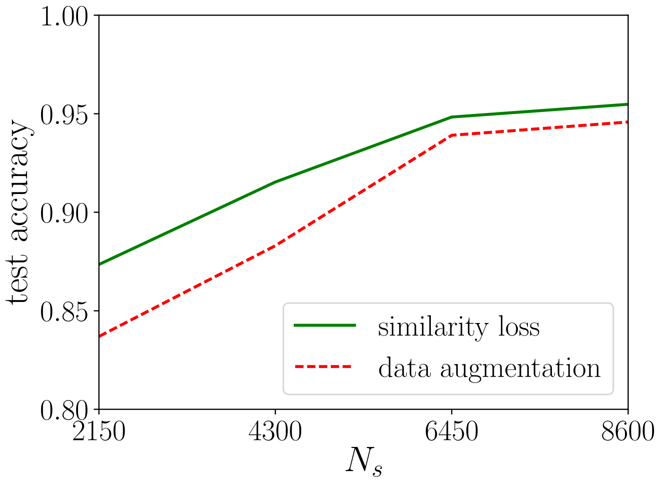


Figure 3.7: Accuracy vs. number of training samples on semi-supervised GTSRB.

In contrast with previous experiments on rotated MNIST, we now train for invariance to projective transformations. Unlike rotations, these cannot be easily encoded into the convolutional filters of a neural network. Here, the use of a similarity loss, which relies on augmenting the training data (as described in Section 3.2.1), offers a simple, yet effective solution to train CNNs for invariance to more complex geometric transformations.

The test results for semi-supervised learning on the GTSRB task for 50 samples per class (*i.e.*, 2150 samples in total) and $\lambda = 10.0$ are displayed in Table 3.3. We observe a clear improvement in the final test accuracy from 14.8% to 9.6% when utilizing the similarity loss, despite the model already being heavily regularized by dropout, weight decay and batch normalization, which again indicates the effectiveness of the similarity loss when little labeled data is available.

As for rotated MNIST, we again perform a data ablation study (see Figure 3.7). The study shows that the improvement when using an additional similarity loss is largest when N_s is small, but that even for larger sizes of the labeled training set the similarity loss outperforms the data augmentation model.

3.3 Discussion

The experiments in Section 3.2 demonstrate the benefits of using the proposed similarity loss in both supervised and semi-supervised learning tasks. Data augmentation works well in practice for fully supervised problems when big labeled training sets are available. However, in this work, we show that an additional similarity loss can act as an effective regularizer, which improves upon data augmentation when little annotated training data is available. The benefit of the proposed similarity loss is not limited to the "toy-like" rotated MNIST task but extends to more complex geometric transformations of natural images where even larger improvements can be obtained.

In addition, another contribution of our work concerns the fact that the proposed similarity loss can be utilized for semi-supervised learning, where it can help to exploit additional unlabeled data. The inclusion of the similarity loss in a semi-supervised ladder network shows particular promise. With our proposed modification, we further improve over an architecture which until recently was the state-of-the-art approach for semi-supervised learning. As the rotated MNIST dataset has not been commonly used to evaluate semi-supervised learning architectures we do not claim to set a new state-of-the-art but consider the ladder network using an additional similarity loss to have highly competitive performance.

In general, the use of an unsupervised similarity loss is a surprisingly simple idea which can easily be integrated into any deep learning model. All it requires is to duplicate the classification stream of the network and tune the λ hyperparameter. In our experiments, we found the performance to generally be very robust to the choice of the weight parameter λ. The similarity loss can be additionally combined with methods which encode invariances directly into convolutional filters [18, 155, 168] and with architectures which enable neural networks to handle geometric transformations more easily [24, 63].

3.4 Conclusions

This chapter proposed an unsupervised similarity loss which penalizes differences between the predictions for transformed copies of an input for improved learning of transformation invariance in deep neural networks on the rotated MNIST and German Traffic Sign Recognition Benchmark classification tasks. We showed that our similarity loss acts as an effective regularizer, which improves model performance when little annotated data is available, in both supervised and semi-supervised learning. Future work could investigate the application of the proposed similarity loss on a combination of network layers or an adjustment of the weight parameter λ over the course of training. In addition, it could consider the applicability and usefulness of a similarity loss for other computer vision tasks besides image classification such as semantic segmentation or object detection.

While this work improves the use of data-driven methods based on augmenting the training data for learning transformation invariance, there still remains a gap to techniques which encode invariances to transformations directly into the filters of a convolutional neural network when training on the full set of labels on rotated MNIST [18, 155, 168]. However, unlike the proposed similarity loss, which can easily be applied for a wide variety of different transformations, these methods are currently limited to simple geometric transformations such as rotations and generally focus on encoding a single additional invariance or equivariance property into the network architecture.

Yet, there are scenarios for which the development of approaches that encode transformation invariance into the architecture of deep models is a promising avenue of research. In particular, this is the case for tasks in which a single dominant transformation exists that occurs regularly in the input data and which has the potential to negatively impact model performance. In such cases, it can be highly beneficial for a model if it does not have to utilize a large part of its model capacity in order to learn the desired invariance. If this invariance instead is encoded into the model, this factor of variation is removed and the model is able to use its full capacity to focus on other important aspects in the data. In this way, encoding invariance into the network architecture can both improve the performance of a model as well as enable faster learning from fewer training samples.

We look at one such scenario in the next chapter, where we present the SphereNet framework for encoding invariance to the distortions which are present in the planar projection of 360° omnidirectional images into deep convolutional neural networks.

Chapter 4

SphereNet: Learning Spherical Representations in Omnidirectional Images

In this chapter, we present the *SphereNet* framework to learn spherical representations in omnidirectional images. Over the last years, omnidirectional imaging devices have gained in popularity due to their wide field of view and their widespread applications ranging from virtual reality to robotics [58, 78, 114, 133, 134]. Today, omnidirectional action cameras are available at an affordable price and 360° viewers are integrated into social media platforms. Given the growing amount of spherical imagery, there is an increasing interest in computer vision models (e.g., image classification, object detection) which are optimized for this kind of data.

The most popular representation of 360° images is the equirectangular projection where latitude and longitude of the spherical image are mapped to horizontal and vertical grid coordinates, see Figs. 4.1+4.2 for an illustration. However, the equirectangular image representation suffers from heavy distortions in the polar regions which implies that an object will appear differently depending on its latitudinal position. This presents a challenge to modern computer vision algorithms, such as convolutional neural networks (CNNs) which are the state-of-the-art solution to many computer vision tasks.

While CNNs are capable of learning invariances to common object transformations and intra-class variations, they require significantly more parameters, training samples and training time to learn invariance to these distortions from data. This is undesirable as data annotation is time-consuming and annotated omnidirectional datasets are scarce and smaller in size than those collected for the perspective case. An attractive alternative is to encode invariance to geometric transformations directly into a CNN, which has been proven highly efficient in reducing the number of model parameters as well as the required number of training samples [18, 155].

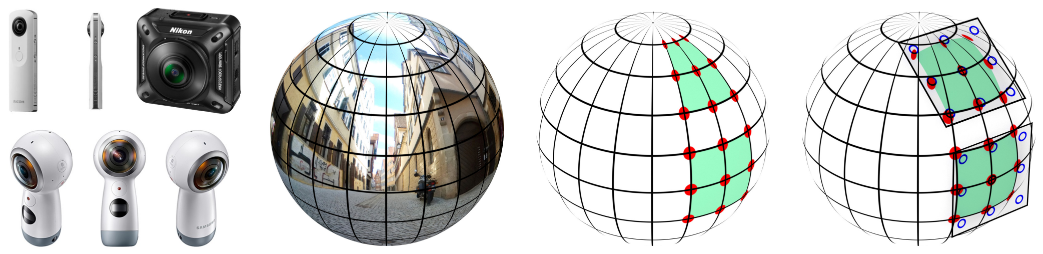

(a) 360° Cameras (b) 360° Image (c) Regular Kernel (d) SphereNet Kernel

Figure 4.1: **Overview.** (b) Capturing images with fisheye or 360° action cameras (see Fig. 4.1a) results in images which are best represented on the sphere. (c) Using regular convolutions (e.g., with 3×3 filter kernels) on the rectified equirectangular representation (see Fig. 4.2b) suffers from distortions of the sampling locations (red) close to the poles. (d) In contrast, our SphereNet kernel exploits projections (red) of the sampling pattern on the tangent plane (blue), yielding filter outputs which are invariant to latitudinal rotations.

However, only few deep neural network architectures have so far been specifically designed to encode invariance against the distortions in omnidirectional images into the architecture of a deep learning model (see Section 2.2).

One is the graph-based approach for omnidirectional image classification by Khasanova et al. [67]. While a graph representation solves the problem of discontinuities at the borders of an equirectangular image, graph convolutional networks are limited to small graphs and image resolutions (50×50 pixels in [68]) and have not yet demonstrated recognition performance comparable to regular CNNs on more challenging datasets. In contrast, our method also restores the spherical image connectivity but builds on regular convolutions, which offer state-of-the-art performance for many computer vision tasks.

In concurrent work, Cohen et al. [16] proposed to use spherical CNNs for classification and encode rotation equivariance into the network. However, often full rotation invariance is not desirable as 360° images are mostly captured in a dominant orientation (i.e., it is rare that the camera is flipped upside-down). Incorporating full rotation invariance in such scenarios reduces discriminative power as evidenced by our experiments. Furthermore, it is non-trivial to integrate either graph or spherical convolutions into network architectures for more complex computer vision tasks like object detection. In fact, no results beyond image classification are provided in the literature. In contrast, our SphereNet framework readily allows for adapting existing CNN architectures as well as trained perspective CNN models for object detection or other higher-level vision tasks to the omnidirectional case.

The most related work to SphereNet is the model by Su et al. [162] which processes equirectangular images with regular convolutions filters but increases their size towards the poles. However, this adaptation is a simplistic approximation of distortions in the equirectangular format and implies that weights can only be shared along each row, resulting in a significant increase in model parameters, which makes it hard to train their model from scratch. In contrast, SphereNet retains weight sharing across all rows and columns and better approximates the distortions in equirectangular images by adapting the sampling locations instead of the size of the convolutional filters.

Several recent works also consider adapting the sampling locations of convolutional networks, either dynamically [25] or statically [64, 96]. Unlike our work, these methods need to learn the sampling locations during training, which requires additional model parameters and training steps. In contrast, we take advantage of the known geometric properties of omnidirectional cameras to inject this knowledge explicitly into a convolutional neural network, thus avoiding distortions as illustrated in Figs. 4.1+4.2. As SphereNet builds on regular convolutional filters, it naturally enables the transfer of CNNs between different image representations by adapting the sampling locations of the convolution kernels. We demonstrate this by training object detectors on perspective images and transferring them to omnidirectional inputs. We provide extensive experiments on semi-synthetic as well as real-world datasets which demonstrate the effectiveness of the proposed approach.

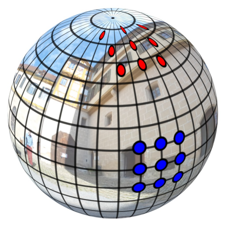
In summary, this chapter makes the following **contributions**:

- We introduce SphereNet, a framework for learning spherical image representations by encoding distortion invariance into convolutional filters. SphereNet retains the original spherical image connectivity and, by building on regular convolutions, enables the transfer of perspective CNN models to omnidirectional inputs.

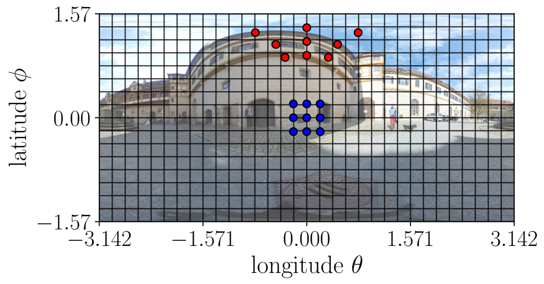
- We improve the computational efficiency of SphereNet using an approximately uniform sampling of the sphere.

- We create novel semi-synthetic and real-world datasets for object detection, semantic segmentation and optical flow in omnidirectional images.

- We demonstrate improved performance as well as SphereNet's transfer learning capabilities on the tasks of image classification, object detection, semantic segmentation as well as optical flow and compare our results to several state-of-the-art baselines.



(a) Sphere



(b) Equirectangular

Figure 4.2: **Kernel Sampling Pattern** at $\phi = 0$ (blue) and $\phi = 1.2$ (red) in spherical (a) and equirectangular (b) representation. Note the distortion of the kernel at $\phi = 1.2$ in (b).

4.1 Method

This section introduces the proposed SphereNet framework. First, we describe the adaptation of the sampling pattern to achieve distortion invariance on the surface of the sphere (Section 4.1.1). Second, we propose an approximation which uniformly samples the sphere to improve the computational efficiency of our method (Section 4.1.2). Next, we present details on the Spherical Transformer Network (Section 4.1.3). Finally, we demonstrate how SphereNet can be incorporated into models for image classification (Section 4.1.4), object detection (Section 4.1.5), semantic segmentation (Section 4.1.6) and optical flow (Section 4.1.7), for which we propose a novel spherical optical flow representation.

4.1.1 Kernel Sampling Pattern

The central idea of SphereNet is to lift local CNN operations (e.g. convolution, pooling) from the regular image domain to the sphere surface where omnidirectional images can be represented without distortions. This is achieved by representing the kernel as a small patch tangent to the sphere (see Fig. 4.1d). Our model focuses on distortion invariance and not rotation invariance, as in practice 360° images are mostly captured in one dominant orientation. Thus, we consider upright patches which are aligned with the great circles of the sphere.

More formally, let S be the unit sphere with S^2 its surface. Every point $\mathbf{s} = (\phi, \theta) \in S^2$ is uniquely defined by its latitude $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and longitude $\theta \in [-\pi, \pi]$. Let further Π denote the tangent plane located at $\mathbf{s}_\Pi = (\phi_\Pi, \theta_\Pi)$. We denote a point on Π by its coordinates $\mathbf{x} \in \mathbb{R}^2$. The local coordinate system of Π is hereby centered at $\mathbf{s}$ and oriented upright. Let Π_0 denote the tangent plane located at $\mathbf{s} = (0,0)$. A point $\mathbf{s}$ on the sphere is related to its tangent plane coordinates $\mathbf{x}$ via a gnomonic projection [111].

While the proposed approach is compatible with convolutions of all sizes, in the following we consider a 3×3 kernel, which is most common in state-of-the-art architectures [53, 129]. We assume that the input image is provided in equirectangular format which is the de facto standard representation for omnidirectional cameras of all form factors (e.g. catadioptric, dioptric or polydioptric). In Section 4.1.2 we consider a more efficient representation that improves the computational efficiency of our method.

The kernel shape is defined so that its sampling locations $\mathbf{s}_{(j,k)}$, with $j,k \in \{-1,0,1\}$ for a 3×3 kernel, align with the step sizes Δ_θ and Δ_ϕ of the equirectangular image at the equator. This ensures that the image can be sampled at Π_0 without interpolation:

$$\begin{aligned}
\mathbf{s}_{(0,0)} &= (0,0) && (4.1)\\
\mathbf{s}_{(\pm1,0)} &= (\pm\Delta_\phi, 0) && (4.2)\\
\mathbf{s}_{(0,\pm1)} &= (0, \pm\Delta_\theta) && (4.3)\\
\mathbf{s}_{(\pm1,\pm1)} &= (\pm\Delta_\phi, \pm\Delta_\theta) && (4.4)
\end{aligned}$$

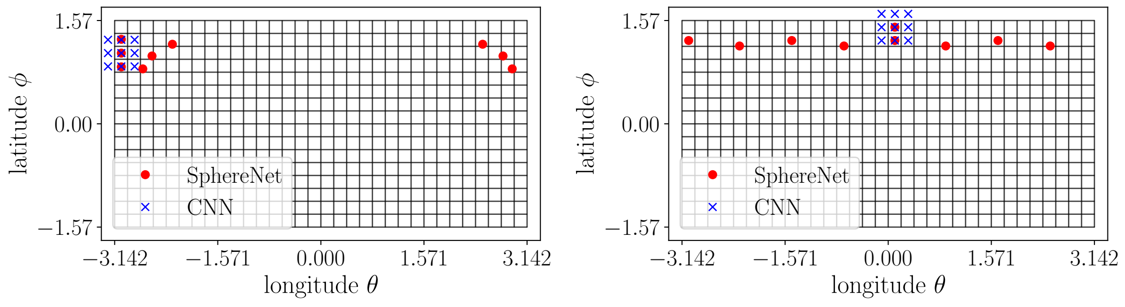
The position of these filter locations on the tangent plane Π_0 can be calculated via the gnomonic projection [111]:

$$\begin{aligned}
x(\phi,\theta) &= \frac{\cos\phi \sin(\theta - \theta_{\Pi_0})}{\sin\phi_{\Pi_0}\sin\phi + \cos\phi_{\Pi_0}\cos\phi\cos(\theta - \theta_{\Pi_0})} && (4.5)\\
y(\phi,\theta) &= \frac{\cos\phi_{\Pi_0}\sin\phi - \sin\phi_{\Pi_0}\cos\phi\cos(\theta - \theta_{\Pi_0})}{\sin\phi_{\Pi_0}\sin\phi + \cos\phi_{\Pi_0}\cos\phi\cos(\theta - \theta_{\Pi_0})} && (4.6)
\end{aligned}$$

For the sampling pattern $\mathbf{s}_{(j,k)}$, this yields the following kernel pattern $\mathbf{x}_{(j,k)}$ on Π_0:

$$\begin{aligned}
\mathbf{x}_{(0,0)} &= (0,0) && (4.7)\\
\mathbf{x}_{(\pm1,0)} &= (\pm\tan\Delta_\theta, 0) && (4.8)\\
\mathbf{x}_{(0,\pm1)} &= (0, \pm\tan\Delta_\phi) && (4.9)\\
\mathbf{x}_{(\pm1,\pm1)} &= (\pm\tan\Delta_\theta, \pm\sec\Delta_\theta\tan\Delta_\phi) && (4.10)
\end{aligned}$$

We keep this pattern fixed. When applying the filter at a different location $\mathbf{s}_\Pi = (\phi_\Pi, \theta_\Pi)$,



(a) Left Boundary

(b) Top Boundary

Figure 4.3: **Sampling Locations.** This figure compares the sampling locations of SphereNet (red) to the sampling locations of a regular CNN (blue) at the boundaries of the equirectangular image. Note how the SphereNet kernel automatically wraps at the left image boundary (a) while correctly representing the discontinuities and distortions at the pole (b). SphereNet thereby retains the original spherical image connectivity which is discarded in a regular convolutional neural network that utilizes zero-padding along the image boundaries.

the inverse gnomonic projection is applied where $\rho = \sqrt{x^2 + y^2}$ and $\nu = \tan^{-1} \rho$:

$$\begin{aligned} \phi(x,y) &= \sin^{-1}\left(\cos\nu \sin\phi_\Pi + \frac{y \sin\nu \cos\phi_\Pi}{\rho}\right) \\ \theta(x,y) &= \theta_\Pi + \tan^{-1}\left(\frac{x \sin\nu}{\rho \cos\phi_\Pi \cos\nu - y \sin\phi_\Pi \sin\nu}\right) \end{aligned} \tag{4.11}$$

The sampling grid locations of the convolutional kernels thus get distorted in the same way as objects on a tangent plane of the sphere get distorted when projected from different elevations to an equirectangular image representation. Fig. 4.2 demonstrates this concept by visualizing the sampling pattern at two different elevations ϕ.

Besides encoding distortion invariance into the filters of convolutional neural networks, SphereNet additionally enables the network to wrap its sampling locations around the sphere. As SphereNet uses custom sampling locations for sampling inputs or intermediary feature maps, it is straightforward to allow a filter to sample data across the image boundary. This eliminates any discontinuities which are present when processing omnidirectional images with a regular convolutional neural network and improves recognition of objects which are split at the sides of an equirectangular image representation or which are positioned very close to the poles, see Fig. 4.3.

By changing the sampling locations of the convolutional kernels while keeping their size unchanged, our model additionally enables the transfer of CNN models between different image representations. In our experimental evaluation, we demonstrate how an object detector trained on perspective images can be successfully applied to the omnidirectional case. Note that our method can be used for adapting almost any existing deep learning architecture from perspective images to the omnidirectional setup. In general, our SphereNet framework can be applied as long as the image can be mapped to the unit sphere. This is true for many imaging models, ranging from perspective over fisheye[1] to omnidirectional models. Thus, SphereNet can be seen as a generalization of regular CNNs which encodes the camera geometry into the network architecture.

Implementation: As the sampling locations are fixed according to the geometry of the spherical image representation, they can be precomputed for each kernel location at every layer of the network. Further, their relative positioning is constant in each image row. Therefore, it is sufficient to calculate and store the sampling locations once per row and then translate them. We store the sampling locations in look-up tables. These look-up tables are used in a customized convolution operation which is based on highly optimized general matrix multiply (GEMM) functions [65]. As the sampling locations are real-valued, interpolation of the input feature maps is required. For an arbitrary sampling location (p_x, p_y) in a feature map f and an interpolation kernel $g(a,b)$, interpolation is defined as:

$$f(p_x, p_y) = \sum_n^H \sum_m^W f(m,n) g(p_x, m) g(p_y, n) \tag{4.12}$$

In our experiments, we compare nearest neighbor interpolation to bilinear interpolation. The bilinear interpolation kernel is defined as:

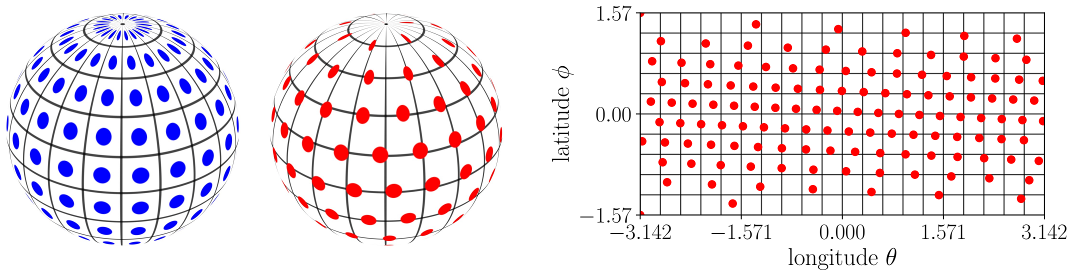
$$g(a,b) = max(0, 1 - |a - b|) \tag{4.13}$$

while the nearest neighbor kernel is defined as follows:

$$g(a,b) = \delta(\lfloor a + 0.5 \rfloor - b) \tag{4.14}$$

where $\delta(\cdot)$ is the Kronecker delta function.

[1]While in some cases the single viewpoint assumption is violated, the deviations are often small in practice and can be neglected at larger distances.



(a) Equirectangular (b) Uniform (c) Uniform Sampling in the Image Plane

Figure 4.4: **Uniform Sphere Sampling.** Comparison of an equirectangular sampling grid on the sphere with $P = 200$ points (a) to an approximation of evenly distributing $P = 127$ sampling points on a sphere with the method of Saff and Kuijlaars [120] (b, c). Note that the sampling points at the poles are much more evenly spaced in the uniform sampling (b) compared to the equirectangular representation (a) which oversamples in these regions.

4.1.2 Uniform Sphere Sampling

In order to improve the computational efficiency of our method, we investigate a more efficient sampling of the spherical image. The equirectangular representation oversamples the spherical image in the polar regions (see Fig. 4.4a), which results in near duplicate image processing operations in this area. We can avoid unnecessary computation in the polar regions by applying our method to a representation where data is stored uniformly on the sphere, in contrast to considering the pixels of the equirectangular image.

To sample points evenly from a sphere, we leverage the method of Saff and Kuijlaars [120] as it is fast to compute and works with an arbitrary number of sampling points P, including large values of P. More specifically, we obtain points along a spiral that encircles the sphere in a way that the distance between adjacent points along the spiral is approximately equal to the distance between successive coils of the spiral. For all points k with $1 \leq k \leq P$, it sets ϕ_k and θ_k for $2 \leq k \leq P-1$ to:

$$h_k = -1 + \frac{2(k-1)}{P-1} \tag{4.15}$$

$$\phi_k = \arccos(h_k) - \frac{\pi}{2} \tag{4.16}$$

$$\theta_k = \left(\theta_{k-1} + \frac{3.6}{\sqrt{P}} \frac{1}{\sqrt{1-h_k^2}}\right) (\bmod 2\pi) - \pi \tag{4.17}$$

while $\theta_1 = \theta_P = -\pi$.

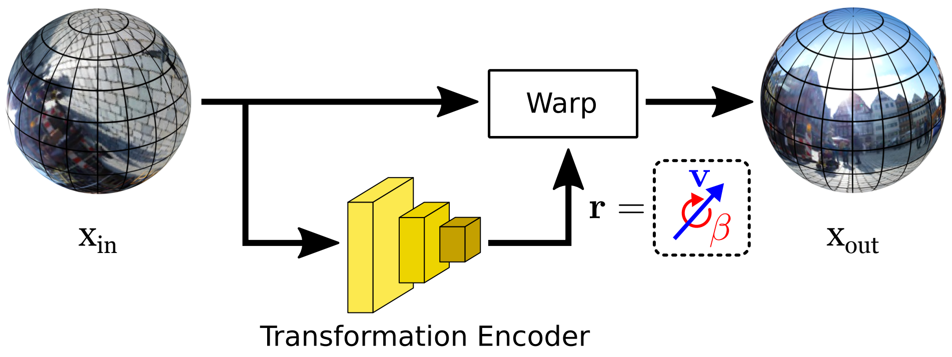


Figure 4.5: **Spherical Transformer Network.** A transformation encoder network outputs the rotation parameters $\mathbf{r}$ specifying the axis $\mathbf{v}$ and angle β of rotation. The input x_{in} is warped to the output x_{out} via bilinear interpolation according to $\mathbf{r}$.

In order to sample omnidirectional images which are represented in the equirectangular format without loss of information along the equator, we define $P = \lfloor \frac{2P_e}{\pi} \rfloor$ where P_e is the number of sampling points in the equirectangular image. As visualized in Fig. 4.4 for an equirectangular image with $P_e = 20 \times 10 = 200$ sampling points, this results in a sampling grid of $P = 127$ points with a similar sampling density at the equator, while significantly reducing the number of sampling points at the poles.

To minimize the loss of information when sampling the equirectangular image we use bilinear interpolation. Afterwards, the image is represented by an $P \times c$ matrix, where c is the number of image channels. Unlike the equirectangular format, this representation no longer encodes the spatial position of each data point. Thus, we save this information in a separate matrix. This location matrix is used to compute the look-up tables for the kernel sampling locations as described in Section 4.1.1. Downsampling of the image is implemented by recalculating a reduced set of sampling points. For applying the kernels and downsampling the image nearest neighbor interpolation is used.

4.1.3 Spherical Transformer Network

As a baseline for the classification and detection tasks, we investigate the use of a *Spherical Transformer Network (SphereTN)*, which aims to undistort objects by performing a global rotation of the spherical image conditioned on the input image. Compared to SphereNet, the Spherical Transformer needs to learn how to undistort an object. Additionally, it only performs a single global transformation of the input and may therefore be unable to undistort multiple objects at once.

Similar to a Spatial Transformer Network (STN) [63], the Spherical Transformer Network uses a localization network to predict a set of transformation parameters which are used to resample the input image. However, unlike a Spatial Transformer which typically outputs an affine transformation, a Spherical Transformer Network predicts and applies a 3D rotation of the spherical image representation.

In order to avoid the gimbal lock problem, we do not represent the axial rotations using Euler angles but instead leverage the axis-angle representation. More formally, we rotate 3D points $\mathbf{y} \in \mathbb{R}^3$ located on the surface of the unit sphere S by applying a rotation $\mathbf{y}' = R\mathbf{y}$ where the rotation matrix $R \in SO(3)$ is given as follows:

$$R = I + (\sin\beta)[\mathbf{v}]_\times + (1 - \cos\beta)[\mathbf{v}]_\times^2 \tag{4.18}$$

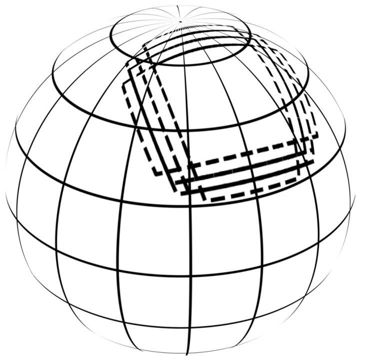
Here, $\mathbf{v} \in \mathbb{R}^3$ is a unit vector defining the rotation axis and β denotes the rotation angle. $[\mathbf{v}]_\times$ denotes the cross-product matrix and is an element of the Lie algebra $so(3)$.

As $\mathbf{v}$ is a unit vector with two degrees of freedom, we are able to encode the rotational component β as its length. More specifically, we parameterize the rotation axis $\mathbf{v}$ and rotation angle β with a 3-dimensional vector $\mathbf{r} = \beta \cdot \mathbf{v}$ which is the output of the localization network conditioned on the input image. Note that after prediction, $\mathbf{r}$ can be easily decomposed into $\beta = \|\mathbf{r}\|_2$ and $\mathbf{v} = \mathbf{r}/\beta$.

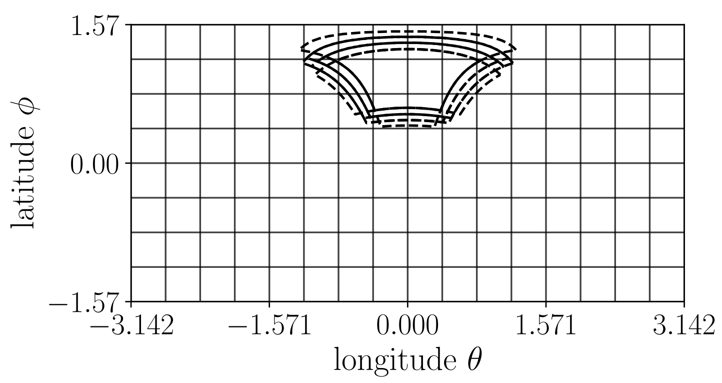
The predicted transformation is applied to the omnidirectional image via differentiable image sampling with bilinear interpolation [63]. See Fig. 4.5 for an illustration.

4.1.4 Spherical Image Classification

SphereNet can be integrated into a convolutional neural network for image classification by adapting the sampling locations of the convolution and pooling kernels as described in Section 4.1.1. Furthermore, it is straightforward to additionally utilize a uniform sphere sampling (see Section 4.1.2), which we will compare to nearest neighbor and bilinear interpolation on an equirectangular representation in the experiments. The integration of SphereNet into an image classification network does not introduce novel model parameters and no changes to the training of the network are required.



(a) Sphere



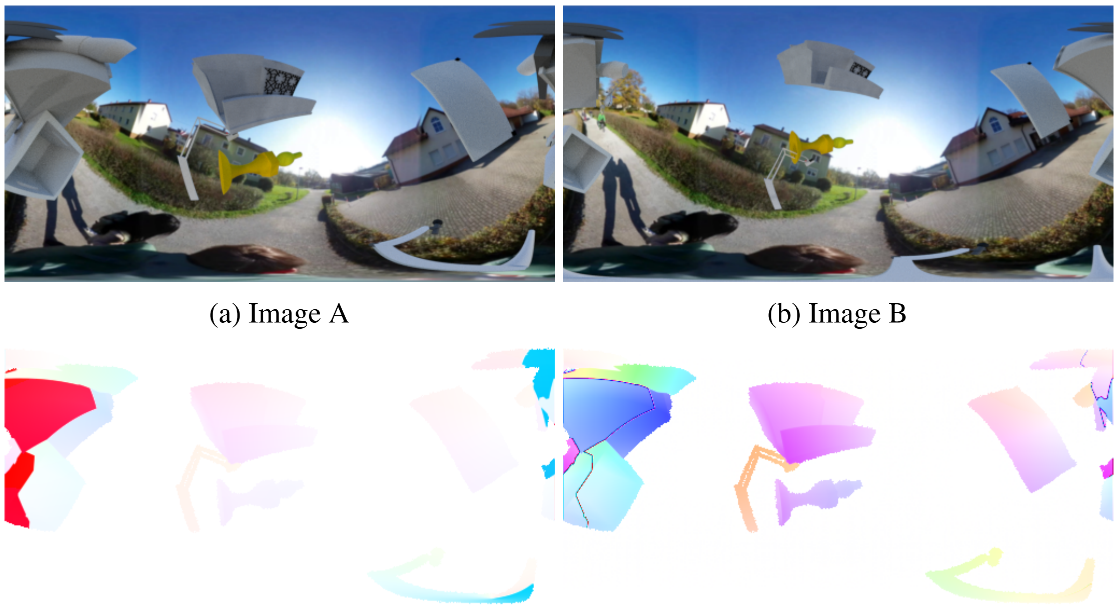
(b) Equirectangular

Figure 4.6: **Spherical Anchor Boxes** are gnomonic projections of 2D bounding boxes of various scales, aspect ratios and orientations on tangent planes of the sphere. The above figure visualizes anchors of the same orientation at different scales and aspect ratios on a 16×8 feature map on a sphere (a) and an equirectangular grid (b).

4.1.5 Spherical Object Detection

In order to perform object detection on the sphere, we propose the *Spherical Single Shot MultiBox Detector (SphereSSD)*, which adapts the popular Single Shot MultiBox Detector (SSD) [89] to objects located on tangent planes of a sphere. SSD exploits a fully convolutional architecture, predicting category scores and box offsets for a set of default anchor boxes of different scales and aspect ratios. We refer the reader to [89] for details. As in the regular SSD, SphereSSD uses a weighted sum between a localization loss and confidence loss. However, in contrast to the original SSD, anchor boxes are now placed on tangent planes of the sphere and are defined in terms of spherical coordinates of their respective tangent plane, the width/height of the box on the tangent plane as well as an in-plane rotation. An illustration of spherical anchor boxes is provided in Fig. 4.6.

In order to match anchor boxes to ground truth detections, we select the anchor box closest to each ground truth box. During inference, we perform non-maximum suppression. For evaluation, we use the Jaccard index computed as the overlap of two polygonal regions which are constructed from the gnomonic projections of evenly spaced points along the rectangular bounding box on the tangent plane.


(a) Image A (b) Image B

(c) Standard Flow Representation (d) Spherical Flow Representation

Figure 4.7: **Spherical Flow** resolves the errors which appear in standard flow at the borders of the equirectangular image.

4.1.6 Spherical Semantic Segmentation

Similarly to the classification task (see Section 4.1.4), SphereNet can be integrated into a convolutional neural network for semantic segmentation by adapting the convolutional and pooling kernels, with no further changes to the training data or training scheme required and no additional model parameters being introduced. Unlike classification networks, semantic segmentation is a structured prediction problem with a high-dimensional output (i.e., the semantic label map). Thus semantic segmentation networks typically comprise an encoder and a decoder, often in combination with skip connections to retain fine details. To avoid checkerboard-like artifacts of transposed convolution operations we use the resize-convolution upsampling scheme proposed by Odena et al. [105], which consists of separate steps of upsampling to a higher resolution and convolution to compute features.

4.1.7 Spherical Optical Flow

In order to perform optical flow prediction on omnidirectional images, we need to adapt the optical flow formulation to the sphere. Regular optical flow is defined as a two-dimensional vector in the image plane. However, for omnidirectional images this definition is flawed as it introduces errors when objects cross the borders of the equirectangular image (see high intensity regions at the borders of Fig. 4.7c). We therefore propose a novel spherical optical flow formulation for omnidirectional images. Given two points on the sphere (ϕ_a, θ_a), (ϕ_b, θ_b) we first compute their corresponding Cartesian coordinates a, b. We then rotate a and b by θ_a around the z-axis and $-\phi_a$ around the y-axis so that the rotated vector a_r is aligned with the x-axis at $(1,0,0)$. The spherical coordinates θ_u, ϕ_v of the rotated vector b_r are then used as our novel spherical optical flow representation. While this representation can handle optical flow across borders of an equirectangular image, it is limited to a range of $\theta_u, \phi_v \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, which we however consider to be sufficient for the majority of omnidirectional optical flow applications. Fig. 4.7 visualizes how this spherical optical flow representation resolves issues of the standard optical flow formulation when objects cross the borders or poles of the equirectangular image.

Below, we briefly compare our proposed spherical optical flow formulation to a number of alternative representations of optical flow for omnidirectional images.

First, a seemingly natural spherical flow representation could be defined as a three-dimensional vector f, which is a product of the angle axis n and the angle α between the two points in 3D:

$$f = n \cdot \alpha \tag{4.19}$$

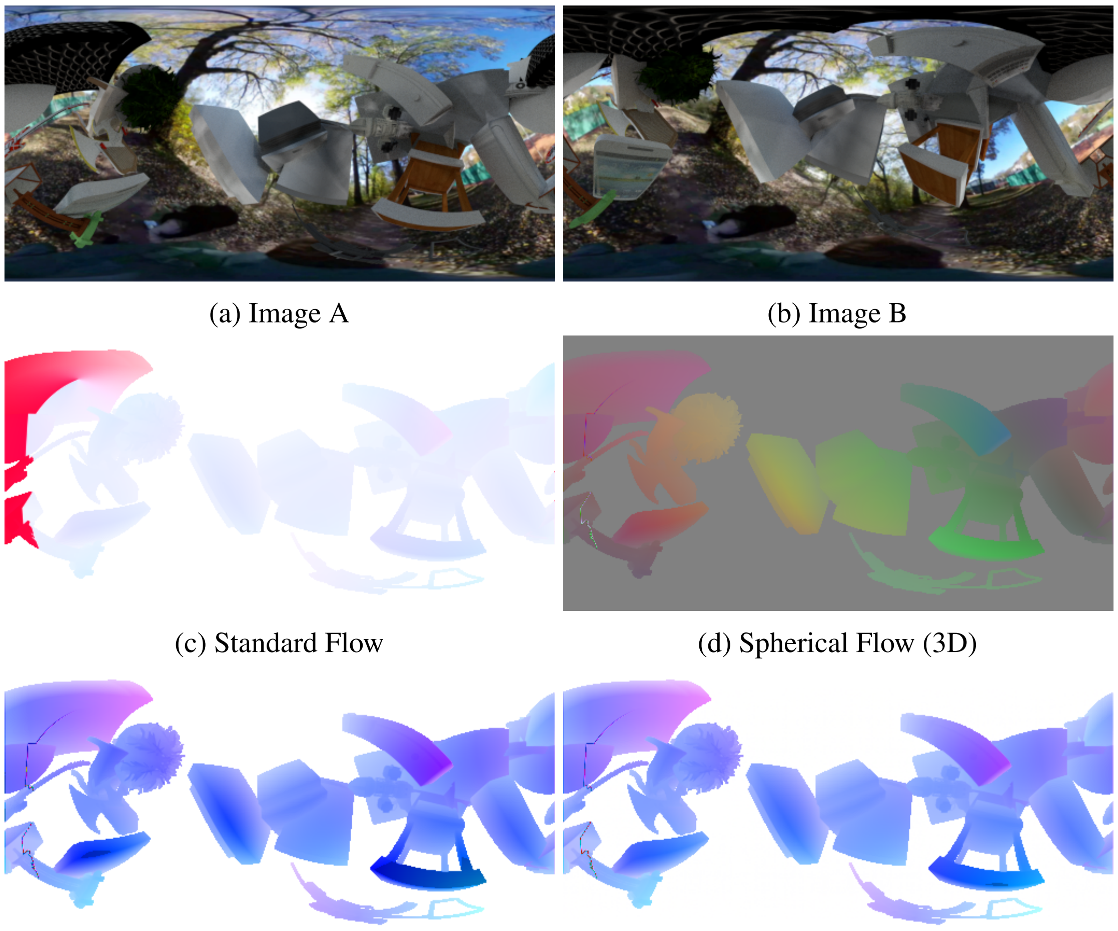
$$n = \frac{a \times b}{||a \times b||_2} \tag{4.20}$$

$$\alpha = \arcsin(||a \times b||_2) \tag{4.21}$$

$$f = \frac{a \times b}{||a \times b||_2} \cdot \arcsin(||a \times b||_2) \tag{4.22}$$

where a, b are the Cartesian coordinates of (ϕ_a, θ_a) and (ϕ_b, θ_b), respectively.

This corresponds to the widely used axis-angle representation for rotations. We visualize this three-dimensional flow representation in Fig. 4.8d for a scene in which all objects rotate in the same direction around the camera. While this optical flow formulation solves the problem of discontinuities for optical flow that crosses the borders or poles of the equirectangular image, this representation, unlike standard optical flow, is not translation invariant. Translation variant representations are difficult for convolutional neural networks to learn as confirmed by our initial experiments. We thus do not consider this to be a suitable representation. In addition, a three-dimensional flow is generally less interpretable as it makes a direct comparison with standard flow images difficult.


(a) Image A (b) Image B

(c) Standard Flow (d) Spherical Flow (3D)

(e) Spherical Flow (tangent) (f) Spherical Flow (proposed)

Figure 4.8: **Alternative Spherical Flow Representations**: An overview of alternatives to the proposed spherical flow representation.

A second alternative is given by considering the spherical flow on tangent planes of the sphere. For a given start point (ϕ_a, θ_a) and end point (ϕ_b, θ_b) with $\phi_b = \phi_a + \Delta_\phi$ and $\theta_b = \theta_a + \Delta_\theta$, the spherical flow is defined as the gnomonic projection of (ϕ_b, θ_b) onto the tangent plane at (ϕ_a, θ_a). As SphereNet already defines its filters on tangent planes of the sphere, this seems like a suitable choice at first glance. While again solving the discontinuities in the standard optical flow in equirectangular images, it additionally is translation invariant. Unlike the first alternative, it is a two-dimensional representation, which is easy to interpret (see Fig. 4.8e). However, the use of the gnomonic projection results in the spacing of the projected spherical coordinate grid on the tangent plane increasing very rapidly away from the center of the projection (ϕ_a, θ_a), thus severely distorting large flows. This is undesirable as it means that larger flows would be penalized more during training than to smaller flows, which could lead to a less accurate flow estimation for smaller optical flows and thus suboptimal model performance.

In comparison to defining the optical flow on tangent planes of the sphere, our proposed spherical flow representation (see Fig. 4.8f) has the same advantages (no discontinuities, translation invariance, two-dimensional) but does not distort large optical flows vectors, thereby making it the most suitable among the discussed formulations for representing optical flow in omnidirectional images.

4.2 Experimental Evaluation

We first validate our model with respect to several existing state-of-the-art methods using a simple omnidirectional MNIST classification task, before evaluating SphereNet on more challenging object detection, semantic segmentation and optical flow tasks.

4.2.1 Image Classification: Omni-MNIST

For the classification task, we create an omnidirectional MNIST dataset (*Omni-MNIST*), where MNIST digits are placed on tangent planes of the image sphere and an equirectangular image of the scene is rendered at a resolution of 60×60 pixels.

We compare the performance of our method to several baselines. First, we compare to a regular convolutional network operating on a equirectangular images (EquiCNN) or on a cube map representation of the input (CubeCNN). We combine the EquiCNN model with a Spherical Transformer Network (SphereTN) which learns to undistort parts of the image by performing a global rotation of the sphere. In addition, we evaluate the graph convolutional network of Khasanova et al. [67] and the spherical convolutional model of Cohen et al. [16], for which we use the code published by the authors. As [67] does not provide code, we reimplement their model based on code by Defferrard et al. [27].

The network architecture for all models consists of two blocks of convolution and max-pooling, followed by a fully-connected layer. We use 32 filters in the first and 64 filters in the second layer and each layer is followed by a ReLU activation. The fully connected layer has 10 output neurons and uses a softmax activation function. In the CNN and SphereNet models, the convolutional filter kernels are of size 5×5 and are applied with stride 1. Max pooling is performed with kernels of size 3×3 and a stride of 2.

Table 4.1: **Classification Results on Omni-MNIST.** Performance comparison on the omnidirectional MNIST dataset.

Method	Test error (%)	# of Parameters
GCNN [67]	17.21	282K
S2CNN [16]	11.86	149K
CubeCNN	10.03	167K
EquiCNN	9.61	196K
EquiCNN+SphereTN	8.22	291K
SphereNet (Uniform)	7.16	144K
SphereNet (NN)	7.03	196K
SphereNet (BI)	**5.59**	196K

The Spherical Transformer Network uses an identical network architecture but replaces the fully-connected output layer with a convolutional layer that outputs the parameters of the rotation. After applying the predicted transformation of the Spherical Transformer the transformed output is then used as input to the EquiCNN model.

Similarly, the graph convolutional baseline (GCNN) uses graph-conv layers with 32 and 64 filters of polynomial order 25 each, while the spherical CNN baseline (S2CNN) uses an S^2-conv layer with 32 filters and a $SO(3)$-conv layer with 64 filters. Downsampling in the S2CNN model is implemented with bandwidths of $30, 10, 6$ as suggested in [16]. Thus, all models have a comparable number of trainable model parameters (see Table 4.1). In addition, all models are trained with identical training parameters using Adam, a base learning rate of 0.0001 and batches of size 100 for 100 epochs.

Results on Omni-MNIST: Table 4.1 compares the performance of SphereNet with uniform sphere sampling (Uniform), nearest neighbor interpolation in the equirectangular image (NN) and bilinear interpolation in the equirectangular image (BI) to the baseline methods. Our results show that all three variants of SphereNet outperform all baselines.

Despite its high number of model parameters, the graph convolutional (GCNN) model struggles to solve the Omni-MNIST task. The spherical convolutional (S2CNN) model performs better but is also outperformed by all CNN-based models. For the CNN-based models, the CubeCNN has a higher test error than EquiCNN, most likely due to the discontinuities at cube borders and varying digit orientation in the top and bottom faces.

Table 4.2: **Digit Scale Evaluation on Omni-MNIST.** Performance comparison on the omnidirectional MNIST dataset for varying digit sizes.

Method	Large	Medium	Small
GCNN [67]	17.21	20.35	28.54
S2CNN [16]	11.86	19.90	38.80
CubeCNN	10.03	11.37	24.46
EquiCNN	9.61	9.10	8.60
EquiCNN+SphereTN	8.22	8.69	8.46
SphereNet (Uniform)	7.16	6.91	8.77
SphereNet (NN)	7.03	6.32	7.51
SphereNet (BI)	**5.59**	**5.03**	**5.89**

Table 4.3: **Digit Elevation Evaluation on Omni-MNIST.** Performance comparison on the omnidirectional MNIST dataset for varying digit elevation.

Method	$\phi_d \in \pm[0, \frac{\pi}{8}]$	$\phi_d \in \pm[\frac{\pi}{8}, \frac{\pi}{4}]$	$\phi_d \in \pm[\frac{\pi}{4}, \frac{3\pi}{8}]$	$\phi_d \in \pm[\frac{3\pi}{8}, \frac{\pi}{2}]$
GCNN [67]	17.48	19.48	18.39	20.63
S2CNN [16]	11.63	11.45	11.41	11.49
CubeCNN	8.70	8.70	9.10	13.80
EquiCNN	7.02	8.64	9.17	11.34
EquiCNN+SphereTN	6.46	7.76	7.74	11.08
SphereNet (Uniform)	6.25	6.87	7.18	9.14
SphereNet (NN)	6.15	6.36	6.23	8.67
SphereNet (BI)	**4.50**	**4.87**	**4.89**	**6.84**

Table 4.4: **Interpolation Evaluation for SphereNet on Omni-MNIST.** Performance comparison of SphereNet when varying the location of the bilinear interpolation (BI) layers.

Bi-layers	*none*	$conv_1$	$conv_2$	$pool_1$	$pool_2$	$conv_{1,2}$	$pool_{1,2}$	*all*
Test error	7.03	6.31	6.25	6.47	6.25	6.18	6.17	**5.59**

The performance of the EquiCNN is improved when combined with a Spherical Transformer Network (EquiCNN+SphereTN), demonstrating that the SphereTN is able to support the classification task by predicting global sphere rotations and undistorting parts of the image. However, it does not reach the performance of SphereNet, thus confirming the benefit of encoding distortion invariance into the network architecture itself.

For SphereNet, the uniform sphere sampling (Uniform) variant performs similar to the nearest neighbor (NN) variant, which demonstrates that the loss of information by uniformly sampling the sphere is negligible. SphereNet with bilinear interpolation (BI) overall performs best, improving upon all baselines by a significant margin.

In order to analyze these results more closely, we conduct further studies. First, we test the effect of digit scale on the performance of the different models. To do so, we generate the Omni-MNIST dataset at three different scales (small, medium, large) and train and evaluate each model on all three variants. The results are shown in Table 4.2 and demonstrates that the performance of the nearest neighbor (NN) and bilinear interpolation (BI) variants of SphereNet is not significantly impacted by changes in digit scale. On the other hand, the uniformly sampled variant (Uniform) drops in performance for smaller digits, as in this case important information is lost at a smaller object scale. In contract, the EquiCNN baseline performs slightly better for small digit scales, as smaller digits minimize the amount of object distortion in the equirectangular image. The CubeCNN, S2CNN and GCNN baselines all show significantly degraded performance for smaller digit sizes, with the S2CNN model particularly struggling with digits of smaller size.

Second, we evaluate the performance of all models for different ranges of digit elevation ϕ_d (see Table 4.3). While the CubeCNN and EquiCNN models perform gradually worse with increasing elevation and object distortion, the SphereNet variants offer near constant performance for elevations $\phi_d \leq \frac{3\pi}{8}$. When further investigating the decrease in SphereNet's performance for elevations $\phi_d \in \pm[\frac{3\pi}{8}, \frac{\pi}{2}]$, we find it to be caused by a sudden drop in performance to nearly 20% test error at the poles ($\phi_d = \pm\frac{\pi}{2}$) at which SphereNet's assumption of an upright object orientation no longer holds true. Unlike SphereNet, S2CNN encodes full rotation equivariance and therefore is the only baseline with near-constant performance over all digits elevations.

Finally, we evaluate which network layers benefit most from bilinear interpolation (BI) over nearest neighbor interpolation (NN). Table 4.4 indicates that all layers benefit from bilinear interpolation. The benefit is slightly larger in the second layers and increases further when both convolutional and pooling layers utilize bilinear interpolation.

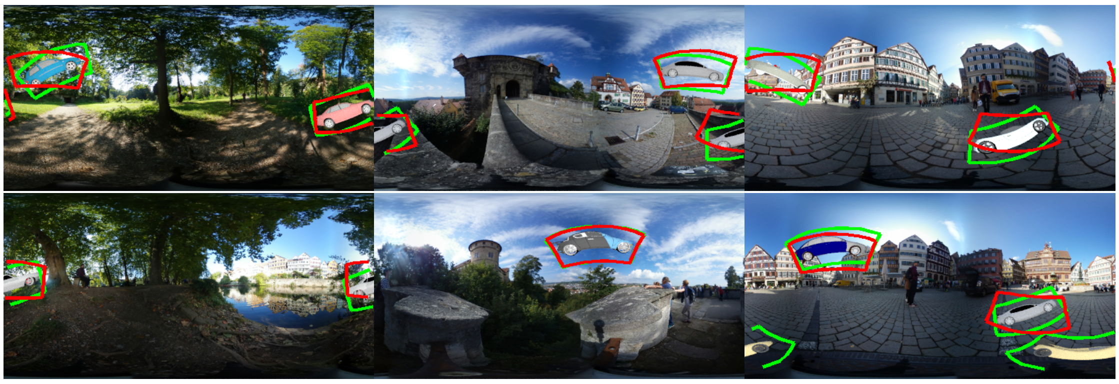


Figure 4.9: **Detection Results on FlyingCars Dataset.** The ground truth is shown in green, our SphereNet (NN) results in red.

Table 4.5: **Detection Results on FlyingCars Dataset.** All models are trained and tested on the FlyingCars dataset.

Method	Test mAP (%)	Training speed	Inference Speed
EquiCNN+SphereTN	38.91	3.0 sec/step	0.232 sec/step
EquiCNN	41.57	1.7 sec/step	0.091 sec/step
EquiCNN++	45.65	3.1 sec/step	0.175 sec/step
CubeCNN	48.42	1.8 sec/step	0.095 sec/step
SphereNet (NN)	**50.18**	2.1 sec/step	0.101 sec/step

4.2.2 Object Detection

4.2.2.1 FlyingCars

We now consider the object detection task. Due to a lack of suitable existing omnidirectional image benchmarks, we create the novel *FlyingCars* dataset. It combines real-world background images of an omnidirectional 360° action camera with rendered 3D car models. For the 3D car models we select a subset of 50 car models from the popular ShapeNet dataset [7], which are rendered onto the background images at different elevations, distances and orientations.

The scenes are rendered using an equirectangular projection to images of dimension 512×256, covering a complete 360° field of view around the camera. Each rendered scene contains between one to three cars, which may be partially occluded. Object bounding boxes are automatically extracted and represented by the lat/lon coordinates (ϕ_i, θ_i) of the object's tangent plane as well as the object width w and height h on the tangent plane and its in-plane rotation α. All ground truth coordinates are normalized to a range of $[-1.0, 1.0]$. In total, the dataset comprises $1,000$ test and $5,000$ training images, out of which a subset of $1,000$ images is used as validation set.

For this task, we integrate the nearest neighbor variant (NN) of SphereNet into the SphereSSD framework (see Section 4.1.5). Because the graph and spherical convolution baselines are not applicable to the object detection task, we compare the performance of SphereNet to a CNN operating on the cube map (CubeCNN) and equirectangular representation (EquiCNN). The latter is again tested in combination with a Spherical Transformer Network (EquiCNN+SphereTN).

Following [162] we evaluate a version of EquiCNN where the size of the convolutional kernels is enlarged towards the poles to approximate the object distortion in equirectangular images (EquiCNN++). Like [162] we limit the maximum kernel dimension to 7×7. However, unlike [162] we keep weight tying in place for image rows with filters of the same dimension, thus reducing the number of model parameters. We thereby enable regular training of the network without kernel-wise knowledge distillation as in [162]. In addition, we utilize pre-trained weights when kernel dimensions match with a pre-trained network architecture so that not all model parameters need to be trained from scratch.

As feature extractor all models use a VGG-16 network [129], which is initialized with weights pre-trained on the ILSVRC-2012-CLS dataset [118]. We change the pooling kernels to size 3×3, use ReLU activations, L_2 regularization with weight $4e^{-5}$ and batch normalization in all layers of the network. Additional convolutional box prediction layers of depth $256, 128, 128, 128$ are attached to layer *conv5_3*. Anchors of scale 0.2 to 0.95 are generated for layer *conv4_3*, *conv5_3* and the box prediction layers. The aspect ratio for all anchor boxes is fixed to the aspect ratio of the side view of the rendered cars $(2:1)$. The full network is trained end-to-end in the SphereSSD framework with the RMSProp optimizer, batches of size 5 and a learning rate of 0.004.

Results on FlyingCars: Table 4.5 presents the results for the object detection task on the FlyingCars dataset after $50,000$ steps of training. Following common practice, we use an intersection-over-union (IoU) threshold of 0.5 for evaluation. Again, our results demonstrate that SphereNet outperforms the baseline methods. Qualitative results of the SphereNet model are shown in Fig. 4.9.

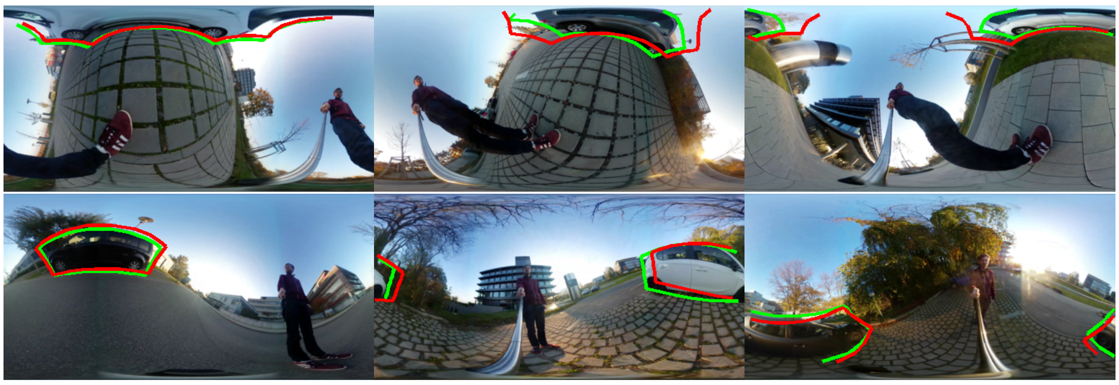


Figure 4.10: **Detection Results on OmPaCa Dataset.** The ground truth is shown in green, our SphereNet (NN) results in red.

Compared to the classification experiments, the Spherical Transformer Network (SphereTN) demonstrates less competitive performance as no transformation is able to account for undistorting all objects in the image at the same time. It is thus outperformed by the EquiCNN. The performance of the EquiCNN model is improved when the kernel size is enlarged towards the poles (EquiCNN++), but all EquiCNN models perform worse than the CNN operating on a cube map representation (CubeCNN). The reason for the improved performance of the CubeCNN compared to the classification task is most likely that discontinuities at the patch boundaries are less often present in the FlyingCars dataset due to the smaller relative size of the objects.

Besides accuracy, another important property of an object detector is its training and inference speed. Table 4.5 therefore additionally lists the training time per batch and inference time per image on an NVIDIA Tesla K20. The numbers show similar runtimes for EquiCNN and CubeCNN. SphereNet has a small runtime overhead of factor 1.1 to 1.2, while the EquiCNN++ and EquiCNN+SphereTN models have a larger runtime overhead of factor 1.8 for training and 1.9 to 2.5 for inference.

4.2.2.2 Transfer Learning: OmPaCa

In addition, we consider the transfer learning task, where a model trained on a perspective dataset is transferred to handle omnidirectional imagery. For this task we record a new real-world dataset of omnidirectional images of real cars with a handheld action camera. The images are recorded at different heights and orientations. The *omnidirectional parked cars (OmPaCa)* dataset consists of $1,200$ labeled images of size 512×256 with more

Table 4.6: **Transfer Learning Results on OmPaCa Dataset.** We transfer detection models trained on perspective images from the KITTI dataset [43] to an omnidirectional representation and finetune the models on the OmPaCa dataset.

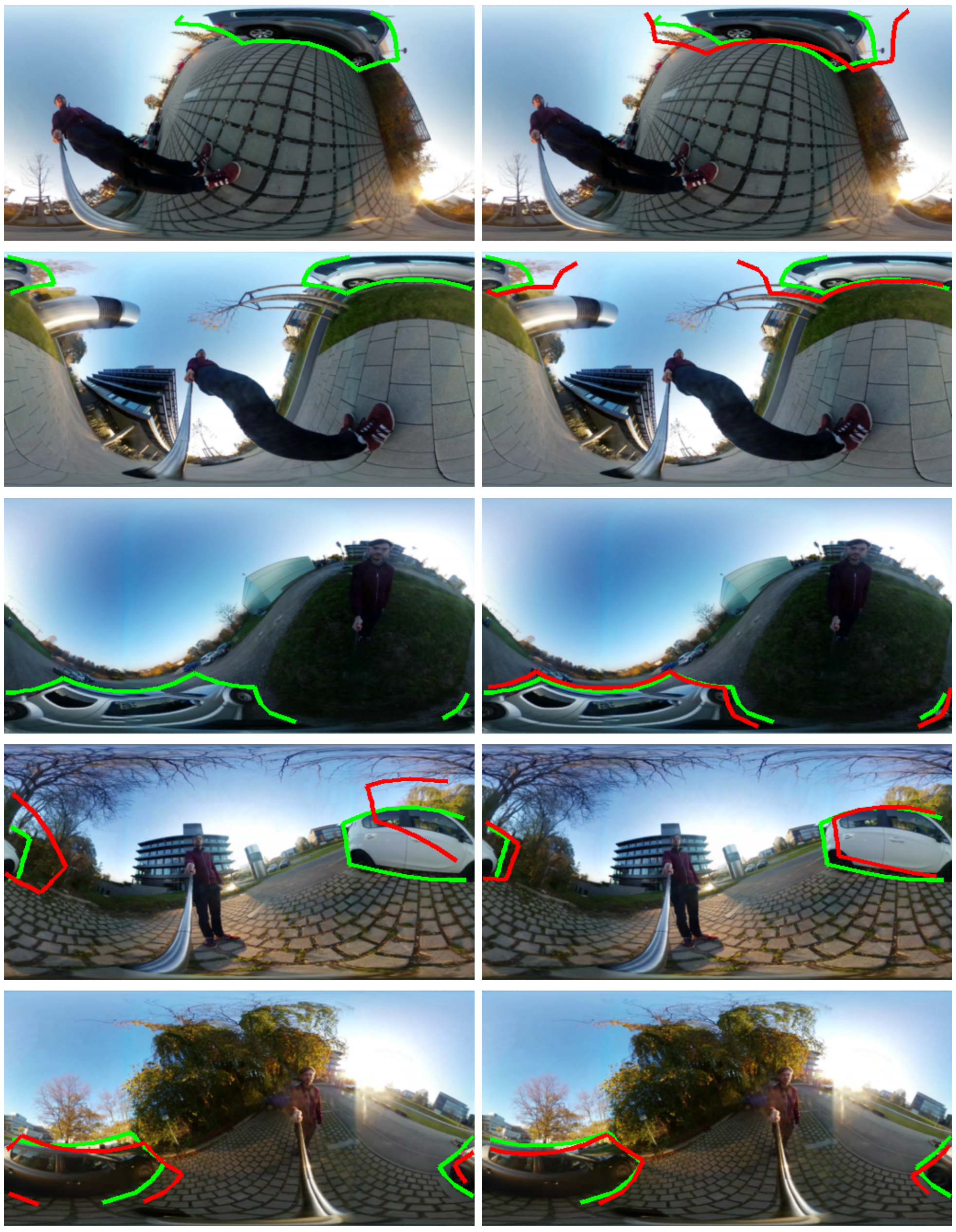
Method	Test mAP (%)
CubeCNN	34.19
EquiCNN	43.43
SphereNet (NN)	**49.73**

than 50 different car models in total. The dataset is split into 200 test and $1,000$ training instances, out of which a subset of 200 is used for validation.

We use the same detection architecture and training parameters as in Section 4.2.2.1 but now start from a perspective SSD model trained on the KITTI dataset [43], convert it to our SphereSSD framework and fine-tune for $20,000$ iterations on the OmPaCa dataset. For this experiment we only compare against the EquiCNN and CubeCNN baselines. Both the EquiCNN+SphereTN as well as the EquiCNN++ are not well suited for the transfer learning task due to the introduction of new model parameters, which are not present in the perspective detection model and which would thus require training from scratch.

Results on OmPaCa: Our results for the transfer learning task on the OmPaCa dataset are shown in Table 4.6 and demonstrate that SphereNet outperforms both baselines. Unlike in the object detection experiments on the FlyingCars dataset, the CubeCNN performs worse than the EquiCNN by a large margin of nearly 10%, indicating that the cube map representation is not well suited for the transfer of perspective models to omnidirectional images. On the other hand, SphereNet performs better than the EquiCNN by more than 5%, which confirms that the SphereNet approach is better suited for transferring perspective models to the omnidirectional case.

A selection of qualitative results for the SphereNet model is visualized in Fig. 4.10. As evidenced by our experiments, the SphereNet model is able to detect cars at different elevations on the sphere including the polar regions where regular convolutional object detectors fail due to the heavy distortions present in the input images. We provide a qualitative comparison of the detection results in Fig. 4.11.



(a) EquiCNN (b) SphereNet (NN)

Figure 4.11: **Qualitative Performance Comparison** between EquiCNN and SphereNet (NN) model on the OmPaCa dataset. The ground truth is shown in green, detections are shown in red. Unlike SphereNet, the baseline EquiCNN model struggles to detect objects in the polar regions of omnidirectional images (row $1-3$) and, in general, outputs less tight bounding boxes (row $4-5$).

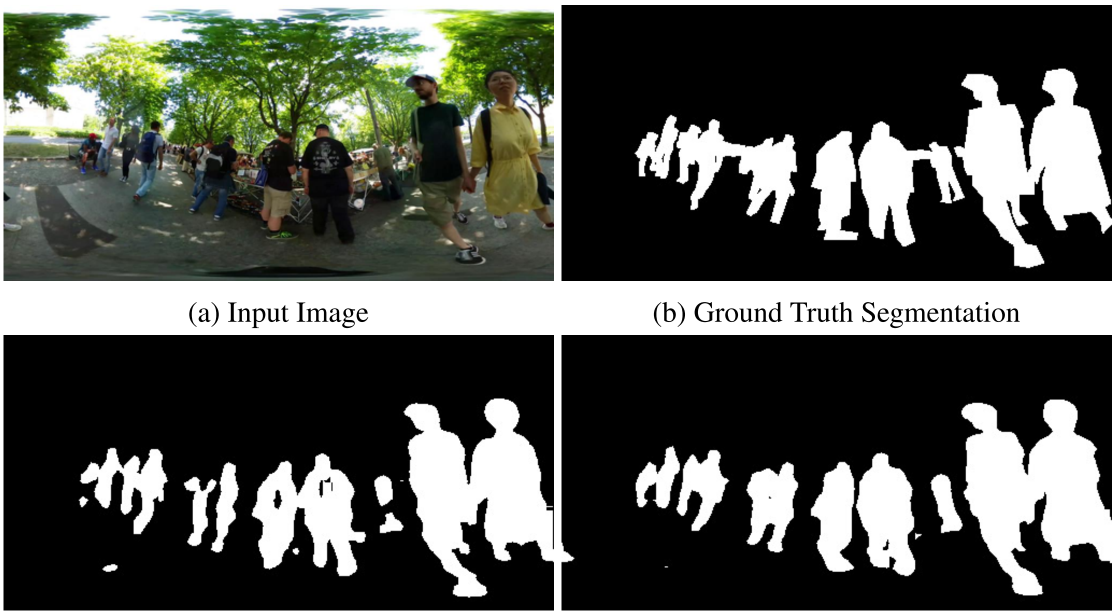

(a) Input Image
(b) Ground Truth Segmentation
(c) EquiCNN Prediction
(d) SphereNet Prediction

Figure 4.12: **Qualitative Performance Comparison** on the Stuttgart360 semantic segmentation dataset. While the EquiCNN prediction features fragmented and incomplete objects with holes, the objects in the SphereNet prediction are more complete and the prediction overall has fewer outliers.

4.2.3 Semantic Segmentation: Stuttgart360

For the semantic segmentation task we record a new real-world dataset of crowded pedestrian scenes in the city center of Stuttgart with a handheld omnidirectional camera. The *Stuttgart360* dataset is pixel-wise labeled with the classes human and background. It consists of 250 training and 50 test images of dimension 512×256. The dataset size is artificially increased during training by performing random rotations of the input images and corresponding labels which do not violate the image's overall upright orientation.

We compare our SphereNet model with nearest neighbor (NN) interpolation to an EquiCNN baseline. Both models use a fully convolutional architecture as described in [90] with a VGG-16 feature extractor [129] that is initialized with weights pre-trained on the ILSVRC-2012-CLS dataset [118].

Results on Stuttgart360: Table 4.7 shows the results after training for $20,000$ steps with batches of size 2 on the Stuttgart360 dataset, demonstrating improved performance for SphereNet compared to the EquiCNN baseline on the semantic segmentation task.

Table 4.7: **Semantic Segmentation Results on Stuttgart360 Dataset.** All models are trained and tested on the Stuttgart360 semantic segmentation dataset.

Method	Test IOU (%)	Test IOU$_{contour}$ (%)
EquiCNN	87.52	27.62
SphereNet (NN)	**88.64**	**32.16**

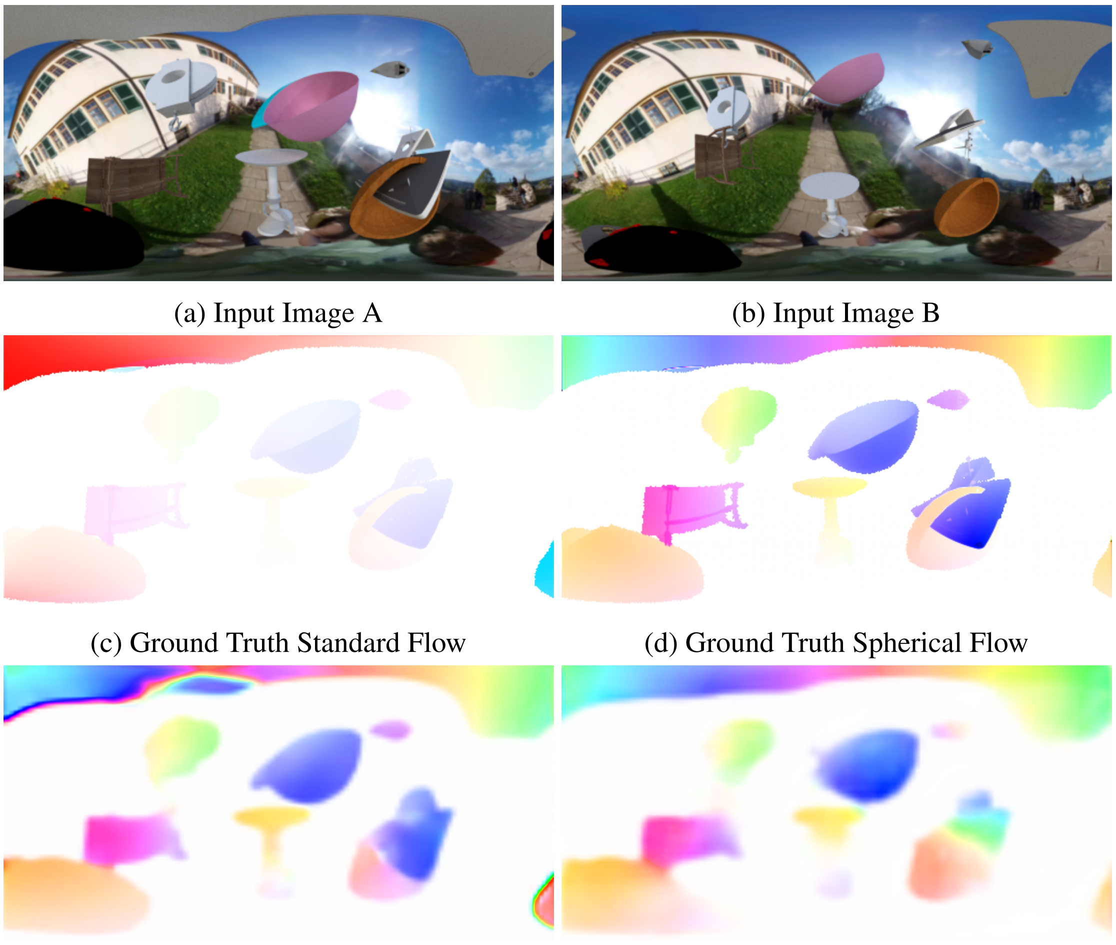
Overall, SphereNet improves upon the EquiCNN baseline by more than 1% IoU. When only evaluating the pixels on the contour of the objects, the improvement is nearly 5% IoU. This improvement is confirmed when visualizing the predictions of both models (see Fig. 4.12), where we can observe that SphereNet produces less artifacts in its output and that objects are more complete.

4.2.4 Optical Flow: FlyingThings

Finally, we consider the optical flow task. For this task we generate a novel synthetic dataset of omnidirectional 360° flow (*FlyingThings*). Similarly to the FlyingCars dataset, it is rendered in Blender at a resolution of 512×256 by placing between 5 to 10 models from the ShapeNet dataset in front of real-world omnidirectional backgrounds.

The objects are initially placed at random locations and orientations on a sphere with radius $r = 0.75$ surrounding the camera. They are moved on the sphere with randomly and uniformly chosen translations $\Delta_\phi, \Delta_\theta$ in the range $[-\frac{\pi}{8}, \frac{\pi}{8}]$ and rotated with Euler angles α, β, γ which are chosen uniformly at random from the interval $[-0.25, 0.25]$. For all rendered images the background does not move but is kept fixed. In total, the dataset consists of $5,000$ training, 500 validation and 500 test instances.

As in the semantic segmentation task, we compare SphereNet (NN) to an EquiCNN model. As a base network architecture we use the FlowNetS model, as described in [31]. For the optical flow task, we disable SphereNet in the refinement part of the FlowNetS model as we observe the use of spherical kernels in the upsampling steps leading to artifacts in the learned optical flow. We train the network for $300,000$ steps with a learning rate of 0.0001 which is halved every $50,000$ steps after the first $100,000$ training steps. While the EquiCNN model is trained with a standard optical flow representation, we utilize the proposed spherical optical flow formulation (see Section 4.1.7) for training the


(a) Input Image A (b) Input Image B

(c) Ground Truth Standard Flow (d) Ground Truth Spherical Flow

(e) EquiCNN Prediction (f) SphereNet Prediction

Figure 4.13: **Qualitative Performance Comparison** on the FlyingThings spherical optical flow dataset, indicating improved performance for SphereNet in the polar and border region of the equirectangular image.

SphereNet model. For evaluation, we transform the output of EquiCNN to the spherical optical flow format and calculate the average end-point-error (EPE).

Results on FlyingThings: The results for the spherical optical flow experiments on the FlyingThings dataset (see Table 4.8) show that the use of SphereNet in combination with the novel spherical optical flow representation improves upon the EquiCNN baseline which is trained on standard optical flow. While the quantitative improvement is small, a qualitative comparison demonstrates that SphereNet noticeably improves over the EquiCNN in the border and polar regions of the equirectangular image (see Fig. 4.13).

Table 4.8: **Optical Flow Results on FlyingThings Dataset.** All models are trained and tested on the FlyingThings optical flow dataset.

Method	Test EPE
EquiCNN	8.07e-05
SphereNet (NN)	**7.84e-05**

We expect further improvement of our model with more work on adapting the loss of the FlowNetS model to the smaller range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ of the novel spherical optical flow representation as well as a more detailed investigation on the optimal layer placement of SphereNet in the FlowNetS model, which has not been rigorously evaluated.

4.3 Conclusions

We presented SphereNet, a framework for deep learning with 360° cameras. SphereNet lifts 2D convolutional neural networks to the surface of the unit sphere. By applying 2D convolution and pooling filters directly on the sphere's surface, our model effectively encodes distortion invariance into the filters of convolutional neural networks. Wrapping the convolutional filters around the sphere further avoids discontinuities at the borders or poles of the equirectangular projection. By updating the sampling locations of the convolutional filters we allow for easily transferring perspective CNN models to handle omnidirectional inputs. Our experiments show that the proposed method improves upon a variety of strong baselines in omnidirectional image classification, object detection, semantic segmentation and optical flow.

We expect that with the increasing availability and popularity of omnidirectional sensors in both the consumer market (e.g., action cameras) as well as in industry (e.g., autonomous cars, robotics, virtual reality), the demand for specialized models for omnidirectional images such as SphereNet will increase in the near future. Here, future research could investigate how the flexibility of SphereNet could be exploited for other computer vision tasks such as scene flow estimation or single image depth estimation. Another area of future work may be to look into extending SphereNet to related camera models such as fisheye cameras or to consider barrel distortion in general.

While SphereNet addresses one important scenario (e.g. the change to a different camera model) in which we are missing labeled data, there remain other scenarios in which we are similarly lacking labeled data but are unable to encode invariance into the model or utilize a similarity loss for learning invariance.

One such scenario is the change to a novel viewpoint in which no data has yet been labeled. In this case, we are unable to use the proposed similarity loss as an augmentation of the training data is no longer trivial but would require a representation of the 3D geometry of a scene in order to perform a warp to the novel viewpoint. Similarly, it is also non-trivial to encode invariance to a change in viewpoint into a model.

An alternative to the use of a regularizer for learning invariance or the encoding of invariance into the network architecture is the adaptation of the labeled training data, which is available in a given source viewpoint, to the target viewpoint. This adaptation can be performed with the help of available prior knowledge about the viewpoint change in a learning-based manner. After adapting the source dataset to the desired target viewpoint, a network can then be trained for viewpoint invariance by combining the adapted source data with data from other viewpoints such as the original source viewpoint. Alternatively, the network can also be trained only with the adapted source data if viewpoint invariance is not required and the best model performance in the target viewpoint is desired.

In the next chapter, we introduce the Novel Viewpoint Adaptation (NoVA) model, which utilizes available prior knowledge about a viewpoint change in order to perform such a data-driven adaptation from a source to a novel target viewpoint.

Chapter 5

NoVA: Learning to See in Novel Viewpoints and Domains

In this chapter, we present the Novel Viewpoint Adaptation (NoVA) model which enables an adaptation from a viewpoint in which a large labeled dataset is available to a novel viewpoint, potentially within a different domain, in which little or no labeled data is available. For this chapter, we focus on the task of semantic segmentation which is highly relevant for both autonomous driving as well as advanced driver-assistance. However, the presented model is also applicable to other computer vision tasks with little to no extensions required.

Deep neural networks for semantic segmentation require huge labeled datasets. However, as labeling is expensive and time-consuming, the re-use and transfer of existing labeled datasets is desirable whenever possible. Yet, in many cases available datasets do not exactly match the setup of the problem we are interested in but instead differ in style (e.g. simulation vs. reality), camera model (e.g. rectilinear vs. omnidirectional) or in camera viewpoint, which is the focus of this chapter.

While the adaptation to a different image style has been extensively addressed in previous domain adaptation work [4, 56, 125, 146], viewpoint adaptation has not yet been widely considered. However, as demonstrated by our experiments, such a change in viewpoint can lead to a dramatic performance drop. Thus, we formally introduce the challenge of domain and viewpoint adaptation and propose the *Novel Viewpoint Adaptation* (NoVA) model. NoVA enables the adaptation of source domain data to the view and style of a target domain in which no labeled data is available.

Specifically, we investigate the adaptation of a semantic segmentation model for the task of autonomous driving. In this field of autonomous driving, large labeled datasets exist [23, 42]. However, most of them are recorded from similar viewpoints, hindering

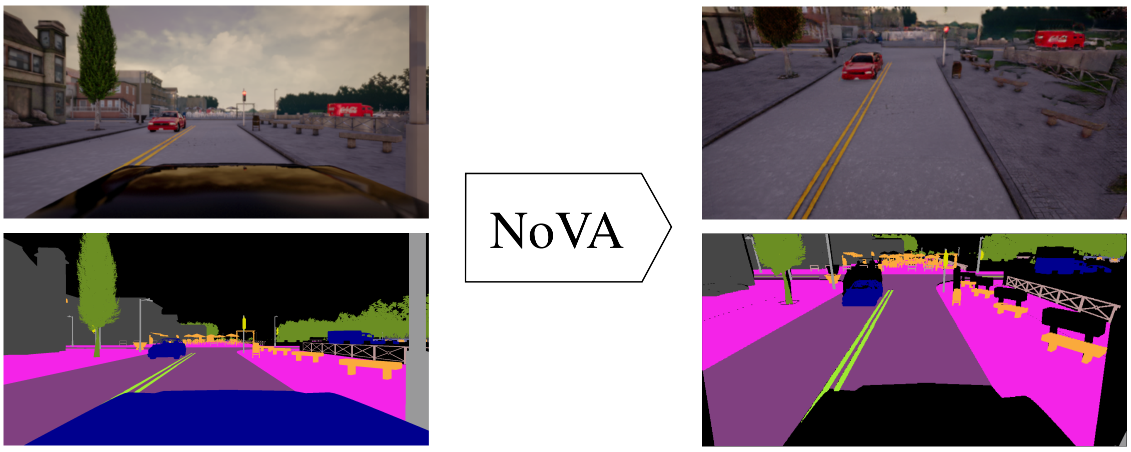


Figure 5.1: NoVA enables the adaptation from a source domain view to a novel viewpoint in a target domain. It performs a geometry-aware image and label translation from a source (left) to a target (right) view, in which no labels exist.

their application to novel viewpoints. Here, NoVA enables the re-use and adaptation of datasets to the novel viewpoints of autonomous buses, trucks or drones.

State-of-the-art domain adaptation approaches such as CyCADA [56] and SPLAT [146] perform an image-level adaptation to translate source images to the style of the target domain. Yet, these approaches struggle when faced with performing a semantically consistent translation from the source domain to the novel viewpoint of the target domain, as evidenced in our experiments. One reason for this is that, unlike NoVA, CyCADA and SPLAT do not utilize an explicit depth representation of the scene but instead need to learn the perspective transformation to a novel viewpoint end-to-end. While some recent image-translation works utilize a depth representation to simulate synthetic foggy images [122] or to preserve semantic information during image translation [11], we exploit depth cues for adapting to a novel viewpoint.

A further limitation of current domain adaptation models is that they do not take a translation of the source view segmentation labels into consideration. This is problematic as the translated source images are no longer compatible with the original source labels, which leads to a mismatch when using the translated source view images in combination with the original source view labels for training the task segmentation network. In contrast, NoVA utilizes its explicit representation of the scene geometry in order to translate both the source view images and the source view labels to the target domain viewpoint.

The adaptation to a different camera is considered in some recent works but is limited to an adaptation to a different camera style [164, 165] or to novel camera intrinsics [36]. With regards to adapting to a different viewpoint, Di Mauro et al. [28] consider the adaptation to a single novel camera view. They do not perform image translation but instead propose an encoder-decoder model in which the latent code corresponds to a semantic segmentation map of the input. It is trained with a segmentation loss (source images only), a reconstruction loss and an adversarial loss. Thus, unlike NoVA, their model is not able to provide any task supervision in the target domain. Our experiments confirm that NoVA compares favorably to their SceneAdapt method.

In very recent work, Tran et al. [142] propose a domain adaptation approach which, similarly to NoVA, draws on ideas from novel view synthesis. They utilize a keypoint-based appearance flow for a perspective transformation of source images to a novel viewpoint and perform a photometric refinement using a CycleGAN. However, unlike NoVA, they do not utilize a dense depth estimation but instead use a sparse representation of a small number (i.e. 36) of 2D object keypoints. In contrast to NoVA, which is able to utilize self-supervision for its depth estimators and supports depth estimation for complex multi-object scenes, their keypoint localization network requires ground truth depth for training and only supports the localization of keypoints for a single foreground object.

In the field of novel view synthesis (NVS), several models have been proposed for generating novel views from a single input image. While some directly predict a novel view using an encoder-decoder architecture [76,140,159] or generative adversarial network [143, 160], others utilize the appearance flow, a dense flow field that specifies how to warp the input to the target view [109, 167]. However, flow-based warping can lead to the distortion of local structures in the output.

Liu et al. [86] demonstrated that an explicit representation of the 3D scene geometry improves upon flow-based view synthesis. The benefit of an explicit depth representation has also been confirmed in other recent NVS works [10, 12]. NoVA builds on this insight but extends the geometry-aware image warping to unsupervised depth estimation models [47, 166] and integrates it into a framework for domain and viewpoint adaptation.

Our NoVA pipeline is split into four stages, which can be trained jointly or independently. First, we estimate the scene geometry by predicting a depth map from a source image. Next, we utilize the predicted depth map as well as prior knowledge about the transformation between the two viewpoints, which we assume to be given, to forward warp the source image and label to the target view. A refinement network then performs inpainting of occluded areas and stylizes the warped image in the style of the target domain. Finally, we train a target segmentation network with the translated source data.

Thereby, NoVA effectively reduces the domain and viewpoint adaptation task to the well-studied problems of depth estimation [33, 47, 166] and image inpainting [110, 158] / stylization [56, 146], for which deep neural networks have already demonstrated remarkable performance. While NoVA builds on recent advances in supervised and self-supervised depth estimation, it utilizes a novel residual refinement network which enables the model to focus on filling in occluded image areas and updating the overall image style without having to synthesize a new image from scratch.

Compared to current image-level domain adaptation approaches, which focus on directly translating the source images to the target domain with a single generative model, NoVA uses a modular architecture that utilizes an explicit representation of the scene geometry. This enables NoVA to perform a geometry-aware translation of both source images and labels to the target domain viewpoint. In addition, it makes it possible to efficiently utilize information about how the source and target domain viewpoints are related. This prior knowledge is commonly available yet not used by current state-of-the-art domain adaptation models that are designed to mainly account for a change in image style and not for a change in viewpoint.

We demonstrate the benefit of using NoVA over existing domain adaptation approaches for adapting to a novel viewpoint within a simulation environment as well as for adapting from simulation to a complex real-world dataset.

In summary, this chapter makes the following **contributions**:

- We introduce the task of domain and viewpoint adaptation, a variant of the domain adaptation task for which the domains do not only differ in style but also correspond to different viewpoints. In particular, we consider the unsupervised adaptation task, in which no labels are available in the target domain viewpoint.

- We improve upon current domain adaptation models by using an explicit representation of the scene geometry that enables NoVA to forward warp source view images and labels to the target domain. Thereby, the viewpoint change itself no longer has to be learned and instead the task is reduced to the well-studied problems of depth estimation and image inpainting/stylization.

- We demonstrate improved performance compared to state-of-the-art domain adaptation models on synthetic and challenging real-world datasets.

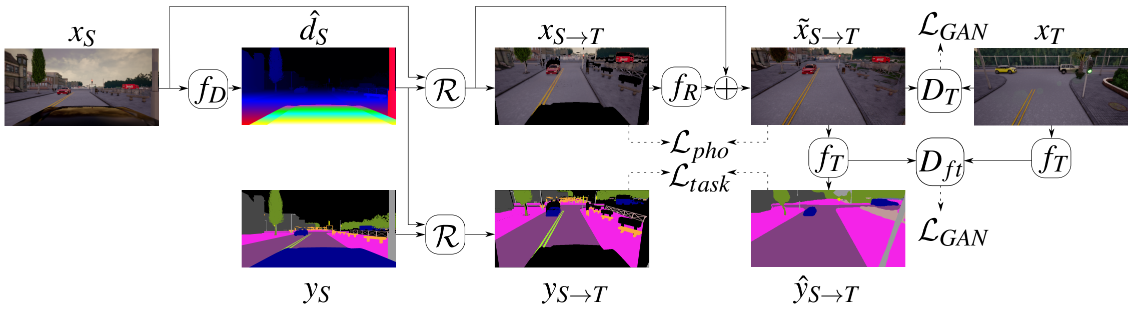


Figure 5.2: **NoVA Pipeline.** A depth estimation model f_D estimates a depth $\hat{d}_S$ given a source image x_S. Based on $\hat{d}_S$, the source image x_S and corresponding label y_S are warped by a differentiable rendering operator $\mathcal{R}$ to the target domain viewpoint. The warped image $x_{S\to T}$ is refined by a residual refinement network f_R to create a refined image $\tilde{x}_{S\to T}$. A discriminator D_T ensures the realism of the refined image, while a photometric consistency loss $\mathcal{L}_{pho}$ ensures consistency between the warped and refined image. A segmentation model f_T that predicts $\hat{y}_{S\to T}$ is trained with a task loss $\mathcal{L}_{task}$ on the refined images $\tilde{x}_{S\to T}$ and warped labels $y_{S\to T}$. During training of f_T, a feature-level discriminator D_{ft} ensures alignment between features of refined images $\tilde{x}_{S\to T}$ and target images x_T.

5.1 Method

This section introduces our Novel Viewpoint Adaptation (NoVA) framework. NoVA performs a geometry-aware translation of source domain data to the target domain. It is an unsupervised and unpaired method, requiring no annotations in the target domain nor corresponding image pairs in the source and target domain.

The NoVA pipeline is split into four stages (see Fig. 5.2). In a first step, a depth map is estimated from a given source domain image. Next, the source image and label are forward warped to the viewpoint of the target domain. Afterwards, occluded areas are inpainted and the style of the warped image is adapted to the style of the target domain by a refinement network. Finally, the translated images and labels are utilized to train a target segmentation network.

Problem Setup. We consider the challenging problem of domain and viewpoint adaptation for the task of semantic segmentation. More precisely, we consider unsupervised adaptation, where we are provided with a set of images X_S and labels Y_S in the source domain and with unpaired images X_T and no labels in the target domain. In addition, we assume to know the transformation between the source and target viewpoint $V_{S\to T} = (K_S, K_T, R_{S\to T}, t_{S\to T})$, where K_S, K_T are the camera intrinsics and $R_{S\to T}, t_{S\to T}$ is the transformation between source and target view. Based on the source dataset (X_S, Y_S)

we can train a source segmentation model f_S, parameterized by a CNN with weights W_s, for K-way classification using a cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{task}(f_S, X_S, Y_S) = \\ -\mathbb{E}_{(x_S, y_S)\sim(X_S, Y_S)} \sum_{k=1}^{K} \mathbb{1}_{k=y_S} \log\left(\sigma(f_S^{(k)}(x_S|W_s))\right)\end{aligned} \tag{5.1}$$

where σ is the softmax function.

However, the source model will not perform well on images from the target viewpoint, as evidenced in our experiments (see Section 5.2). Thus, we aim to train a target segmentation network f_T that is optimized for target domain images.

Depth Estimation. In order to utilize the prior knowledge about the transformation between the source and target viewpoints, which is encapsulated in $V_{S\to T}$, we utilize an explicit depth representation of the scene. In addition, this enables us to not only translate the source domain images but also the corresponding source labels. Given a source view image $x_S \sim X_S$, a depth estimation network f_D, parameterized by a CNN with weights W_d, estimates a depth map $\hat{d}_S = f_D(x_S|W_d)$.

Rendering. Given a predicted source view depth map $\hat{d}_S$, we warp the source view image x_S as well as the corresponding source view label $y_S \sim Y_S$ to the target view. Using a differentiable rendering operator $\mathcal{R}$ we generate a target view image $x_{S\to T} = \mathcal{R}(x_S, \hat{d}_S, V_{S\to T})$ and target view label $y_{S\to T} = \mathcal{R}(y_S, \hat{d}_S, V_{S\to T})$ according to $V_{S\to T}$. While we allow backpropagation through $\mathcal{R}$ for warping the source images x_S , we stop the gradients from backpropagating through $\mathcal{R}$ when warping the source labels y_S.

While self-supervised monocular depth estimation methods utilize inverse warping to provide supervision via view synthesis [41, 166], our problem setup, which considers unpaired images, necessitates the use of forward warping. As in [144], we utilize a forward-splatting approach. We first perform a forward projection of each pixel p_S in the source image to the pixels in the target frame p_T using the inverse predicted depth $\hat{d}_S^{-1}$, the camera intrinsics K_S, K_T and the transformation between the views $R_{S\to T}, t_{S\to T}$:

$$\begin{bmatrix} p_T^x \\ p_T^y \\ 1 \\ \hat{d}_T^{-1} \end{bmatrix} \sim \begin{bmatrix} K_T & \hat{0} \\ \hat{0} & 1 \end{bmatrix} \begin{bmatrix} R_{S\to T} & t_{S\to T} \\ \hat{0} & 1 \end{bmatrix} \begin{bmatrix} K_S^{-1} & \hat{0} \\ \hat{0} & 1 \end{bmatrix} \begin{bmatrix} p_S^x \\ p_S^y \\ 1 \\ \hat{d}_S^{-1} \end{bmatrix} \tag{5.2}$$

The target image is initialized with an empty canvas onto which the projected source pixels are splatted. Several source pixels may map to the same target pixel, thus we require the use of z-buffering to deal with occlusions. For this, we use a differentiable *soft* z-buffer where the contribution of each source pixel to a target pixel is weighted according to its inverse depth in the target view $\hat{d}_T^{-1}$. Finally, the image is normalized by a weighted average of the contributions of points which splat to a given target pixel. For more details, we refer the reader to [144].

Target View Refinement. As the target view may contain new scene content, which was occluded or outside of the camera frame in the source view image, we refine the warped image $x_{S \to T}$ by inpainting blank image areas and stylizing the image in the target domain style. This task is performed by a refinement network f_R, which is parameterized by a CNN with weights W_r. This network is not tasked with synthesizing an image from scratch but instead only needs to output a residual r which is added to the warped source image $x_{S \to T}$ before the hyperbolic tangent activation in the network's last layer to create the refined image $\tilde{x}_{S \to T} = \tanh(x_{S \to T} + r)$. By modeling the refinement with a residual connection, we make the inpainting task easier for the refinement network to learn and encourage the network to keep the overall image structure of the warped image in the refined image.

The supervision for training the refinement network is provided by a discriminator network D_T that is trained with an adversarial loss $\mathcal{L}_{GAN}$:

$$\begin{aligned} &\mathcal{L}_{GAN}(\tilde{G}_{S \to T}, D_T, X_T, X_S) \\ &\quad = \mathbb{E}_{x_T \sim X_T}[\log D_T(x_T)] \\ &\quad + \mathbb{E}_{x_S \sim X_S}[\log(1 - D_T(\tilde{G}_{S \to T}(x_S)))] \end{aligned} \tag{5.3}$$

where the depth estimation, forward rendering and refinement steps are encapsulated into a single virtual generator step $\tilde{G}_{S \to T} = f_R(\mathcal{R}(x_S, f_D(x_S|W_d), V_{S \to T})|W_r)$.

As we do not assume to have any target labels available, we cannot apply the same unsupervised refinement approach to the warped source labels $\hat{y}_{S \to T}$. Instead, we use the warped source labels directly without refinement to provide a sparse supervision signal for training the target segmentation network f_T, in which the task loss is only applied in regions with label information.

Enforcing Photometric Refinement Consistency. An important aspect of the refinement step is that the original scene structure and content of the warped source image should be preserved. In order to enforce consistency between the warped and the refined image, we propose a lightweight photometric refinement loss that penalizes differences between the warped and the refined source pixels:

$$\begin{aligned}\mathcal{L}_{pho}(G_{S\to T}, \tilde{G}_{S\to T}, X_S) = \\ \lambda_{pho}\,\mathbb{E}_{x_S\sim X_S}[||G_{S\to T}(x_S) - \tilde{G}_{S\to T}(x_S)||_1]\end{aligned} \tag{5.4}$$

where $G_{S\to T} = \mathcal{R}(x_S, f_D(x_S|W_d), V_{S\to T})$ and where λ_{pho} is a binary pixel-wise mask. The weight λ_{pho} is 0 for empty pixels in the warped image $x_{S\to T}$, onto which no source pixel was mapped, and 1 for all non-empty pixels.

While the refinement network has the freedom to change any warped source pixel if desired, $\mathcal{L}_{pho}$ encourages the refinement network to effectively act as an inpainting model. However, even when the source and target domains do not only differ in viewpoint but also in their overall image style, we find the photometric refinement loss to be a lightweight and effective alternative to a semantic [146] or cycle consistency loss [56].

Target Network Training. The target task network f_T, which is parameterized by a CNN with weights W_t, is trained with the translated source domain data. Let us denote the translated source view dataset as $(\tilde{X}_{S\to T}, Y_{S\to T})$ where $\tilde{X}_{S\to T} = \{f_R(\mathcal{R}(x_S, f_D(x_S|W_d), V_{S\to T})|W_r)|x_S \in X_S\}$, $Y_{S\to T} = \{\mathcal{R}(y_S, f_D(x_S|W_d), V_{S\to T})|x_S \in X_S, y_S \in Y_S\}$.

For training f_T we again utilize the cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{task}(f_T, \tilde{X}_{S\to T}, Y_{S\to T}) = \\ -\,\mathbb{E}_{(\tilde{x}_{S\to T}, y_{S\to T})\sim(\tilde{X}_{S\to T}, Y_{S\to T})} \\ \sum_{k=1}^{K} \mathbb{1}_{k=y_{S\to T}} \log\left(\sigma(f_T^{(k)}(\tilde{x}_{S\to T}|W_t))\right)\end{aligned} \tag{5.5}$$

In addition, we perform feature-level alignment of f_T between the the target images X_T and the refined images $\tilde{X}_{S\to T}$. For this, we add a discriminator D_{ft} to distinguish between features of target images and refined images:

$$\begin{aligned}\mathcal{L}_{GAN}(f_T, D_{ft}, f_T(\tilde{X}_{S\to T}|W_t), X_T) = \\ \mathbb{E}_{\tilde{x}_{S\to T}\sim\tilde{X}_{S\to T}}[\log D_{ft}(f_T(\tilde{x}_{S\to T}|W_t))] + \\ \mathbb{E}_{x_T\sim X_T}[\log(1 - D_{ft}(f_T(x_T|W_t)))]\end{aligned} \tag{5.6}$$

Overall Learning Objective. Our complete learning objective encapsulates the above losses, which optimize for target view segmentation ($\mathcal{L}_{task}$, see Eq. (5.5)), image refinement ($\mathcal{L}_{GAN}$, see Eq. (5.3)), photometric consistency ($\mathcal{L}_{pho}$, see Eq. (5.4)) and feature alignment ($\mathcal{L}_{GAN}$, see Eq. (5.6)):

$$
\begin{aligned}
&\mathcal{L}_{NoVA}(f_T, \tilde{G}_{S\to T}, D_T, D_{ft}, X_S, X_T, Y_S) \\
&\quad = \mathcal{L}_{task}(f_T, \tilde{X}_{S\to T}, Y_{S\to T}) \\
&\quad + \mathcal{L}_{GAN}(\tilde{G}_{S\to T}, D_T, X_T, X_S) \\
&\quad + \mathcal{L}_{pho}(G_{S\to T}, \tilde{G}_{S\to T}, X_S) \\
&\quad + \mathcal{L}_{GAN}(f_T, D_{ft}, f_T(\tilde{X}_{S\to T}|W_t), X_T)
\end{aligned}
\tag{5.7}
$$

5.2 Experiments

In order to demonstrate the effectiveness of NoVA, we perform experiments to adapt to a novel viewpoint within a simulation environment (see Section 5.2.2) as well as from simulation to a complex real environment (see Section 5.2.3). In Section 5.2.1 we present an overview of our experimental setup for both sets of experiments.

5.2.1 Experimental Setup

Datasets We utilize synthetic data generated in CARLA [32] as well as the real-world dataset CityScapes [23]. In the CARLA simulation framework, we generate data from a car and a truck viewpoint. For both views we generate 30 train, 15 test and 5 validation sequences of $1,000$ frames each, where every frame consists of a stereo RGB image pair, a semantic segmentation label and a depth map of resolution 2048×1024. In CityScapes, we use 2975 train and 500 test frames with fine annotations. Details on the semantic segmentation classes and on the viewpoint transformation for both experiments is presented in the Appendix (see Section A.1).

Baselines. The naïve baseline for all of our experiments is to train a segmentation model f_S on source data only. In addition, we use two state-of-the-art image-level domain adaptation models, CyCADA [56] and SPLAT [146], as well as the SceneAdapt model by Maura et al. [28] which is aims at adapting to a novel scene (i.e. a novel static viewpoint) within the same domain. It should be noted that CyCADA and SPLAT have been proposed for the general task of domain adaptation and not for domain and viewpoint adaptation.

For CyCADA, we follow the generator and discriminator architectures of [169]. The input to both networks is resized to 512×256. We train CyCADA with the Adam optimizer, single image batches and a learning rate of 0.0002 for 20 epochs after which the learning rate is linearly decayed to zero over the course of the next 20 epochs. We adopt the same training scheme for SPLAT but replace the cycle-consistency with a semantic-consistency loss that uses the source segmentation model. The SceneAdapt model is constructed from an encoder based on our base segmentation network (see below) and a decoder based on the generator models of CyCADA and SPLAT. The discriminator architecture and training scheme are consistent with CyCADA and SPLAT.

NoVA. While our differentiable rendering formulation enables a joint end-to-end training of the complete NoVA pipeline, we found it beneficial to train NoVA in stages.

For depth estimation, we evaluate three classes of estimators: A self-supervised, monocular approach [47], a supervised monocular approach [59] and a supervised stereo approach [8]. Each approach is trained on source images of resolution 512×256 as outlined in the respective paper. For rendering, we bilinearly upsample the predicted depth maps to match the source resolution of 2048×1024. In order to avoid empty pixels in the warped output, the rendering operator $\mathcal{R}$ outputs images and labels which are downscaled by a factor of 4. The forward warped images are refined by a residual refinement network f_R that is based on the CyCADA generator architecture which is modified to include a residual connection. The overall training scheme remains unchanged.

Segmentation Model. For the segmentation model we use a VGG16-FCN8s [90]. We train it for $100,000$ steps with batches of size 4 using a learning rate of $1e-3$ with SGD and momentum of 0.9. Feature-level adaptation is performed as described in [56].

Evaluation Metrics. We consider the metrics of mean intersection-over-union (mIoU), frequency-weighted intersection-over-union (fwIoU) and pixel accuracy (pixAcc):

$$\text{mIoU} = \frac{1}{C}\sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{5.8}$$

$$\text{fwIoU} = \frac{1}{\sum_k t_k}\sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{5.9}$$

$$\text{pixAcc} = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{5.10}$$

where C is the number of segmentation classes, n_{ij} is the number of pixels of class i predicted as class j and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i.

5.2.2 Sim2Sim

For a first set of experiments, we aim to evaluate the task of viewpoint adaptation only. To this effect, we select two domains which only differ in viewpoint but not in style.

Table 5.1 presents the results for adapting from a car to a truck viewpoint in the CARLA simulation [32]. It shows that NoVA outperforms the scene and domain adaptation baselines by a large margin on the task of viewpoint adaptation.

In fact, given ground truth depth, NoVA is able to get very close to reaching the performance of a target oracle model, which is trained on the labeled target data. For predicted depth estimation, we find that all variants of NoVA still compare favorably to the baselines. Here, the supervised stereo model (*stereo-sup*) outperforms the monocular self-supervised (*mono-self*) and the monocular supervised (*mono-sup*) approach.

For SceneAdapt, we find that it improves over the source segmentation model for the viewpoint adaptation task. As for CyCADA and SPLAT, we find that both struggle with the task. Because they are designed for domain adaptation and not for viewpoint adaptation, they do not take a translation of the source labels to the target viewpoint into consideration. This leads to a mismatch between the translated source images and the original source labels. Indeed, we see that the performances of CyCADA and SPLAT improve when we combine their translated images with the actual target labels. However, even when using target view labels, their performances do not reach the level of NoVA. This indicates that NoVA's image translation pipeline is overall better adapted to the task of viewpoint adaptation.

When inspecting the translated images of CyCADA and SPLAT (see Fig. 5.5), we find that a shortcoming for both models is that the semantics of translated images are not always consistent with the semantics of the source images. Interestingly, we find that CyCADA is nonetheless able to reconstruct the source from the translated image well, which suggests that it has learned to encode some of the source semantics in the noise of the translated image (see Fig. 5.9) [13].

On the other hand, qualitative results for NoVA (see Fig. 5.3 and Fig. 5.4) demonstrate that NoVA is effective in retaining the source image semantics in the refined images. As shown by the ablation study in Table 5.2, training with forward warped image-label pairs $(x_{S\to T}, y_{S\to T})$ already results in a large performance gain in comparison to the source segmentation model. NoVA's residual refinement further improve NoVA's performance over a default refinement model that synthesizes its output image from scratch.

Table 5.1: **Results for Viewpoint Adaptation on Sim2Sim.** When tasked with adapting a semantic segmentation model from a car to a truck viewpoint, in which no labels are available, NoVA outperforms current state-of-the-art domain and scene adaptation baselines and closes the gap to a target oracle model, which is trained on labeled target data.

Method	mIoU	fwIoU	pixAcc
Source Only	26.54	43.56	55.82
SceneAdapt [28]	26.63	54.65	68.15
CyCADA [56]	10.57	21.44	30.36
CyCADA [56] + *trgt-labels*	16.31	45.89	62.55
SPLAT [146]	13.63	22.77	32.26
SPLAT [146] + *trgt-labels*	18.81	45.12	59.29
NoVA$_{mono-self}$	42.54	69.99	79.99
NoVA$_{mono-sup}$	45.27	71.20	80.49
NoVA$_{stereo-sup}$	49.67	76.44	84.97
NoVA$_{GT}$	**51.89**	**78.66**	**86.69**
Target Oracle	52.72	79.96	87.81

Table 5.2: **Ablation Study for NoVA$_{GT}$ on Sim2Sim.** When the source and the target domain are separated by a change in viewpoint only, NoVA's forward warping of source images and labels to the target domain yields the largest performance improvement over training with source data only. A residual refinement, which inpaints occluded areas in the forward warped images, improves over a default refinement, that needs to synthesize a complete new image from scratch.

Method	mIoU	fwIoU	pixAcc
Source Only	26.54	43.56	55.82
+ Forward Warping	47.81	75.74	84.11
+ Default Refinement	49.24	76.91	85.08
+ Residual Refinement	**51.89**	**78.66**	**86.69**

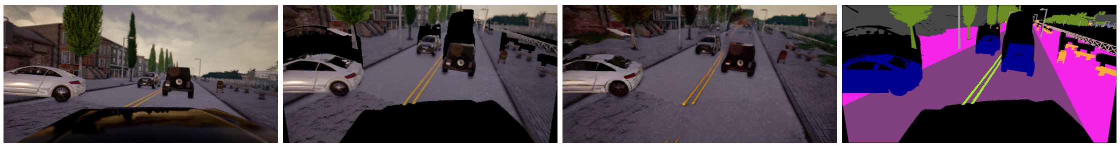

(a) Source Image (b) Warped Image (c) Refined Image (d) Warped Label

Figure 5.3: **NoVA$_{GT}$ Performance on Sim2Sim.** Given a source view frame, NoVA forward warps the source image and label to the target viewpoint and refines the warped image by inpainting occluded image areas with a residual refinement network.

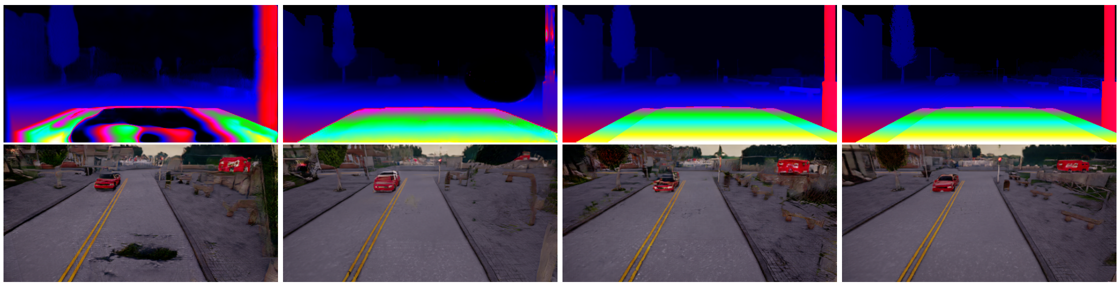

(a) NoVA$_{mono-self}$ (b) NoVA$_{mono-sup}$ (c) NoVA$_{stereo-sup}$ (d) NoVA$_{GT}$

Figure 5.4: **NoVA Performance for Different Depth Estimators.** NoVA can utilize self-supervised and supervised monocular or stereo depth estimation models as well as ground truth depth maps (top row: predicted depth, bottom row: refined images).

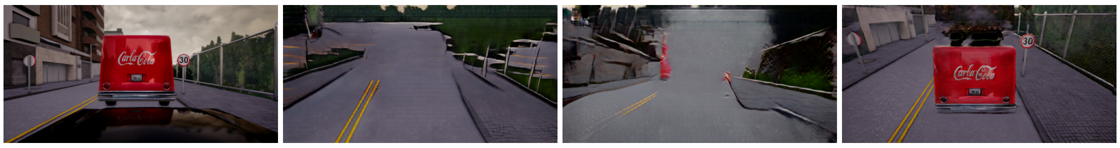

(a) Source Image (b) CyCADA (c) SPLAT (d) NoVA$_{GT}$

Figure 5.5: **Qualitative Comparison of NoVA$_{GT}$ to the Baselines on Sim2Sim.** In contrast to NoVA, the CyCADA and SPLAT baseline models do not ensure a semantic consistency between the source image and the translated source image.

5.2.3 Sim2Real

In a second set of experiments we investigate the adaptation to a novel viewpoint in a novel domain. For this, we aim to adapt from a truck viewpoint in the CARLA simulation to a car viewpoint in the complex real-world dataset of CityScapes. In this setup, the domain gap is now not only caused by a change in camera viewpoint but also by a change in image style and overall scene complexity. We restrict our evaluation to the subset of CityScapes classes which are present in CARLA.

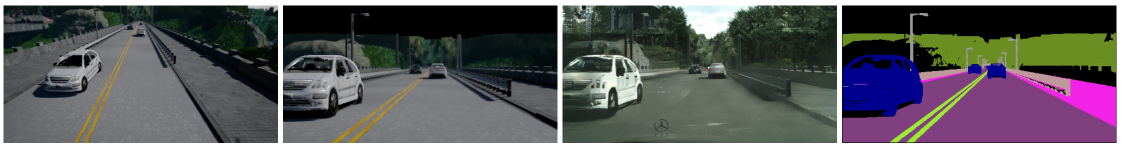

(a) Source Image (b) Warped Image (c) Refined Image (d) Warped Label

Figure 5.6: **NoVA$_{GT}$ Performance on Sim2Real.** NoVA is able to adapt to the novel viewpoint and style of the CityScapes dataset by forward warping to the target view and refining the warped images in the style of the target domain.

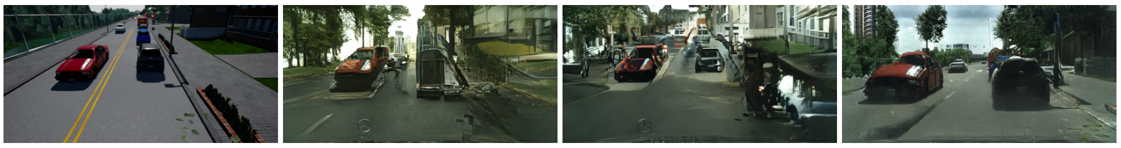

(a) Source Image (b) CyCADA (c) SPLAT (d) NoVA$_{GT}$

Figure 5.7: **Qualitative Comparison of NoVA$_{GT}$ to the Baselines on Sim2Real.** In contrast to NoVA, the CyCADA and SPLAT baseline models fail to correctly adapt semantic objects (e.g. cars) from the source to the target domain viewpoint.

Table 5.3 demonstrates that NoVA also improves upon the baseline methods in closing the domain gap from a synthetic source domain to the novel viewpoint of a challenging real-world target domain. Unlike for the viewpoint adaptation experiments, SceneAdapt now yields no performance improvements over training a segmentation model with source data only. This suggests that the SceneAdapt model is better suited for the viewpoint adaptation task than for the joint domain and viewpoint adaptation task.

CyCADA and SPLAT demonstrate performance improvements with respect to the source model on the Sim2Real task. However, they still perform worse than all NoVA variants and a qualitative comparison to NoVA (see Fig. 5.7) reveals that they struggle to correctly warp the appearance of semantic objects (e.g. cars) to the viewpoint of the target domain. In the case of SPLAT, this can be explained by its semantic consistency loss, which encourages semantic objects to reappear in the translated image at the same spatial location as in the original source image.

Despite not using a cycle or semantic consistency loss, qualitative results in Fig. 5.6 and Fig. 5.7 confirm that NoVA preserves the scene semantics well when adapting to a novel domain. This suggests that NoVA's explicit forward warping in combination with its residual refinement and photometric refinement loss offer a lightweight yet effective alternative for ensuring consistency between source images and translated source images.

Table 5.3: **Results for Domain and Viewpoint Adaptation on Sim2Real.** When the source and target domain differ in both viewpoint and style, NoVA again significantly outperforms the state-of-the-art adaptation baseline models.

Method	mIoU	fwIoU	pixAcc
Source Only	18.84	37.34	47.59
SceneAdapt [28]	11.54	30.65	37.23
CyCADA [56]	19.26	43.90	56.43
SPLAT [146]	21.01	49.42	60.99
NoVA$_{mono-self}$	30.23	60.32	72.09
NoVA$_{mono-sup}$	34.36	66.83	78.25
NoVA$_{stereo-sup}$	32.96	63.95	75.09
NoVA$_{GT}$	**35.91**	**69.52**	**80.84**
Target Oracle	51.30	79.82	88.36

Table 5.4: **Ablation Study for NoVA$_{GT}$ on Sim2Real.** While we again find forward warping to be highly beneficial, residual refinement now yields a large improvement over training with the forward warped data as the refinement adapts the warped images to the style of the target domain.

Method	mIoU	fwIoU	pixAcc
Source Only	18.84	37.34	47.59
+ Forward Warping	26.95	55.23	69.72
+ Default Refinement	30.41	58.30	68.97
+ Residual Refinement	**35.91**	**69.52**	**80.84**

In a second ablation study (see Table 5.4), we find NoVA's forward warping component again to be highly beneficial. However, compared to the results on the Sim2Sim task, the residual refinement now improves upon the forward warping significantly, as the domains are now also separated by a change in image style.

As opposed to the results on the Sim2Sim task, NoVA is now unable to fully close the gap to the target oracle model that is trained on the labeled target data. We suspect this may be due to CARLA lacking CityScapes' overall diversity. Here, semi-supervised adaptation with a limited number of labeled CityScapes examples can further improve NoVA's performance and help to close the gap to the target oracle model.

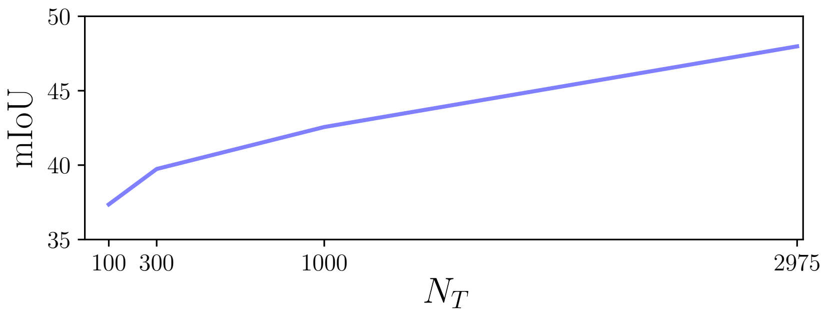


Figure 5.8: **Semi-Supervised Adaptation on Sim2Real.** Performance improves when we combine NoVA$_{GT}$'s translated source domain dataset of size $N_S = 30,000$ with a set of labeled target examples from CityScapes of size N_T.

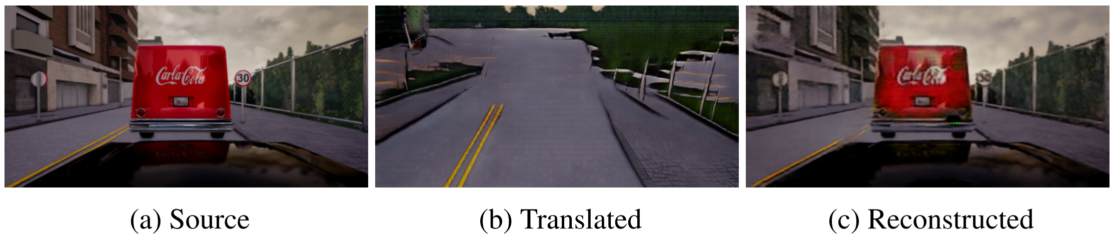


Figure 5.9: **CyCADA on Sim2Sim.** CyCADA learns to encode some of the source image semantics in the noise of the translated image for the reconstruction of the source image.

As Fig. 5.8 visualizes, combining $N_T = 300$ labeled CityScapes frames with NoVA's $N_S = 30,000$ translated source frames can already boost NoVA's mIoU-performance by about 5% wrt. an unsupervised adaptation.

We present further image translation examples as well as qualitative semantic segmentation results for NoVA and the baselines on both Sim2Sim and Sim2Real in Section A.2.

5.3 Conclusions

In this chapter, we introduced NoVA, a new model for adapting to novel viewpoints and domains. NoVA performs a geometry-aware translation of source domain images and labels to a target domain, in which no labeled examples are available. Our experiments on the task of semantic segmentation demonstrate that NoVA significantly improves over state-of-the-art domain adaptation models for adapting to novel views in simulation and complex real world datasets.

An area of future work is to adapt the NoVA pipeline to support the generation of higher resolution target domain data, possibly by incorporating an additional upscaling step in the pipeline. Furthermore, the realism of the refined images is an obvious area for improvement, where it could be beneficial to take over some of the recent advances from the field of generative modeling. Lastly, while NoVA supports an end-to-end training of its pipeline, we found it more stable to train the individual NoVA networks in stages. Here, it may be interesting to investigate if and how an end-to-end training of NoVA can be made more stable and performant.

NoVA is the third and final of our proposed approaches for incorporating prior knowledge in the form of invariances into deep convolutional neural networks. In the next chapter, we will conclude this thesis and give a brief outlook on possible directions of future work.

Chapter 6

Summary

This thesis has proposed three approaches for the incorporation of prior knowledge in the form of invariance into the training or architecture of deep convolutional neural networks for tasks such as image classification, object detection, semantic segmentation or optical flow. All three presented approaches have focused on the application areas of autonomous driving and advanced driver-assistance systems in which the training of robust computer vision models even from little labeled training data is of particularly high importance.

Our first proposed unsupervised similarity loss uses the invariance properties of image labels under a given set of geometric transformations of the input data. Thereby, the similarity loss acts as an effective regularizer when performing supervised learning from small labeled datasets and additionally makes it possible to utilize unlabeled training data in order to perform semi-supervised learning of an image classification model.

Second, the proposed SphereNet framework enables the learning of spherical representations for a variety of computer vision tasks in omnidirectional images. By encoding invariance to the distortions which are present in the equirectangular representation of omnidirectional images into the filters of convolutional neural networks, SphereNet enables a transfer of models from the perspective domain, in which large labeled datasets are commonly available, to the omnidirectional domain, in which large datasets are scarce.

Third, the Novel Viewpoint Adaptation (NoVA) model performs an adaptation from a viewpoint, in which a large labeled dataset is available, to a novel viewpoint, potentially in a different domain, in which little or even no labeled data is available. By utilizing the prior knowledge about the transformation between the two viewpoints, NoVA performs a geometrically-aware translation of source view images and labels to the target domain viewpoint and thus enables the training of a task network that performs well for target view images.

We expect future work to continue to explore all three of the presented research directions as well as flexible combinations of them for incorporating prior knowledge of invariances or equivariances into deep neural networks. Indeed, the field of invariant and equivariant deep neural networks is currently gaining more attention in the computer vision research community and the number of published papers on invariant or equivariant neural networks continues to grow.

While many of the current invariant network architectures have so far focused on the task of classification, we have shown that invariance can also be highly beneficial for more complex vision tasks. In fact, the extension of invariant and equivariant deep models to tasks such as object detection or semantic segmentation is a fundamental requirement to enable their use in many real-world applications. These include autonomous driving and advanced driver-assistance systems, where the detection of other traffic participants or the segmentation of the drivable path are core parts of any perception stack.

Yet, the adaptation of deep invariant/equivariant models in real-world applications does not only depend on their ability to handle more complex vision tasks but also on whether they can be fast and resource-efficient enough for the desired use-case. In the automotive context, this generally means that a perception model must be real-time capable and also that it must often fit on an embedded system with limited processing power and memory. Here, additional research and engineering work will be required to enable the transfer of newly developed research ideas into real-world production environments.

In general, a challenge remains to adapt established and proven network architectures, learning procedures or software frameworks to support these novel models and training approaches. However, this will be a crucial step in order to enable their widespread use by practitioners outside of research circles and thereby establish these models as parts of the common deep learning toolboxes and as cornerstones of modern computer vision.

Appendix A

Appendix

In this Appendix, we present more details on the datasets which were used in our NoVA experiments and provide additional qualitative results for NoVA as well as the baselines.

A.1 NoVA Datasets

In this section, we provide additional details on the simulation and real-world datasets which were used for our Sim2Sim and Sim2Real experiments.

A.1.1 Semantic Segmentation Classes

In total, we use 9 semantic classes. The mapping from dataset IDs and classes to training labels for CARLA [32] and CityScapes [23] is defined in Table A.1, where -1 denotes classes which are ignored during training. For CARLA, the pedestrian class is removed as there are no pedestrians in the generated dataset. For CityScapes, classes which are semantically equivalent to CARLA classes are mapped to the respective CARLA training labels. For example, as CARLA features both cars and trucks in its vehicles class, the respective CityScapes classes are mapped to the same training label as CARLA's vehicles class.

A.1.2 Viewpoint Transformation

NoVA utilizes the transformation $V_{S\to T}$ between the source and target domain viewpoints in order to warp the source images and labels to the target domain viewpoint. $V_{S\to T}$ is defined wrt. a coordinate system in which the x-axis points in the driving direction, the y-axis points left and the z-axis points up.

(a) CARLA Mapping

ID	**Class**	**Label**
0	*None*	-1
1	Buildings	0
2	Fences	1
3	*Other*	-1
4	*Pedestrians*	-1
5	Poles	2
6	RoadLines	3
7	Roads	3
8	Sidewalks	4
9	Vegetation	5
10	Vehicles	6
11	Walls	7
12	TrafficSigns	8

(b) CityScapes Mapping

ID	**Class**	**Label**
-1	*License Plate*	-1
0	*Unlabeled*	-1
1	*Ego Vehicle*	-1
2	*Rectification Border*	-1
3	*Out of RoI*	-1
4	*Static*	-1
5	*Dynamic*	-1
6	*Ground*	-1
7	Road	3
8	Sidewalk	4
9	*Parking*	-1
10	*Rail Track*	-1
11	Building	0
12	Wall	7
13	Fence	1
14	*Guard Rail*	-1
15	*Bridge*	-1
16	*Tunnel*	-1
17	Pole	2
18	*Polegroup*	-1
19	Traffic Light	8
20	Traffic Sign	8
21	Vegetation	5
22	*Terrain*	-1
23	*Sky*	-1
24	*Person*	-1
25	*Rider*	-1
26	Car	6
27	Truck	6
28	*Bus*	-1
29	*Caravan*	-1
30	*Trailer*	-1
31	*Train*	-1
32	*Motorcycle*	-1
33	*Bicycle*	-1

Table A.1: Mapping of dataset IDs and classes to training labels in CARLA and CityScapes.

A.1.2.1 Sim2Sim

For Sim2Sim, the transformation between the car viewpoint in CARLA and the truck viewpoint in CARLA is given by $V_{Sim2Sim} = (K_{CARLA}, K_{CARLA}, R_{Sim2Sim}, t_{Sim2Sim})$ where K_{CARLA} is:

$$K_{CARLA} = \begin{bmatrix} 1024 & 0 & 1024 \\ 0 & 1024 & 512 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

The rotation and translation between the two viewpoints is derived from the extrinsic parameters of the two cameras. The car camera points straight forward (no rotation along any axis) and is translated by $t_{car} = \begin{bmatrix} 0.30 & -0.11 & 1.30 \end{bmatrix}^T$. In contrast, the truck camera is rotated around the pitch axis by $\beta_y = -22.5°$ and translated by $2m$ wrt. the car camera $t_{truck} = \begin{bmatrix} 0.30 & -0.11 & 3.30 \end{bmatrix}^T$. Thus, the extrinsic transformation between the two viewpoints for the Sim2Sim task is $R_{Sim2Sim} = R_y(22.5°)$ where R_y is the rotation matrix around the y-axis and $t_{Sim2Sim} = \begin{bmatrix} 0 & 0 & 2 \end{bmatrix}^T$

A.1.2.2 Sim2Real

For Sim2Real, the transformation between the truck viewpoint in CARLA and the car viewpoint in CityScapes is given by $V_{Sim2Real} = (K_{CARLA}, K_{CityScapes}, R_{Sim2Real}, t_{Sim2Real})$ where K_{CARLA} is defined as in Eq. (A.1) and $K_{CityScapes}$ is:

$$K_{CityScapes} = \begin{bmatrix} 2262.52 & 0 & 1096.98 \\ 0 & 2265.30 & 513.14 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

The extrinsic parameters of the truck camera in CARLA remain unchanged. In CityScapes, the camera is rotated around the pitch axis by $\beta = 2.18°$ and around the yaw axis by $\gamma = 1.12°$. It is translated by $t_{CityScapes} = \begin{bmatrix} 1.70 & 0.10 & 1.22 \end{bmatrix}^T$. Thus, the extrinsic transformation between the truck viewpoint in CARLA and the car viewpoint in CityScapes is given by $R_{Sim2Real} = R_z(1.12°)R_y(-20.32°)$, $t_{Sim2Real} = \begin{bmatrix} 1.40 & 0.21 & -2.08 \end{bmatrix}^T$.

A.2 Additional Qualitative Results for NoVA

In this section, we present additional qualitative results for image translation and semantic segmentation of the NoVA model as well as the baselines for the Sim2Sim and Sim2Real tasks.

A.2.1 Sim2Sim

Figures A.1 to A.4 visualize the intermediary forward warped images, refined warped images and forward warped labels for the different variants of NoVA. For the NoVA variants that use depth estimation models instead of ground truth depth maps, we find that a bilinear upsampling of the predicted depth maps from a resolution of 512×256 to a resolution of 2048×1024 causes some artifacts in the warped images and labels. However, the refinement model is able to remove some of these artifacts in the subsequent image refinement step.

As Fig. A.5 shows, the CyCADA [56] and SPLAT [146] domain adaptation baselines struggle with the viewpoint adaptation task. In contrast to NoVA, they are not designed for viewpoint adaptation but for the more general task of domain adaptation. Thus, they do not utilize an explicit depth representation but instead need to learn the perspective transformation end-to-end to perform an image translation to a novel viewpoint. Furthermore, they only consider a translation of the source images to the target domain viewpoint, which leads to a mismatch when using the translated source images in combination with the original source labels for training the task segmentation network.

A comparison of the semantic segmentation performance of the baselines and NoVA is visualized in Fig. A.7 for a VGG16-FCN8s network architecture [90], which was utilized in the experiments of the main paper, as well as in Fig. A.6 for a DRN-26 model [161], which was trained and evaluated in addition for this supplementary material. For the Sim2Sim viewpoint adaptation task, we find that neither the source model that was trained with the original source view dataset nor the domain adaptation baselines perform well. In contrast, NoVA demonstrates good performance and comes close to matching the output of a target oracle model that was trained with the labeled target viewpoint dataset.

A.2.2 Sim2Real

For the Sim2Real task, Figures A.8 to A.11 show the image and label translation for the different variants of NoVA. We find that as the style of the CARLA and CityScapes domains differs significantly, the refinement network does not only inpaint the warped images but also adapts their overall style to the style of the target domain. Importantly, the refinement retains the overall semantic structure of the warped images so that the refined images are still consistent with the warped source labels.

The image translation for the CyCADA and SPLAT baselines is visualized in Fig. A.12. We find that while it retains some of the source image semantics in the translated image and successfully adapts the source images to the style of the target domain, it does not correctly warp semantic objects (i.e. cars) to the target domain viewpoint. In the case of SPLAT, this can be explained by its semantic consistency loss which encourages semantic objects to reappear in the same image locations as in the original source segmentation maps.

The semantic segmentation performance for the baselines and NoVA is visualized in Fig. A.14 for a VGG16-FCN8s [90] and in Fig. A.13 for a DRN-26 [161] model. In contrast to Sim2Sim, CyCADA and SPLAT now show improved performance over a source segmentation model, whereas SceneAdapt [28] does not appear to be well suited for adapting to a view in a novel domain. As before, NoVA compares favorably to the baselines and gets close to matching the performance of a target oracle.

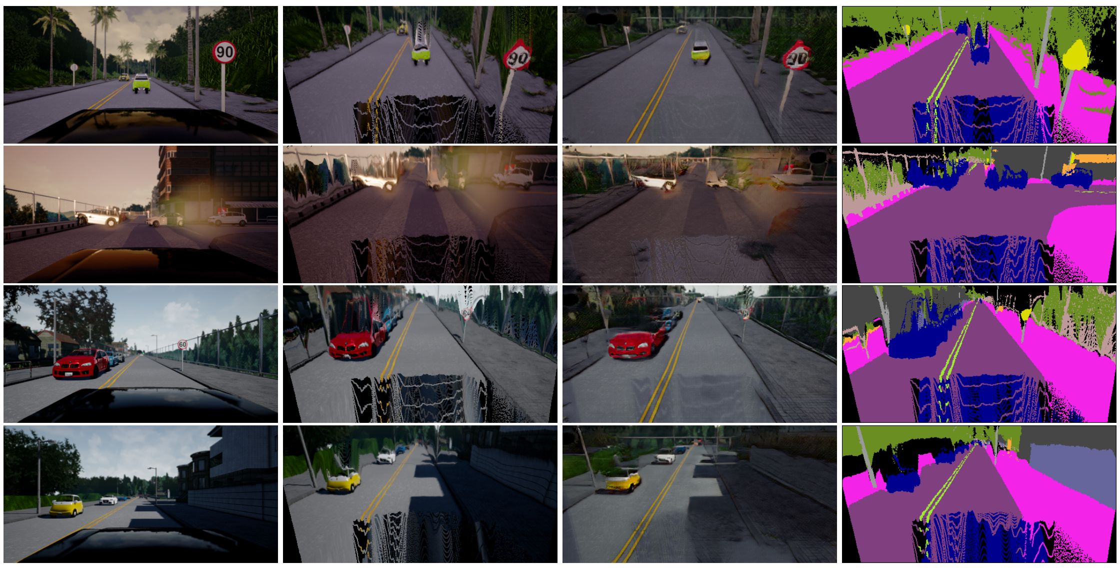

(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.1: **NoVA$_{mono-self}$ Translation on Sim2Sim.**

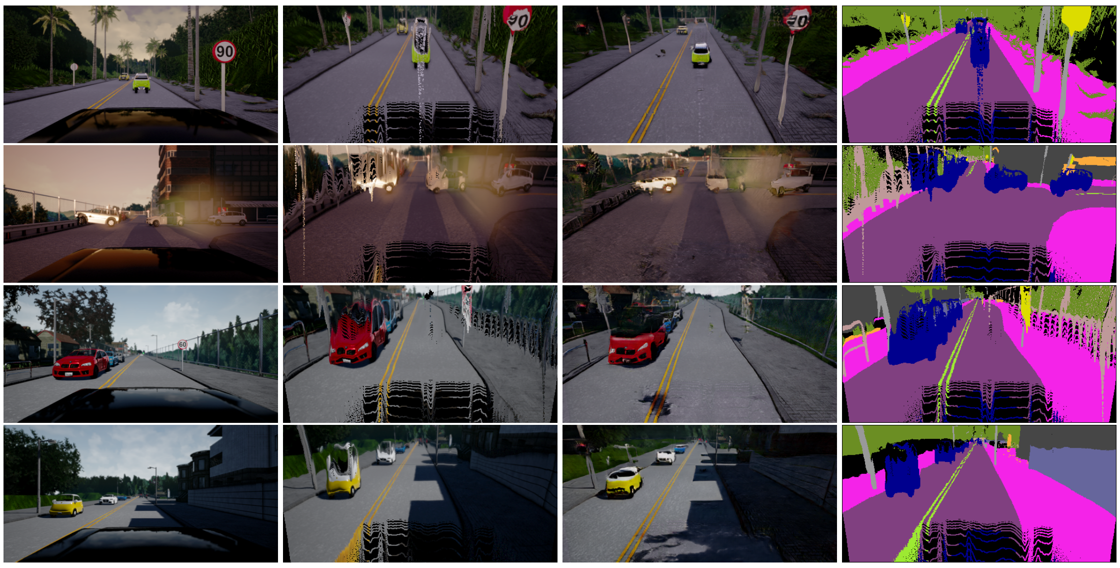
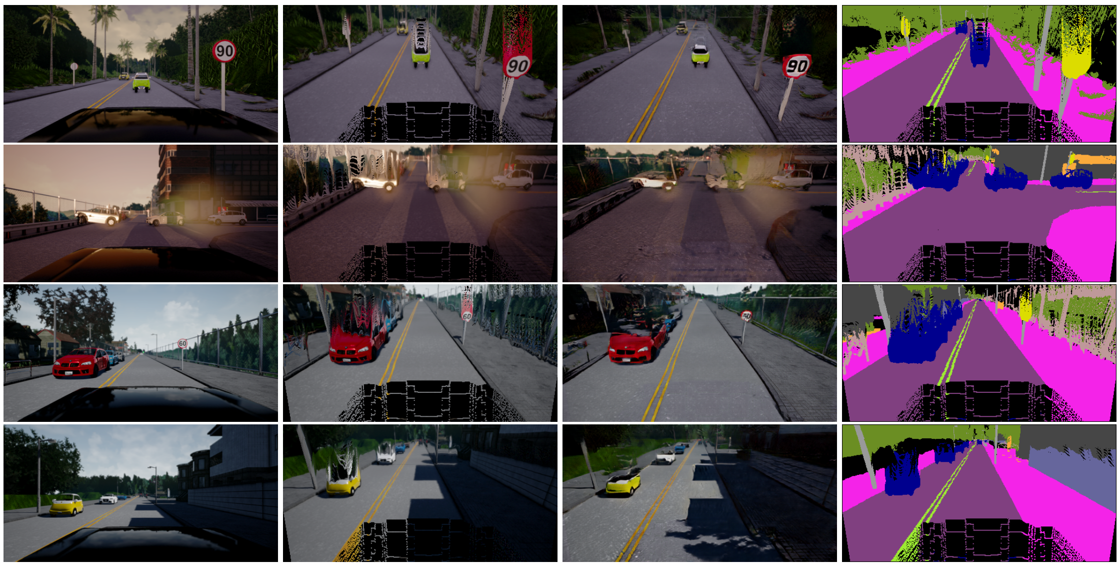
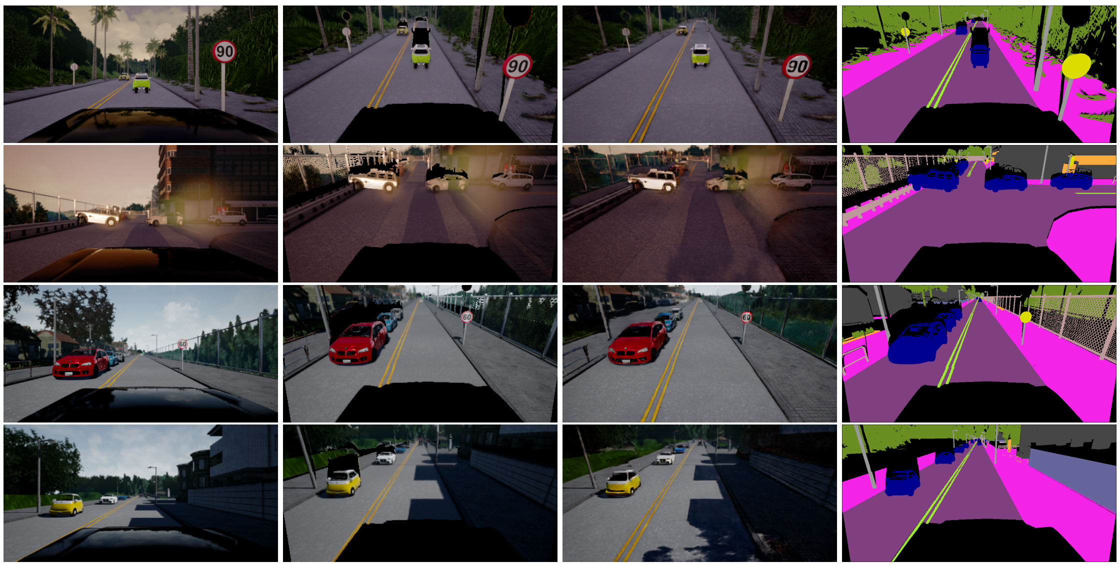

(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.2: **NoVA$_{mono-sup}$ Translation on Sim2Sim.**



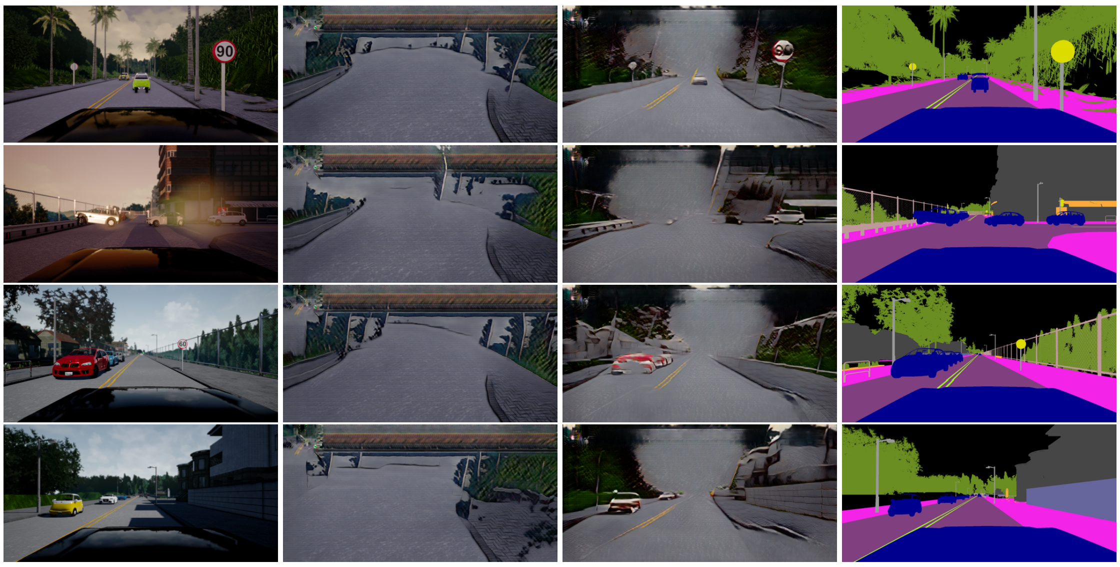
(a) Input Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.3: **NoVA$_{stereo-sup}$ Translation on Sim2Sim.**



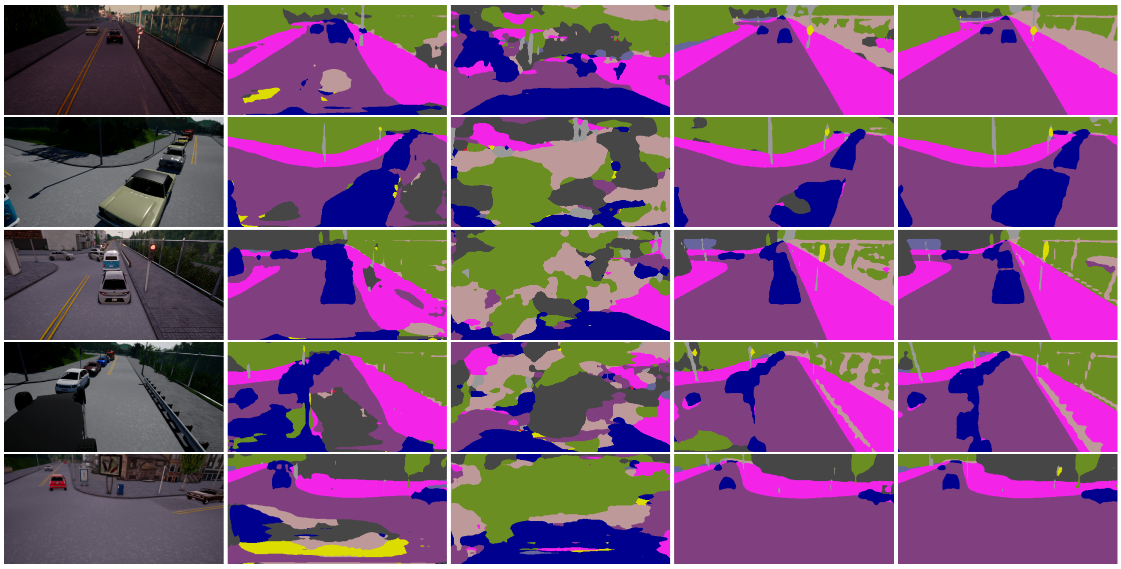
(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.4: **NoVA$_{GT}$ Translation on Sim2Sim.**



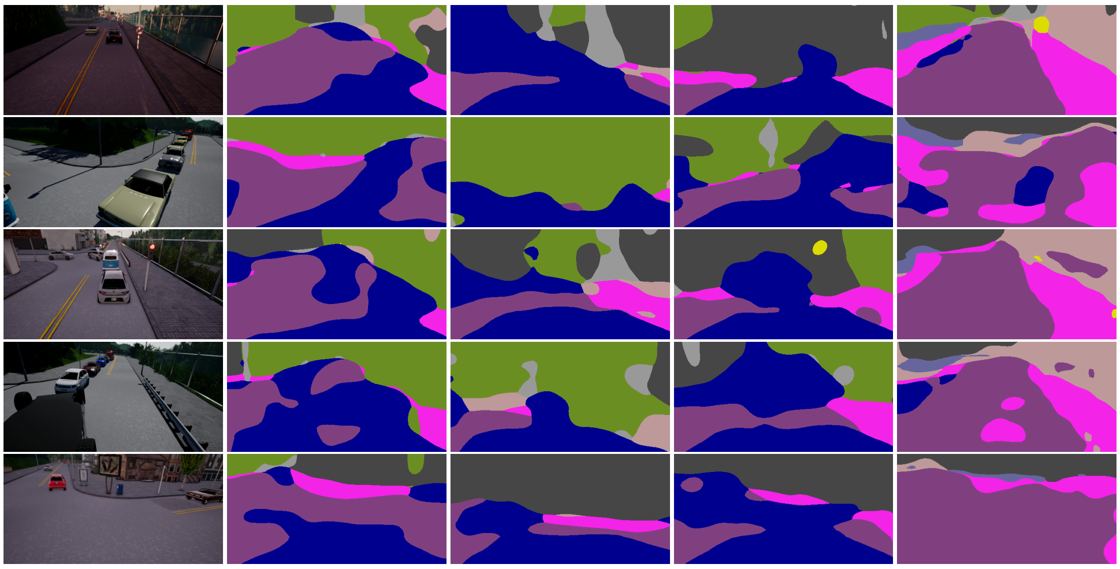
(a) Source Images (b) CyCADA (c) SPLAT (d) Source Labels

Figure A.5: **CyCADA and SPLAT Translation on Sim2Sim.**

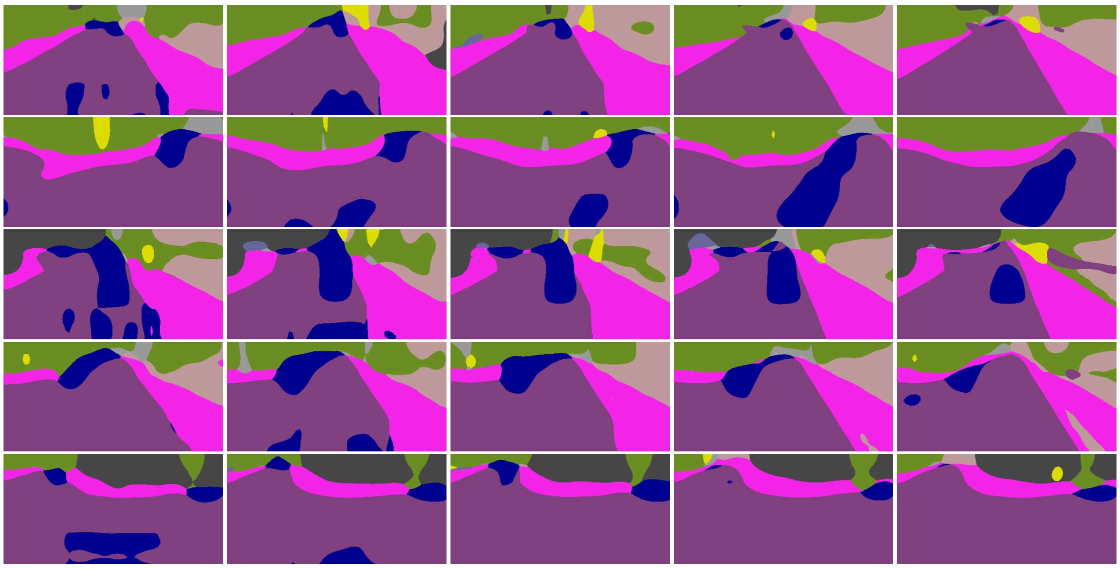


(a) Target Images (b) Source Model (c) CyCADA (d) NoVA$_{GT}$ (e) Target Oracle

Figure A.6: **Comparison of Semantic Segmentation Performance on Sim2Sim for a DRN-26 Model.**



(a) Target Images (b) Source Model (c) CyCADA (d) SPLAT (e) SceneAdapt



(f) NoVA$_{mono-self}$ (g) NoVA$_{mono-sup}$ (h) NoVA$_{stereo-sup}$ (i) NoVA$_{GT}$ (j) Target Oracle

Figure A.7: **Comparison of Semantic Segmentation Performance on Sim2Sim for a VGG16-FCN8s Model.**

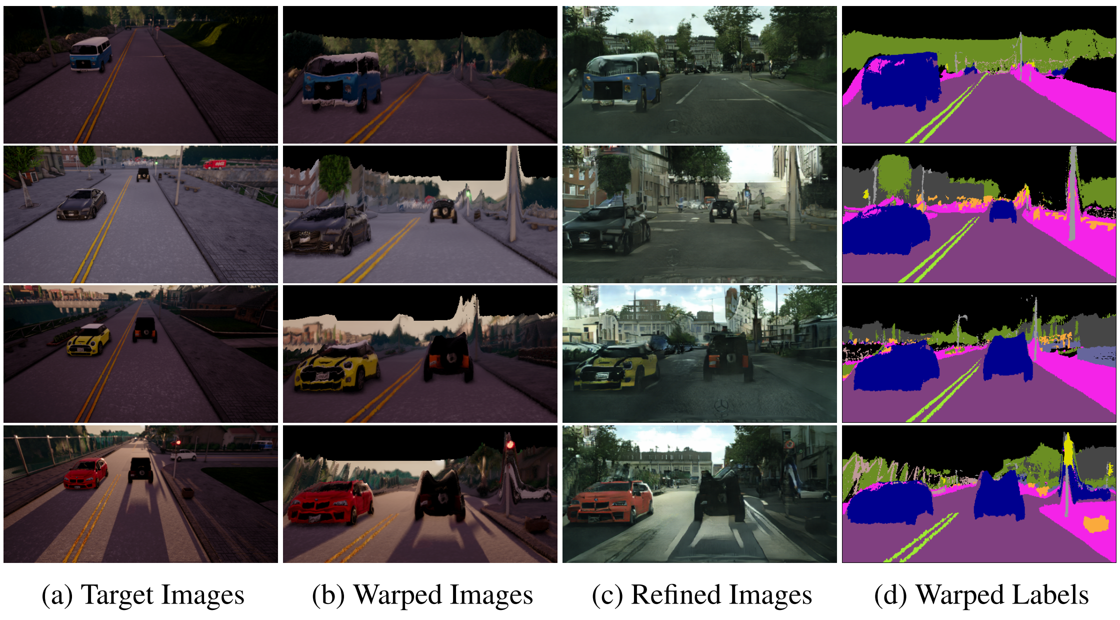

(a) Target Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.8: **NoVA$_{mono-self}$ Translation on Sim2Real.**

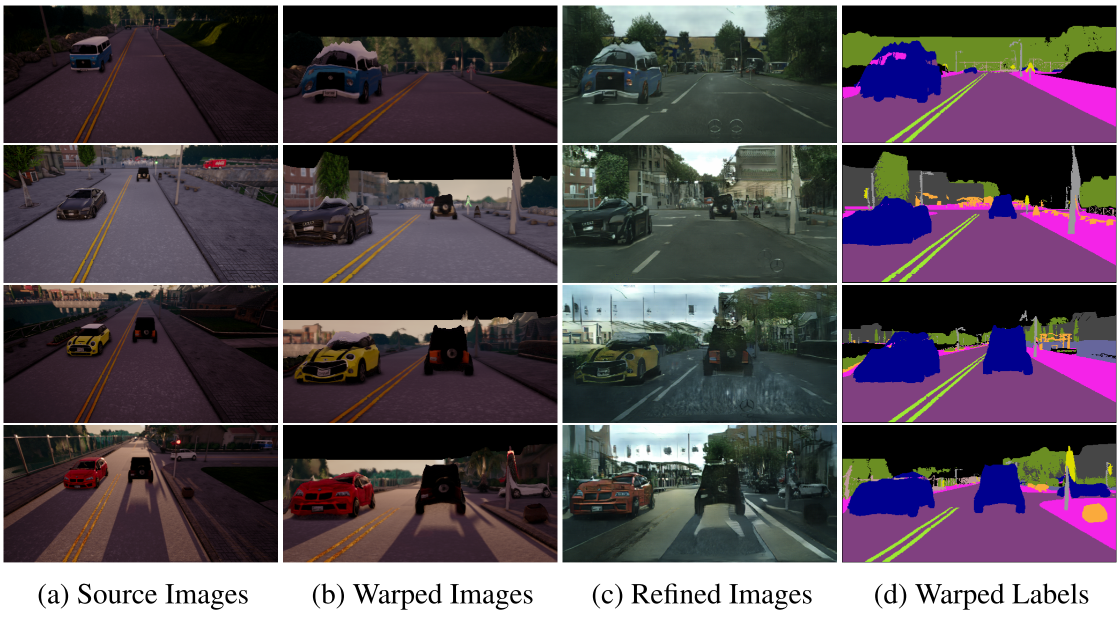

(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.9: **NoVA$_{mono-sup}$ Translation on Sim2Real.**

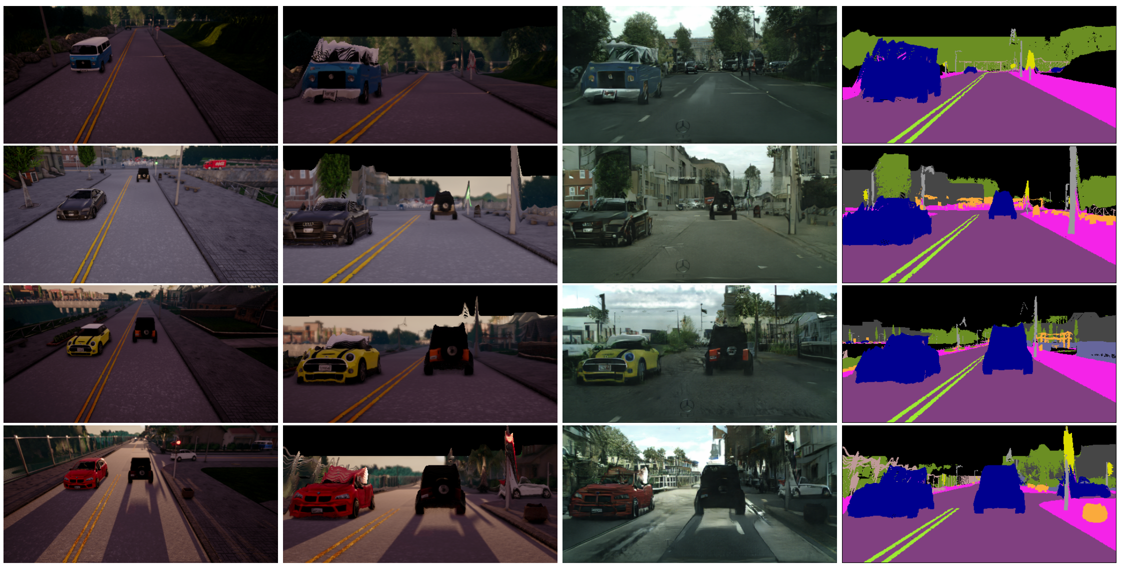

(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.10: **NoVA$_{stereo-sup}$ Translation on Sim2Real.**

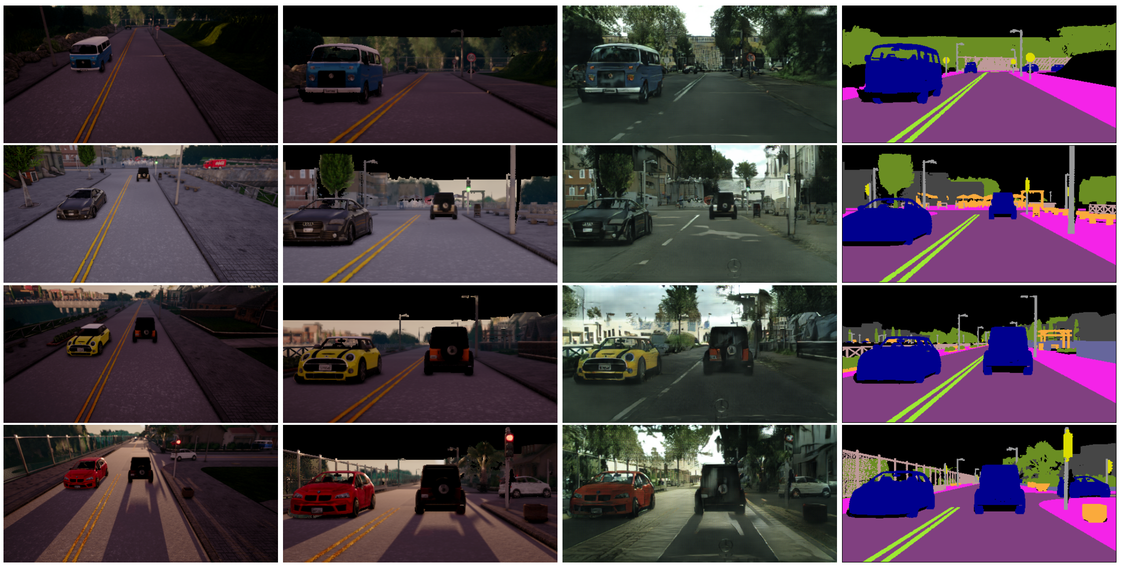

(a) Source Images (b) Warped Images (c) Refined Images (d) Warped Labels

Figure A.11: **NoVA$_{GT}$ Translation on Sim2Real.**

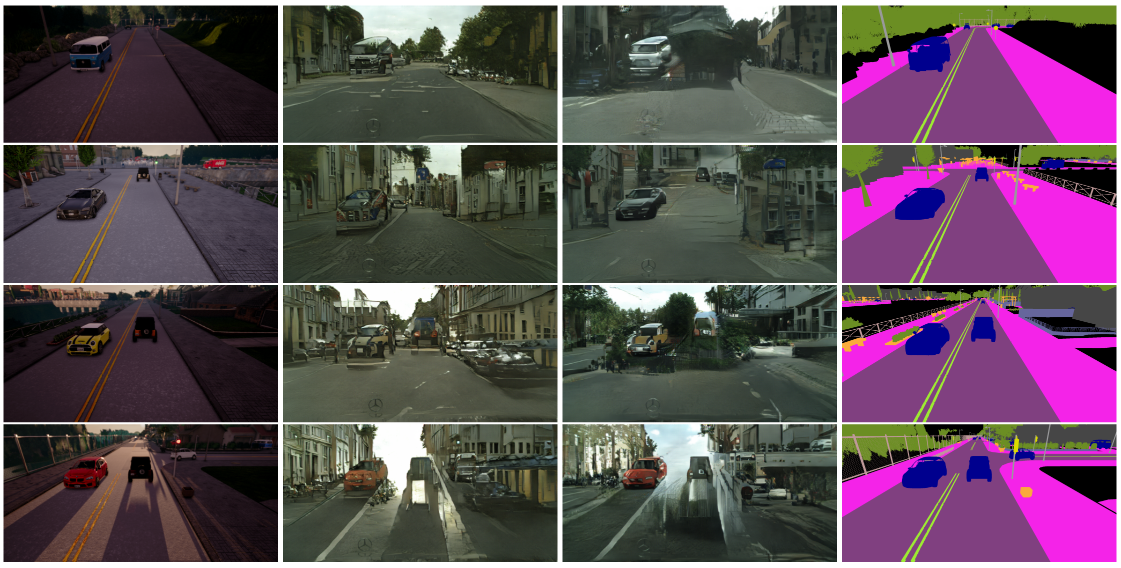

(a) Source Images (b) CyCADA (c) SPLAT (d) Source Labels

Figure A.12: **CyCADA and SPLAT Translation on Sim2Real.**

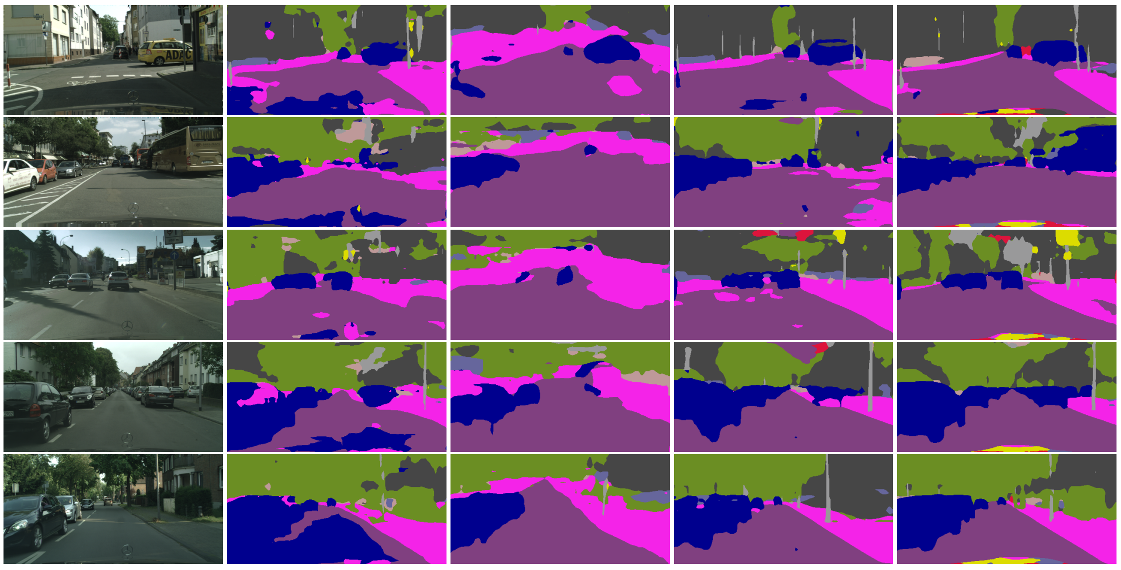
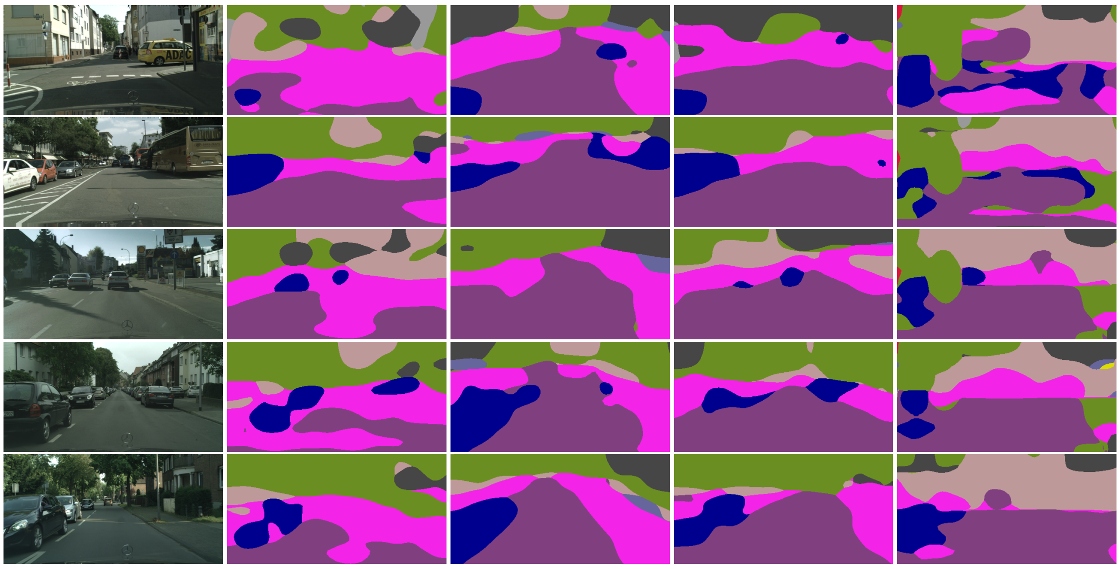
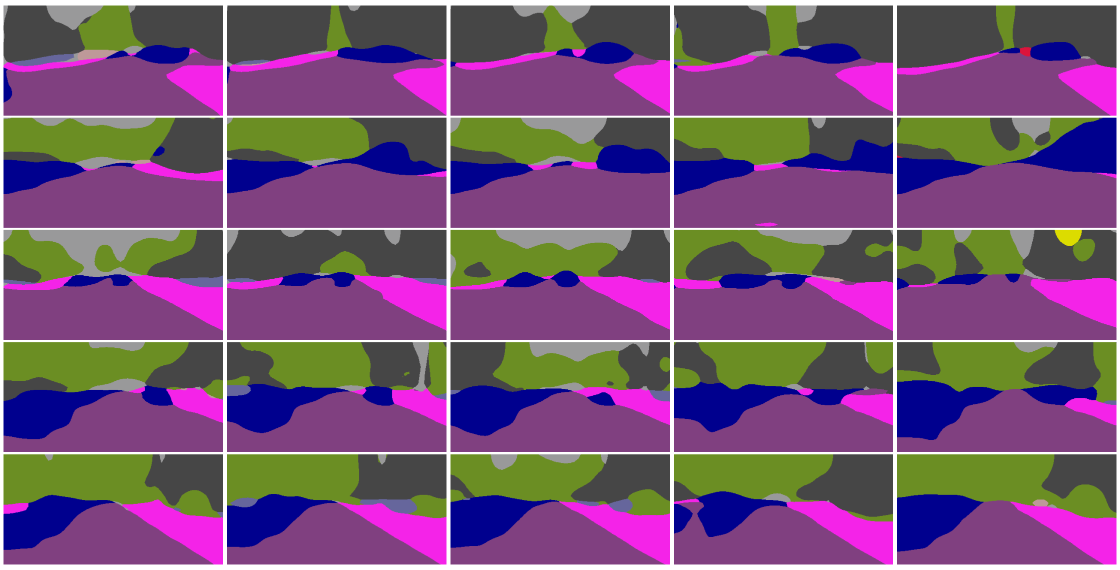

(a) Target Images (b) Source Model (c) CyCADA (d) NoVA$_{GT}$ (e) Target Oracle

Figure A.13: **Comparison of Semantic Segmentation Performance on Sim2Real for a DRN-26 Model.**



(a) Target Images (b) Source Model (c) CyCADA (d) SPLAT (e) SceneAdapt



(f) NoVA$_{mono-self}$ (g) NoVA$_{mono-sup}$ (h) NoVA$_{stereo-sup}$ (i) NoVA$_{GT}$ (j) Target Oracle

Figure A.14: **Comparison of Semantic Segmentation Performance on Sim2Real for a VGG16-FCN8s Model.**

Symbols

This list provides an overview of the notation that is shared across chapters.

X	Input dataset
Y	Label dataset
N	Dataset size
I	Identity matrix
K	Intrinsic matrix
R	Rotation matrix
W	Model weights
x	Input sample $x \in X$
y	Label sample $y \in Y$
f	Function
g	Interpolation kernel
t	Transformation
$\mathbb{I}$	Identity function
$\mathbb{R}$	Real numbers
$\mathcal{D}$	Distance metric
$\mathcal{L}$	Loss function
$\mathcal{N}$	Normal distribution
$\mathcal{T}$	Set of transformations
α, β, γ	Rotation angles
ϕ, θ	Spherical coordinates
δ	Kronecker delta function
λ	Weight parameter
σ	Softmax function
Δ	Step size

Abbreviations

BI	Bilinear Interpolation
CAD	Computer-Aided Design
CARLA	Car Learning to Act
CNN	Convolutional Neural Network
CoGAN	Coupled GAN
CubeCNN	Cube Map CNN
CyCADA	Cycle-Consistent Adversarial Domain Adaptation
CycleGAN	Cycle-Consistent Adversarial Network
DA	Domain Adaptation
DC-IGN	Deep Convolution Inverse Graphics Network
DCN	Deformable Convolutional Network
DRN	Dilated Residual Network
DSS	Deep Scale-Spaces
EPE	End-Point-Error
EquiCNN	Equirectangular CNN
FCN	Fully Convolutional Network
fwIoU	Frequency-weighted Intersection-over-Union
G-CNN	Group Equivariant Convolutional Neural Network
GAN	Generative Adversarial Network
GCNN	Graph Convolutional Neural Network
GRL	Gradient Reversal Layer
GTSRB	German Traffic Sign Recognition Benchmark
H-Net	Harmonic Network
HOG	Histogram of Oriented Gradients
IoU	Intersection-over-Union
mAP	Mean Average Precision
mIoU	Mean Intersection-over-Union

MMD	Maximum Mean Discrepancy
MNIST	Modified National Institute of Standards and Technology database
NN	Nearest Neighbor
NBN	N-body Network
NoVA	Novel Viewpoint Adaptation
NVS	Novel View Synthesis
Omni-MNIST	Omnidirectional MNIST
OmPaCa	Omnidirectional Parked Cars
ORN	Oriented Response Network
pixAcc	Pixel Accuracy
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
RotEqNet	Rotation Equivariant Vector Field Network
S2CNN	Spherical CNN
SFCNN	Steerable Filter CNN
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SphereSSD	Spherical Single Shot MultiBox Detector
SphereTN	Spherical Transformer Network
SPLAT	Semantic Pixel-Level Adaptation Transforms
SSD	Single Shot MultiBox Detector
STN	Spatial Transformer Network
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
TI-POOLING	Transformation-Invariant Pooling
TFN	Tensor Field Network

Bibliography

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[2] H. S. Baird. Document image defect models. In L. O'Gorman and R. Kasturi, editors, *Document Image Analysis*, pages 315–325. IEEE Computer Society Press, 1995.

[3] D. Beymer and T. Poggio. Face recognition from one example view. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1995.

[4] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[6] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8):1872–1886, 2013.

[7] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015.

[8] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] O. Chapelle and B. Schölkopf. Incorporating invariances in non-linear support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[10] X. Chen, J. Song, and O. Hilliges. NVS machines: Learning novel view synthesis with fine-grained view control. *arXiv.org*, 1901.01880, 2019.

[11] Y. Chen, W. Li, X. Chen, and L. V. Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] I. Choi, O. Gallo, A. J. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. *arXiv.org*, 1812.04777, 2018.

[13] C. Chu, A. Zhmoginov, and M. Sandler. Cyclegan, a master of steganography. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2017.

[14] T. Cohen, M. Geiger, and M. Weiler. A general theory of equivariant nns on homogeneous spaces. *arXiv.org*, 1811.02017, 2018.

[15] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019.

[16] T. S. Cohen, M. Geiger, J. Khler, and M. Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

[17] T. S. Cohen, M. Geiger, and M. Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks). *arXiv.org*, 1803.10743, 2018.

[18] T. S. Cohen and M. Welling. Group equivariant convolutional networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2016.

[19] T. S. Cohen and M. Welling. Steerable cnns. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

[20] B. Coors, A. Condurache, and A. Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[21] B. Coors, A. Condurache, and A. Geiger. Nova: Learning to see in novel viewpoints and domains. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2019.

[22] B. Coors, A. Condurache, A. Mertins, and A. Geiger. Learning transformation invariant representations with weak supervision. In *Proc. of the Conf. on Computer Vision Theory and Applications (VISAPP)*, 2018.

[23] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene

understanding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[25] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1703.06211, 2017.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[28] D. Di Mauro, A. Furnari, G. Patanè, S. Battiato, and G. M. Farinella. Scene adaptation for semantic segmentation using adversarial learning. In *Proc. of International Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, 2018.

[29] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2016.

[30] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[31] A. Dosovitskiy, P. Fischer, E. Ilg, P. Haeusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017.

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

[34] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis. Learning so(3) equivariant representations with spherical cnns. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[35] C. Esteves, C. Allen-Blanchette, X. Zhou, and K. Daniilidis. Polar transformer networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018.

[36] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera. CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[37] B. Fasel and D. Gatica-Perez. Rotation-invariant neoperceptron. In *Proc. of the International Conf. on Pattern Recognition (ICPR)*, 2006.

[38] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[39] Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015.

[40] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 17:2096–2030, 2016.

[41] R. Garg, B. G. V. Kumar, G. Carneiro, and I. D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[43] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[44] R. Ghosh and A. K. Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. In *Proc. of the International Conf. on Machine learning (ICML) Workshops*, 2019.

[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[46] R. B. Girshick. Fast R-CNN. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[47] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[49] I. J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, and A. Y. Ng. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[50] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[51] P. Haeusser, A. Mordvintsev, and D. Cremers. Learning by association - a versatile semi-supervised training method for neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[53] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[54] J. a. F. Henriques and A. Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. In *Proc. of the International Conf. on Machine learning (ICML)*, 2017.

[55] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *Proc. of the International Conf. on Artificial Neural Networks (ICANN)*, 2011.

[56] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018.

[57] J. Hoffman, D. Wang, F. Yu, and T. Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv.org*, 1612.02649, 2016.

[58] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360° sports video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[59] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[60] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, 195:215–243, 1968.

[61] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015.

[62] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[63] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[64] Y. Jeon and J. Kim. Active convolution: Learning the shape of convolution for image classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[65] Y. Jia. *Learning Semantic Image Representations at a Large Scale*. PhD thesis, EECS Department, University of California, Berkeley, May 2014.

[66] A. Kanazawa, A. Sharma, and D. W. Jacobs. Locally scale-invariant convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2014.

[67] R. Khasanova and P. Frossard. Graph-based classification of omnidirectional images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2017.

[68] R. Khasanova and P. Frossard. Graph-based isometry invariant representation learning. In *Proc. of the International Conf. on Machine learning (ICML)*, 2017.

[69] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[70] J. J. Kivinen and C. K. I. Williams. Transformation equivariant boltzmann machines. In *Proc. of the International Conf. on Artificial Neural Networks (ICANN)*, 2011.

[71] R. Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv.org*, 1803.01588, 2018.

[72] R. Kondor and S. Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018.

[73] P. Koniusz, Y. Tas, and F. Porikli. Domain adaptation by mixture of alignments of second- or higher-order scatter tensors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[74] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[76] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[77] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[78] W. Lai, Y. Huang, N. Joshi, C. Buehler, M. Yang, and S. B. Kang. Semantic-driven generation of hyperlapse from 360 degree video. *Proc. of IEEE Transactions on Visualization and Computer Graphics*, 2017.

[79] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[80] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2007.

[81] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec. 1989.

[82] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[83] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[84] C.-H. Lin and S. Lucey. Inverse compositional spatial transformer networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[85] C.-H. Lin, E. Yumer, O. Wang, E. Shechtman, and S. Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[86] M. Liu, X. He, and M. Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[87] M. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[88] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[89] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[90] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[91] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015.

[92] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[93] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2017.

[94] A. M. López, J. Xu, J. L. Gomez, D. Vázquez, and G. Ros. From virtual to real world visual perception using domain adaptation - the DPM as example. In *Domain Adaptation in Computer Vision Applications.*, pages 243–258. Springer, 2017.

[95] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[96] J. Ma, W. Wang, and L. Wang. Irregular convolutional neural networks. *Proc. of Asian Conference on Pattern Recognition*, 2017.

[97] D. Marcos, B. Kellenberger, S. Lobry, and D. Tuia. Scale equivariance in cnns with vector fields. In *Proc. of the International Conf. on Machine learning (ICML) Workshops*, 2018.

[98] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. Rotation equivariant vector field networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[99] D. Marcos, M. Volpi, and D. Tuia. Learning rotation invariant convolutional filters for texture classification. In *Proc. of the International Conf. on Pattern Recognition (ICPR)*, 2016.

[100] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv.org*, 1411.1784, 2014.

[101] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing by virtual adversarial examples. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2016.

[102] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[103] P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 86(11):2196–2209, 1998.

[104] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by

solving jigsaw puzzles. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[105] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[106] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[107] E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[108] E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[109] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[110] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[111] F. Pearson. *Map Projections: Theory and Applications*. Taylor & Francis, 1990.

[112] X. Peng and K. Saenko. Synthetic to real adaptation with deep generative correlation alignment networks. *arXiv.org*, 1701.05524, 2017.

[113] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[114] L. Ran, Y. Zhang, Q. Zhang, and T. Yang. Convolutional neural network-based robot navigation using uncalibrated spherical images. *Sensors*, 17(6), 2017.

[115] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[116] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[117] A. Rozantsev, M. Salzmann, and P. Fua. Beyond sharing weights for deep domain adaptation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[118] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[119] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[120] E. B. Saff and A. B. J. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, 1997.

[121] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[122] C. Sakaridis, D. Dai, S. Hecker, and L. Van Gool. Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[123] B. Schoelkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[124] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems (NIPS)*, 1998.

[125] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[126] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[127] P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop - a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems (NIPS)*, 1992.

[128] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. of the International Conference on Document Analysis and Recognition*, 2003.

[129] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[130] K. Sohn and H. Lee. Learning invariant representations with local transformations. In *Proc. of the International Conf. on Machine learning (ICML)*, 2012.

[131] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. In *International Conf. on Learning Representations (ICLR) (workshop track)*, 2015.

[132] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[133] Y.-C. Su and K. Grauman. Making 360deg video watchable in 2d: Learning videography for click free viewing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[134] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2vid: Automatic cinematography for watching 360 degree videos. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, 2016.

[135] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2016.

[136] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2014.

[137] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV 2016 Workshops*, 2016.

[138] K. S. Tai, P. Bailis, and G. Valiant. Equivariant Transformer Networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019.

[139] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

[140] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[141] N. Thomas, T. Smidt, S. M. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv.org*, 1802.08219, 2018.

[142] L. Tran, K. Sohn, X. Yu, X. Liu, and M. Chandraker. Gotta adapt em all: Joint pixel and feature-level domain adaptation for recognition in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[143] L. Tran, X. Yin, and X. Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[144] S. Tulsiani, R. Tucker, and N. Snavely. Layer-structured 3d scene inference via view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[145] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[146] E. Tzeng, K. Burns, K. Saenko, and T. Darrell. SPLAT: semantic pixel-level adaptation transforms for detection. *arXiv.org*, 1812.00929, 2018.

[147] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[148] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[149] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv.org*, 1412.3474, 2014.

[150] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. Rotation equivariant cnns for digital pathology. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2018.

[151] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[152] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant cnns. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[153] M. Winkels and T. S. Cohen. 3d g-cnns for pulmonary nodule detection. In *International Conference on Medical Imaging with Deep Learning*, 2018.

[154] D. Worrall and G. Brostow. Cubenet: Equivariance to 3d rotation and translation. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[155] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[156] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[157] D. E. Worrall and M. Welling. Deep scale-spaces: Equivariance over scale. *arXiv.org*, 1905.11697, 2019.

[158] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[159] J. Yang, S. Reed, M.-H. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[160] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker. Towards large-pose face frontalization in the wild. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[161] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[162] K. G. Yu-Chuan Su. Flat2sphere: Learning spherical convolution for fast features from 360° imagery. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[163] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep

neural networks via stability training. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[164] Z. Zhong, L. Zheng, S. Li, and Y. Yang. Generalizing a person retrieval model hetero- and homogeneously. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[165] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang. Camera style adaptation for person re-identification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[166] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[167] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[168] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[169] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.