UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MATHEMATICS AND
IMAGE COMPUTING

Diffeomorphism-Equivariant Neural Networks

Diffeomorphismusäquivariante Neuronale Netze

Masterarbeit

verfasst am
Institute of Mathematics and Image Computing, Universität zu Lübeck

im Rahmen des Studiengangs
Mathematik in Medizin und Lebenswissenschaften
der Universität zu Lübeck

vorgelegt von
Josephine Elisabeth Oettinger

ausgegeben und betreut von
Prof. Dr. Jan Lellmann

mit Unterstützung von
Zakhar Shumaylov und **Prof. Dr. Carola Bibiane Schönlieb,**
Department of Applied Mathematics and Theoretical Physics, University of Cambridge

Lübeck, den 06. November 2025

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

_____

Josephine Elisabeth Oettinger

Zusammenfassung

Deep-Learning-Methoden haben sich zu einem Eckpfeiler der modernen datengesteuerten Forschung entwickelt und finden in den verschiedensten wissenschaftlichen Bereichen Anwendung. Die meisten Netzwerke sind jedoch stark von den im Training verwendeten Daten abhängig und erfordern große Mengen an hochwertigen Daten sowie erhebliche Rechenressourcen.

In dieser Arbeit schlagen wir einen Ansatz vor, mit dem sich ein beliebiges neuronales Netz in ein diffeomorphismusäquivariantes neuronales Netz transformieren lässt, und analysieren ihn. Wir verwenden ein energiebasiertes Kanonisierungs-Framework, das von LieLAC inspiriert ist, einem Verfahren zur Erzeugung äquivarianter Netzwerke für Lie-Gruppen. Wir erweitern dieses Framework und passen es an die Gruppe der Diffeomorphismen an. Unser Netzwerk erreicht eine annähernde Äquivarianz, ohne auf umfangreiche Datenaugmentation oder erneutes Training angewiesen zu sein, da es nur einmal auf einem einfachen Datensatz trainiert werden muss, aber dennoch gut auf unbekannte Transformationen generalisiert.

Wir liefern eine theoretische Analyse der Generalisierungseigenschaften der energiebasierten Kanonisierung und leiten Schranken für den erwarteten Verlust her. Diese theoretischen Erkenntnisse spiegeln sich in unserer praktischen Realisierung wider, in der wir beispielhaft eine diffeomorphismusäquivariante Segmentierung implementieren. Die experimentelle Validierung anhand eines synthetischen Datensatzes mit verschachtelten Quadraten bestätigt die Wirksamkeit unseres Ansatzes, da unser Netzwerk einen naiven Ansatz übertrifft und eine nahezu vergleichbare Segmentierungsgenauigkeit wie ein erweitertes U-Net erreicht, dabei jedoch deutlich weniger Trainingsdaten benötigt. Darüber hinaus weist unser Netzwerk weniger drastische Ausreißer auf. Wir evaluieren die Leistung unseres Netzwerks außerdem auf realen Thorax-Röntgenaufnahmen zur Lungensegmentierung. Hier erreicht oder übertrifft unser Netzwerk den naiven Ansatz bei etwa 80% der ausgewerteten Bilder.

Insgesamt stellt diese Arbeit einen theoretisch fundierten und praktisch validierten Rahmen vor, um Diffeomorphismusäquivarianz in neuronalen Netzen durch energiebasierte Kanonisierung zu erreichen und ebnet damit den Weg für dateneffiziente und transformationskonsistente Deep-Learning-Modelle.

Abstract

Deep learning models have become a cornerstone of modern data-driven research, finding applications across various scientific domains. Most networks rely heavily on the data seen in training and require large amounts of high-quality data as well as computational resources.

In this work, we propose and analyse a strategy for turning any neural network into a diffeomorphism-equivariant neural network. We use an energy-based canonicalisation framework, inspired by LieLAC, a framework for creating equivariant networks for Lie groups. We extend the framework to the group of diffeomorphisms. Our network achieves approximate equivariance without relying on extensive data augmentation or retraining, as it only needs to be trained once on a simple dataset. Nevertheless, it generalises well to unseen transformations.

We provide a theoretical analysis of the generalisation properties of energy-based canonicalisation, and derive bounds on the expected loss. The theoretical insights are reflected in a practical network architecture, which achieves approximate diffeomorphism-equivariance for an exemplary segmentation task. The experimental evaluation on a synthetic dataset of nested squares confirms the effectiveness of our approach, as we outperform a naïve approach and achieve a segmentation accuracy that is close to that of an augmented U-Net, while requiring significantly less training data. In addition, our network has less drastic outliers. We also evaluate the performance of our network on real-world chest X-ray images for lung segmentation. Here, our network matches or outperforms the naïve approach on approximately 80% of the evaluated images.

Overall, this work introduces a theoretically motivated and practically validated framework for achieving diffeomorphism-equivariance in neural networks through energy-based canonicalisation, paving the way for data-efficient and transformation-consistent deep learning models.

Acknowledgements

First of all, I would like to thank everyone who gave me the opportunity to write my thesis in Cambridge. Thank you, Carola, for taking me in; Jan, for making the connection and supervising my thesis in Lübeck; Zak, for supervising me in Cambridge; and everyone else who supported me along the way. I had a splendid time and learnt so much.

Furthermore, I would like to thank everyone who helped me write this thesis. It involved rather a lot of "trail and error" (and yes, at one point I used the word "trail" instead of "trial" in my thesis, my apologies to everyone who had to proofread my sometimes chaotic drafts). Special thanks go to Peter and Zak, who spent many Fridays with me wrestling through the theoretical parts; missing lunch and/or happy hour in the process. Thank you also for patiently proofreading everything. In addition, I would like to thank everyone else, especially Ole, Johannes, Flo, and Ulvo, for their proofreading and mental support.

And now, after countless misspellings and mispronunciations of "canonicalisation", it is finally done. So buckle your seatbelts everyone, and (hopefully) enjoy the ride...

AI Disclaimer

While working on this thesis, AI models such as ChatGPT and Claude AI were used to improve writing and correct errors. Additionally, the code writing was assisted by the GitHub Copilot in VS Code and ChatGPT. All texts/outputs were manually scrutinised and validated.

Contents

1 Introduction **3**
- 1.1 Motivation 3
- 1.2 Canonicalisation 5
- 1.3 Contributions 7
- 1.4 Structure of this Work 8
- 1.5 Related Work 8

2 Theoretical Background **15**
- 2.1 Topology, Manifolds, and Diffeomorphisms 15
- 2.2 Lie Group Theory 22
- 2.3 Measure Theory 26

3 On Generalisation of Canonicalisation **31**
- 3.1 Setup of the Learning Scenario 32
- 3.2 Bounding the Expected Generalisation Loss 38

4 Diffeomorphism-Equivariant Neural Network **47**
- 4.1 Problem Setup 48
- 4.2 Canonicalisation 49
- 4.3 Segmentation 55
- 4.4 Reverse Canonicalisation 57
- 4.5 Theoretical Connection and Summary 57

5 Experiments and Results **59**
- 5.1 Synthetic Datasets 59
- 5.2 Hyperparameter Tuning and Implementation of DiffeoNN 62
- 5.3 Benchmarking 67
- 5.4 Invariance of the Canonicalisation 68
- 5.5 DiffeoNN for Lung Segmentation 69

6 Conclusion and Discussion **73**

Bibliography **75**

A Appendix **82**
A.1 Artificial Neural Networks 82
A.2 Synthetic Dataset 86
A.3 DiffeoNN on the Synthetic Dataset 87
A.4 Experiments on Invariance of the Canonicalisation 90
A.5 DiffeoNN for Lung Segmentation 91

List of Used Symbols

$\mathfrak{g}$	Lie algebra.
A_δ^c	complement of A_δ, i.e., set of elements with poorly-sampled orbits.
A_δ	set of elements x for which the orbits $\mathcal{O}_x$ is well-sampled.
C_{Lip}	Lipschitz-constant of $x \mapsto L(f_\theta(x), y)$.
E_{X_E}	image similarity energy.
E_{VAE}	VAE-based energy.
E_{adv}	adversarial energy.
E_{can}	canonicalisation energy, $E_{\text{can}} : \mathcal{X} \times \mathcal{D}_{\text{SVF}}(\Omega) \to \mathbb{R}$.
E_{reg}	regularising energy.
G_x	stabiliser of x.
G	group, here usually a compact Lie group.
S_σ	set of $x \in X$, where a small loss can be ensured.
V_{max}	upper bound of orbit volume $\text{vol}_{\mathcal{O}}(\mathcal{O}_x)$.
X/G	orbit space.
X_E	training dataset, in experiments split into X_E^{train}, X_E^{val}, and X_E^{test} for training networks.
X_{TE}	diffeomorphically transformed synthetic training dataset, mainly used to test DiffeoNN, like the training data X_E: in experiments split into X_{TE}^{train}, X_{TE}^{val}, and X_{TE}^{test}.
X	diffeomorphically transformed training dataset, input dataset.
Y_E	set of ground truth segmentations of X_E.
Y_{TE}	set of ground truth segmentations of X_{TE}.
Ω	image domain.
$\tilde{f}_\theta$	diffeomorphism-equivariant neuronal network, references DiffeoNN.
$\det(\mathcal{J}_g)$	Jacobian determinant of function g.
$\hat{\mu}_E$	smooth estimation of the empirical measure μ_E with density $\hat{\rho}_E$.
$\mathbb{1}_A$	indicator function of set A.
$\mathcal{B}(X)$	Borel σ-algebra of a topological space X.
$\mathcal{D}(\mathcal{X})$	groups of diffeomorphisms mapping fom the manifold $\mathcal{X}$ to $\mathcal{X}$.
$\mathcal{D}_{\text{SVF}}(\mathcal{X})$	group of SVF-based diffeomorphisms.
$\mathcal{O}_x$	group orbit of x.
$\mathcal{X}$	manifold, describes the data manifold.
$\mathcal{Y}$	output dataset.
μ_E	discrete empirical measure that places equal weights on all training samples X_E.

μ_T	"true" distribution over manifold $\mathcal{X}$ in the sense that it is the distribution, under which our data is drawn independently.
$\omega_{\mathcal{O}_x}$	orbit measure.
π	projection map, mapping x to its orbit $\mathcal{O}_x$, $\pi : X \to X/G$.
$\text{vol}_{\mathcal{O}}(\mathcal{O}_x)$	volume of the orbit $\mathcal{O}_x$.
$c(x)$	set of optimisers for the canonicalisation problem.
$f_*\mu$	pushforward measure $f_*\mu(B) := \mu(f^{-1}(B))$.
f_θ	inner neuronal network, here only trained on the small training dataset X_E.
g_x^{-1}	reverse canonicalising element.
g_x	canonicalising element for the input x.
m_E	empirical orbit mass.
x_c	canonicalised input x.
$x_i \sim \text{Law}(\mu_T)$	x_i is drawn independently under the distribution μ_T.
y_x	output for input x.
$\lvert G \rvert$	power of the group G.
DiffeoNN	our diffeomorphism-equivariant neuronal network.
SVF	stationary velocity field.
VAE	variational autoencoder.

1

Introduction

In the last few decades, deep learning models have become more and more popular and have been applied to numerous scientific fields, such as biology [1], physics [59], and engineering [29]. Especially in computer vision [83] and medical imaging [84, 23], deep learning has become a central tool, as it can be used to automatically solve many tasks such as classification [54], object detection [88], and segmentation [67].

Two of the main obstacles in training deep learning models are the large amounts of high-quality data and computing resources needed [35, 77]. But even with sufficient data, generalisation properties cannot be guaranteed, as they highly depends on the data seen in training and testing [58, 87].

1.1 Motivation

In the quest for more robust and generalisable machine learning models, the focus has shifted towards exploiting group symmetries of the task's problem through *invariant* or *equivariant neural networks* [21]. While *group invariance* means that a model's output remains unchanged when the input is transformed by any element of the group, *group equivariance* means that the model's output transforms in a predictable way under the group action that is applied to the input. More formally, let $\mathcal{X}$ and $\mathcal{Y}$ be manifolds. Furthermore, let G be a group of transformations with the actions $\cdot_{\mathcal{X}}$ and $\cdot_{\mathcal{Y}}$, defined as maps $\cdot_{\mathcal{X}} : G \times \mathcal{X} \to \mathcal{X}$ and $\cdot_{\mathcal{Y}} : G \times \mathcal{Y} \to \mathcal{Y}$ that satisfy composition properties. A function $\tilde{f}_\theta : \mathcal{X} \to \mathcal{Y}$ is called:

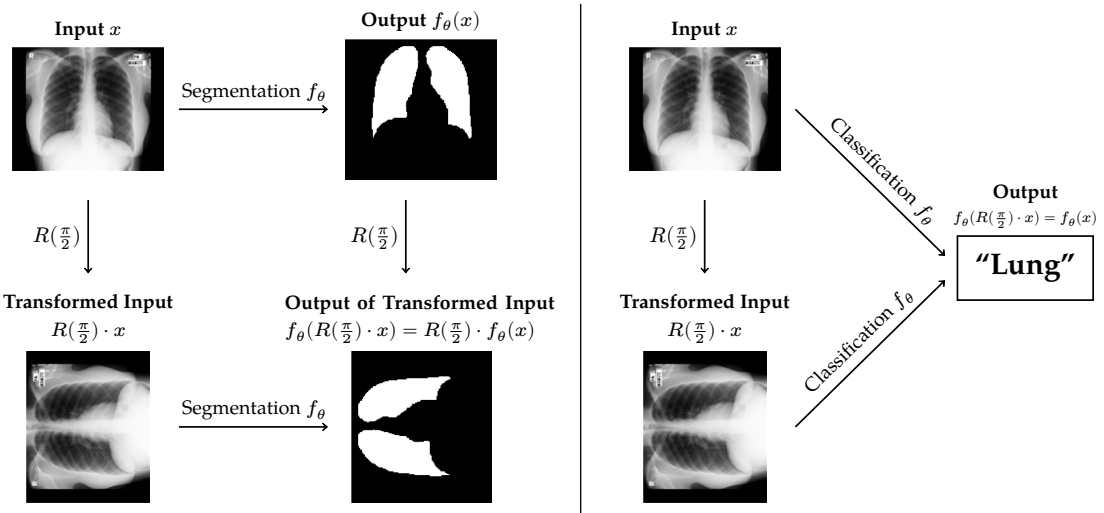
- G-**invariant** if for all $g \in G$ and all $x \in \mathcal{X}$,

$$\tilde{f}_\theta(g \cdot_{\mathcal{X}} x) = \tilde{f}_\theta(x), \tag{1.1}$$

- G-**equivariant** if for all $g \in G$ and all $x \in \mathcal{X}$,

$$\tilde{f}_\theta(g \cdot_{\mathcal{X}} x) = g' \cdot_{\mathcal{Y}} \tilde{f}_\theta(x), \tag{1.2}$$

  where g' is the corresponding output transformation induced by g. The corresponding output transformation g' depends on the action $\cdot_{\mathcal{Y}}$ and g. When $\mathcal{X} = \mathcal{Y}$ and $\cdot_{\mathcal{X}} = \cdot_{\mathcal{Y}}$, it could, for example, be the classical inverse of g.



(a) Group-equivariance on the example of segmentation.

(b) Group-invariance on the example of classification.

Figure 1.1: Difference between group-equivariance (left) and group-invariance (right) for the group of rotations in the 2D plane $SO(2)$ using lung images from [76]. In segmentation, equivariance is desired, as rotating the input image by $R(\frac{\pi}{2}) \in SO(2)$ should result in a correspondingly rotated segmentation output, which means $f_\theta(R(\frac{\pi}{2}) \cdot x) = R(\frac{\pi}{2}) \cdot f_\theta(x)$. Group-invariance is, for example, desired for a classification task: A rotated input image should yield the same class label, i.e., $f_\theta(R(\frac{\pi}{2}) \cdot x) = f_\theta(x)$.

For simplicity, we will refer to $\cdot_\mathcal{X}$ and $\cdot_\mathcal{Y}$ as $\cdot$ if it is clear from the context. Depending on the task, either group-invariance (e.g., classification) or group-equivariance (e.g., segmentation) is desired. This is illustrated in Figure 1.1, where an exemplary group-equivariant segmentation and an exemplary group-invariant classification for the group of rotations in the 2D plane $SO(2)$ is visualised on lung images from [76].

Enforcing invariance or equivariance with respect to a relevant transformation group can not only increase robustness but also reduce the need for extensive annotated data augmentation, and, therefore, the training time [26, 21, 15, 20].

Previous works have primarily focused on small, discrete symmetry groups or very specific learning settings [26, 31]. However, in many applications, the relevant transformations are not simple rigid motions but smooth, non-linear deformations of the input domain. These transformations are naturally modelled as *diffeomorphisms*, which form an infinite-dimensional group:

Definition 1.1 (Diffeomorphism). Let $\mathcal{X}$ and $\mathcal{Y}$ be differentiable manifolds. A map $f : \mathcal{X} \to \mathcal{Y}$ is called a diffeomorphism if it is bijective, differentiable, and its inverse $f^{-1} : \mathcal{Y} \to \mathcal{X}$ is also differentiable. The set of all such maps is denoted as $\mathcal{D}(\mathcal{X}, \mathcal{Y})$.

This set forms an infinite-dimensional group under composition, when $\mathcal{X} = \mathcal{Y}$. We call that group $\mathcal{D}(\mathcal{X})$. A more detailed definition and more background can be found in Section 2.1. Informally, diffeomorphisms are smooth, invertible mappings between manifolds with smooth inverses. They are frequently used in imaging applications, particularly medical image registration [12, 28, 19, 8], where they model realistic anatomical transformations while preserving topological properties [60, 75]. A neural network that is equivariant to diffeomorphisms would therefore be highly valuable for real-world tasks.

For example, in medical image segmentation, two anatomically identical lungs may appear very different across scans. Variations in patient positioning or changes in breathing phase can stretch, compress, or warp the anatomy. A naïve network trained without special precautions would often fail to segment both inputs consistently. By contrast, a diffeomorphism-equivariant network could capture the equivalence between such scans and adapt the segmentation accordingly.

In this work, we focus on the infinite-dimensional group of diffeomorphisms $\mathcal{D}(\mathcal{X})$ and introduce a diffeomorphism-equivariant neural network. However, achieving diffeomorphism-equivariance is very challenging: unlike finite groups or simple transformations, such as rotations or translations, the group of diffeomorphisms is infinite-dimensional, and cannot be parametrised compactly in a way that is suitable for building it into deep learning architectures directly. Most classical strategies fail:

- **Data augmentation**, where training data is synthetically transformed to mimic group symmetries [71, 37], cannot guarantee equivariance for infinite-dimensional groups, due to their dependence on finitely sampled data and resources.
- **Group-equivariant layers**, which encode symmetries directly into the network's architecture [26, 25, 44], need a known finite or at least compact group structure.
- Methods focusing on group orbits by **averaging** over predictions of transformed inputs, e.g., group and frame averaging [64, 66], become impractical for non-compact and high-dimensional groups.

For more details on these approaches and their limitations, see Section 1.5. A feasible strategy to achieve diffeomorphism-equivariance is *canonicalisation* [40, 72]. **Canonicalisation** is based on group orbits, and maps each input to a fixed representative in its group orbit. Besides being able to handle diffeomorphisms, canonicalisation has the advantage of creating a diffeomorphism-equivariant neural network that can easily be adapted for various tasks, e.g., object detection and image segmentation, as the equivariance is achieved without changing the network itself. Our diffeomorphism-equivariant neural network therefore uses a canonicalisation strategy, which is inspired by LieLAC [72].

1.2 Canonicalisation

Let $\mathcal{X}$ be a data manifold. The central idea of our method is to train a simple model $\tilde{f}_\theta$ only on a small labelled training dataset $X_E \subseteq \mathcal{X}$ and then extend it to perform well on the entire dataset $X \subseteq \mathcal{X}$, containing all diffeomorphic transformations of the training dataset X_E without additional training. This allows us also to adapt pretrained networks without changing them. In analogy with [28], we parametrise the diffeomorphisms using *stationary velocity fields* (SVFs). We call this set of *SVF-based diffeomorphisms* $\mathcal{D}_{\text{SVF}}(\mathcal{X}) \subset \mathcal{D}(\mathcal{X})$.

The extended model $\tilde{f}_\theta$ consists of three steps: a *canonicalisation step*, an *inner model trained on the training data* X_E, and a *reverse canonicalisation step*, illustrated in Figure 1.2.

1. In the **canonicalisation step**, we find a representation x_c for a given input $x \in X$ such that x_c is "close" to the training data X_E. A canonicalising element $g_x \in \mathcal{D}_{SVF}(\mathcal{X})$ is

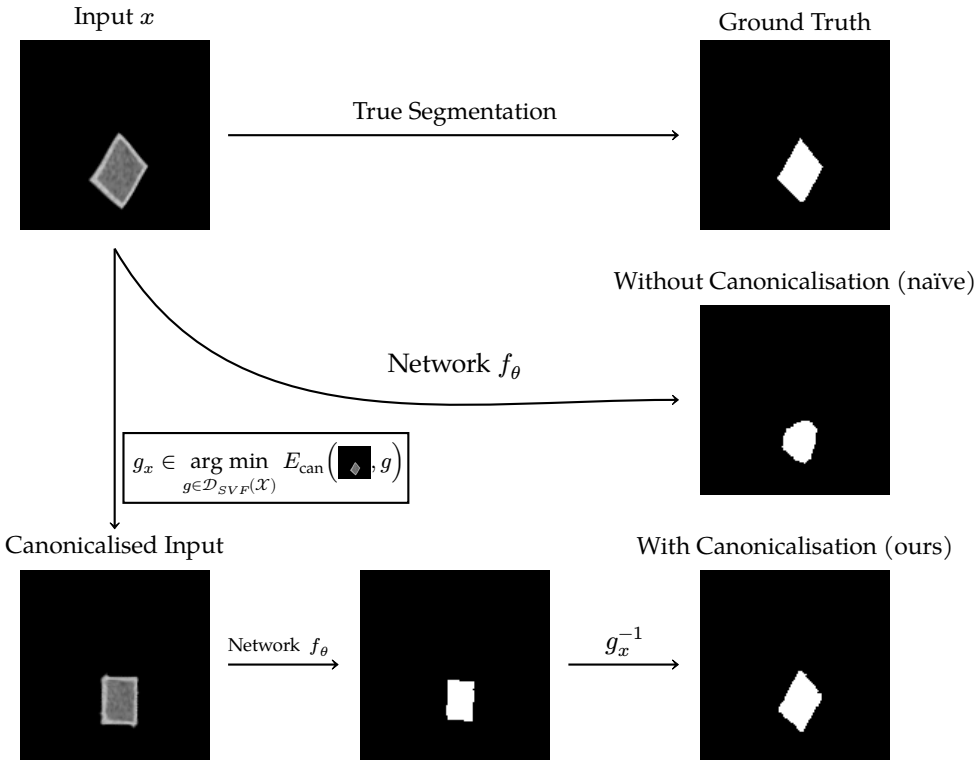


Figure 1.2: Canonicalisation strategy for segmentation. The task is to segment the inner square. The network f_θ is pretrained on a dataset of squares. Applying the model f_θ to a diffeomorphically transformed square image without canonicalisation results in an Intersection-over-Union (IoU) of 0.6887 (naïve strategy). One can see obvious classification errors. Using canonicalisation before applying the network f_θ, and then transforming it back, increases the IoU to 0.9299 (our method). The predicted output is nearly identical to the ground-truth segmentation. Our segmentation is significantly closer to a perfect segmentation, which would have an IoU of 1, as the naïve approach.

found by minimising a task specific canonicalisation energy:

$$g_x \in \underset{g \in \mathcal{D}_{SVF}(\mathcal{X})}{\arg\min} E_{\text{can}}(x, g). \tag{1.3}$$

The canonicalised input is then $x_c := g_x \cdot x$.

2. At the core is a **model** $f_\theta : \mathcal{X} \to \mathcal{Y}$, $x \mapsto f_\theta(x) =: y_x \in \mathcal{Y}$ that is **trained on the dataset** X_E. The model performs the task that is desired for the whole model and outputs $y_{x_c} = f_\theta(x_c) \in \mathcal{Y}$ for the canonicalised input x_c. The output space $\mathcal{Y}$ depends on the task of the whole network.
3. To obtain the final output, a **reverse canonicalisation** is applied:

$$\tilde{f}_\theta(x) := y_x := g_x^{-1} \cdot y_{x_c} = g_x^{-1} \cdot (f_\theta(g_x \cdot x)), \tag{1.4}$$

where $g_x^{-1} \in \mathcal{D}_{SVF}(\mathcal{Y})$ reverses the map corresponding to the canonicalising element g_x on $\mathcal{Y}$. This could, for example, be the inverse of g_x; but reverse canonicalisation depends on the action of G on $\mathcal{Y}$, which depends on the task, and could thus be something completely else.

While this framework is general, we illustrate its relevance in the context of image segmentation: For segmentation, *diffeomorphism-equivariance* means that if the input image is

deformed by a diffeomorphism g, the predicted segmentation output should deform exactly the same way: Let $\tilde{f}_\theta : \mathcal{X} \to \mathcal{Y}$ be a segmentation network for a set of images $X \subseteq \mathcal{X}$ with an output set of segmentation $\mathcal{Y} \subseteq \mathcal{X}$. For this, a network $\tilde{f}_\theta$ is diffeomorphism-equivariant if for all $g \in \mathcal{D}(\mathcal{X})$ and all $x \in X$

$$\tilde{f}_\theta(g \cdot x) = g \cdot \tilde{f}_\theta(x) \tag{1.5}$$

holds. In this case, the reverse canonicalisation is exactly the inverse of the canonicalising element g_x, i.e., $\tilde{f}_\theta(x) = g^{-1} \cdot \tilde{f}_\theta(g \cdot x)$

An example setup of our method for a segmentation task is illustrated in Figure 1.2. The segmentation model f_θ is trained on a dataset X_E of images that contain two nested squares. The task is to find a segmentation for the inner square. When following a naïve approach, applying the model f_θ directly to diffeomorphically deformed squares, we have an *Intersection-over-Union* (IoU) of 0.6887. The segmentation with canonicalisation has an IoU of 0.9299, which is significantly closer to the IoU of a perfect segmentation of 1. This demonstrates the benefit of our approach.

1.3 Contributions

The main contributions of this thesis can be summarised as follows:

We introduce a **diffeomorphism-equivariant neural network**. For this, we adapt LieLAC [72] to the infinite-dimensional group of diffeomorphisms, i.e., instead of an arbitrary Lie group, we construct equivariance over the group of diffeomorphisms. We parametrise the diffeomorphisms through stationary velocity fields. While following its general framework, we modify LieLAC so that it is usable for the group of diffeomorphisms. This is a novel approach, as LieLAC has primarily been studied for finite- or low-dimensional continuous groups. We extend the idea to an infinite-dimensional group: For the canonicalisation, we construct our own canonicalisation energy combining a variational autoencoder (VAE) loss with an adversarial discriminator [57, 63], and a deformation regularisation following [19]. Furthermore, we replace the optimisation strategy from LieLAC with a gradient-based optimisation strategy [19], which is specifically designed to find SVF-based diffeomorphisms. To be able to use this gradient-based strategy, we replace the typically supervised energy with our canonicalisation energy. Our method is a general framework, and can be used for various tasks requiring diffeomorphism-equivariance. As an example application, we perform a **segmentation task**, which has not been done by Shumaylov et al. [72] or anyone else to our knowledge.

To evaluate our diffeomorphism-equivariant neural network, we perform several experiments with the example diffeomorphism-equivariant segmentation. This is done on a synthetic dataset of nested squares that we created for this purpose, and on a real-world dataset of chest X-ray images. After hyperparameter tuning, we **compare our method on the synthetic dataset to a naïve approach and a data augmentation approach**. Furthermore, the **performance on a real-world dataset** is **compared to a naïve approach**. The **canonicalisation step** is theoretically **diffeomorphism-invariant** by construction. We confirm this in practice by comparing the canonicalised input of an original image and a diffeomorphically transformed one visually and by energy level.

Next to the practical investigations, we examine the **generalisation properties of canonicalisation**: We provide an informal proof that canonicalisation (as done in [72]) is generalisable. Under mild assumptions, training on canonicalised data ensures low expected loss on new canonicalised samples. This supports the choice of using the strategy of canonicalisation in a theoretical manner.

1.4 Structure of this Work

In this section, we provide an overview of the organisation of this work: After motivating the goal of this work as well as giving a summary of our proposed diffeomorphism-equivariant neural network and our distributions, we will finish this chapter (Section 1.5) by introducing a selection of related methods.

In **Chapter 2**, we introduce the mathematical foundations required throughout this work. Those include key concepts from topology, manifold theory, Lie groups, and measure theory.

Chapter 3 focuses on the theoretical generalisation properties of the canonicalisation method that we use. We first formalise the problem setup and then present results on generalisation and robustness of energy-based canonicalisation in a finite-dimensional and compact setting.

Our diffeomorphism-equivariant neural network is properly introduced in **Chapter 4**. We focus on an example segmentation task, which makes our general framework more tangible. Starting with the proposed model architecture, we then detail the canonicalisation process, segmentation step, and reverse canonicalisation.

We evaluate the proposed approach in **Chapter 5**. For this, a synthetic dataset is created. We describe the data generation process, implementation details, and hyperparameter tuning. This is followed by a presentation and discussion of the results, in which we compare our method with a naïve and a data augmentation approach on a synthetic dataset. Furthermore, we verify the invariance properties of the learned canonicalisation. We also evaluate the performance of DiffeoNN compared to a naïve approach for lung segmentation on real-world chest X-ray images.

In **Chapter 6**, we summarise the main findings, discuss limitations, and outline possible directions for future research.

1.5 Related Work

In the last decade, several strategies have been proposed to incorporate group symmetries into neural networks. These approaches range from task-specific designs for segmentation to more general architectural or orbit-based approaches. In the following, we discuss four classical approaches for equivariant networks: **data augmentation**, **group-equivariant layers**, **group-/frame-averaging** and **canonicalisation**. We present a selection of examples for these, and evaluate their suitability for achieving diffeomorphism-equivariance. An overview of these strategies is provided in Figure 1.3.

As we adapt our diffeomorphism-equivariant network exemplary for a segmentation

strategy	training data size	computational cost	group-equivariance theoretically guaranteed?	diffeomorphism group possible?	requires retraining if extending pretrained model?
data augmentation [71, 37]	large	high	no	yes	yes
group-equivariant layers [26, 25, 44]	small	low + architecture realisation	yes	no	yes
group-/ frame-averaging [64, 66]	small	low + averaging	approx.	no	no
canonicalisation [40, 72]	small	low + canonicalisation step	approx.	yes	no

Figure 1.3: Different strategies to realise group-equivariant neural networks. Even though data augmentation allows the group of diffeomorphisms, group-equivariance cannot be theoretically guaranteed. In addition, there is a large dataset needed for training and the computational costs are high, mainly because of the large (augmented) training data. The other strategies only require a relatively low amount of data in training. Group-equivariant layers can guarantee the group-equivariance by design but do not allow the group of diffeomorphisms, as the latter is not finite or compact. The computational cost for group-equivariant layers is only a low training cost and the cost needed to realise the network's architecture. In contrast, group- and frame-averaging do not need a modification of the network, but extend it, which results in a low general cost and the additional cost of the averaging step. In addition, group-equivariance can be theoretically guaranteed. A drawback is that the group of diffeomorphisms is not implementable, as it is infinite-dimensional, and does not have a Haar measure. On the contrary, the canonicalisation strategy does not have such restrictions and allows diffeomorphisms while still guaranteeing its equivariance approximately. Moreover, its computational cost is low, and it only the training of a simple model (that is also not modified) and the canonicalisation step are required. Another advantage of canonicalisation (and group-/ frame-averaging) is that those strategies can extend pretrained models to be group-equivariant without requiring retraining of them. Overall, the only strategy that is feasible to create a diffeomorphism-equivariant neural network is therefore canonicalisation.

task, we further review two strategies that are specialised for diffeomorphism-equivariant segmentation. Both approaches are based on image registration, a task closely connected to diffeomorphisms and image segmentation.

1.5.1 Data Augmentation

A common approach to improve a model's generalisation capacity and to avoid overfitting is data augmentation [71]. In this approach, additional training samples are generated synthetically by applying transformations to the original training data. When the transformations are symmetry-preserving, this method can be used to make the model invariant or equivariant to them.

Mainly used in computer vision, a classical data augmentation approach applies affine transforms such as scalings, rotations, translations, and reflections to the training data [45]. This approach has been extended to elastic deformation and noise in-

jection, as well as cropping and random erasing, and even modality changes (contrast, brightness) [22, 30, 73, 62]. A crucial requirement is that the transformations are label-preserving: for instance, rotating the digit six ("6") by 180 degrees turns it into the digit nine ("9"), which changes its class.

The transformation of the data is usually performed on-the-fly while training. The influence of data augmentation on the performance of a network has been widely studied in [71]: While it is highly effective in reducing overfitting, data augmentation cannot overcome all biases present in a small dataset. Nevertheless, it has been shown that data augmentation can prevent or at least reduce several forms of biases, e.g., lighting, scale, background, and noise.

Equivariance via Data Augmentation. Group-equivariance can be induced through data augmentation by selecting transformations from a symmetry group G. The network is then not only trained on the training data, but also on the same data transformed by the group G. In this way, group-equivariance is approximately achieved without the need to modify the network architecture. An example of creating a diffeomorphism-equivariant neural network through data augmentation can be found in [37].

The main drawback of data augmentation is that, by creating more data, the training is also computationally more expensive: Firstly, it requires applying a transformation to each training image, which can be quite expensive for diffeomorphisms. Secondly, with augmentation, the training data increases substantially. In addition, finitely sampling from a high-dimensional group of transformations, such as the group of diffeomorphisms, cannot capture the whole variability of the group. As a result, group-equivariance cannot be guaranteed in theory. Furthermore, the approach relies on the assumption that the network has the capacity to learn the symmetry in the first place. This makes data augmentation not suitable for the infinite-dimensional group of diffeomorphisms.

1.5.2 Group Equivariant Layers

In contrast to data-based methods, incorporating symmetries through equivariant architectures can guarantee group-equivariance by design. Recent years have seen increasing interest in such approaches, as they provide a way to encode known symmetries in neural networks directly and reduce the need for data augmentation or extra-processing, such as averaging [66, 64] or canonicalisation [40, 72]. This line of research falls under the umbrella of *geometric deep learning* [21], which studies models that respect symmetries and invariances induced by group actions on data.

Group-Equivariant Convolutional Networks. Convolutional neural networks (CNNs) are famously (approximately) translation-equivariant due to the weight-sharing in their convolutional layers [34, 51]. A standard convolution of a signal $f : \mathbb{Z} \to \mathbb{R}$ with a kernel $k : \mathbb{Z} \to \mathbb{R}$ is defined as

$$[f * k](x) := \sum_{y \in \mathbb{Z}} f(y)k(x - y), \tag{1.6}$$

and is equivariant under the group of translations. It has been suggested that this is an

important reason for the efficiency of CNNs in tasks such as image classification [46, 39]. As there might be other symmetries in the data, this has motivated several other neural networks, where group-equivariance or group-invariance is incorporated in the network layers. Dieleman et al. [31] proposed, for example, a rotation-invariant convolutional neural network.

This idea has been generalised to arbitrary finite groups via Group Convolutional Networks (G-CNNs) [26], where the standard convolutions are replaced with *group convolutions* over discrete symmetry groups. Given a finite group G acting on X, a feature map $f : G \to \mathbb{R}^d$ and a kernel $k : G \to \mathbb{R}^d$, the group convolution is defined as

$$[f *_G k](g) := \sum_{h \in G} f(h)k(g^{-1} \cdot h), \quad g \in G. \tag{1.7}$$

Group convolutions are G-equivariant by construction, and therefore, group convolutional networks (G-CNNs) guarantee exact equivariance to the group G. This setup generally assumes "nice" group actions, usually linear actions. Since their introduction, G-CNNs have been extended in multiple directions: to compact groups such as $SO(2)$ and $SO(3)$ via spherical CNNs [25], and to efficient implementations using fast Fourier transforms on groups [44].

In general, group convolutions require the group G to be finite (or at least compact and tractably parametrised [26]), as the convolution is defined by a discrete sum over G. This makes G-CNNs incompatible with continuous and infinite groups. This assumption fails for the diffeomorphism group, which is infinite-dimensional and cannot be represented by a finite set of group elements. Furthermore, the group of diffeomorphisms does not behave linearly. Therefore, G-CNNs cannot be used to build a diffeomorphism-equivariant network.

Lie Group Approaches. An approach to tackle these limitations is generalising to arbitrary Lie groups, which are continuous and differentiable groups. The idea is to replace the discrete sum by an integral with respect to the Haar measure η on a Lie group G, yielding the *Lie group convolution*

$$[f *_{\text{Lie}} k](g) = \int_G f(h)k(g^{-1} \cdot h)\mathrm{d}\eta(h). \tag{1.8}$$

This definition preserves equivariance under the continuous group G in the same way as in the discrete case. As Lie groups are locally compact, they always admit a Haar measure [82]. To make the Lie group convolutions computationally manageable, several strategies have been proposed:

LieConv [33] suggests parametrisations of Lie groups via their Lie algebras. Other approaches have focused on specific groups such as roto-translation groups [14], or used B-spline parametrisations to approximate filters on Lie groups [13]. A more recent method exploits the local structure of the Lie algebra to approximate group convolutions more efficiently [48].

While Lie group convolutions extend the idea to continuous and infinite groups via integration with respect to the Haar measure, the group needs to have a Haar measure. While the existence of a Haar measure is guaranteed for locally compact groups, it is generally not the case for non-compact groups. The group of diffeomorphisms is not locally

compact and does not have a Haar measure [61]. The approaches above can therefore not be used for diffeomorphisms. Furthermore, even if a fitting measure existed, we still needed to integrate over the whole group G, which is rather expensive for the group of diffeomorphisms. This motivates orbit-based approaches such as frame-averaging (Section 1.5.3) and canonicalisation (Section 1.5.4).

1.5.3 Averaging-based Approaches

An alternative to explicitly modifying the network architecture is to exploit symmetries by operating directly on the *orbits* of the group action. Instead of building equivariance into the layers, these methods either average over the orbit or parts of the orbit [64, 66, 32]. A group orbit of a given element x is defined as the set of all elements that can be reached from x by the action of all elements of a group G:

$$\mathcal{O}_x := \{g \cdot x \mid g \in G\}. \tag{1.9}$$

The main averaging-based strategies are *group averaging* [64], *frame averaging* [66] and *weighted frames* [32]. These strategies have the advantage that the network's architecture itself does not need to be adjusted. Equivariance is achieved solely by the averaging step.

Group Averaging and Frames. A common approach to achieve G-equivariance for a finite group G is frame averaging [66]. Frame averaging generalises the classical group averaging or Reynolds operator [64], which for a finite group G acts on a bounded continuous function $f : X \to Y$ as

$$R_G(f)(x) \;=\; \frac{1}{|G|} \sum_{g \in G} f(g^{-1} \cdot x). \tag{1.10}$$

The Reynolds operator is G-equivariant, but requires a summation over the whole group G, which makes it impractical when $|G|$ is large or infeasible when it is infinite.

To overcome this, *frames* restrict the averaging to a smaller, input-dependent subset: Let $x \in X$ be the input. The subset $F(x) \subseteq G$ is defined through a G-equivariant function $F : X \to 2^G/\{\emptyset\}$, i.e., $F(g \cdot x) = g \cdot F(x)$ for all $g \in G$. The resulting *frame average* [66] is then

$$F(f)(x) \;=\; \frac{1}{|F(x)|} \sum_{g \in F(x)} f(g^{-1} \cdot x). \tag{1.11}$$

Weighted frames [32] extend this idea further by allowing an input-dependent measure μ_x on G such that

$$\mu(f)(x) = \int_G f(g^{-1} \cdot x)\, d\mu_x(g). \tag{1.12}$$

This helps to define frames for non-finite groups and can guarantee continuity of the representation. But for non-compact or high-dimensional groups, it may be difficult to compute μ_x, as there exists no Haar measure, and therefore, the measure needs to be approximated. This can be computationally expensive, especially as the measure μ_x is input-dependent. Furthermore, we need to integrate over the whole group G. Therefore, group averaging, frame averaging, and weighted frames are not suitable for achieving diffeomorphism-equivariance.

1.5.4 Canonicalisation

An approach that is closely related to, but distinct from frame averaging is orbit canonicalisation [40]. Here, each input $x \in X$ is mapped to a representative element of its group orbit under the action of a transformation group G. The assumption is that the network performs well on those *canonical representatives*, and hence, once every input is replaced by its canonical form, a standard architecture can be applied without modifications. The step of finding the canonical representatives is called canonicalisation step and is usually realised by solving an optimisation problem that depends on the desired task and given data. We denote the set of optimal solutions to that problem $c(x)$. A canonical representative of an input x is then $x_c \in c(x)$. By reducing the input x to a canonical representative, the problem complexity of the network is reduced, as it only has to work well on the canonical representatives. Therefore, the approach is particularly suitable for large or continuous groups, where constructing fully equivariant architectures is expensive or even infeasible.

Canonicalisation is computationally efficient and easy to integrate with arbitrary models, but depending on the group, it cannot always guarantee continuity at the orbit boundaries [32]. It is also well-suited for handling *approximate symmetries*, since small perturbations can be absorbed by choosing stable canonical representatives [40]. Canonicalisation has been shown to outperform group averaging in sample efficiency when the dataset is sufficiently large [78].

The representational power of the canonical representative can be further improved through *probabilistic symmetry breaking* [50], which replaces equivariant functions with equivariant conditional distributions. In [50], this is implemented by introducing randomness into the canonicalisation step.

Lie Algebra Canonicalization (LieLAC). A promising approach is *Lie Algebra Canonicalization* (*LieLAC*) [72] that applies canonicalisation to arbitrary Lie groups. Given an energy function $E : X \to [0, \infty]$, we define the canonical representative of an orbit as the minimiser of the energy function E over the group orbit $\mathcal{O}_x$

$$C_E(x) := \arg\min_{y \in \mathcal{O}_x} E(y). \tag{1.13}$$

Intuitively, the energy E is chosen such that its minima are "close" to the training data X_E. The minimisation problem can be lifted to a distribution over the group orbit instead of the group orbit. LieLAC uses Lie algebra descent [72] to find a minimiser. The canonicalisation step can be combined with a pretrained model on simple training data X_E, which achieves equivariance without modifying the model's architecture. This makes it highly attractive for several applications, as existing models only need to be extended, and not modified.

Our work builds on the LieLAC approach [72] and adapts it to achieve diffeomorphism-equivariance. We use this energy-based canonicalisation idea but use another method to find the minimisers. Furthermore, we apply the method to a segmentation task for the first time, to our knowledge. So far, LieLAC has only been used for solving PDEs and in image classification [72]. Although the group of diffeomorphisms is not a Lie group, we can still apply the LieLAC strategy by changing the optimisation method.

In contrast to the other approaches mentioned above, the main principle of LieLAC does not depend on the group G being finite or locally compact, and we do not need to integrate over the whole group G. Through a specific choice of parametrisation of the group of diffeomorphisms, we can justify, why the theoretical guarantees on generalisation of the LieLAC strategy still hold approximately for diffeomorphisms.

1.5.5 Registration-based Approaches for Image Segmentation

As an example, we apply our diffeomorphism-equivariant method to an image segmentation task. In addition to the more general approaches from above, we therefore discuss two methods that present a diffeomorphism-equivariant network that is created task-specific for segmentation. As diffeomorphisms are widely used in medical image registration [19, 12], a combination of segmentation and image registration naturally arises to create diffeomorphism-equivariant networks. Furthermore, registration bears similarity to canonicalisation. As such, we use a modified registration method to find the minimiser of the energy in the canonicalisation step of our network.

An approach by Beekman et al. [11] for diffeomorphism-equivariant segmentation is to join image registration and segmentation in one network. The input image is segmented by voxel-wise classification through registering a prior segmentation. The diffeomorphic transformations are modelled via stationary velocity fields [28], and integrated using a Scaling and Squaring method [7]. For our method, we adopt this strategy to parametrise diffeomorphisms.

Similarly, Sheikhjafari et al. [70] combined registration and segmentation in a supervised framework. This framework is based on a deep learning parametrisation of diffeomorphic image registration. There, a deep learning network learns transformations from image pairs and corresponding segmentation outputs while ensuring diffeomorphisms through heavy constraints. The segmentation is achieved by transforming a "phantom" prior with the learned diffeomorphism, whereas canonicalisation maps the input to a common representative.

A limitation of both approaches is that they rely on labelled image pairs for training. This is not the case for our method. Moreover, they are tailored to segmentation tasks. Our strategy to achieve diffeomorphism-equivariance is more general and follows classical strategies to build equivariant neural networks.

2

Theoretical Background

This chapter introduces the mathematical background required for our diffeomorphism-equivariant framework, which extends the LieLAC approach [72] from finite-dimensional Lie groups to the infinite-dimensional group of diffeomorphisms. To establish both the theoretical and practical foundations of our method, we draw on concepts from differential geometry, Lie theory, and measure theory. The first two provide the geometric framework necessary for defining diffeomorphisms and equivariance, while measure theory is essential for the theoretical generalisation results discussed in Chapter 3.

Section 2.1 introduces the fundamental notions of topological spaces and manifolds, while also introducing diffeomorphisms. Section 2.2 then formalises Lie groups and related constructions, which form the basis of the equivariant architecture of LieLAC. Finally, Section 2.3 provides a brief introduction to measure theory, focusing on the concepts required for analysing the generalisation properties of energy-based canonicalisation in Chapter 3.

2.1 Topology, Manifolds, and Diffeomorphisms

To analyse an energy-based canonicalisation approach theoretically, we follow the setup of LieLAC [72], where the data space is modelled as a compact manifold. LieLAC focuses on Lie groups, which have the special property of being both smooth manifolds and groups simultaneously. While Lie groups and their related theory are introduced later in Section 2.2, we begin here by introducing smooth manifolds and the underlying geometric structure.

In our work, we extend the LieLAC framework, which achieves Lie group-equivariance to the infinite-dimensional group of diffeomorphisms. Diffeomorphisms are smooth, invertible maps between manifolds whose inverses are also smooth. They describe continuous deformations of a space and provide the natural framework for modelling smooth geometric transformations of images or shapes.

This section defines compact and smooth manifolds, introduces diffeomorphisms as smooth mappings between them, and establishes the background required for the theoretical and practical developments in later chapters. The presentation mainly follows [53] with contributions from [49, 43, 86, 2, 3].

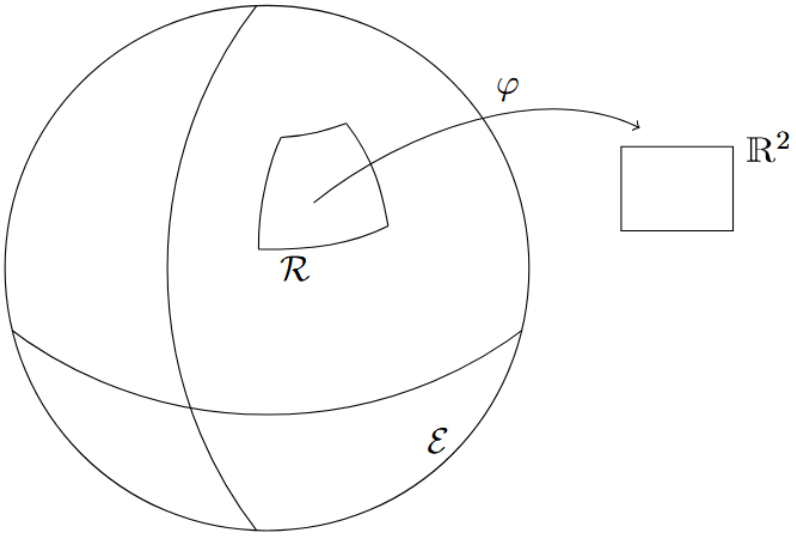


Figure 2.1: The surface of the Earth $\mathcal{E}$ modelled as the sphere in $\mathbb{R}^3$ is a manifold from [89]. A subset $\mathcal{R} \subset \mathcal{E}$ can be mapped with φ to a plane in $\mathbb{R}^2$.

2.1.1 Compact and Smooth Manifolds

A manifold is a *topological space* that locally resembles a real Euclidean space. A tangible example of a manifold is the surface of the Earth, modelled as the sphere in $\mathbb{R}^3$. There exists a map, for example, a geographic map, for every region of Earth, such that this region can be mapped to a plane in $\mathbb{R}^2$, as visualised in Figure 2.1 There also exists a collection of maps covering the whole Earth. This collection is called an atlas. In order to define a topological manifold, the concepts of a map and an atlas must be transferred to it. For this, some theoretical concepts are necessary. We begin with the most fundamental concept, a topological space:

Definition 2.1 (Topology, Topological Space, [53]). A topology on a set X is a collection $\mathcal{T}$ of subsets of X such that:

1. $\emptyset, X \in \mathcal{T}$,
2. any arbitrary union of elements of $\mathcal{T}$ belongs to $\mathcal{T}$, and
3. any finite intersection of elements of $\mathcal{T}$ belongs to $\mathcal{T}$.

The elements of $\mathcal{T}$ are called *open sets*. A subset $C \subseteq X$ is *closed* if its complement $X \setminus C$ is an open set. A pair $(X, \mathcal{T})$ consisting of a set X and a topology $\mathcal{T}$ on X is called a topological space. If the topology $\mathcal{T}$ is clear from the context, we refer to the topological space $(X, \mathcal{T})$ as X.

Note that arbitrary unions in a topology may be infinite, whereas intersections are required to be finite.

Hausdorff Space. The definition of a topological space is very general and flexible. This can lead to undesirable properties and complications. For example, every sequence

in X converges to every point in X in the *trivial topology* $\{\emptyset, X\}$ [53]. To avoid cases like this, one usually focuses their attention on *Hausdorff spaces*.

Definition 2.2 (Hausdorff Space, [53]). A Hausdorff space T_2 is a topological space in which distinct points have disjoint neighbourhoods (open subsets). More specifically, if $x, y \in T_2$ with $x \neq y$, then

$$\exists\, U_x, V_y \in T_2 : U_x \cap V_y = \emptyset. \tag{2.1}$$

An important property of a Hausdorff space is that every convergent sequence has a unique limit. A classic example of a Hausdorff space is the n-dimensional Euclidean space.

Example 2.3 (Euclidean Space, [53]). For any integer $n \geq 1$, the n-dimensional Euclidean space is the set $\mathbb{R}^n$ of ordered n-tuples of real numbers. We denote a point in $\mathbb{R}^n$ by $(x_1, ..., x_n)$, $(x_i)_{i=1}^n$ or x. The real numbers x_i (with $i = 1, ..., n$) are called the *coordinates* of x.

We use the Hausdorff space to define a compact manifold.

Second Axiom of Countability. Another property that we need to define the latter is *second countability*, which is based on the concept of *open bases*.

Definition 2.4 (Open Base, [53]). Let $(X, \mathcal{T})$ be a topological space. A collection of open sets $\mathcal{B} \subseteq \mathcal{T}$ is called a *base* for $\mathcal{T}$ if every open set $U \in \mathcal{T}$ can be written as a union of elements of $\mathcal{B}$. This means: for every open set $U \in \mathcal{T}$, there exists $\mathcal{A} \subseteq \mathcal{B}$ such that $U = \bigcup_{V \in \mathcal{A}} V$.

A topological space $(X, \mathcal{T})$ can be defined through a base. The *second axiom of countability* is then defined as follows:

Definition 2.5 (Second Countable Space, [53]). A topological space $(X, \mathcal{T})$ is said to satisfy the second axiom of countability if there exists a countable open base for the topology $\mathcal{T}$. A topological space $(X, \mathcal{T})$ satisfying the second axiom of countability is called a *second countable space*.

Continuous Maps. A central notion in topology, and later in the study of manifolds, is that of *continuous maps*. Intuitively, continuity formalises the idea that nearby points in one space are mapped to nearby points in another without changing the underlying topological structure. This concept allows us to compare spaces, as well as define, for example, differentiable maps between manifolds, which we need to define the "atlas" of a manifold. Moreover, continuity is the basis for many later constructions, as it underlies the definition of measurable and smooth maps, as well as diffeomorphisms, when we move from topology to measure theory and analysis on manifolds.

Definition 2.6 (Continuous Map, [53]). Let X and Y be topological spaces. A map $f : X \to Y$ is continuous if $f^{-1}(U)$ is open in X for all open sets $U \subseteq Y$.

Combining the notion of compactness with continuous maps results in the *extreme value theorem* on topological spaces, which guarantees that continuous functions on compact spaces attain their maximum and minimum values.

Theorem 2.7 (Extreme Value Theorem on Compact Spaces, [69]). *Let $(X, \mathcal{T})$ be a compact topological space and let*

$$f : X \to \mathbb{R} \tag{2.2}$$

be a continuous function. Then f attains both a maximum and a minimum value on X, i.e., there exist points $x_{\min}, x_{\max} \in X$ such that

$$f(x_{\min}) \leq f(x) \leq f(x_{\max}), \quad \forall x \in X. \tag{2.3}$$

Proof. See [69]. □

The classical Weierstrass theorem for continuous functions on closed and bounded subsets of $\mathbb{R}^n$ is a special case of Theorem 2.7, since such sets are compact in the standard topology. The generalised form applies equally to continuous functions on compact manifolds that are compact topological spaces (see Definition 2.10). This ensures the existence of global extrema for smooth functions defined on compact manifolds, which we need in Chapter 3 to prove the existence of the canonicalising elements.

A bijective continuous map f whose inverse f^{-1} is also continuous is a homeomorphism.

Definition 2.8 (Homeomorphism, [53]). Let X and Y be two topological spaces. The function $f : X \to Y$ is a homeomorphism if

(i) f is a bijection,
(ii) f is continuous,
(iii) the inverse $f^{-1} : Y \to X$ is also continuous.

Locally Euclidean. We can now use homeomorphisms to define what it means to be *locally Euclidean*. This is used to formalise the intuitive notion that a manifold behaves locally like an Euclidean space, even though the global structure might differ drastically.

Definition 2.9 (Locally Euclidean, [53]). A topological space X is called locally Euclidean of dimension n if every point x of X has a neighbourhood U_x that is homeomorphic to a subset of $\mathbb{R}^n$. In other words: for each $x \in X$, there exist

- an open set $U_x \subset X$,
- an open set $V_x \subset \mathbb{R}^n$, and
- a homeomorphism $\psi_x : U_x \to V_x$.

Having introduced all the necessary properties, we are now able to define topological manifolds:

Definition 2.10 (Topological Manifold, Compact Manifold, [53]). Let X be a topological space. We say X is an n-dimensional topological manifold X (or n-manifold) if it has the following properties:

- X is a Hausdorff space (Definition 2.2),
- X is second countable (Definition 2.5), and
- X is locally Euclidean of dimension n (Definition 2.9).

A topological manifold X is said to be *compact* if it is compact as a topological space, i.e., if every open cover of X admits a finite subcover. If the dimension is clear from the context or does not need to be specified, we refer to an n-manifold as a manifold.

Very useful subclasses of manifolds are smooth and differentiable manifolds. A differentiable manifold is locally similar enough to a vector space to have a globally defined differential structure.

Definition 2.11 (Smooth Manifold, Differentiable Manifold, [43]). A smooth manifold X, is a second countable Hausdorff space with a collection C of maps (U, ψ) such that:

(i) Every map $(U, \psi) \in C$ is a homeomorphism $\psi : U \to V$ with $U \subset X$ and $V \subset \mathbb{R}^n$ open. We call such a pair *chart*.
(ii) Every point $x \in X$ is in the domain of some chart.
(iii) For two charts $(U_i, \psi_i), (U_j, \psi_j) \in C$, $\psi_i : U_i \to V_i$ and $\psi_j : U_j \to V_j$ with $U_i \cap U_j \neq \emptyset$, the map $(\psi_i \circ \psi_j^{-1}) : \psi_j(U_i \cap U_j) \to \psi_i(U_i \cap U_j)$ is $\mathcal{C}^\infty$.
(iv) The collection C of charts is maximal under all collections satisfying (i) - (iii) (meaning, if $\phi : U \to V$ is a homeomorphism with $U \subset X$ and $V \subset \mathbb{R}^n$ open and fulfils (iii) for all $\psi \in C$ (commutatively), then $\phi \in C$).

A set of charts satisfying (i) - (iii) is called an *atlas*. If the map in $\psi_i \circ \psi_j^{-1}$ in (ii) is in $\mathcal{C}^1$, then X is called a differentiable manifold X.

2.1.2 Differentiable Maps and Diffeomorphisms

The notion of smooth manifolds provides a way to extend Euclidean principles to spaces that are only locally similar to $\mathbb{R}^n$. After defining smooth manifolds, we can now transfer the idea of differentiable functions to the manifold setting by introducing smooth maps. An important subclass of smooth maps is the class of diffeomorphisms. Diffeomorphisms preserve the manifold's differentiable structure and serve as the mathematical foundation for describing smooth deformations, coordinate changes, and symmetries. Before defining diffeomorphisms formally, we first introduce the general notion of differentiable and smooth maps between manifolds using charts:

Definition 2.12 (Differentiable Map, Smooth Map, [53]). Let X and Y be smooth manifolds. A function $f : X \to Y$ is said to be k-times differentiable if for every $x \in X$, there exist smooth charts (U, ψ) containing x and (V, ϕ) containing $f(x)$ such that $f(U) \subset V$ and the composite map

$$\phi \circ f \circ \psi^{-1} : \psi(U) \subset \mathbb{R}^n \to \mathbb{R}^m \tag{2.4}$$

is k-times differentiable in the Euclidean sense. If the composite map is infinitely differentiable, the function f is called a smooth map.

Smooth maps generalise differentiable functions to manifolds. To describe reversible transformations that preserve smooth structure, we now introduce the concept of *diffeomorphisms*. While homeomorphisms (see Definition 2.8) preserve only topological properties, they do not necessarily preserve smooth structures. This motivates investigating the set of diffeomorphisms.

Definition 2.13 (Diffeomorphism, $\mathcal{C}^k$-Diffeomorphism, [86]). Let X and Y be two differentiable manifolds. The differentiable map $f : X \to Y$ is a diffeomorphism if it is a bijection, and its inverse $f^{-1} : Y \to X$ is differentiable as well. We denote the set of diffeomorphisms by $\mathcal{D}(X, Y)$. If f and f^{-1} are k times continuously differentiable, f is called a $\mathcal{C}^k$-diffeomorphism.

In our application, diffeomorphisms represent smooth, invertible deformations of images. In this work, we introduce a neural network that is diffeomorphism-equivariant, ensuring consistent predictions under smooth, invertible deformations of the input.

Smooth and differentiable maps allow us to describe how one manifold can be smoothly deformed into another. To analyse such deformations quantitatively, however, we need a way to measure how much one configuration differs from another.

2.1.3 Tangent Space and Geodesic Distance

The notion of distance on smooth manifolds is crucial for our later theoretical investigations. To define such a distance, we first require a local linear structure that allows us to measure lengths and angles. To capture the local behaviour of the manifold around each point, we introduce the tangent space. We can then define an inner product on tangent spaces, which leads to the concept of a *Riemannian metric*. This metric induces a natural notion of distance between points, the *geodesic distance*.

We begin by introducing *tangent vectors* and the *tangent space* at a point x in a smooth manifold X.

Definition 2.14 (Tangent Vector, Tangent Space, [53]). Let X be a smooth manifold. A tangent vector at $x \in X$ is a linear map

$$v_x : \mathcal{C}^\infty(X) \to \mathbb{R}, \tag{2.5}$$

satisfying the *Leibniz rule*

$$v_x(fg) = f(x)\, v_x(g) + g(x)\, v_x(f), \quad \forall f, g \in \mathcal{C}^\infty(X). \tag{2.6}$$

The set of all tangent vectors at x forms a real vector space [53], denoted by T_xX, called the tangent space of X at x.

One can visualise a tangent vector v_x to an abstract smooth manifold X as an "arrow" that is tangent to X with a "starting point" that is attached to X at the given point x. We can combine all the tangent spaces to a *tangent bundle*. The tangent bundle collects all tangent spaces of the manifold into a single smooth structure.

Definition 2.15 (Tangent Bundle, [53]). The tangent bundle of a smooth manifold X is the disjoint union

$$TX = \bigsqcup_{x \in X} T_x X, \tag{2.7}$$

together with the natural projection $\pi : TX \to X$, $(x, v) \mapsto x$.

Before we are able to define a distance, we need to introduce a Riemannian metric and a *Riemannian manifold*. In those, the Euclidean concept of an inner product is transferred to manifolds.

Definition 2.16 (Riemannian Metric, Riemannian Manifold, [53]). Let X be a smooth manifold. A Riemannian metric g on X is a smooth family of inner products on the tangent spaces of X. Namely, g associates to each $x \in X$ a positive definite symmetric bilinear form on $T_x X$,

$$g_x : T_x X \times T_x X \to \mathbb{R}, \tag{2.8}$$

which is smooth in the sense that the function

$$x \in X \mapsto g_x(U_x, V_x) \in \mathbb{R} \tag{2.9}$$

must be smooth for every locally defined smooth vector field U_x, V_x in X. A Riemannian manifold is a pair (X, g), where X is a smooth manifold and g is a Riemannian metric on X.

A useful fact is that every smooth manifold admits a Riemannian metric [53]. This makes every smooth manifold a Riemannian manifold. We can define a notion of distance between points on a Riemannian manifold by using curves:

Definition 2.17 (Geodesic Distance, [53]). Let (X, g) be a Riemannian manifold. The *length* of a smooth curve $\gamma : [0, 1] \to X$ is defined as

$$L(\gamma) := \int_0^1 \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))}\, dt. \tag{2.10}$$

The *geodesic distance* between two points $x, y \in X$ is then given by

$$d_g(x, y) := \inf_{\gamma \in \Gamma(x,y)} L(\gamma), \tag{2.11}$$

where $\Gamma(x, y)$ denotes the set of all smooth curves $\gamma : [0, 1] \to X$ with $\gamma(0) = x$ and $\gamma(1) = y$.

The geodesic distance $d_g(x, y)$ generalises the Euclidean distance to curved spaces and measures the length of the shortest smooth path connecting two points on the manifold. We will use this distance in our theoretical investigations in Chapter 3 to measure how "close" an image is to an element x_i in the training dataset X_E.

2.2 Lie Group Theory

Having defined the underlying theoretical background, we are now able to introduce Lie groups. This section is based on [43, 53, 2, 3, 82]. Lie groups are a central component of the introduced canonicalisation method used in [72], whose generalisation behaviour we will investigate in Chapter 3, and on which our method DiffeoNN is based. These groups are particularly interesting because they have all the properties of a differentiable manifold while also having a group structure.

Definition 2.18 (Group, [3]). A group $(G, *)$ is a set G in combination with a binary operator $* : G \times G \to G$, $(f, g) \mapsto f * g$, such that:

(i) *Associativity*: For all $f, g, h \in G$ one has $(f * g) * h = f * (g * h)$,
(ii) *Identity element*: There exists a unique element $e \in G$ such that for every $f \in G$ one has $e * f = f * e = f$. This element is the so-called *identity element* (or *neutral element*) of the group G.
(iii) *Inverse element*: For all $f \in G$, there exists an unique *inverse element* $g \in G$ such that $f * g = g * f = e$.

If the underlying space is compact, G is called a *compact group*.

Compact groups play an important role in analysis and geometry because they admit a (up to scaling) unique bi-invariant measure, called the *Haar measure* (see Definition 2.41), which allows integration of functions over G in a group-invariant manner. Typical examples include the circle group $\mathrm{SO}(2)$ and the special orthogonal group $\mathrm{SO}(n)$. An example of locally compact groups are Lie groups:

Definition 2.19 (Lie Group, [43]). A (finite-dimensional) Lie group is a set G that is a group and simultaneously a smooth n-manifold such that the *multiplication map*

$$(f, g) \mapsto f * g : G \times G \to G \tag{2.12}$$

and the *inversion map*

$$f \mapsto f^{-1} : G \to G \tag{2.13}$$

are smooth.

Example 2.20. Examples of Lie groups include:

- $(\mathbb{R}^n, +)$, the additive group of vectors,
- $GL(n, \mathbb{R})$, the *general linear group* of invertible $n \times n$ matrices with the classical matrix multiplication,
- $SO(n)$, the special orthogonal group of rotations in $\mathbb{R}^n$ with the classical matrix multiplication.

An example of a group that is not a Lie group, is the group of diffeomorphisms:

Example 2.21. The set of diffeomorphisms that maps a smooth manifold X to itself; so $X = Y$ in Definition 2.13, forms an infinite-dimensional group under composition [86]. We denote this group as $\mathcal{D}(X)$. As the group of diffeomorphisms is infinite-dimensional, it is not a Lie group.

The concept of Lie groups can be extended to an infinite-dimensional setting. In the following, we will focus on the finite-dimensional Lie groups. For the infinite-dimensional case, we refer to [65]. Every Lie group G defines a *Lie algebra*:

Definition 2.22 (Lie Algebra, [53]). A Lie algebra is a real vector space $\mathfrak{g}$ endowed with a map, the *Lie bracket*,

$$[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}, \tag{2.14}$$

that satisfies the following properties: For all $u, v, w \in \mathfrak{g}$ and for all $a, b \in \mathbb{R}$ holds

(i) $[u, v+w] = [u, v]+[u, w]$, $[u, av] = a[u, v]$ and $[v+w, u] = [v, u]+[w, u]$, $[au, v] = a[u, v]$ (bilinearity),
(ii) $[u, v] = -[v, u]$ (antisymmetry),
(iii) $[u, [v, w]] + [v, [w, u]] + [w, [u, v]] = 0$ (Jacobi Identity).

In the following, we will refer to a Lie algebra with only $\mathfrak{g}$ instead of $(\mathfrak{g}, [\cdot,\cdot])$ for simplicity reasons.

Example 2.23. The Lie algebra for the example Lie group $GL(n, \mathbb{R})$ (see Example 2.20) is $M(n, \mathbb{R})$. The Lie algebra $M(n, \mathbb{R})$ is the vector space of all $n \times n$ real matrices with the Lie bracket $[A, B] = AB - BA$. This Lie bracket is also called *commutator*.

The Lie algebra associated with a Lie group G corresponds to the tangent space of G at the identity element, i.e. $T_e(G)$ where $e \in G$ is identity element.

2.2.1 One-Parameter Subgroup and Exponential Map

After introducing the basic concepts of Lie groups and their associated Lie algebras, we now turn to the connection between the two. While the Lie group captures more global transformations, its Lie algebra describes the local structure around the identity element. To move smoothly between these two representations, we introduce the concept of a *one-parameter subgroup* and define the *exponential map*. Intuitively, the exponential map translates a vector in the Lie algebra to a curve on the Lie group. One can think of this as starting at the identity and "flowing" through the group according to a constant velocity given by the Lie algebra element. We also use a similar concept later on, when we parametrise diffeomorphisms through stationary velocity fields, see Section 4.2.1.

Let us start with the one-parameter subgroup:

Definition 2.24 (One-Parameter Subgroup, [3]). Let $(G, *)$ be a Lie group. A one-parameter subgroup of G is a smooth function $f : \mathbb{R} \to G$ with a continuous derivative satisfying

$$f(x_1 + x_2) = f(x_1) * f(x_2) \tag{2.15}$$

for all $x_1, x_2 \in \mathbb{R}$.

The *exponential map* plays an important role, as it is a smooth map between the linear structure of the Lie algebra and the non-linear structure of the Lie group. The exponential map defines a unique one-parameter subgroup for every element v in the Lie algebra $\mathfrak{g}$ of a Lie group G.

Definition 2.25 (Exponential Map, [43]). Let G be a Lie group with the Lie algebra $\mathfrak{g}$. For every $v \in \mathfrak{g}$, there exists a unique smooth one-parameter subgroup $\phi_v : \mathbb{R} \to G$ such that

(i) $\phi_v(0) = e$,
(ii) $\phi_v'(0) = v$,

where e is the identity element of G and ϕ_v' is the derivative of ϕ_v. The exponential map is defined by $\exp(v) := \phi_v(1)$.

Equivalently, it holds $\exp(tv) = \phi_v(t)$. As an intuition: the exponential map gives the point reached by "flowing" from the identity along v for time t. Following the Lie group $GL(n, \mathbb{R})$ from Example 2.20 and Example 2.23, we can define an exponential map:

Example 2.26. For the matrix Lie group $GL(n, \mathbb{R})$, the exponential map is the usual matrix exponential, so for $t \in \mathbb{R}$ and $A \in \mathfrak{g} = M(n, \mathbb{R}) \subset \mathbb{R}^{n \times n}$ the map exp is defined as

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k. \tag{2.16}$$

Having introduced smooth maps between Lie groups and their Lie algebra, we will now focus on Lie groups acting on another space.

2.2.2 Lie Group Action and Group Orbits

The LieLAC setup [72] relies on a Lie group G acting on a compact manifold that represents the data space on which our network operates. This "action" describes how each group element transforms points on the manifold, and therefore defines the notion of equivariance, which is the goal of our network. In the following, we therefore introduce the formal definition of a *Lie group action* and the related concepts of *group orbits* and *stabilisers*, which provide the foundation for the following sections:

Definition 2.27 (Group Action, Lie Group Action, [43]). Let G be a Lie group and X be a smooth manifold. A (left) group action of G on X is a map

$$\alpha : G \times X \to X, \quad (g, x) \mapsto g \cdot x, \tag{2.17}$$

such that

1. $e \cdot x = x$ for all $x \in X$ (identity property) and
2. $g_1 \cdot (g_2 \cdot x) = (g_1 \cdot g_2) \cdot x$ for all $g_1, g_2 \in G$ and $x \in X$ (associativity).

If the map α is differentiable, it is called Lie group action. We will also say that the Lie group G acts smoothly on the manifold X.

Group actions partition the space X into orbits, which play a crucial role in our theoretical investigations in Chapter 3.

Definition 2.28 (Group Orbit, Stabiliser, [43]). Let α be a group action of the Lie group G on the smooth manifold X. Let $x \in X$.

- The group orbit of x is defined as

$$\mathcal{O}_x = \{g \cdot x | g \in G\} \subseteq X. \tag{2.18}$$

- The stabiliser of x is defined as

$$G_x = \{g \in G | g \cdot x = x\}. \tag{2.19}$$

While an orbit $\mathcal{O}_x$ represents the set of all points that can be reached from $x \in X$ by applying elements of the group G, the stabiliser G_x describes the subgroup that leaves x unchanged. The following theorem shows how these two concepts are intrinsically linked:

Theorem 2.29 (Orbit Stabiliser Theorem, [43]). *Let a Lie group G act smoothly on a manifold X. The stabiliser G_x is a closed Lie subgroup, and the natural map $G/G_x \to \mathcal{O}_x$ is a diffeomorphism onto the orbit $\mathcal{O}_x$.*

Proof. For a detailed proof, see [43, Theorem 3.29]. □

Note that the Lie group partitions a manifold into equivalence classes of points, when we consider two elements $x, y \in X$ equivalent under the Lie group action if $y \in \mathcal{O}_x$ [43]. Orbits are especially relevant to understand and investigate canonicalisation, as the idea is to find for every input x a representative within its orbit $\mathcal{O}_x$ that is "close" to the training data. The space containing all group orbits is therefore particularly interesting for us:

Definition 2.30 (Orbit Space, [53]). The set of all orbits is called *orbit space*,

$$X/G := \{\mathcal{O}_x | x \in X\}. \tag{2.20}$$

The map sending $x \in X$ to its orbit $\mathcal{O}_x$ is called *projection map*.

Definition 2.31 (Projection Map, [53]). Let the Lie group G act smoothly on the manifold X. The projection map $\pi : X \to X/G$ sends $x \in X$ to its orbit $\mathcal{O}_x$:

$$x \mapsto \mathcal{O}_x = \{g \cdot x | g \in G\}. \tag{2.21}$$

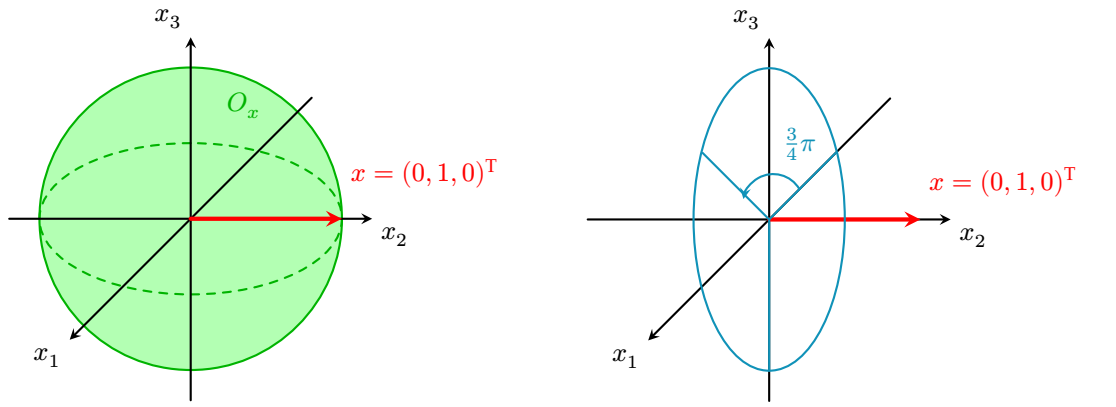
Let us explain the concept of group orbits and stabilisers on a simple example further:

Example 2.32. Let the manifold $\mathcal{X}$ be the three-dimensional Euclidean space $\mathbb{R}^3$ with the norm $\|\cdot\|$ that is induced by the scalar product of $\mathbb{R}^3$. Furthermore, let the Lie group G be the rotations in this space, so

$$SO(3) := \{R \in \mathbb{R}^{3\times 3} | R^\top R = I, \det(R) = 1\}. \tag{2.22}$$

As an example, we choose $x = (0, 1, 0)^\top \in \mathbb{R}^3$. The group orbit of this x is then the $3D$ unit sphere:

$$\mathcal{O}_x = \{y \in \mathbb{R}^3 | \, \|y\| = 1\}, \tag{2.23}$$



(a) The group orbit $\mathcal{O}_x$ of $x = (0,1,0)^\top$ is the $3D$ unit sphere.

(b) The stabilisers G_x are the group of rotations in the x_1–x_3 plane.

Figure 2.2: Example of the group orbit $\mathcal{O}_x$ (a) and the stabilisers G_x for $x = (0,1,0)^\top$.

because precisely for all $y \in \mathbb{R}^3$ with $\|y\| = 1$, there exists a rotation $R \in SO(3)$ that maps x to y, i.e., $Rx = y$. The group stabilisers are all the rotations that map x back to x:

$$G_x = \left\{ R_\alpha := \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \;\middle|\; \alpha \in [0, 2\pi) \right\}, \tag{2.24}$$

which is the set of all rotations that rotate within the plane with x as normal vector and the identity matrix. The group orbit and stabiliser of x are visualised in Figure 2.2. For an arbitrary $x \in \mathbb{R}^3$, we can define the group orbit as

$$\mathcal{O}_x = \{y \in \mathbb{R}^3 | \; \|y\| = \|x\|\}. \tag{2.25}$$

Thus, the orbits are all the three-dimensional spheres with varying radii.

The example above shows how the action of a Lie group sections a manifold into group orbits, where each orbit contains all points that can be reached from one another by a transformation out of a group, here rotations.

We now want to shift our focus from points on the manifold to distributions defined over them. Instead of considering the space itself, we are interested in how a probability mass or a measure is spread across the manifold and its orbits. This step is important for our later theoretical work, where we analyse how functions, more specifically losses, behave under transformations of the underlying data distribution.

2.3 Measure Theory

To formalise these ideas, we first recall some basic concepts from measure theory. Those notions allow us then to define measures on Lie groups and on group orbits, which will play a central role in the theoretical analysis in Chapter 3. We follow the notation and conventions of [17, 18, 52].

Definition 2.33 (σ-Algebra, [17]). Let X be a set with the *power set* 2^X; the set of all subsets of X. A σ-algebra Σ_X is a subset of 2^X such that

(i) $X \in \Sigma_X$,
(ii) if $A \in \Sigma_X$ then $X/A \in \Sigma_X$, and
(iii) for any countable collection $(A_n)_{n\in\mathbb{N}}$ of sets in Σ_X, the union is contained in Σ_X, i.e., $\bigcup_{n\in\mathbb{N}} A_n \in \Sigma_X$.

The σ-algebra allows us to define what "measurable" means formally.

Definition 2.34 (Measurable Set/Space, [17]). Let Σ_X be the σ-algebra of the set X.

- A set $A \in \Sigma_X$ is called a *measurable set*.
- The pair (X, Σ_X) is called a *measure space*.

For any topological space, there exists a standard σ-Algebra, the so-called Borel σ-Algebra.

Definition 2.35 (Borel σ-Algebra, [17]). Let X be a topological space. The Borel σ-algebra $\mathcal{B}(X)$ is the smallest σ-algebra such that all open sets are measurable. The pair $(X, \mathcal{B}(X))$ is called *Borel space*. If it is clear from the context, we will simply write X to denote the Borel space $(X, \mathcal{B}(X))$.

So, there exists a measurable structure for any topological space, which allows us to define measures on it. The concept of "measurability" is also applicable to functions.

Definition 2.36 (Measurable Function, [17]). Let (X, Σ_X) and (Y, Σ_Y) be measure spaces. A function $f : X \to Y$ is a *measurable function* if for all measurable sets $B \in \Sigma_Y$, the set $A = f^{-1}(B)$ is measurable.

Having specified measurable spaces and functions, we can now introduce the concept of a measure itself.

Definition 2.37 (Measure, [17]). Given a measure space (X, Σ_X), a measure is a function $\mu : \Sigma_X \to \mathbb{R}_{\geq 0}$ such that

(i) $\mu(\emptyset) = 0$ and
(ii) for all countable collections $(A_n)_{n\in\mathbb{N}}$ of pairwise disjoint sets in Σ_X, it holds that

$$\mu\Big(\bigcup_{n\in\mathbb{N}} A_n\Big) = \sum_{n\in\mathbb{N}} \mu(A_n). \tag{2.26}$$

A measure is called finite if $\mu(X) < \infty$. A *probability measure* is a measure μ such that $\mu(X) = 1$.

Remark 2.38. Let (X, Σ_X) be a measure space and $\mu : \Sigma_X \to \mathbb{R}_{\geq 0}$ a finite measure such that there exists a $A \in \Sigma_X$ with $\mu(A) \neq 0$, i.e., $\mu(X) \neq 0$. One can always define a probability measure μ' on (X, Σ_X) as

$$\mu'(A) := \frac{\mu(A)}{\mu(X)} \tag{2.27}$$

for any $A \in \Sigma_X$. This probability measure is called the *normalised measure associated with* μ.

Measures can also be "moved" between spaces via pushforward construction of measurable functions. This is particularly useful for our work, as we want to move between a manifold X and its orbit space X/G.

Definition 2.39 (Pushforward Measure, [17]). Suppose (X, Σ_X) and (Y, Σ_Y) are measure spaces. Let $f : X \to Y$ be a measurable function and μ a measure on (X, Σ_X). The *pushforward measure* $f_*\mu$ is defined as follows:

$$f_*\mu(B) := \mu(f^{-1}(B)) \tag{2.28}$$

for all $B \in \Sigma_Y$.

We are particularly interested in measures that are compatible with the underlying topology of Lie groups and manifolds. These allow us to integrate functions over such spaces in a well-defined way and to study how probability mass behaves under smooth transformations. Let us therefore introduce specific types of spaces and measures that are relevant for our later constructions: The measure on a *Radon space* helps us to define a measure on a (locally) compact group, such as a Lie group.

Definition 2.40 (Radon Space, [18]). Let X be a topological space. Furthermore, let the set $\Pi_B(X)$ be the set of all Borel probability measure on X, meaning all the probability measures defined on the Borel σ-algebra $\mathcal{B}(X)$ of X. The space X is a Radon space if all the measures $\mu \in \Pi_B(X)$ are inner regular, i.e., for all $A \in \mathcal{B}(X)$ holds

$$\mu(A) = \sup\{\mu(F) | F \subseteq A, F \in \mathcal{B}(X), F \,\text{compact}\}. \tag{2.29}$$

A probability measure on a Radon space is called a *Radon measure*.

Definition 2.41 (Haar Measure, [18]). Let G be a locally compact topological group. A *Haar measure* η on G is a Radon measure on $\mathcal{B}(G)$ that is left-invariant under the group action, i.e.,

$$\eta(gA) = \eta(A), \quad \forall g \in G,\ A \in \mathcal{B}(G). \tag{2.30}$$

If G is compact, the Haar measure can be normalised to satisfy $\eta(G) = 1$, in which case it is a probability measure.

For Lie groups, Haar measure corresponds to the canonical "volume element" compatible with the group structure. We furthermore define the *Riemannian volume measure* that generalises the Lebesgue measure from $\mathbb{R}^n$ to curved manifolds.

Definition 2.42 (Riemannian Volume Measure, [52]). Let (X, g) be an oriented Riemannian manifold of dimension n. There exists a unique smooth n-form, called the *Riemannian volume form* and denoted $d\mu_V$, determined by the metric g and the chosen orientation. In local coordinates $(x^1, \dots, x^n)$ around a point $x \in X$, it is given by

$$d\mu_V(x) = \sqrt{\det(g_{ij}(x))}\, dx^1 \wedge \cdots \wedge dx^n, \tag{2.31}$$

where (g_{ij}) are the components of the metric tensor in these coordinates. The corresponding measure on the Borel σ-algebra $\mathcal{B}(X)$ is called the *Riemannian volume measure*. If X is compact, its *volume* is defined by

$$\mathrm{Vol}(X) = \int_X 1 \, d\mu_V = \int_X d\mu_V. \tag{2.32}$$

With these concepts, we can now describe how measures behave on Lie groups and smooth manifolds. In particular, the Haar measure provides an invariant way to integrate over the group itself, while the Riemannian volume measure extends this idea to general manifolds. In the following, we combine these notions with the previously introduced concept of group orbits. This allows us to define how a measure on a manifold can be projected to the orbit space and how it can be decomposed into orbit-wise components.

2.3.1 Pushforward Orbit Measure and Disintegration

When a Lie group G acts on a compact manifold X, one can push forward measures to the quotient space.

Definition 2.43 (Pushforward Orbit Measure). Let G be a compact Lie group acting on the compact manifold X. Let μ be a measure on X and let $\pi : X \to X/G$ be the projection map that sends $x \in X$ to its orbit $\mathcal{O}_x$ (Definition 2.31). The orbit measure (or pushforward measure on the orbit space) is defined as

$$\mu_{\mathcal{O}}(A) = \pi_* \mu(A) = \mu(\pi^{-1}(A)), \tag{2.33}$$

where $A \subseteq X/G$ is a measurable set.

This measure ensures that integration over the quotient space corresponds to integration over X:

$$\int_{X/G} f(\mathcal{O}_x) \, d\pi_* \mu(\mathcal{O}_x) = \int_X f(\pi(x)) \, d\mu(x) \tag{2.34}$$

for every measurable function $f : X/G \to \mathbb{R}$. This follows directly from the definition of the pushforward measure (Definition 2.39), as it is a simple change of variables.

Using this pushforward measure and the concept of disintegration [18, Section 10.6], one can also decompose μ into orbit components:

Theorem 2.44 (Disintegration Theorem for Orbits). *Let G be a compact Lie group acting on the compact manifold X. Let $\Pi_B(X)$ denote the collection of Borel probability measures on X. Let μ be a Borel probability measure on X, i.e., $\mu \in \Pi_B(X)$, and let $\pi : X \to X/G$ be the projection map that sends $x \in X$ to its orbit $\mathcal{O}_x$. If the spaces X and X/G are Radon spaces, and the projection map $\pi \in \Pi_B(X)$, then there exists a $\pi_*\mu$-almost everywhere uniquely determined family of probability measures $\{\mu_{\mathcal{O}_x}\}_{\mathcal{O}_x \in X/G} \subseteq \Pi_B(X)$ such that:*

1. *The map $\mathcal{O}_x \mapsto \mu_{\mathcal{O}_x}$ is measurable in the sense that $\mathcal{O}_x \mapsto \mu_{\mathcal{O}_x}(B)$ is Borel-measurable for each $B \subseteq X$.*

2. *For $\pi_*\mu$-almost all $\mathcal{O}_x \in X/G$,*

$$\mu_{\mathcal{O}_x}(A) = \mu_{\mathcal{O}_x}(A \cap \mathcal{O}_x), \quad \forall A \in \mathcal{B}(X), \tag{2.35}$$

i.e., each conditional measure is supported on its orbit.

3. *For every Borel-measurable function $f : X \to \mathbb{R}_{\geq 0}$, the* disintegration formula *holds:*

$$\int_X f(x)\, d\mu(x) = \int_{X/G} \int_{\mathcal{O}_x} f(y)\, d\mu_{\mathcal{O}_x}(y)\, d\pi_*\mu(\mathcal{O}_x). \tag{2.36}$$

In particular, choosing $f = \mathbb{1}_A$ for $A \subseteq X$, i.e., as the indicator function of A, yields

$$\mu(A) = \int_{X/G} \mu_{\mathcal{O}_x}(A)\, d\pi_*\mu(\mathcal{O}_x). \tag{2.37}$$

This *measure disintegration* shows how a measure on X can be decomposed into orbit-wise conditional measures and a pushforward measure on X/G. It provides the foundation for the theoretical evaluation of the generalisation properties of the canonicalisation framework in Chapter 3. For further notions on disintegration and measures, see also [18, 27].

After introducing the theoretical background, we can now focus on the generalisation properties of canonicalisation models, which we investigate in the next chapter.

3

On Generalisation of Canonicalisation

Canonicalisation has been proposed as a general strategy for handling symmetries in learning problems, especially with respect to high-dimensional groups. Despite its practical importance, the theoretical foundations of canonicalisation remain poorly understood. Current methods, while empirically successful, lack generalisation guarantees. One reason for this is that canonicalisation is a relatively new concept. In this chapter, we analyse canonicalisation models following the LieLAC approach [72], where canonical representatives are selected by minimising an energy that is chosen task-specific and learned from training data. The energy-based approach allows us to give theoretical guarantees that were not possible with other setups. The main goal is to formalise the generalisation behaviour and determine the conditions under which a canonicalisation rule learned from a finite dataset can be expected to generalise to unseen samples from the set of training data that was transformed by a group G, i.e., group-equivariance is actually achieved by the investigated network.

Since canonicalisation was introduced, there have been several attempts at theoretical analysis. While some works have evaluated its generalisation properties and sample complexity under restrictive assumptions, such as finite orthogonal groups and polynomial function classes [79], others have focused on the regularity and stability of canonicalised models [80]. Even though setting a good foundation, they have strong constraints, for example, they require large training datasets. Here, we continue this line of work in the context of energy-based canonicalisation, rather than polynomial approximation or continuity constraints: In the following, we give an informal proof that illustrates the expected generalisation error in our setup: For energy-based canonicalisation following LieLAC, we prove that the generalisation error is bounded as follows:

$$\begin{aligned}\text{generalisation error} < \ &\text{training error} + \text{mass of poorly-sampled orbits} \\ &\cdot (\text{maximal loss} - \text{training error}).\end{aligned}$$

Note that this is the generalisation error before we apply the reverse canonicalisation. We also show that, under additional assumptions, this bound can provide a bound for the loss of the overall network. To our knowledge, this is the first time a generalisation bound for energy-based canonicalisation has been proposed in a theoretical manner. We intend to extend this informal proof to a formal one in future works, but a proper formalisation exceeded this work's frame.

In the following section (Section 3.1), we describe the general setup. That section mainly focuses on the used measures, the formalisation of the canonicalisation and well-sampled orbits. We then introduce the needed assumptions (Section 3.2.1), and the parameter selection (Section 3.2.2). In Section 3.2.3, we prove the main theorem of this chapter. In the end, we investigate under which conditions the bounded expectation can give us a bound for the expectation after the application of the reverse canonicalisation.

3.1 Setup of the Learning Scenario

We follow the setup from [72] and formalise the necessary measures, orbits, and canonicalising elements. For more details on the theoretical background of group and measure theory, see Chapter 2. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a compact d-dimensional manifold (e.g., the space of discretised images) and let G be a compact Lie group that acts smoothly on $\mathcal{X}$. For each $x \in \mathcal{X}$, the orbit under the group action is defined as

$$\mathcal{O}_x = \{g \cdot x | g \in G\}. \tag{3.1}$$

The corresponding orbit space is denoted as $\mathcal{X}/G$. Based on the disintegration theorem (Theorem 2.44 and [27]), we can assume that there exists an (orbit) measure $\omega_{\mathcal{O}_x}$ on each orbit. We also have a G-invariant measure μ_T on $\mathcal{X}$. As a model can only be trained on finite samples, we furthermore consider the finite training set

$$X_E = X_E^N = \{x_1, ..., x_N\} \subseteq \mathcal{X}, \tag{3.2}$$

where $x_i \sim \text{Law}(\mu_T)$ for $i = 1, ...N$, meaning every x_i is drawn independently under the distribution μ_T.

3.1.1 Model Architecture

Let us now define the set X as

$$X := \{g \cdot x | g \in G, x \in X_E\} \subseteq \mathcal{X}, \tag{3.3}$$

which contains all the orbits of the training samples. The goal is to find a model $\tilde{f}_\theta$ that fulfils a specific task for all $x \in X$, e.g., a segmentation, while only training on the empirical dataset X_E. The model $\tilde{f}_\theta$ maps the input space $\mathcal{X}$ to an output space $\mathcal{Y}$ that depends on the specific task.

Example 3.1. Let us simplify the setup with an example. This example is technically an infinite-dimensional manifold $\mathcal{X}$. The overall goal of our theoretical investigation is to provide a bound for an infinite-dimensional data manifold, but this goes beyond the scope of this thesis. Ergo, we focus only on compact d-dimensional manifolds. Still, we give the intuition following the overall goal of an infinite-dimensional setting for a specific example:

- Let $\mathcal{X}$ be the space of all images, modelled as functions that map an image domain, say, $\Omega = [-1, 1] \times [-1, 1] \subset \mathbb{R}^2$ to $[0, 1]$, i.e.,

$$\mathcal{X} = \{x | x : \Omega \to [0, 1]\}. \tag{3.4}$$

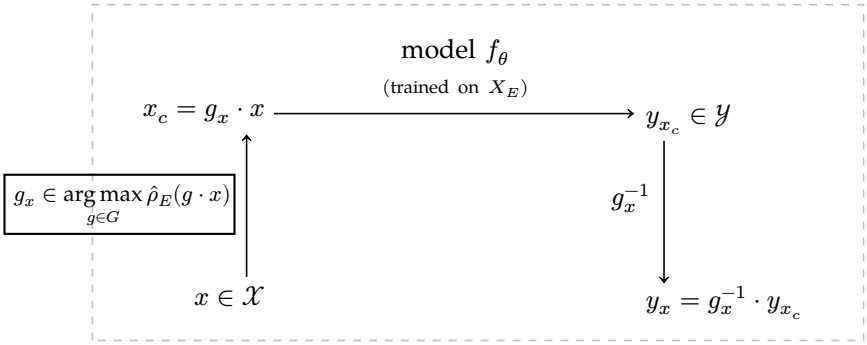


Figure 3.1: Setup of the learning scenario. An input $x \in \mathcal{X}$ is canonicalised to $x_c = g_x \cdot x$, where g_x is a canonicalising element that is determined by solving an optimisation problem, which depends on the task and on the training data X_E. Then the model f_θ is applied to x_c. Afterwards, the output from the central model $f_\theta(x_c) = y_{x_c}$ is transformed by the reverse canonicalisation g_x^{-1}. The overall output for x is then $y_x = g_x^{-1} \cdot y_{x_c}$.

- Let G be the group of rotations in the 2D-plane:

$$G = SO(2) := \{R(\theta) | \theta \in [0, 2\pi)\}, \quad \text{with } R(\theta) := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{3.5}$$

- The empirical dataset X_E is a randomly but well sampled, finite subset of all the images of two nested squares of varying sizes and colours, where the squares' edges are parallel to the image edges.
- The set X is defined as

$$X = \{R(\theta) \cdot x | x \in X_E, \theta \in [0, 2\pi)\}. \tag{3.6}$$

The overall task is, for example, finding a segmentation for the inner square. In this example, the output space $\mathcal{Y}$ can be defined as binary images, where 1 indicates the foreground, and the background is 0:

$$\mathcal{Y} = \{y \mid y : [-1, 1] \times [-1, 1] \to \{0, 1\}\} \subset \mathcal{X}. \tag{3.7}$$

For the segmentation, a model f_θ is trained on the set X_E, where the ground truth for the elements of the training set X_E is either known or created. The model is then extended to the model $\tilde{f}_\theta$ that works well on the whole X and not only on training data X_E.

In general, the extended model $\tilde{f}_\theta$ consists of three parts: a canonicalisation step, a classical neural network that is trained on the training data X_E, and a reverse canonicalisation step.

1. First, we perform a **canonicalisation step** to find a representation $x_c \in \mathcal{O}_x$ for every input $x \in \mathcal{X}$ in such a way that x_c is "close" to the training data X_E. We find a canonicalising element $g_x \in G$ by solving a task-specific chosen optimisation problem that also depends on the training data X_E. The canonicalised input is then $x_c = g_x \cdot x$.
2. At the core is a **model** $f_\theta : \mathcal{X} \to \mathcal{Y}$, $x \mapsto f_\theta(x) =: y_x \in \mathcal{Y}$ that is **trained on the set** X_E. We apply the model f_θ to the canonicalised input x_c and yield the output $y_{x_c} = f_\theta(x_c) \in \mathcal{Y}$.

3. Afterwards, the output from the central model $f_\theta(x_c) = y_{x_c}$ is transformed by the **reverse canonicalisation**. The reverse canonicalisation $g_x^{-1} \in G$ depends on how G acts on the output space $\mathcal{Y}$, the output space itself, and the performed task of f_θ. It is typically a version of the classical inverse of g_x that acts similar on $\mathcal{Y}$ to the classical inverse on $\mathcal{X}$.

Combining the three steps, the overall network $\tilde{f}_\theta : \mathcal{X} \to \mathcal{Y}$ is

$$\tilde{f}_\theta(x) := g_x^{-1} \cdot f_\theta(g_x \cdot x) \text{ with } g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x). \tag{3.8}$$

An explanatory diagram of the network can be found in Figure 3.1.

As a rule, we do not explicitly know the measure μ_T under which the training data X_E is drawn. Therefore, our setup needs more measures to approximate μ_T. Let us introduce these measures further:

3.1.2 Measures

1. The measure μ_T describes the "true" distribution on the data manifold $\mathcal{X}$. The distribution is "true" in the sense that it is the distribution from which the data is drawn independently. We would like to estimate μ_T, as the measure is crucial for bounding the generalisation error in our setup. This "true" distribution is assumed to be invariant under the group G, so that for all $g \in G$ and all measurable $A \subseteq \mathcal{X}$,

$$\mu_T(g \cdot A) = \mu_T(A) \tag{3.9}$$

   holds.
2. The finite training set X_E induces a discrete empirical measure μ_E, which is again a distribution over $\mathcal{X}$. To model this, the measure μ_E places equal weights on all training samples X_E. It is defined as

$$\mu_E(A) := \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}(A), \tag{3.10}$$

   where δ_{x_i} is the Dirac measure of x_i and $A \subseteq \mathcal{X}$ is a measurable set.
3. We also define a smooth estimation of the empirical measure μ_E called $\hat{\mu}_E$. This measure has the density $\hat{\rho}_E(x) := \frac{d\hat{\mu}_E}{d\mu_V}(x)$, where μ_V is the Riemannian volume measure (Definition 2.42) on $\mathcal{X}$. It is smooth in the sense that $\hat{\rho}_E$ is absolutely continuous with respect to μ_V. This density can be chosen freely and plays a crucial part for the generalisation. A smooth estimation of the empirical measure μ_E is necessary, as the empirical measure μ_E does not have a density.

The density $\hat{\rho}_E$ can also be used to induce the empirical orbit mass

$$m_E(\mathcal{O}_x) := \int_{\mathcal{O}_x} \hat{\rho}_E(y) d\mu_{\mathcal{O}_x}(y), \tag{3.11}$$

where $\mu_{\mathcal{O}_x}$ is the orbit measure that we assume exists, see also Theorem 2.44 and [27].

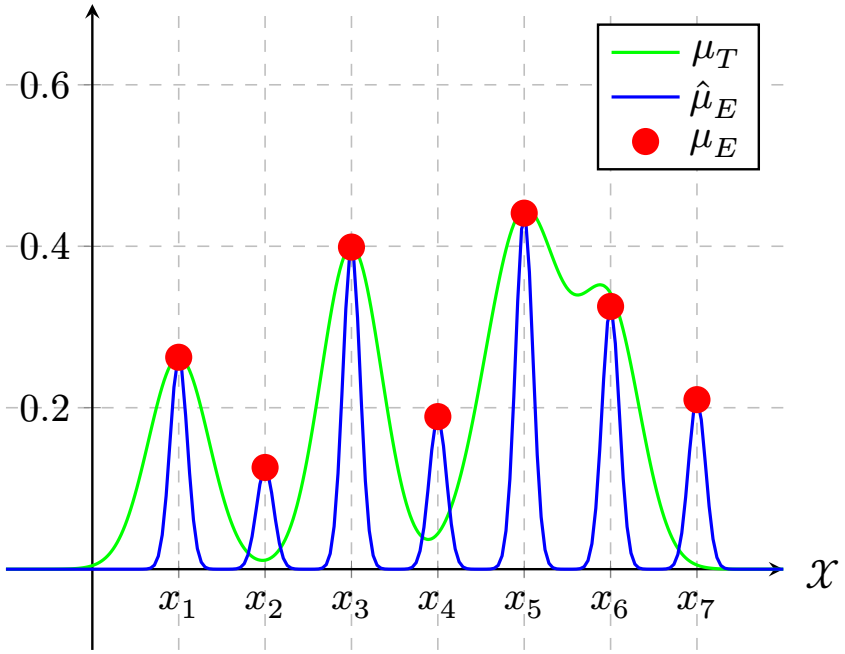


Figure 3.2: Sketch of μ_E, $\hat{\mu}_E$ and μ_T. The red points represent μ_E, which is a discrete model on the finite training data set $X_E = \{x_1, \ldots, x_7\} \subseteq \mathcal{X}$. The blue curve $\hat{\mu}_E$ is constructed by convolving μ_E with Gaussian kernels. This smooth approximation spreads the discrete mass of each sample over a local neighbourhood, producing a continuous density that can be used to estimate orbit masses or integrate over regions of the data space $\mathcal{X}$. The green curve μ_T is the true distribution of the data.

Let us propose the following definition for the smooth approximation of $\hat{\mu}_E$. We define $\hat{\mu}_E$ by combining μ_E with a Gaussian kernel:

$$\hat{\rho}_E(x) = \frac{K}{N} \sum_{i=1}^{N} \exp\left(-\frac{|x - x_i|^2}{2\sigma^2}\right), \tag{3.12}$$

where K is a normalisation factor. It follows the idea of a *Gaussian mixture*. The parameter $d = \dim(\mathcal{X})$ is the dimension of the manifold $\mathcal{X}$, and $|\cdot|$ is the geodesic distance on $\mathcal{X}$ (see Definition 2.17). In Section 3.2.2, we describe our choice of the variance σ. The above defined density $\hat{\rho}_E$ is continuous (even C^∞) and positive for all $x \in \mathcal{X}$. The measure $\hat{\mu}_E$ is then defined as

$$\hat{\mu}_E(A) := \int_A \hat{\rho}_E(x) d\lambda(x), \tag{3.13}$$

where $A \subseteq \mathcal{X}$ is any measurable subset. This applies especially to $A = \mathcal{O}_x$. The construction of $\hat{\mu}_E$ is visualised in Figure 3.2.

3.1.3 Canonicalisation

The chosen canonicalisation strategy is particularly crucial for the model's performance. In the canonicalisation step, a representation $x_c \in \mathcal{O}_x$ for every input $x \in \mathcal{X}$ is found in such a way that x_c is "close" to the training data X_E. The canonicalised input x_c is constructed by applying a canonicalising element $g_x \in G$ to the input x:

$$x_c = g_x \cdot x. \tag{3.14}$$

In practice, a canonicalising element g_x is found by minimising an energy E_{can} that depends on the task and the training data X_E:

$$g_x \in \arg\min_{g \in G} E_{\text{can}}(g \cdot x). \tag{3.15}$$

One way to choose the canonicalisation energy is through the density $\hat{\rho}_E$: For all $x \in \mathcal{X}$ define

$$E_{\text{can}}(x) := -\log \hat{\rho}_E(x) \tag{3.16}$$

up to a scaling factor. This relation between the energy and density allows us to select g_x by maximising the following problem:

$$g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x). \tag{3.17}$$

The maximisers of $\hat{\rho}_E$ are the minimisers of E_{can}. This connection between the density-based formulation and the energy-based implementation allows us to realise the canonicalisation step in praxis by minimising over the energy E_{can}, but make theoretical guarantees based on the density-based canonicalisation. This energy $E_{\text{can}}(g \cdot x)$ or density $\hat{\rho}_E(g \cdot x)$ must not be strictly convex or concave, and can be very similar for several $g \in G$, therefore the canonicalising element g_x is not necessarily unique. Intuitively, the non-uniqueness is not problematic because as long as the energy $E_{\text{can}}(g_x \cdot x)$ is low enough or the density $\hat{\rho}_E(g \cdot x)$ is high enough, the canonicalised input $x_c = g_x \cdot x$ is close enough to the training data X_E.

Note that the canonicalisation is G-invariant, in the sense that for all $x \in \mathcal{X}$ and any transformation $g' \in G$,

$$\arg\max_{g \in G} \hat{\rho}_E(g \cdot x) = \arg\max_{g \in G} \hat{\rho}_E(g \cdot (g' \cdot x)) \tag{3.18}$$

holds. This means that for $x \in \mathcal{X}$ and an arbitrary $\tilde{x} = g' \cdot x$:

$$E_{\text{can}}(x_c) = E_{\text{can}}(\tilde{x}_c) \tag{3.19}$$

and

$$\hat{\rho}_E(x_c) = \hat{\rho}_E(\tilde{x}_c). \tag{3.20}$$

Lemma 3.2 (Existence of the Canonicalising Element). *Let G be a compact Lie group that acts smoothly on the compact manifold $\mathcal{X}$. Let the density $\hat{\rho}_E$ be continuous. For each $x \in \mathcal{X}$, assume also that $\hat{\rho}_E(\circ \cdot x) : G \to \mathbb{R}$ is continuous. Then*

$$\arg\max_{g \in G} \hat{\rho}_E(g \cdot x) \tag{3.21}$$

is non-empty. Henceforth, there exists at least one canonicalising element $g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$ for all $x \in \mathcal{X}$.

Proof. Let $x \in \mathcal{X}$ be arbitrary. $\hat{\rho}_E(\circ \cdot x) : G \to \mathbb{R}$ is continuous and G is compact. The generalisation of the extreme value theorem (Th. 2.7) says that continuous functions assume their maximum on compact topological space. A compact Lie group is a compact topological space by definition. Therefore, for each $x \in \mathcal{X}$, the maximum in (3.17) is assumed and the set of maximisers $\arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$ is non-empty. □

After introducing the setup and the necessary measures, let us look back at the overall network

$$\tilde{f}_\theta(x) = g_x^{-1} \cdot f_\theta(g_x \cdot x) \quad \text{with } g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x). \tag{3.22}$$

The goal of this section is to bound

$$\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt})], \tag{3.23}$$

where $y_x^{gt} \in \mathcal{Y}$ is the ground-truth output of $x \in \mathcal{X}$, g_x the canonicalising element, and L is the loss function for the inner network f_θ, introduced in Section 3.1.

3.1.4 Well-Sampled Orbits

Since we only observe a finite training set X_E, we cannot guarantee that the empirical distribution μ_E covers the entire space $\mathcal{X}$. In particular, some orbits may be represented by many training samples, while others are covered only sparsely or not at all. To formalise this distinction, we introduce the notion of *well-sampled* and *poorly-sampled* orbits. We define the set of elements with well-sampled orbits as

$$A_\delta := \{x \in \mathcal{X} \mid m_E(\mathcal{O}_x) > \delta\}, \tag{3.24}$$

and the complementary set of elements with poorly-sampled orbits as

$$A_\delta^c := \mathcal{X} \setminus A_\delta = \{x \in \mathcal{X} \mid m_E(\mathcal{O}_x) \leq \delta\}. \tag{3.25}$$

We use $m_E(\mathcal{O}_x)$, the mass of the smoothed density in the orbit of x, as a measurement of the orbit coverage. The parameter $\delta > 0$ serves as a threshold that distinguishes between sufficient and insufficient orbit coverage. Its choice is discussed in Section 3.2.2. Equivalently, A_δ can be written as a union of orbits whose empirical orbit mass is larger than δ:

$$A_\delta = \{x \in \mathcal{X} \mid m_E(\mathcal{O}_x) > \delta\} \tag{3.26}$$

$$= \bigcup_{\mathcal{O}_x \in \mathcal{X}/G,\, m_E(\mathcal{O}_x) > \delta} \mathcal{O}_x. \tag{3.27}$$

We therefore refer to A_δ as the set of *well-sampled orbits*, and to A_δ^c as the set of *poorly-sampled orbits*.

Lemma 3.3. *The set of well-sampled orbits A_δ and the set of poorly-sampled orbits A_δ^c are Borel measurable subsets of $\mathcal{X}$, i.e., $A_\delta, A_\delta^c \in \mathcal{B}(\mathcal{X})$.*

Proof. The projection map π and the empirical orbit mass m_E are measurable. Therefore, their composition $m_E \circ \pi$ is measurable as well. Furthermore, the set $\{x > \delta\} \subseteq \mathbb{R}$ is Borel measurable, i.e., $\{x > \delta\} \in \mathcal{B}(\mathbb{R})$. The preimage of measurable function (Definition 2.36) of a measurable set is also measurable, therefore

$$A_\delta = (m_E \circ \pi)^{-1}(\{x > \delta\}) \tag{3.28}$$

is also Borel measurable. □

The property of Borel measurability of the set of well-sampled and poorly-sampled orbits is crucial to ensure that we can integrate over those sets and that the measures on those sets are defined.

3.2 Bounding the Expected Generalisation Loss

Having established the learning setting in Section 3.1, we now turn to the central question of this chapter: Under what conditions does energy-based canonicalisation generalise from finite training data to the set $\mathcal{X}$.

3.2.1 Assumptions

To bound the generalisation loss, a few assumptions need to be made: As mentioned in the introduction of the true distribution μ_T, one assumption is that μ_T is invariant under the group G.

Assumption 3.4 (G-Invariance of μ_T). Let $A \subseteq \mathcal{X}$ be measurable. Then, for all $g \in G$, we assume that

$$\mu_T(g \cdot A) = \mu_T(A) \tag{3.29}$$

holds.

Assumption 3.5 (G-Equivariance of Ground Truth). We assume that the ground-truth mapping $x \mapsto y_x^{gt}$ is G-equivariant, i.e., that for all $g \in G$ and $x \in \mathcal{X}$

$$y_{g \cdot x}^{gt} = g \cdot y_x^{gt} \tag{3.30}$$

holds.

This assumption is very important, as it sets the ground for our approach and describes the way we want our overall network $\tilde{f}_\theta$ to behave.

Assumption 3.6 (Improbable Poorly-Sampled Orbits). There exists an $\alpha \in [0, 1)$, $\alpha \ll 1$, such that the probability with respect to the "true" distribution of the poorly sampled orbits is bounded by α, i.e.,

$$\mu_T(A_\delta^c) \leq \alpha \ll 1. \tag{3.31}$$

Intuitively, this means that we sampled well enough such that we covered nearly all the orbits and we only did not look at very few, improbable orbits. Furthermore, assume that the orbit volume is bounded:

Assumption 3.7 (Bounded Orbit Volume). For all $x \in \mathcal{X}$, the volume of the orbit $\mathcal{O}_x$ is bounded: There exists an $V_{\max} \in \mathbb{R}$ such that, for all $x \in \mathcal{X}$,

$$0 < \mathrm{vol}_{\mathcal{O}}(\mathcal{O}_x) := \mu_{\mathcal{O}_x}(\mathcal{O}_x) := \int_{\mathcal{O}_x} d\mu_{\mathcal{O}_x}(y) \leq V_{\max} \tag{3.32}$$

holds.

For the loss function L assume the following:

Assumption 3.8 (Bounded Loss Function). The loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is bounded. There exists an $L_{\max} \in \mathbb{R}$, such that, for all $y, y' \in \mathcal{Y}$,

$$0 \leq L(y, y') \leq L_{\max} \tag{3.33}$$

holds.

Assumption 3.9 (Lipschitz Continuity). Assume f_θ is Lipschitz continuous with the Lipschitzconstant C_f and L is Lipschitz continuous in its first argument with the Lipschitzconstant C_L. Therefore, $L(\cdot, y)$ is C_L-Lipschitz for any fixed $y \in \mathcal{Y}$ (independent of y). Then $x \mapsto L(f_\theta(x), y)$ (with $x \in \mathcal{X}$) is Lipschitz with Lipschitzconstant $C_{\mathrm{Lip}} := C_L \cdot C_f$.

We further assume that the model f_θ is well-trained on the training data X_E, achieving low loss on all samples:

Assumption 3.10 (Bounded Training Error). For the model f_θ, trained on X_E, exists a small $\epsilon > 0$, such that

$$\sup_{x \in X_E} L(f_\theta(x), y_x^{gt}) \leq \tfrac{\epsilon}{2}, \tag{3.34}$$

holds, where $y_x^{gt} \in \mathcal{Y}$ is the ground truth for $x \in X_E$.

3.2.2 Parameter Selection

In this section and the preceding section, several thresholds and parameters were introduced. The only one that can be properly controlled and distinguished in practice is the bound for the training error $\frac{\epsilon}{2}$. In the following, we assume that we have a fixed $\epsilon > 0$ (following Ass. 3.10). Furthermore, assume the Lipschitzconstant C_{Lip} is known. The other parameters are chosen depending on ϵ and C_{Lip}.

First, we choose the variance σ of $\hat{\rho}_E$ as

$$\sigma := \sigma(\epsilon, C_{\mathrm{Lip}}) := \frac{\epsilon}{2C_{\mathrm{Lip}}}. \tag{3.35}$$

This ensures that for any $x \in \mathcal{X}$ within distance σ of a training data point $x_i \in X_E$, the loss is bounded by ϵ. Because, under Assumption 3.9, for all $x \in \mathcal{X}$ (with corresponding ground truth $y_x \in \mathcal{Y}$) with an $x_i \in X_E$ such that $|x - x_i| \leq \sigma$, holds

$$L(f_\theta(x), y_x) \overset{\text{Ass. 3.9}}{\leq} L(f_\theta(x_i), y_x) + C_{\text{Lip}}\sigma \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon, \tag{3.36}$$

where L is the loss from (3.33) and (3.34). Using this σ, we define a region, where a small loss can be ensured: The set, where the loss is bounded by ϵ is

$$S_\sigma := \bigcup_{i=1}^{N} B_\sigma(x_i) \tag{3.37}$$

with

$$B_\sigma(x_i) := \{x \in \mathcal{X} \mid \|x - x_i\| \leq \sigma\}, \tag{3.38}$$

which denotes a ball around x_i with radius σ. The parameter σ in combination with the maximal orbit volume V_{max} and the normalisation factor of $\hat{\rho}_E$ can be used to define the threshold that distinguishes between well-sampled and poorly-sampled orbits:

Definition 3.11 (Well-Sampled and Poorly-Sampled Orbits). Set the threshold $\delta > 0$ as

$$\delta := \delta(\sigma, V_{\text{max}}) := \delta(\epsilon, C_{\text{Lip}}, K, V_{\text{max}}) := KV_{\text{max}}\mathrm{e}^{-\frac{1}{2}} \tag{3.39}$$

and define $\hat{\rho}_E$ as in the preceding section.

This threshold ensures that orbits with $m_E(\mathcal{O}_x) > \delta$ have sufficient empirical mass with high probability. Furthermore, the following Lemma about the relation between the well-sampled orbits A_δ and set with bounded loss S_σ holds:

Lemma 3.12. *Set $\delta = KV_{max}\mathrm{e}^{-\frac{1}{2}}$ and $\sigma = \frac{\epsilon}{2C_{Lip}}$. For any $x \in A_\delta$, a canonicalised representation of x, $x_c = g_x \cdot x$ with $g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$, lies in S_σ.*

The main idea of this lemma is visualised in Figure 3.3.

Proof. Assume $x \in A_\delta$, i.e., $m_E(\mathcal{O}_x) > \delta$, where $m_E(\mathcal{O}_x)$ is defined as

$$m_E(\mathcal{O}_x) = \int_{\mathcal{O}_x} \hat{\rho}_E(y) d\mu_{\mathcal{O}_x}(y) \tag{3.40}$$

$$= \int_{\mathcal{O}_x} \frac{K}{N} \sum_{i=1}^{N} \exp\left(-\frac{|y - x_i|^2}{2\sigma^2}\right) d\mu_{\mathcal{O}_x}(y). \tag{3.41}$$

We claim that there exists a $x' \in \mathcal{O}_x$ with

$$\hat{\rho}_E(x') = \frac{K}{N} \sum_{i=1}^{N} \exp\left(-\frac{|x' - x_i|^2}{2\sigma^2}\right) > \frac{\delta}{\text{vol}_{\mathcal{O}}(\mathcal{O}_x)}. \tag{3.42}$$

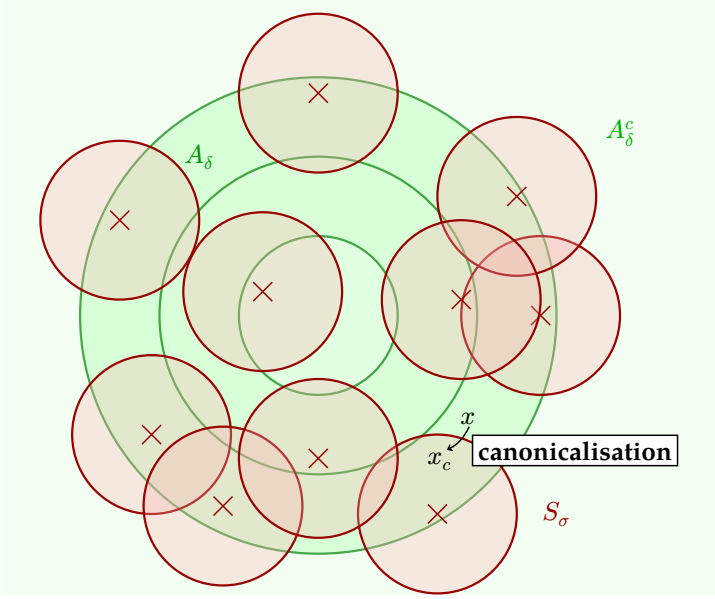


Figure 3.3: Intuition of Lemma 3.12. The set of well-sampled orbits A_δ is shown in green; the set of poorly-sampled orbits A_δ^c in light green. The region S_σ around X_E, where we know that the model f_θ performs well, is shown in red. The red crosses indicate the locations of the samples in the training dataset X_E. Lemma 3.12 states that any input $x \in A_\delta$ is canonicalised to a x_c that lies in S_σ.

Indeed if not, then $\hat{\rho}_E(y) \leq \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)}$ for all $y \in \mathcal{O}_x$, thus

$$m_E(\mathcal{O}_x) = \int_{\mathcal{O}_x} \underbrace{\hat{\rho}_E(y)}_{\leq \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)}} d\mu_{\mathcal{O}_x}(y) \tag{3.43}$$

$$\leq \int_{\mathcal{O}_x} \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)} d\mu_{\mathcal{O}_x}(y) \tag{3.44}$$

$$= \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)} \int_{\mathcal{O}_x} d\mu_{\mathcal{O}_x}(y) \tag{3.45}$$

$$= \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)} \text{vol}_\mathcal{O}(\mathcal{O}_x) = \delta. \tag{3.46}$$

This would contradict $m_E(\mathcal{O}_x) > \delta$, which ensures that there must be at least one $x' \in \mathcal{O}_x$ with $\hat{\rho}_E(x') > \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)}$. Consequently, by a similar argument applied to (3.42), there must also exist some $x_i \in X_E$, such that

$$K \exp\left(-\frac{|x' - x_i|^2}{2\sigma^2}\right) > \frac{\delta}{\text{vol}_\mathcal{O}(\mathcal{O}_x)} \tag{3.47}$$

holds. Including the bound from Assumption 3.7 and dividing by normalisation factor K, the following inequality holds:

$$\exp\left(-\frac{|x' - x_i|^2}{2\sigma^2}\right) > \frac{\delta}{K\text{vol}_\mathcal{O}(\mathcal{O}_x)} \geq \frac{\delta}{KV_{\max}}. \tag{3.48}$$

And by the definition of δ

$$\frac{\delta}{KV_{\max}} = \mathrm{e}^{-\frac{1}{2}}, \tag{3.49}$$

so

$$\exp\left(-\frac{|x' - x_i|^2}{2\sigma^2}\right) > \mathrm{e}^{-\frac{1}{2}}. \tag{3.50}$$

Reducing this further, we get

$$-\frac{|x' - x_i|^2}{2\sigma^2} > -\tfrac{1}{2} \tag{3.51}$$

$$\Leftrightarrow \frac{|x' - x_i|^2}{\sigma^2} < 1. \tag{3.52}$$

Thus, if $m_E(\mathcal{O}_x) > \delta$, there exists $x' \in \mathcal{O}_x$ and $x_i \in X_E$, such that $|x' - x_i| \leq \sigma$. We will now show that $x_c = g_x \cdot x$ with $g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$ lies in S_σ by contradiction. We know that g_x maximises the density, thus:

$$\hat{\rho}_E(x_c) \geq \hat{\rho}_E(x') > \frac{\delta}{KV_{max}}. \tag{3.53}$$

Assume x_c does not lie in S_σ, then $|x_c - x_i| \geq \sigma$ for all $x_i \in X_E$. Then

$$\hat{\rho}_E(x_c) = \frac{K}{N}\sum_{i=1}^{N} \underbrace{\exp\left(-\frac{|x_x - x_i|^2}{2\sigma^2}\right)}_{\leq \exp(-\frac{1}{2})} \tag{3.54}$$

and so

$$\hat{\rho}_E(x_c) \leq \frac{K}{N}\sum_{i=1}^{N} \mathrm{e}^{-\frac{1}{2}} = K\mathrm{e}^{-\frac{1}{2}} = \frac{\delta}{V_{max}}. \tag{3.55}$$

This contradicts (3.53) and it follows that $x_c \in S_\sigma$. □

The previous Lemma 3.12 ensures that the canonicalisation step acts as intended: For all x lying on well-sampled orbits, the canonicalised version $x_c = g_x \cdot x$ is guaranteed to be sufficiently "close" to the training data X_E. This means that, after our canonicalisation step, the input is effectively transformed to a region, where the model f_θ is known to perform well, which is the main mechanism that our approach is based on, and it is crucial for generalisation in our setting.

3.2.3 Main Theorem: Generalisation Bound

Theorem 3.13. *Let the setup be as described in Section 3.1. Assume the following:*

1. *G-invariance of μ_T (Assumption 3.4),*
2. *G-equivariance of the ground truth (Assumption 3.5),*

3. *The poorly-sampled orbits are improbable under the true distribution* (*Assumption 3.6*),
4. *Bounded orbit volume* (*Assumption 3.7*),
5. *Bounded loss function* (*Assumption 3.8*),
6. *Lipschitz continuity of* $L(f_\theta(x), y)$ (*Assumption 3.9*) *and*
7. *Bounded training error* (*Assumption 3.10*).

We define the density explicitly as in (3.12). *Furthermore, we set the parameters*

$$\sigma := \frac{\epsilon}{2C_{Lip}} \tag{3.56}$$

with C_{Lip} *from Assumption 3.9 and* ϵ *from Assumption 3.10 and*

$$\delta := KV_{max}\mathrm{e}^{-\frac{1}{2}}. \tag{3.57}$$

Then the expected generalisation loss is bounded by:

$$\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt})] < \epsilon + (L_{\max} - \epsilon)\,\alpha, \tag{3.58}$$

where $y_x^{gt} \in \mathcal{Y}$ *denotes the ground truth corresponding to* $x \in \mathcal{X}$ *and*

$$g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x) = \arg\max_{y \in \mathcal{O}_x} \hat{\rho}_E(y) \tag{3.59}$$

is a canonicalising element.

Proof. By definition of the expectation:

$$\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt})] = \int_{\mathcal{X}} L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt}) d\mu_T(x) \tag{3.60}$$

With Assumption 3.5, it follows:

$$\int_{\mathcal{X}} L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt}) d\mu_T(x) = \int_{\mathcal{X}} L(f_\theta(g_x \cdot x), y_{g_x \cdot x}^{gt}) d\mu_T(x) \tag{3.61}$$

Splitting the expectation over well-sampled and poorly-sampled orbits (defined as in Definition 3.11)

$$\begin{aligned}\int_{\mathcal{X}} L(f_\theta(g_x \cdot x), y_{g_x \cdot x}^{gt}) d\mu_T(x) = &\int_{A_\delta} L(f_\theta(g_x \cdot x), y_{g_x \cdot x}^{gt}) d\mu_T(x) \\ &+ \int_{A_\delta^c} L(f_\theta(g_x \cdot x), y_{g_x \cdot x}^{gt}) d\mu_T(x).\end{aligned} \tag{3.62}$$

Those integrals are well-defined as μ_T is a probability measure, the loss function L has an upper and lower bound (Assumption 3.8), and A_δ and A_δ^c are Borel measurable sets (Lemma 3.3), which makes them integrable sets in this setup.

Well-Sampled Orbits. Let us first look at the integral over the well-sampled orbits: Let $x \in A_\delta$ with $\delta = KV_{\max}\mathrm{e}^{-\frac{1}{2}}$. Using Lemma 3.12, we know have $x_c = g_x \cdot x \in S_\sigma$ with $g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$ and so there exists some $x_i \in X_E$ such that $|x_c - x_i| \leq \sigma$ and so

$$L(f_\theta(g_x \cdot x), y_{g_x \cdot x}^{gt}) \leq \epsilon. \tag{3.63}$$

Therefore,

$$\int_{A_\delta} L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x}) d\mu_T(x) \leq \epsilon \int_{A_\delta} d\mu_T(x) = \epsilon\, \mu_T(A_\delta). \tag{3.64}$$

Poorly-Sampled Orbits. On the poorly-sampled orbits, there is no information about the loss function, therefore the upper bound $L_{\max}$ (see Assumption 3.8) has to be used:

$$\int_{A^c_\delta} L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x}) d\mu_T(x) \leq L_{\max} \int_{A^c_\delta} d\mu_T(x) = L_{\max} \mu_T(A^c_\delta), \tag{3.65}$$

with $g_x \in \arg\max_{g \in G} \hat{\rho}_E(g \cdot x)$.

Combining the results on the well- and poorly-sampled orbits (3.64), (3.65) the full expectation is bounded by

$$\begin{aligned}
\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y^{gt}_x)] &\overset{(3.61)}{=} \mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x})] && (3.66)\\
&= \int_{\mathcal{X}} L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x}) d\mu_T(x) && (3.67)\\
&\overset{(3.62)}{=} \int_{A_\delta} L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x}) d\mu_T(x) && (3.68)\\
&\quad + \int_{A^c_\delta} L(f_\theta(g_x \cdot x), y^{gt}_{g_x \cdot x}) d\mu_T(x)\\
&\overset{(3.64),(3.65)}{\leq} \epsilon\, \mu_T(A_\delta) + L_{\max} \mu_T(A^c_\delta). && (3.69)
\end{aligned}$$

Using $\mu_T(A_\delta) = 1 - \mu_T(A^c_\delta)$ we get

$$\begin{aligned}
\epsilon\, \mu_T(A_\delta) + L_{\max} \mu_T(A^c_\delta) &= \epsilon\, (1 - \mu_T(A^c_\delta)) + L_{\max} \mu_T(A^c_\delta) && (3.70)\\
&= \epsilon + (L_{\max} - \epsilon) \mu_T(A^c_\delta). && (3.71)
\end{aligned}$$

Assumption 3.6 gives us

$$\mu_T(A^c_\delta) < \alpha, \tag{3.72}$$

henceforth

$$\epsilon + (L_{\max} - \epsilon) \mu_T(A^c_\delta) < \epsilon + (L_{\max} - \epsilon)\, \alpha, \tag{3.73}$$

and so

$$\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y^{gt}_x)] < \epsilon + (L_{\max} - \epsilon)\, \alpha. \tag{3.74}$$

□

This result is the central theoretical contribution of this chapter. We showed that the expected loss after the canonicalisation step and applying the inner network is bounded under mild assumptions. To our knowledge, this is the first generalisation bound derived for canonicalisation. This provides a theoretical foundation for why such models work in practice.

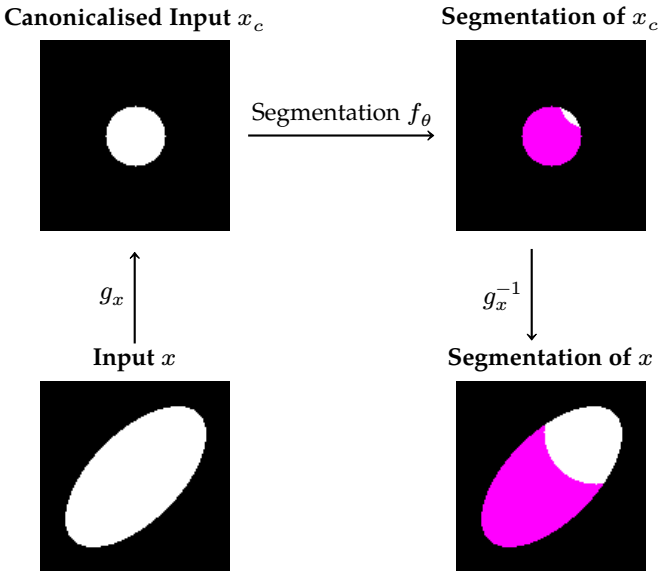


Figure 3.4: Example illustrating the difficulties of bounding the error after applying the reverse canonicalisation g_x^{-1}. The goal is to segment an ellipse. For this, the input image x is canonicalised to $x_c = g_x \cdot x$, where the ellipse looks now like a circle. This canonicalised input x_c is then segmented with f_θ. The segmentation output $f_\theta(x_c)$ is shown as purple. There is a relatively small error on the upper right of the circle. When this segmentation output is then transformed through the reverse canonicalisation to determine a segmentation for the input x, this small error can be greatly amplified as the circle is deformed to the ellipsoid shape.

3.2.4 Expected Loss after Reverse Canonicalisation

We successfully bounded

$$\mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt})], \tag{3.75}$$

in the last section. But, the expectation of the overall loss of the whole network $\tilde{f}_\theta$ is

$$\mathbb{E}_{\mu_T}[L(g_x^{-1} \cdot f_\theta(g_x \cdot x), y_x^{gt})]. \tag{3.76}$$

The latter term is more difficult to bound, as applying the reverse canonicalisation g_x^{-1} to the prediction can greatly amplify errors. A visual example is given in Figure 3.4.

However, there are a few conditions under which we can deduce a bound for (3.76) through a bound for (3.75). In addition to the assumptions in Section 3.2.1, we assume that the loss function L is bounded by the geodesic distance, and the canonicalising element g_x is Lipschitz continuous:

Assumption 3.14. Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the loss function defined above, and the norm $\|\cdot\|$ is the geodesic distance (Definition 2.17) on $\mathcal{Y}$. We assume the following:

- For the loss $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, assume that there exist two constants $C_{L_1}, C_{L_2} \in \mathbb{R}_{>0}$, such that for all $y_1, y_2 \in \mathcal{Y}$,

$$C_{L_1} \cdot \|y_1 - y_2\| \leq L(y_1, y_2) \tag{3.77}$$

and

$$L(y_1, y_2) \leq C_{L_2} \cdot \|y_1 - y_2\| \tag{3.78}$$

hold.

- Furthermore, assume that the canonicalising element g_x and the reverse canonicalisation g_x^{-1} on $\mathcal{Y}$ are jointly C_g-Lipschitz, i.e., there exists $C_g > 0$ such that

$$\|g_x \cdot y_1 - g_x \cdot y_2\| \leq C_g \cdot \|y_1 - y_2\|, \tag{3.79}$$
$$\|g_x^{-1} \cdot y_1 - g_x^{-1} \cdot y_2\| \leq C_g \cdot \|y_1 - y_2\|. \tag{3.80}$$

Following the assumptions, we can now use these properties of L and g_x^{-1} to bound the expected generalisation loss after applying the reverse canonicalisation g_x^{-1}. First, we use that $y_x^{gt} = g_x^{-1} \cdot (g_x \cdot y_x^{gt})$ and get

$$L(g_x^{-1} \cdot f_\theta(g_x \cdot x), y_x^{gt}) = L(g_x^{-1} \cdot f_\theta(g_x \cdot x), g_x^{-1} \cdot (g_x \cdot y_x^{gt})). \tag{3.81}$$

We can then apply the upper bound for our loss function L:

$$L(g_x^{-1} \cdot f_\theta(g_x \cdot x), g_x^{-1} \cdot (g_x \cdot y_x^{gt})) \overset{(3.78)}{\leq} C_{L_2} \cdot \|g_x^{-1} \cdot f_\theta(g_x \cdot x) - g_x^{-1} \cdot (g_x \cdot y_x^{gt})\|. \tag{3.82}$$

Using the Lipschitz continuity of g_x^{-1}, it follows that

$$C_{L_2} \cdot \|g_x^{-1} \cdot f_\theta(g_x \cdot x) - g_x^{-1} \cdot (g_x \cdot y_x^{gt})\| \overset{(3.80)}{\leq} C_{L_2} \cdot C_g \cdot \|f_\theta(g_x \cdot x) - g_x \cdot y_x^{gt}\|. \tag{3.83}$$

When applying the lower bound of L, we furthermore derive

$$\|f_\theta(g_x \cdot x) - g_x \cdot y_x^{gt}\| \overset{(3.77)}{\leq} \frac{1}{C_{L_1}} \cdot L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt}). \tag{3.84}$$

Therefore, there exists a constant $C := \frac{C_{L_2}}{C_{L_1}} \cdot C_g$ such that

$$L(g_x^{-1} \cdot f_\theta(g_x \cdot x), y_x^{gt}) \leq C \cdot L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt}). \tag{3.85}$$

Using this for the expectations, we get

$$\mathbb{E}_{\mu_T}[L(g_x^{-1} \cdot f_\theta(g_x \cdot x), y_x^{gt})] \leq C \cdot \mathbb{E}_{\mu_T}[L(f_\theta(g_x \cdot x), g_x \cdot y_x^{gt})]. \tag{3.86}$$

Under specific conditions, we can therefore also bound the expected loss after the reverse canonicalisation.

In this chapter, we have defined our theoretical learning setup in detail and then provided an upper bound for the expected loss after canonicalisation and inner segmentation under mild assumptions. This guarantees generalisation but is restricted to settings following the assumptions and general setup. We furthermore investigated under what conditions the derived bound can also provide a bound for the expected loss after applying the reverse canonicalisation. All our theoretical statements rely on a finite-dimensional setup.

4

Diffeomorphism-Equivariant Neural Network

In Chapter 3, we proved that a canonicalisation approach inspired by LieLAC [72] generalises under mild assumptions. In addition, this canonicalisation strategy ensures approximate group-equivariance by construction and requires only a small training dataset. However, LieLAC has only been applied to compact manifolds under the action of finite- or low-dimensional Lie groups. In contrast, an important group of transformations in the field of image computing is the infinite-dimensional group of diffeomorphisms.

In this chapter, we therefore propose an approximately diffeomorphism-equivariant neural network. We introduce DiffeoNN, a method that adapts the LieLAC framework to the infinite-dimensional group of diffeomorphisms acting on image domains. While we focus on image segmentation as an example, the framework naturally extends to other tasks such as object detection. The proposed method falls outside the scope of the theoretical guarantees from Chapter 3, as the setup is not compact and finite-dimensional. Nevertheless, the framework is still similar to the LieLAC framework:

1. We **canonicalise** an input by minimising a smooth energy function over a group orbit,
2. we perform a **segmentation on the canonicalised input**,
3. we finally map the segmentation result back by using the **inverse canonicalising element**.

The overall setup of DiffeoNN is illustrated in Figure 4.1. With DiffeoNN, we adapt the LieLAC framework to the diffeomorphic setting as follows.

1. We focus on the group of diffeomorphisms instead of an arbitrary Lie group.
2. We define a canonicalisation energy that combines learned density modelling (VAE), adversarial discriminators, and deformation regularisation following [19].
3. Instead of using a Lie algebra optimisation strategy, we use a gradient-based optimisation strategy [19].
4. We apply our method to a segmentation task based on the LieLAC approach, which has not been done by Shumaylov et al. [72] or anyone else to our knowledge.

In the following, we describe DiffeoNN in detail. We split the chapter along the steps of the framework: First, we explain the problem setup in Section 4.1. After that, in Section 4.2, the canonicalisation strategy is introduced. This section includes the choice of energy and its training method, the parametrisation of the diffeomorphisms by SVFs,

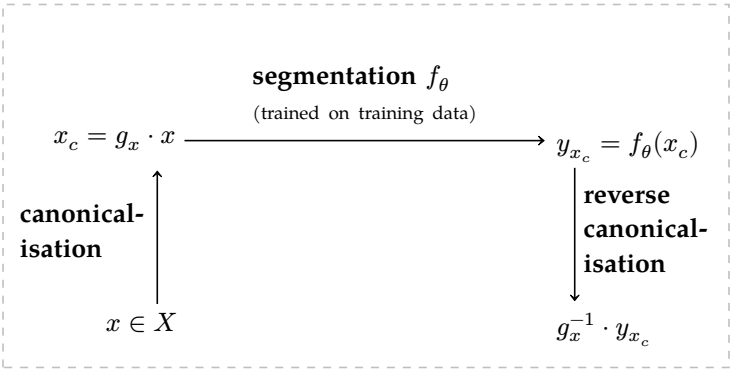


Figure 4.1: Setup of DiffeoNN. An input image $x \in X$ is canonicalised to $x_c = g_x \cdot x$. Then, a segmentation model f_θ is applied to x_c, resulting in the segmentation $y_{x_c} = f_\theta(x_c)$. Afterwards, the segmentation y_{x_c} is transformed by the inverse of the canonicalisation g_x^{-1}. The transformed segmentation $y_x = g_x^{-1} \cdot y_{x_c}$ is a segmentation of the input image $x \in X$.

and the optimisation strategy, which we use to find a canonicalising element g_x. The segmentation model used is further explained in Section 4.3. In Section 4.4, we describe the reverse canonicalisation step. In the last section (Section 4.5), we connect our method to the theoretical investigations of the previous chapter and give a brief summary.

4.1 Problem Setup

Similar to Example 3.1, let manifold $\mathcal{X}$ denote the space of greyscale images defined on the image domain $\Omega \subseteq \mathbb{R}^2$ with pixel intensities in $[0, 1]$, modelled as functions that map from Ω to $[0, 1]$,

$$\mathcal{X} = \{x \mid x : \Omega \to [0, 1]\}. \tag{4.1}$$

We choose the group G, for which we seek the equivariance, as the set of SVF-based diffeomorphisms $\mathcal{D}_{\text{SVF}}(\Omega)$, which act on the image domain Ω. A more detailed definition of SVF-based diffeomorphisms $\mathcal{D}_{\text{SVF}}(\Omega)$ is provided in Section 4.2.1. We define the group action of G on $\mathcal{X}$ as

$$(g \cdot x)(p) := (x \circ g)(p) = x(g(p)) \tag{4.2}$$

for all $p \in \Omega$, $x \in \mathcal{X}$ and $g \in \mathcal{D}_{\text{SVF}}(\Omega)$.

The goal of our proposed method DiffeoNN is to yield a segmentation map for an input $x \in \mathcal{X}$. Image segmentation is a task in computer vision that is performed pixel-wise, meaning that a class label is assigned to every point in an image. This work focuses on binary segmentation, i.e., two classes: foreground/background.

Definition 4.1 (Binary Segmentation). Binary segmentation is the process of creating a binary "image" $y_x : \Omega \to \{0, 1\}$ for a given input image $x \in \mathcal{X}$. We say a point $p \in \Omega$ is:

- segmented as part of the foreground if $y_x(p) = 1$.
- segmented as part of the background if $y_x(p) = 0$.

For our method, we require a finite training dataset of images X_E consisting of $N \in \mathbb{N}$ samples:

$$X_E := X_E^N := \{x_i \mid x_i : \Omega \to \mathbb{R}, i = 1, ..., N\} \subseteq \mathcal{X}. \tag{4.3}$$

Let the dataset X be the set of all images that can be generated by transforming an element of X_E with a diffeomorphism in $\mathcal{D}_{\text{SVF}}(\Omega)$:

$$X := \{g \cdot x \mid x \in X_E, g \in \mathcal{D}_{\text{SVF}}(\Omega)\} \subseteq \mathcal{X}. \tag{4.4}$$

The idea is to find a segmentation for an element $x \in X$ without training a segmentation on the whole dataset (or a finite subset of X). Instead, the whole network is only trained on the dataset X_E. This setting also allows us to use a pretrained model and extend it to be diffeomorphism-equivariant, as the setup is identical and f_θ can just be replaced by the pretrained model. We define the output space $\mathcal{Y}$ as the set of binary images, where 1 represents the foreground and the background is represented by 0:

$$\mathcal{Y} = \{y \mid y : \Omega \to \{0, 1\}\} \subseteq \mathcal{X}. \tag{4.5}$$

With this problem setup, we can now focus on the three steps of our method. We start with the first step, the canonicalisation.

4.2 Canonicalisation

The canonicalisation approach that we use is very similar to the one introduced by Shumaylov et al. [72]. Given the image space X and a transformation group G, the goal is to find a canonicalising element $g_x \in \mathcal{D}_{\text{SVF}}(\Omega)$ for each $x \in X$ such that $x_c = g_x \cdot x$ is a good representation of x and "close" to the training set X_E, on which the network f_θ was trained. How "close" x_c is to the training dataset is modelled by an energy $E_{X_E} : \mathcal{X} \to \mathbb{R}$. Furthermore, we want to prefer physically plausible diffeomorphisms, e.g., orientation-preserving ones, as the canonicalising element. We therefore combine the image similarity energy E_{X_E} with a regularisation energy $E_{\text{reg}} : \mathcal{D}_{\text{SVF}}(\Omega) \to \mathbb{R}$ into a canonicalisation energy:

$$E_{\text{can}} : \mathcal{X} \times \mathcal{D}_{\text{SVF}}(\Omega) \to \mathbb{R}, \tag{4.6}$$

$$E_{\text{can}}(x, g) := E_{X_E}(g \cdot x) + E_{\text{reg}}(g). \tag{4.7}$$

Intuitively, the energy E_{can} should satisfy the following properties:

- $E_{\text{can}}(x, g)$ is small, whenever $g \cdot x$ is "close" to the training data X_E and the transformation g is physically plausible.
- $E_{\text{can}}(x, g)$ is large, when $g \cdot x$ is very different from the training data or the transformation g is not physically plausible.

In Section 4.2.2, we propose a specific choice for E_{can}. An "ideal" canonicalising element g_x for an image $x \in X$ is then defined as a minimiser:

$$g_x \in \operatorname*{arg\,min}_{g \in \mathcal{D}_{\text{SVF}}(\Omega)} E_{\text{can}}(x, g). \tag{4.8}$$

For solving (4.8), we use a modified version of the algorithm introduced in [19], which is described in detail in Section 4.2.3.

4.2.1 SVF-based Diffeomorphisms

As a parametrisation of the group of diffeomorphisms $G = \mathcal{D}(\Omega)$ we consider the set of SVF-based diffeomorphisms $\mathcal{D}_{\text{SVF}}(\Omega)$ that is a structured and computationally efficient subset of the full group of diffeomorphic transformations. The transformations of $\mathcal{D}_{\text{SVF}}(\Omega)$ are widely used in deformable image registration due to their smoothness, easily approximated inverses, and well-defined theoretical structure [12, 8, 28, 86]. Even though the set of SVF-based diffeomorphisms does not include all possible diffeomorphisms mapping from Ω to itself, it is sufficiently expressive for our purposes.

Definition 4.2 (Stationary Velocity Field (SVF)). A *velocity field* is defined as a vector field $v : \Omega \times [0, 1] \to \mathbb{R}^d$, where $v(p, t)$ gives the velocity of a fluid at position p and time t. If the velocity is constant over time, i.e., $v(p, t) = v(p)$ for all t, we call v a *stationary velocity field* (SVF).

A velocity field v can be used to define a *flow* g:

Definition 4.3 (Flow). Given a velocity field v, the associated *flow* $\varphi_t(p) : \Omega \to \Omega$ is defined as the solution to the ordinary differential equation (ODE)

$$\frac{\mathrm{d}\varphi_t(p)}{\mathrm{d}t} = v(\varphi_t(p)), \quad \varphi_0(p) = p \tag{4.9}$$

with $t \in [0, 1]$.

For a $v \in \mathcal{C}_0^1(\Omega, \mathbb{R}^d)$, the solution of the ODE (4.9) at time $t = 1$ yields a diffeomorphism $\varphi_1 = \exp(v)$ [5], where the exponential map $\exp : \text{SVF} \to \mathcal{D}_{\text{SVF}}(\Omega)$ associates to a stationary velocity field (SVF) a flow. Applied to a point $p \in \Omega$, this gives

$$p \mapsto \varphi_1(p) = (\exp(v))(p). \tag{4.10}$$

This exponential map connects the set of smooth vector fields to the group of diffeomorphisms [86, 81] similarly to the exponential map that connects a Lie algebra to its Lie group. SVFs are time-invariant; therefore, the diffeomorphisms defined through them are inherently smooth and invertible. The inverse transformations can be determined by negating the velocity field, i.e., $\varphi_1^{-1} = \exp(v)^{-1} = \exp(-v)$. Note that the set of SVF-based diffeomorphisms is not a group: even though there exists the identity element $id := \exp(0)$, and for all transformations in $\mathcal{D}_{\text{SVF}}(\Omega)$, there exists an inverse transformation in $\mathcal{D}_{\text{SVF}}(\Omega)$, the closure condition is not fulfilled. Still, SVF-based diffeomorphisms are particularly attractive in large-scale or learning-based settings, as they provide an efficient way for parametrising.

4.2.2 Energy Function for Canonicalisation

The canonicalisation energy E_{can} measures the similarity of a transformed image with the training data and encourages more "realistic" transformations within the allowed set of transformations $\mathcal{D}_{\text{SVF}}(\Omega)$. In this work, the energy function $E_{\text{can}} : X \times \mathcal{D}_{\text{SVF}}(\Omega) \to \mathbb{R}$ consists of a weighted sum of different loss terms:

$$E_{\text{can}}(x, g) := \underbrace{E_{X_E}(g \cdot x)}_{\text{image similarity}} + \underbrace{E_{\text{reg}}(g)}_{\text{regularisation}}, \tag{4.11}$$

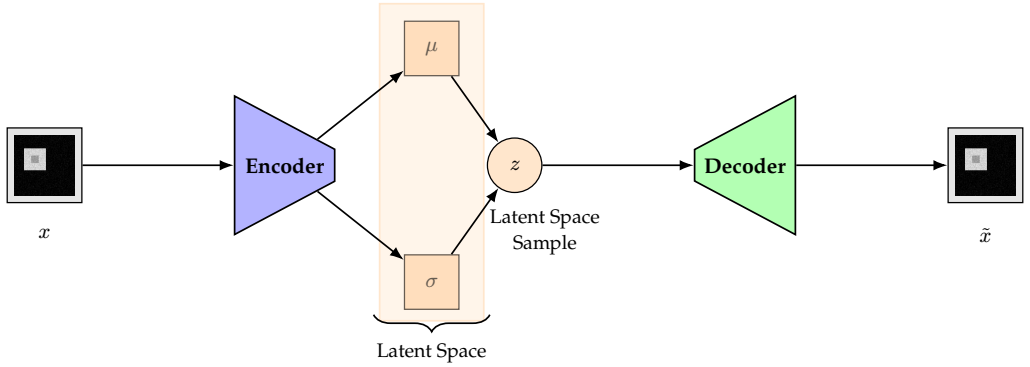


Figure 4.2: VAE architecture. The encoder maps each input to a distribution within the latent probability space, which is a multivariate Gaussian defined by a mean μ and a variance vector σ. A random latent representation is sampled from the latent probability space and passed to the decoder that then reconstructs the initial input from the lower-dimensional representation. We use a VAE to derive the energy function E_{VAE} that is defined as the VAE loss (Definition 4.14).

where the image similarity energy is defined as

$$E_{X_E}(g \cdot x) := \lambda_{\text{VAE}} E_{\text{VAE}}(g \cdot x) + \lambda_{\text{adv}} E_{\text{adv}}(g \cdot x) \tag{4.12}$$

and the regularising energy (following [19]) as

$$E_{\text{reg}}(g) := \underbrace{\lambda_{\text{grad}} \sum_{p \in \Omega} \sum_{i=1}^{d} (\nabla v_i(p))^2}_{\text{gradient loss}} + \underbrace{\lambda_{\text{jac}} \sum_{p \in \Omega} \max(0, -\det(\mathcal{J}_g(p)))}_{\text{Jacobian determinant loss}} . \tag{4.13}$$

Here, λ_{VAE}, λ_{adv}, λ_{grad}, and λ_{jac} are positive scalar weights. The energy functions E_{VAE} (VAE-based energy) and E_{adv} (adversarial energy) evaluate the similarity of an input to the training dataset X_E. The regularising energy E_{reg} penalises implausible or undesired transformations. This choice of energy is merely an example and can be exchanged completely. Nevertheless, our proposed energy is widely usable, especially as it is not specialised for the segmentation task and only depends on the training data and the group of diffeomorphisms. In the following, we discuss the individual parts of our energy choice in more detail.

VAE-based Energy E_{VAE}. One of the central parts of E_{can} is the energy function E_{VAE} derived from a *variational autoencoder* (*VAE*) [42] trained on the training images.

A VAE is a generative model that contains two parts: an *encoder* and a *decoder* [42]. The encoder maps the input into a (usually) lower-dimensional latent space, and the decoder approximately reconstructs the initial input from the lower-dimensional representation of the input in the latent space.

The special feature of a VAE compared to other autoencoders is that the latent space has a probabilistic structure. The latent space is typically a multivariate Gaussian defined by a mean and variance vector. The encoder maps each input (e.g., an image) out of a large dataset into a distribution within the latent space, rather than a single element. Instead of passing the distribution directly to the decoder, a random latent representation is sampled from the latent probability space. This sample is then fed to the decoder to

reconstruct the original input. An illustration of the VAE architecture can be found in Figure 4.2. The input images are passed through the network. Then, the VAE loss is computed and backpropagation is used to update the model weights.

Definition 4.4 (VAE Loss/ELBO). We consider a VAE with a latent space of dimension $d < n$. For each input $x \in [0,1]^n$, we assume a Gaussian latent space distribution $q(\cdot|x) \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$, parametrised by $\mu := (\mu_j)_{j=1}^d \in \mathbb{R}^d$ and $\sigma^2 := (\sigma_j)_{j=1}^d \in \mathbb{R}^d$. We define the VAE loss as a combination of *binary cross entropy* (BCE) and *Kullback–Leibler divergence* (D_{KL}, also called KL divergence) between the encoder distribution and the standard normal distribution:

$$\mathcal{L}_{\mathrm{VAE}}(x) := \underbrace{\mathrm{BCE}(x, \hat{x})}_{\text{reconstruction loss}} + \underbrace{D_{\mathrm{KL}}\left(q(\cdot|x) \,||\, \mathcal{N}(0, I)\right)}_{\text{KL divergence}}, \tag{4.14}$$

where $\hat{x} \in [0,1]^n$ is the output of the VAE. Here, the binary cross entropy is defined as

$$\mathrm{BCE}(x, \hat{x}) := -\sum_{i=1}^{n} \left[x_i \ln \hat{x}_i + (1 - x_i) \ln(1 - \hat{x}_i)\right] \tag{4.15}$$

with $0 \ln(0) := 0$, and the KL divergence is

$$D_{\mathrm{KL}}\left(q(\cdot|x) \,||\, \mathcal{N}(0, I)\right) := -\frac{1}{2} \sum_{j=1}^{d} \left(1 + \ln \sigma_j^2 - \mu_j^2 - \sigma_j^2\right). \tag{4.16}$$

The latter helps to bring the approximate posterior close to a standard normal distribution, while the binary cross entropy helps to minimise the difference between the input x and the reconstructed input $\hat{x}$. The VAE loss is also called *evidence lower bound* (*ELBO*). Classically, the first term is more complicated than the BCE. However, as our inputs are normalised to $[0,1]^n$ we can use the BCE.

We use the trained VAE to compute a scalar energy value for new inputs. The energy E_{VAE} corresponds to the VAE loss and indicates the distance from an input to the training dataset X_E, on which we train the VAE. Hence, we define VAE energy $E_{\mathrm{VAE}} : \mathbb{R}^n \to [0, \infty)$ as

$$E_{\mathrm{VAE}}(x) := \mathcal{L}_{\mathrm{VAE}}(x). \tag{4.17}$$

A low energy means that the input can be reconstructed well by the VAE.

Adversarial Energy E_{adv}. To further increase the similarity between a transformed image and the training data, we additionally use an adversarial energy function. The energy function is based on the adversarial discriminator that is learned by a *discriminator network*, trained to distinguish between the training images X_E and the transformed ones.

The adversarial discriminator is determined by a *Wasserstein generative adversarial network* (*WGAN*) with a *gradient penalty* (*WGAN-GP*) framework [6]. The idea of a WGAN is that it measures how well a transformed image aligns with the distribution of the training data by approximating the Wasserstein distance between them. The gradient penalty ensures smooth and stable training. We base our implementation on adversarial regularisation [57, 63] similarly to the implementation in [72]. A discriminator $D : X \to \mathbb{R}$, is trained in a way that

- $D(x)$ is small if x is "close" to the training data X_E ("real" images), and
- $D(x)$ is large when x is different from the training data X_E ("fake" images).

We generate the "fake" images by applying random SVF-based diffeomorphisms to "real" images. This setup defines two implicit data distributions, p_{real} and p_{fake}. Those correspond to the dataset X_E and to the distribution of randomly transformed images, and are accessed only via sampling rather than by explicit probability densities.

Let x be an image of the training data X_E and $\hat{x} = g \cdot x$ a generated "fake" image. Furthermore, let the image

$$\tilde{x} = \alpha x + (1 - \alpha)\hat{x}, \quad \alpha \sim \text{Uniform}(0, 1) \tag{4.18}$$

be a linear interpolation between x and $\hat{x}$, where $\tilde{p}$ denotes the implicit distribution of interpolated samples $\tilde{x}$ between real and fake images. The WGAN-GP loss for training the discriminator D is defined as

$$\mathcal{L}_{\text{D}}(x, \hat{x}) := \mathbb{E}_{\hat{x} \sim p_{\text{fake}}}[D(\hat{x})] - \mathbb{E}_{x \sim p_{\text{real}}}[D(x)] + \mu \cdot \mathbb{E}_{\tilde{x} \sim \tilde{p}}\left[(|\nabla_{\tilde{x}} D(\tilde{x})|_2 - 1)^2\right], \tag{4.19}$$

where $\mu \in \mathbb{R}$ is the gradient penalty weight. The term $\mathbb{E}_{x \sim p_{\text{real}}}[D(x)]$ denotes the expectation of the discriminator output over data samples,

$$\mathbb{E}_{x \sim p_{\text{real}}}[D(x)] = \int D(x)\, dp_{\text{real}}(x). \tag{4.20}$$

In practice, we approximate the expectation through averaging over the batch size B,

$$\mathbb{E}_{x \sim p_{\text{real}}}[D(x)] \approx \frac{1}{B} \sum_{i=1}^{B} D(x_i), \quad x_i \sim p_{\text{real}}(x). \tag{4.21}$$

We define the expectation analogously for $\hat{x}$ and $\tilde{x}$ with their respective distributions p_{fake} and $\tilde{p}$. The trained discriminator can be used directly to define the adversarial loss:

$$E_{\text{adv}}(x) := D(x). \tag{4.22}$$

The "closer" the input x to the training dataset X_E, the smaller the energy $E_{\text{adv}}(x)$. For more details, see [57, 63, 72].

Regularising Energy E_{reg}. Following [19], we use a regularising energy E_{reg} to encourage physically plausible transformations. As in (4.13), we define the regularising energy E_{reg} as

$$E_{\text{reg}}(g) := \underbrace{\lambda_{\text{grad}} \sum_{p \in \Omega} \sum_{i=1}^{d} (\nabla v_i(p))^2}_{\text{gradient loss}} + \underbrace{\lambda_{\text{jac}} \sum_{p \in \Omega} \max(0, -\det(\mathcal{J}_g(p)))}_{\text{Jacobian determinant loss}}, \tag{4.23}$$

where v_i is the i-th component of the SVF v that induces g, and $\det(\mathcal{J}_g)$ the Jacobian determinant of g. The gradient loss is large for deformations with high spatial variation in their flow field. Therefore, it is small when the deformation is smooth. The Jacobian determinant loss penalises mappings with at least one $p \in \Omega$ with negative Jacobian

determinant. We require $\det(\mathcal{J}_g(p)) > 0$ for all $p \in \Omega$, as such diffeomorphisms are orientation-preserving, which ensures physical plausibility. A negative determinant indicates local reversal of orientation, such as folding or tearing, which does not make sense in most image processing tasks. Therefore, restricting to diffeomorphisms with a positive Jacobian determinant is a common practice in image processing and has been widely researched [55, 47].

The regularising energy E_{reg} is based on the regularisation used in [19]. This energy is only one option to regularise, and it should be adjusted task specific. One could, for instance, leave out the Jacobian determinant loss if an orientation-reversing transformation is a possible and desirable solution of the minimisation problem.

In summary, we propose the canonicalisation energy

$$E_{\text{can}}(x, g) = \underbrace{\lambda_{\text{VAE}} E_{\text{VAE}}(g \cdot x) + \lambda_{\text{adv}} E_{\text{adv}}(g \cdot x)}_{\text{image similarity } E_{X_E}} + \underbrace{\lambda_{\text{grad}} \sum_{p \in \Omega} \sum_{i=1}^{d} (\nabla v_i(p))^2}_{\text{gradient loss}} + \underbrace{\lambda_{\text{jac}} \sum_{p \in \Omega} \max(0, -\det(\mathcal{J}_g(p)))}_{\text{Jacobian determinant loss}}, \tag{4.24}$$

that we can now use to find a suitable canonicalising element g_x.

4.2.3 Canonicalisation via Gradient-based Optimisation

In contrast to the approach of LieLAC [72], we estimate the image-specific canonicalising element by using a gradient-based optimisation method similar to [19, 28, 10] to minimise the canonicalisation energy E_{can}. We use the framework of Bostelmann et al. [19], which was developed for image registration, and adjust the energy term to make it suitable for our purposes. We replace the classical registration term of a data term and a regulariser by our canonicalisation energy E_{can}, making it only dependent on one input image and the transformation. We find that the approach by Bostelmann et al. is fitting for our setup, as it allows us to find SVF-based diffeomorphisms, but, in general, any optimisation method that is able to solve (4.8) could be used for the canonicalisation step.

In the following, we give a short overview of the used optimisation strategy. For a more detailed description, see [19].

SIREN Network. Following Bostelmann et al. [19], we use a *Sinusoidal Representation Network* (*SIREN*) network [74] to parametrise the SVFs, which underlie our diffeomorphisms. Rather than using traditional activation functions, periodic (sinusoidal) activation functions (see Definition A.7) are used. This makes those networks particularly suitable for modelling complex continuous signals.

A SIREN network consists of several fully-connected layers. At each layer, a sinusoidal activation function is applied. Typically, a large frequency (e.g., $\omega_0 = 30$) is chosen for the first layer, which is then followed by smaller frequencies for the remaining layers (e.g., $\omega_0 = 1$) [74]. With this setting, high-frequency components are detected while still encouraging smoothness of the output. Like in [19], the final layer outputs the velocity field $v_\theta(p)$ for each position $p \in \Omega$, a coordinate-based representation of the SVF.

Scaling and Squaring. Bostelmann et al. calculate the flow $g_\theta = \exp(v_\theta)$ using a *Scaling and Squaring* approach [7], which we also employ. Starting with a small deformation $g_{1/2^n}(p) \approx p + \frac{v(p)}{2^n}$, the complete transformation is recovered by recursively composing this map with itself. Specifically, for $k = n-1, ..., 0$, we define

$$g_{1/2^k}(p) = g_{1/2^{k+1}}(g_{1/2^{k+1}}(p)), \tag{4.25}$$

such that after n recursive steps, the deformation g_θ is obtained. In this manner, we approximate the solution of the ODE (4.9) at $t = 1$.

After an initialisation, the network from [19] updates the flow g_θ N times. Each time, the flow is first calculated using the SIREN network and Scaling and Squaring. The flow g_θ and the input x are then used to compute a loss, in our case the canonicalising energy E_{can} by which θ and therefore g_θ are updated in a backpropagation step. We can backpropagate even though our energy function contains the "max" operator, by "differentiating" this operator, following the concept of subgradients like in [19]. Similar to the ReLU function (Definition A.4), we set the gradient at the critical point $x = 0$ to a specific value (here, in PyTorch, set to 0). After N update steps, the transformation g_θ is returned.

We use the diffeomorphic transformation $g_x := g_\theta$ that we obtain by applying the algorithm of Bostelmann et al. [19] to transform the input image x into a canonical representation

$$x_c = g_x \cdot x = x \circ g_x. \tag{4.26}$$

In the second step, we then apply the segmentation model f_θ to this canonical representation x_c. In the following section, we describe the training of the model f_θ. Note that segmentation is just an exemplary application of our network. DiffeoNN can be used for arbitrary other tasks where diffeomorphism-equivariance is desired. One only needs to replace the inner model f_θ accordingly. We can even use a pretrained model for f_θ and make it diffeomorphism-equivariant.

4.3 Segmentation

For the inner segmentation model f_θ, we use a U-Net [67] that is trained only on the training data X_E. A U-Net is a supervised machine learning method for image segmentation, which needs annotated (i.e., labelled) data for training. The name stems from the characteristic (symmetric) "U"-shape of the network architecture that is a modification of a *fully convolutional network* [56] (FCN), i.e., a convolutional neural network without fully connected layers that operates directly on image grids. The network's architecture consists mainly of two parts: a *contracting path* and an *expanding path*.

The contracting path reduces the resolution by repeated application of convolutions, ReLU activation functions, and max pooling (Section A.1). This successively increases the feature information while decreasing the spatial information. At the bottleneck (the lowest resolution), the feature maps represent a compact semantic encoding of the input image.

The expanding path mirrors the contracting path. It increases the spatial resolution. The features and spatial information are combined using an upsampling operation

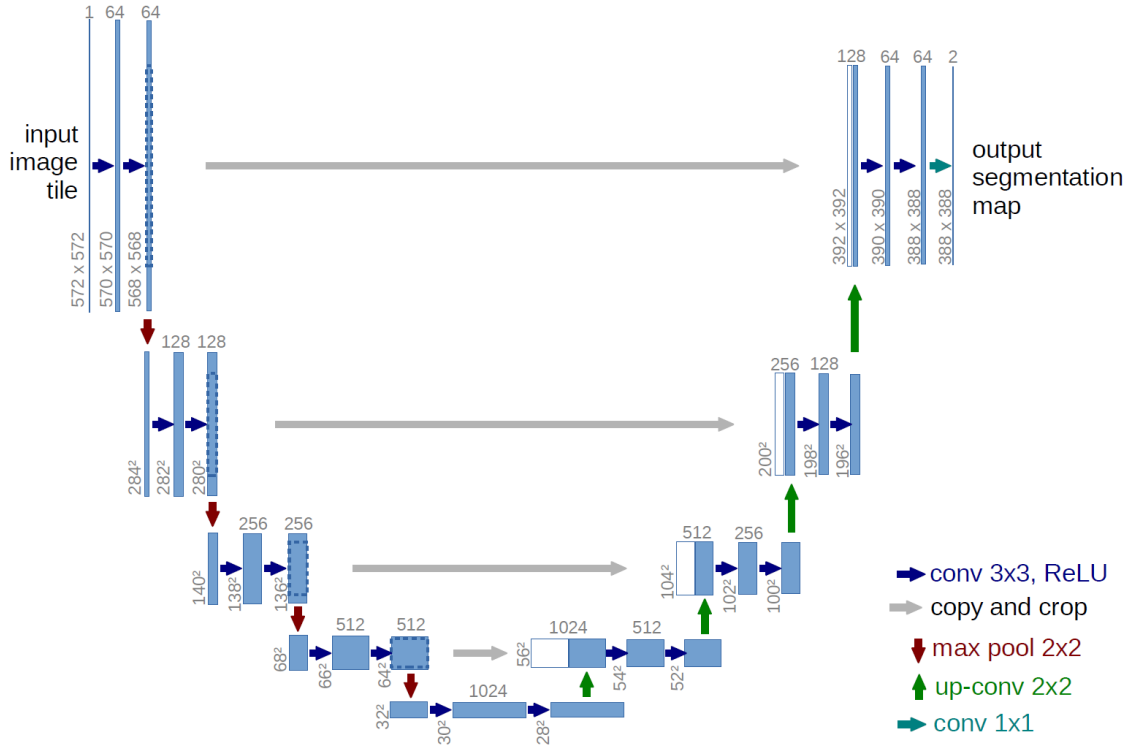


Figure 4.3: Example of a U-Net architecture from [67]. The contracting path and expanding path form the classical "U"-shape. The contracting path reduces the resolution by repeated sequences of convolutions, ReLU activation functions, and max pooling. At the lowest level (the bottleneck), the feature maps contain a compact semantic representation of the input image, which is then expanded again as the resolution increases along the expanding path that mirrors the contracting path. The features and spatial information are combined by an upsampling operation and concatenations with high-dimensional features from the contracting path through skip connections.

(e.g., transposed convolutions) and concatenated with high-dimensional features from the contracting path trough *skip connections*. The latter helps to preserve fine spatial information that might have been lost during downsampling. This improves the localisation accuracy. After a final 1×1 convolution, a *softmax* or *sigmoid* activation function is applied, depending on the segmentation task (multi-class or binary). To supervise the training, a *pixel-wise cross-entropy* loss is classically used [67]:

Definition 4.5 (Pixel-Wise Cross-Entropy). Let k be the number of pixels in an image x, let $y_i^{gt} \in \{0,1\}$ be the ground-truth class label at pixel i, and let $f_\theta(x)_{i,c} \in \mathbb{R}$ be the predicted output of the network for class $c \in \{0,1\}$ at pixel i. Then, the pixel-wise cross-entropy is defined as

$$\mathcal{L}_{CE}(x) := -\tfrac{1}{k} \sum_{i=1}^{k} \ln \left(\frac{\exp(f_\theta(x)_{i,y_i^{gt}})}{\exp(f_\theta(x)_{i,0}) + \exp(f_\theta(x)_{i,1})} \right). \tag{4.27}$$

This formulation corresponds to the categorical (softmax-based) cross-entropy for two classes. It treats the segmentation task as a two-class classification problem, and is therefore equivalent to the binary cross-entropy when $c \in \{0,1\}$. The categorical version is used here for consistency with the multi-class case, where the denominator generalises

naturally to $\sum_{c=1}^{C} \exp(f_\theta(x)_{i,c})$. An example of a U-Net architecture can be found in Figure 4.3.

Ronneberger et al. [67] demonstrated that due to its architectural design and data augmentation, a U-Net performs well even when trained on very few annotated images. Since its first introduction, U-Nets have been widely used and modified [9, 24], and still remain the go-to method for various segmentation tasks [9]. Another advantage is that Convolutional Neural Networks (CNNs), and therefore also U-Nets, are translation-equivariant by design [34, 51].

4.4 Reverse Canonicalisation

In the third and last step of DiffeoNN, we reverse the canonicalisation. In this step, the segmentation output for x_c,

$$y_{x_c} = f_\theta(x_c) = f_\theta(g_x \cdot x), \tag{4.28}$$

is transformed into a segmentation of x by applying the inverse of the canonicalising element g_x to y_{x_c}. The segmentation of x, and therefore the final output of DiffeoNN, is

$$y_x = g_x^{-1} \cdot y_{x_c}, \tag{4.29}$$

where g_x^{-1} is the inverse of the canonicalising element g_x, computed in the canonicalisation step (Section 4.2). This inverse exists and can, up to discretisation, easily be determined from the negative velocity field, as g_x is an SVF-based diffeomorphism.

4.5 Theoretical Connection and Summary

Our formal proof of generalisation in Chapter 3 only holds for Lie groups acting on compact manifolds and does therefore not apply to the group of diffeomorphisms. However, the structural parallels and practical regularisation of diffeomorphisms suggest that the same principles transfer effectively to the image domain:

- The data manifold of images $\mathcal{X}$ is not finite-dimensional and compact, but as we discretise the image domain and pixel intensities in practice, the realisation of $\mathcal{X}$ is in fact finite-dimensional and compact.
- Diffeomorphisms are not a compact Lie group. However, they have similar structural properties that allow us to use the canonicalisation strategy: Diffeomorphisms act smoothly on the image domain Ω and preserve topological properties. We parametrise the diffeomorphisms by stationary velocity fields (SVFs), which connects them even closer to Lie group theory. As the image domain is discretised in practice, the velocity field is represented on a discrete grid with finitely many points. Therefore, optimisation over the set of SVF-based diffeomorphisms $\mathcal{D}_{\text{SVF}}(\Omega)$ is effectively over a finite-dimensional vector space, which makes it more similar to the optimisation over a finite-dimensional Lie group.

- We add a regularising energy E_{reg} to our image similarity energy E_{X_E} to create the canonicalisation energy E_{can}. By adding the regularising energy, we can technically no longer guarantee exact equivariance. Nevertheless, we achieve approximate equivariance, which is reflected in our empirical results in the following chapter, see Section 5.3 and Section 5.5. We also verified the approximate diffeomorphism-invariance of the canonicalisation step in DiffeoNN empirically, see Section 5.4.

All in all, in this chapter, we introduced a diffeomorphism-equivariant segmentation method, called DiffeoNN. DiffeoNN contains a canonicalisation, a classical neural segmentation network (U-Net), and a reverse canonicalisation. While it is only trained on a comparatively small training dataset X_E, DiffeoNN finds segmentations on the whole set X that covers the images that can be obtained by transforming the images of X_E diffeomorphically. DiffeoNN serves as an example and can easily be adapted for other tasks than image segmentation, as only the inner task-specific network needs to be exchanged. It is therefore also possible to use DiffeoNN to turn a pretrained model diffeomorphism-equivariant without retraining.

The optimal choice of parameters of DiffeoNN and its performance are examined in the next chapter. We furthermore give an example of an explicit implementation.

5

Experiments and Results

In this chapter, we evaluate the performance of DiffeoNN. For this, we create a synthetic dataset (see Section 5.1). The training of the segmentation U-Net as well as the training of the canonicalising energy, their hyperparameter tuning, and the general implementation of DiffeoNN, are described in Section 5.2. For the implementation of DiffeoNN, we use code from [19, 67, 42, 57, 63]. In Section 5.3, we benchmark the performance of DiffeoNN against a data-augmented U-Net and the inner U-Net in DiffeoNN without canonicalisation (naïve approach). Furthermore, we evaluate if the canonicalisation is in fact diffeomorphism invariant in Section 5.4. In Section 5.5, we additionally apply DiffeoNN to real-world chest X-ray images to segment lungs.

5.1 Synthetic Datasets

To train and evaluate DiffeoNN, we create synthetic datasets. In general, we need data for two different tasks:

1. "canonical" training data X_E to pretrain the segmentation model and canonicalisation energy for the network,
2. diffeomorphically transformed data to evaluate the performance of the overall approach.

In the following two sections, the generation of both datasets is described in detail.

5.1.1 Training Dataset X_E

For training and evaluation of the different models contained in DiffeoNN (segmentation, VAE, adversarial), we generate a synthetic dataset of labelled "canonical" images. The dataset is specifically created for pixel-wise segmentation and contains an image of two squares nested within each other and a corresponding binary ground-truth segmentation, denoting the inner square.

Generating Images. We choose the images in a similar way to in Example 3.1. All images are of size 128×128. Each image contains two nested squares of varying size and colour with the squares' edges parallel to the image edges. We draw the squares'

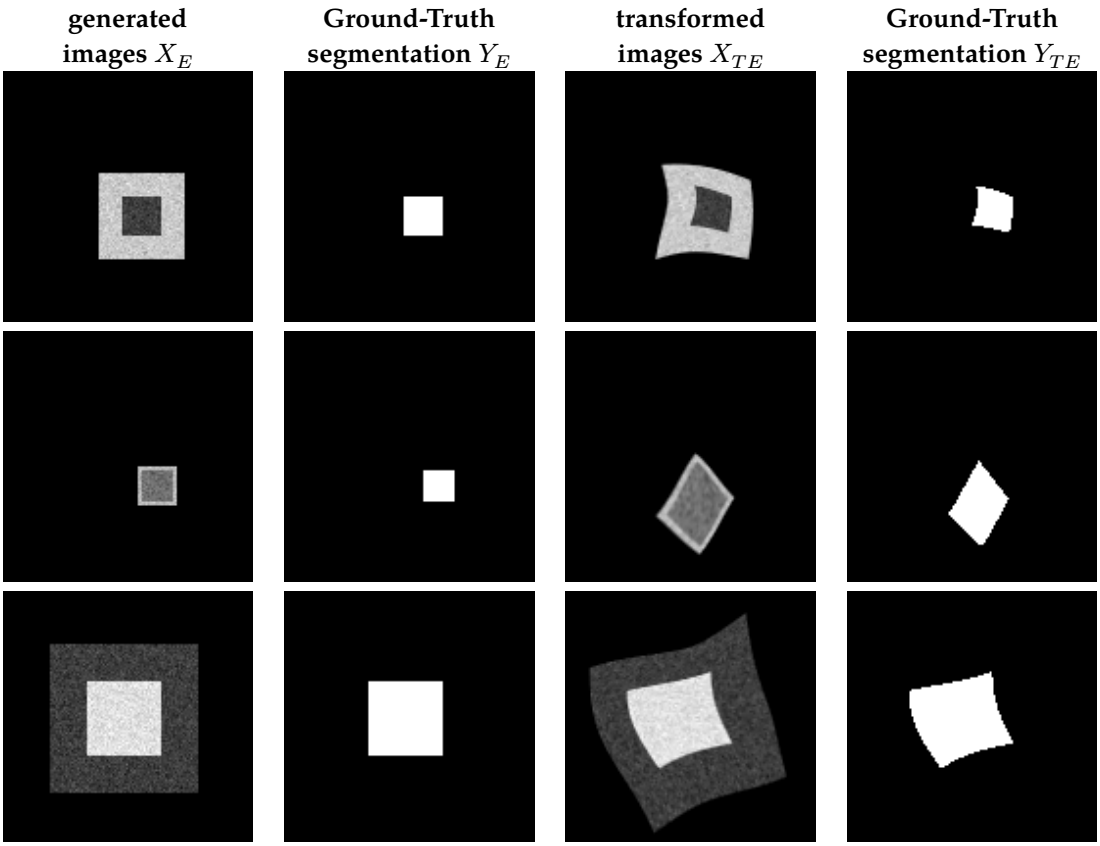


Figure 5.1: Examples from the synthetic dataset that contains the "canonical" images X_E (**left**) and their binary segmentation Y_E (**second column**), as well as diffeomorphically transformed versions of them, X_{TE} (**third column**), and of the corresponding segmentation Y_{TE} (**right**). The images in X_E contain two nested squares of varying size with added Gaussian noise and axis-aligned edges. The binary segmentation Y_E indicates the inner square.

sizes independently from a uniform range, such that the smaller square is always strictly smaller than the larger. This ensures variability in scale of the generated data. We set the centre of both squares to the same location, which we select randomly from a uniform range. The uniform range depends on the larger square's size, to ensure that both squares remain within the image domain. We assign each square a different random greyscale intensity in the range $[1, 255]$ with a minimum distance of 10. The background intensity is set to 0. For this dataset, we create 12 000 images.

To loop back to the theoretical setup: The training dataset is the set that is generated above. In the following, we therefore refer to this dataset as X_E. The change from $[0, 1]$ to $[0, 255]$ is only a scaling issue that does not affect the network structure significantly.

Noise Augmentation. To create less homogeneous surfaces, we add Gaussian noise to the images. We layer the noise only over the squares, leaving the image background constant 0. In many imaging techniques, Gaussian noise is a common occurrence due to sensor sensitivity and other visual effects. So, the data is more realistic and suitable for evaluating the robustness of DiffeoNN.

Corresponding Segmentations. For each image, we create a binary ground-truth segmentation. It serves as the segmentation target for the central model of DiffeoNN. To generate a suitable target for training binary U-Net models, we render the smaller, inner square in the segmentation with a constant value of 1, while the rest of the image remains 0. No noise is added to the segmentation.

The set of 12 000 segmentations will be called Y_E from now on. Examples of X_E and Y_E are shown in Figure 5.1. We split the dataset X_E and its corresponding segmentations Y_E into

- a training subset $(X_E^{\text{train}}, Y_E^{\text{train}})$ (contains 8 000 images/segmentations each),
- a validation subset$(X_E^{\text{val}}, Y_E^{\text{val}})$ (contains 2 000 images/segmentations each), and
- a test subset $(X_E^{\text{test}}, Y_E^{\text{test}})$ (contains 2 000 images/segmentations each).

5.1.2 Dataset for Testing DiffeoNN

To evaluate our DiffeoNN approach and the invariance of the canonicalisation, we generate diffeomorphically transformed data by randomly transforming image and segmentation pairs from X_E and Y_E. The implementation of these transformations relies on the code of [19].

Transformed Data. To mimic realistic transformations, we transform each image of X_E with a random SVF-based diffeomorphism. To avoid an "inverse crime" [85], we do not use the same technique as in our canonicalisation step to parametrise diffeomorphisms for the creation of our diffeomorphically transformed synthetic data: We generate the transformed synthetic data through a concatenation of transformations. First, we apply a scaling matrix and a random rotation. In addition, we sample a deformation field that is created by interpolating 12 randomly displaced control points using radial basis functions (RBFs). The control points are uniformly placed across the image domain, and for each control point, we construct its displacement with a random two-dimensional vector within a maximum displacement norm of 0.12.

Even though the use of RBFs does not guarantee that the transformation is an SVF-based diffeomorphism, we make it "smooth enough" through a smart choice of parameters, which makes extreme transformations highly unlikely. This suggests that there exists a matching SVF-based diffeomorphism to canonicalise all of the transformed images. This assumption is supported by our experiments in Section 5.3, which show that, with the right hyperparameters, the canonicalisation step finds a suitable canonicalising element in the set of SVF-based diffeomorphisms for all considered synthetically transformed input images. Furthermore, to ensure that the transformation is orientation-preserving, only deformations with a positive Jacobian determinant are applied.

We do not apply a translation, as translation-equivariance is relatively straightforward to achieve by using suitable convolutional layers [34, 51], and without translation it is less likely that the transformation moves the squares outside the image frame.

We apply the same transformation to the images of X_E and the corresponding binary segmentation Y_E, to ensure the existence of a correct ground-truth segmentation for testing. The images in X_E are transformed by using a spatial warping function with bilinear

interpolation, and the segmentation using nearest-neighbour interpolation. This ensures that the values stay 0 or 1, as we want to create a binary segmentation as ground-truth.

Ensuring Objects Stay in Domain. To guarantee that we do not move the squares outside the image domain by the transformation, we verify that no non-background point touches the image border after the transformation. Otherwise, the transformation is discarded, and a new one is randomly created. If, after 200 attempts, no suitable transformation is found, the image in X_E and its transformed version are discarded. This happens very rarely, and does not occur in the data used in the subsequent performance evaluation.

In the following, we split the transformed data analogously to the data X_E and refer to the sets of transformed images and their segmentations as $(X_{TE}^{\text{train}}, Y_{TE}^{\text{train}})$ (contains 8 000 images/segmentations each), $(X_{TE}^{\text{val}}, Y_{TE}^{\text{val}})$ (contains 2 000 images/segmentations each), and $(X_{TE}^{\text{test}}, Y_{TE}^{\text{test}})$ (contains 2 000 images/segmentations each). A few examples of the datasets are given in Figure 5.1. More examples of the datasets can be found in Appendix A.2.

5.2 Hyperparameter Tuning and Implementation of DiffeoNN

DiffeoNN has several free variables. In this section, we evaluate their influence on the different models that we use in DiffeoNN. We address implementation details and the hyperparameter choices.

5.2.1 Segmentation

Crucial to the performance of DiffeoNN is the performance of the inner network f_θ, as it determines the upper bound. If the inner network f_θ performs poorly on the training data (X_E, Y_E), the whole network $\tilde{f}_\theta$ can not perform well on the transformed dataset (X_{TE}, Y_{TE}) by construction of $\tilde{f}_\theta$. It is therefore useful to optimise the performance of the inner model f_θ, which is a segmentation U-Net. In this section, the training setup is further described. Moreover, we investigate the effect of the hyperparameters on the segmentation quality to improve the segmentation model. We conduct hyperparameter tuning to determine the optimal batch size and learning rate.

Training. For each combination of hyperparameters, we fully train a U-Net by using the training $(X_E^{\text{train}}, Y_E^{\text{train}})$ and validation datasets $(X_E^{\text{val}}, Y_E^{\text{val}})$ from Section 5.1 without augmentation. We use the Adam optimiser [41] and train for 10 epochs. To supervise the training, we use a pixel-wise cross-entropy loss (see Definition 4.5). The implementation of the U-Net is a lightly modified version of the original U-Net implementation [67].

Evaluation. We conduct a total of 12 experiments, varying the learning rate between $\{5 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 1 \cdot 10^{-5}\}$, and the batch size between $\{2, 4, 8\}$. Furthermore, we evaluate the performance of the models on the test datasets $(X_E^{\text{test}}, Y_E^{\text{test}})$ using the Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy:

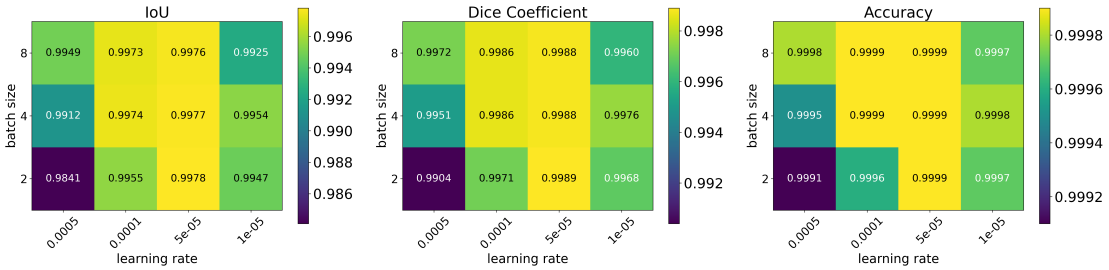


Figure 5.2: Hyperparameter tuning of the U-Net is conducted, varying the batch size between $\{2, 4, 8\}$ and the learning rate between $\{5 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 1 \cdot 10^{-5}\}$. We evaluate the performance on the testing dataset X_E^{test} using the Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy. The model trained with a learning rate of $5 \cdot 10^{-5}$ and a batch size of 2 has the maximal average IoU, average Dice coefficient, and average accuracy.

Definition 5.1 (Intersection-over-Union (IoU)). Let $\Omega_h := \{1, ..., 128\} \times \{1, ..., 128\}$ the discrete image domain. Let $y_x \in \{0, 1\}^{128 \times 128}$ be the predicted segmentation of an input x and $y_x^{gt} \in \{0, 1\}^{128 \times 128}$ be the binary ground-truth segmentation of x. The Intersection-over-Union (IoU) measures the agreement of the two segmentations:

$$\text{IoU}(y_x, y_x^{gt}) := \frac{\sum_{(i,j)\in\Omega_h} \min\{(y_x)_{i,j}, (y_x^{gt})_{i,j}\}}{\sum_{(i,j)\in\Omega_h} \max\{(y_x)_{i,j}, (y_x^{gt})_{i,j}\}}, \tag{5.1}$$

where $(y_x)_{i,j}$ is the value of y_x at $(i, j) \in \Omega_h$ and $(y_x^{gt})_{i,j}$ is defined analogously.

Definition 5.2 (Dice Coefficient). We are using the setup from Definition 5.1. The Dice coefficient measures the similarity of two segmentations:

$$\text{Dice}(y_x, y_x^{gt}) := \frac{2\sum_{(i,j)\in\Omega_h} \min\{(y_x)_{i,j}, (y_x^{gt})_{i,j}\}}{\sum_{(i,j)\in\Omega_h} (y_x)_{i,j} + \sum_{(i,j)\in\Omega_h} (y_x^{gt})_{i,j}}, \tag{5.2}$$

Definition 5.3 (Pixel-wise Accuracy). We are using the setup from Definition 5.1. The pixel-wise accuracy Acc measures the correctly predicted pixels (foreground/background)

$$\text{Acc}(y_x, y_x^{gt}) := \frac{\sum_{(i,j)\in\Omega_h} \mathbb{1}_{\{(y_x)_{i,j} = (y_x^{gt})_{i,j}\}}}{|\Omega_h|}. \tag{5.3}$$

All these scores have values in $[0, 1]$, where a maximal value is more desirable and the optimal value is 1. We visualise the results in Figure 5.2, where the average metric value across all test samples $(X_E^{\text{test}}, Y_E^{\text{test}})$ is shown for varying batch sizes and learning rates. We choose a model trained with a learning rate of $5 \cdot 10^{-5}$ and a batch size of 2 for further experiments, as the average IoU, average Dice coefficient, and average accuracy are best for this choice of parameters.

5.2.2 Canonicalisation Step

Following Section 4.2, we compute a canonicalising element g_x via gradient-based optimisation, where we minimise the canonicalisation energy E_{can}.

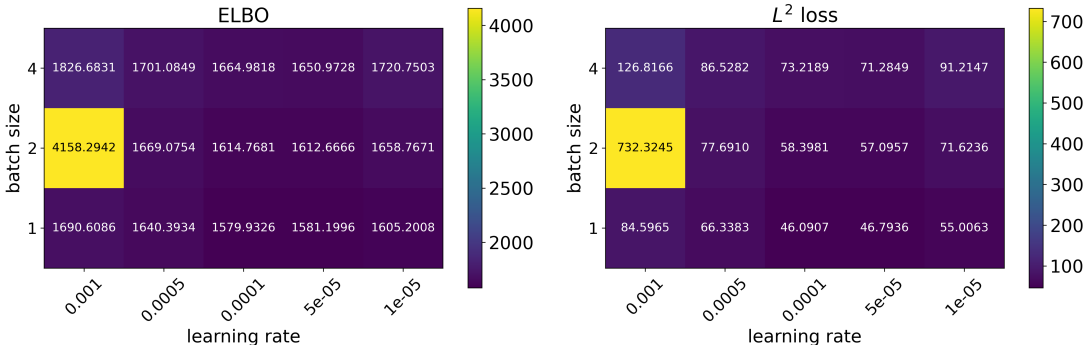


Figure 5.3: Hyperparameter tuning of the VAE on the testing dataset X_E^{test}. We vary the batch size between $\{1, 2, 4\}$ and the learning rate between $\{1 \cdot 10^{-3}, 5 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 1 \cdot 10^{-5}\}$. As the ELBO and L^2 reconstruction loss are minimal with a batch size of 1 and a learning rate of $1 \cdot 10^{-4}$, we choose those training parameters for further experiments.

Gradient-based Optimisation. The implementation of the gradient-based optimisation is based on [19, 28, 10]. We parametrise the transformation through a stationary velocity field (SVF) over the image domain, and calculate the transformation using Scaling and Squaring [7]. We mainly use the code from [19] and adjust the energy that is minimised. Following Bostelmann et al., we initialise transformation randomly, optimise using Adam [41], and perform 5000 update steps. For more detailed explanations on the used networks, and the optimisation process, see Section 4.2.3 and [19].

Canonicalisation Energy. As described in Section 4.2.2, the energy function E_{can} (4.24) consists of a weighted sum of different loss terms with scalar weights. In conducted experiments, described in the following, we choose the scalar weights as $\lambda_{\text{VAE}} = 1 \cdot 10^{-5}$, $\lambda_{\text{grad}} = 1$, and $\lambda_{\text{jac}} = 10$.

As especially the adversarial energy term E_{adv} plays a crucial role for the closeness of the canonicalised inputs to the training dataset X_E, the value of λ_{adv} has a large influence on the performance of the whole network. Therefore, we determine λ_{adv} by comparing different values, described in Section 5.2.4. More details on the training of the models and hyperparameter tuning of the VAE energies and adversarial energy can be found in the following two sections.

5.2.3 VAE Energy

To get a suitable energy E_{VAE} for the canonicalisation step, we conduct hyperparameter tuning of the convolutional variational autoencoder (VAE-CNN). For this, we investigate the influence of the batch size and learning rate on the performance of the VAE-CNN. Our implementation is based on code from [42, 72].

Training. We fully train a VAE-CNN for each combination of hyperparameters using the training dataset X_E^{train} from Section 5.1 without augmentation. The architecture of the VAE-CNN consists of an encoder with six convolutional layers, a decoder with six transposed, fully connected, convolutional layers, and a 10-dimensional latent space. As explained in Section 4.2, we use the ELBO (see Definition 4.4) as loss for training the VAE.

We optimise the model via the Adam optimiser [41] and set the number of epochs to 50. The performance is monitored on the validation dataset X_E^{val}.

Evaluation. We evaluate all combinations of varying the batch size between $\{1, 2, 4\}$ and the learning rate between $\{1 \cdot 10^{-3}, 5 \cdot 10^{-4}, 1 \cdot 10^{-4}, 5 \cdot 10^{-5}, 1 \cdot 10^{-5}\}$. We analyse the performance of the models on the test dataset $(X_E^{\text{test}}, Y_E^{\text{test}})$ by examining the ELBO (Definition 4.4) and the L^2 reconstruction loss:

Definition 5.4 (L^2 Reconstruction Loss). Let $x \in \mathbb{R}^n$ be the input and $\hat{x} \in \mathbb{R}^n$ its reconstruction (the final output). The L^2 loss is defined as

$$L^2(x, \hat{x}) = \|x - \hat{x}\|_F^2 = \sum_{i,j=1}^{128} ((x)_{i,j} - (\hat{x})_{i,j})^2, \tag{5.4}$$

where $(x)_{i,j}$ and $(\hat{x})_{i,j}$ denote the value at the position $(i, j) \in \{1, ..., 128\} \times \{1, ..., 128\}$. It measures the pixel-wise squared error between the input and its reconstruction.

Both metrics map to $\mathbb{R}_{\geq 0}$, and are optimal when their value is 0. The results are visualised in Figure 5.3, where the average metric values across all test samples X_E^{test} are shown depending on the batch size and learning rate. The models trained with a learning rate below $5 \cdot 10^{-3}$ all perform in the same magnitude, while a learning rate of $1 \cdot 10^{-3}$ with a batch size of 2 produces particularly high losses. The ELBO and L^2 reconstruction loss is minimal when we choose a batch size of 1 and a learning rate of $5 \cdot 10^{-5}$. We therefore select those parameters for further experiments.

5.2.4 Adversarial Energy

The adversarial energy term E_{adv} plays an important role in the canonicalisation energy E_{can} to ensure the canonicalised input is "close" to the training dataset X_E. As described in Section 4.2.2, we use the discriminator of a Wasserstein GAN to determine the adversarial energy term E_{adv}. The implementation closely follows [57, 63, 72]. Furthermore, we conduct hyperparameter tuning to find the optimal weight λ_{adv} for E_{adv} in the canonicalisation energy E_{can}.

Training. We train the adversarial discriminator to distinguish between the original square images X_E (see Section 5.1.1) and their diffeomorphically transformed counterparts, which are generated during each training iteration (on-the-fly). This avoids storing or precomputing transformed images. Furthermore, it makes sure that the model does not memorise a static transformation set. We generate the diffeomorphic transformations as described in Section 5.1.2.

We train the adversarial discriminator (see Section 4.2.2) on the dataset X_E^{train}, using the Adam optimiser [41], and validate the training on X_E^{val}. We choose the gradient penalty weight $\mu = 10$ and set the learning rate to $1 \cdot 10^{-4}$, the batch size to 16, and the number of epochs to 50.

Adversarial Weight. To ensure that the canonicalised input is "close" to the training

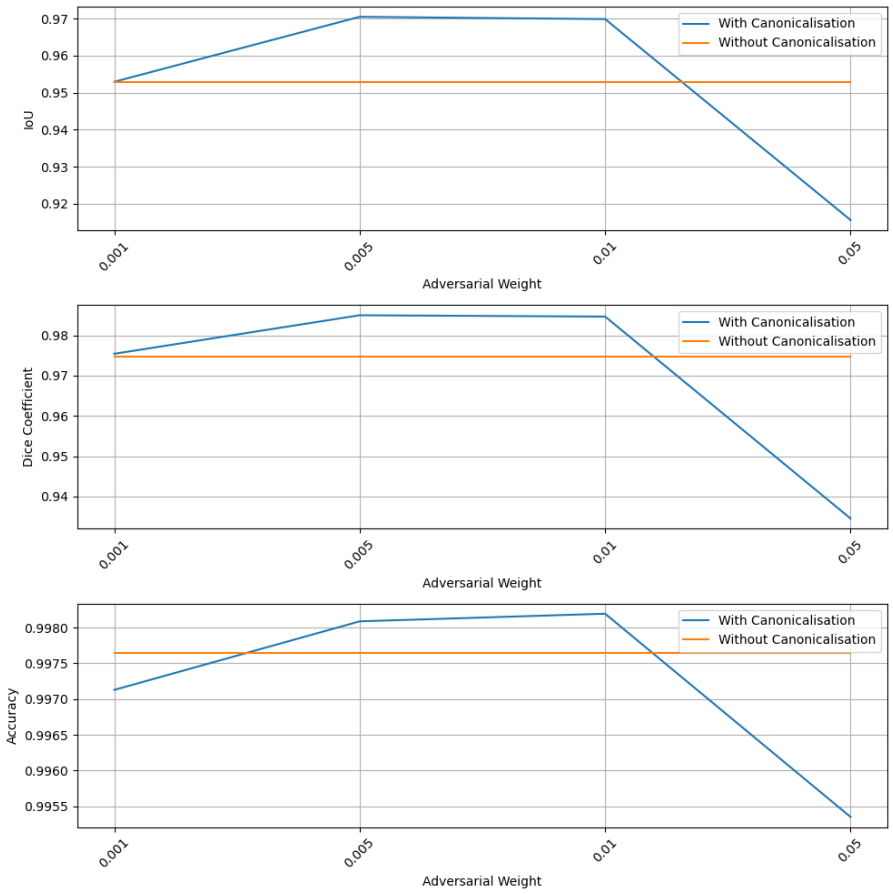


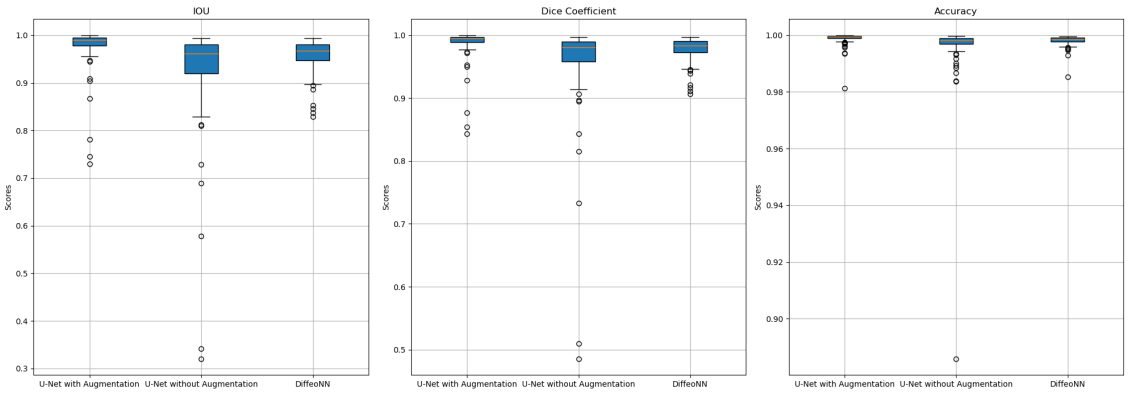
Figure 5.4: Comparison of different values for the adversarial weight λ_{adv} in the canonicalisation energy of DiffeoNN. We evaluate the performance of DiffeoNN using the Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy and compare it with applying only the inner U-Net without canonicalisation. With adversarial weights λ_{adv} in $[0.003, 0.02]$, the performance with canonicalisation is better than that without canonicalisation. The results for λ_{adv} between 0.005 and 0.01 are very similar. Only the accuracy is noticeably better for $\lambda_{\text{adv}} = 0.01$. Therefore, we use $\lambda_{\text{adv}} = 0.01$ for further experiments.

dataset X_E, while still maintaining a good regularisation of the transformation, we conduct hyperparameter tuning of the adversarial weight λ_{adv} for the energy term E_{adv}. We test values in the range $[0.001, 0.05]$ for the adversarial weight λ_{adv}.

We apply DiffeoNN to 20 images of X_{TE}^{test}, and use the mean Dice coefficient (Definition 5.2), Intersection-over-Union (IoU) (Definition 5.1), and pixel-wise accuracy (Definition 5.3) for evaluation. In an ablation experiment, the same segmentation U-Net is also applied directly to the input images without any canonicalisation (naïve approach). The results are visualised in Figure 5.4. One can see that the value of λ_{adv} had a large influence on the performance of DiffeoNN. The performance of the segmentation with canonicalisation is approximately better than without canonicalisation for adversarial weights in $[0.003, 0.02]$. To achieve an optimal segmentation, we choose the adversarial weight λ_{adv} as 0.01 for further experiments.

Model	IoU	Dice Coefficient	Accuracy
U-Net with Augmentation	0.9770	0.9878	0.9989
U-Net without Augmentation	0.9276	0.9582	0.9962
DiffeoNN	0.9571	0.9777	0.9981

(a) Mean Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy.



(b) Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy for DiffeoNN, an augmented U-Net, and the inner U-Net of DiffeoNN in the form of box plots.

Figure 5.5: Performance of DiffeoNN, an augmented U-Net, and the inner U-Net of DiffeoNN without augmentation (naïve approach) in comparison. We apply the methods to 100 images from the dataset X_{TE}^{test}. Subfigure 5.5a summarises the performances to the mean Dice coefficient (Definition 5.2), Intersection-over-Union (IoU) (Definition 5.1), and pixel-wise accuracy (Definition 5.3), while Subfigure 5.5b shows the scores in the form of box plots. DiffeoNN outperforms the segmentation of the naïve approach. When looking at the average, the augmented U-Net still performs best, but DiffeoNN has the least extreme outliers.

5.3 Benchmarking

To evaluate the performance of DiffeoNN, we conduct several experiments. In the previous section, we compared the results of DiffeoNN and the inner U-Net of DiffeoNN. We call the latter method the naïve approach, as it does not use any method to achieve diffeomorphism-equivariance.

To provide a comparison, we include an augmented U-Net, trained with standard data augmentation techniques to approximate deformation invariance. Data augmentation is another approach to achieve approximate diffeomorphism-equivariance (see Section 1.5). We train the augmented U-Net in the same way as the inner U-Net of DiffeoNN, setting the learning rate to $5 \cdot 10^{-5}$, the batch size to 2, and the number of epochs to 10. The dataset used to train the augmented U-Net is a combination of our synthetic datasets X_E and X_{TE}:

- training data: $X_{\text{aug}}^{\text{train}} = X_{TE}^{\text{train}} \cup X_E^{\text{train}}$ and $Y_{\text{aug}}^{\text{train}} = Y_{TE}^{\text{train}} \cup Y_E^{\text{train}}$ (contains 16 000 images/segmentations each),
- validation data: $X_{\text{aug}}^{\text{val}} = X_{TE}^{\text{val}} \cup X_E^{\text{val}}$ and $Y_{\text{aug}}^{\text{val}} = Y_{TE}^{\text{val}} \cup Y_E^{\text{val}}$ (contains 4 000 images/segmentations each).

After training the augmented U-Net, we evaluate and compare the performance of DiffeoNN, the augmented U-Net, and the inner U-Net of DiffeoNN (naïve approach): We

apply the three methods to 100 images from the dataset X_{TE}^{test}, and compute the Dice coefficient (Definition 5.2), Intersection-over-Union (IoU) (Definition 5.1), and pixel-wise accuracy (Definition 5.3). Figure 5.5 shows the results in the form of box plots (b) and the mean scores (a). DiffeoNN outperforms the naïve approach significantly, but, on average, its performance is still slightly lower than that of the augmented U-Net. Nevertheless, the box plots show that although DiffeoNN has slightly lower mean values, it also has less extreme outliers. This suggests that our approach is actually more robust than the augmented U-Net.

We assume that the nearly perfect performance of the augmented U-Net in the mean is based on the fact that our segmentation problem is relatively simple, and we are therefore able to cover its complexity by data augmentation. However, the data augmented U-Net needs large amounts of training data, whereas the other two methods can be trained on the smaller training dataset X_E. In our experiment, for example, the dataset for training the augmented U-Net is twice the size of the training set X_E, which is used for training our method. In addition, the data augmentation requires a much longer training time and much higher computational costs than the simple U-Net that we use in the other two methods. Furthermore, the results show that DiffeoNN improves the performance of the naïve U-Net, which indicates the effectiveness of our method. In Appendix A.3, we present some visual examples of DiffeoNN applied to images from X_{TE}^{test} and compare the results of DiffeoNN, the naïve U-Net, and the augmented U-Net visually.

5.4 Invariance of the Canonicalisation

A central assumption of the canonicalisation strategy is that the canonicalisation is approximately diffeomorphism-invariant, e.g., for an input $x \in X_E$ and a transformed input $g' \cdot x$ with an arbitrary diffeomorphism g it holds

$$\underset{g \in \mathcal{D}_{\text{SVF}}(\Omega)}{\arg\min}\ E_{\text{can}}(x, g) \approx \underset{g \in \mathcal{D}_{\text{SVF}}(\Omega)}{\arg\min}\ E_{\text{can}}(g' \cdot x, g). \tag{5.5}$$

To verify this property, we combine the canonical representations of images in X_E^{test} and their transformed counterpart in X_{TE}^{test}. Since the solution of the canonicalisation optimisation problem is not unique, we compare the results visually and by energy level.

We conduct the experiment on 20 image pairs $(x, g \cdot x) \in X_E^{\text{test}} \times X_{TE}^{\text{test}}$, where g is a randomly chosen transformation, as described in Section 5.1. We perform the canonicalisation step on both images. In addition, we compare the canonicalisation energies before performing the canonicalisation with the final canonicalisation energies. We present a few exemplary results in Figure 5.6. Additional visual results can be found in Appendix A.4. The canonicalised images look very similar to the training dataset X_E. This is also reflected in the energy levels. The energies of an input pair $(x, g \cdot x) \in X_E^{\text{test}} \times X_{TE}^{\text{test}}$ are very different, whereas the canonicalised pair $(x_c, (g \cdot x)_c)$ has approximately the same energy. The average difference between the energies of the 20 canonicalised pairs $(x_c, (g \cdot x)_c)$ is approximately 0.0197, which shows that the canonicalisation step is approximately invariant.

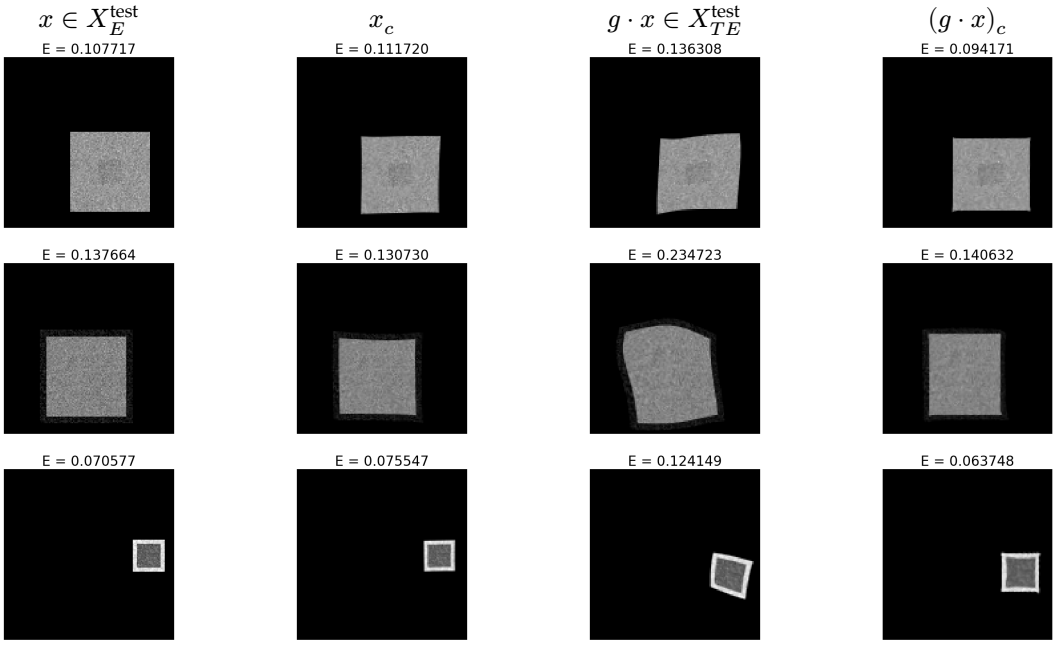


Figure 5.6: Pairs of $(x, g \cdot x) \in X_E^{\text{test}} \times X_{TE}^{\text{test}}$ and their canonicalisation with corresponding canonicalisation energies. **Column one** shows an input $x \in X_E^{\text{test}}$ and **column three** the randomly transformed input $g \cdot x \in X_{TE}^{\text{test}}$ (see Section 5.1). The **columns two and four** contain the canonicalised pair. The energy is shown above each image. While the energies of the input pairs are very different, the energies of the canonicalised pairs $(x_c, (g \cdot x)_c)$ are similar.

5.5 DiffeoNN for Lung Segmentation

To evaluate the performance of DiffeoNN beyond a synthetic dataset, we conduct an experiment on real-world data. For this, we use a dataset with chest X-ray images and their ground-truth lung segmentation from [68].

Dataset. The original dataset contains images and corresponding ground-truth segmentations into three different classes ("Non-Covid", "Covid", and "Non-Covid-Pneumonia"). We combine the images and corresponding ground-truth segmentations of the initial three classes into one dataset, which is then split into a training dataset (containing 234 images and corresponding segmentations), a validation dataset (containing 60 images and corresponding segmentations), and a test dataset (containing 60 images and corresponding segmentations). We then proceed as in Section 5.1.2 to create a dataset of diffeomorphically transformed images. To ensure that the transformation does not move the lungs out of the image domain, we check whether the ground-truth segmentation of the lung is within the image domain, and we randomly draw new transformations until this condition is fulfilled.

Network. We train the inner U-Net of DiffeoNN as well as the adversarial network and VAE with the same parameters as in Section 5.3. We set the weights in the canonicalisation energy E_{can} in (4.24) to

- $\lambda_{\text{VAE}} = 1 \cdot 10^{-4}$,

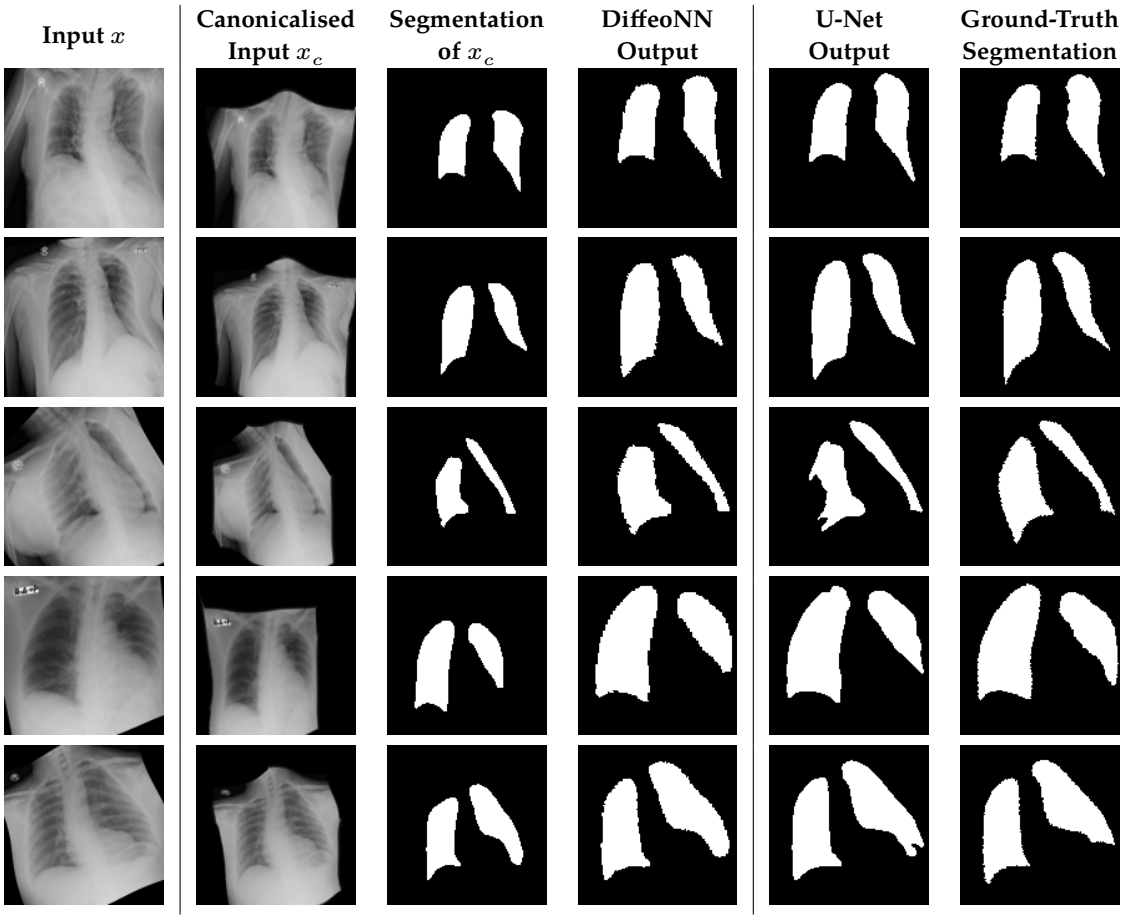


Figure 5.7: Lung segmentation of diffeomorphically transformed chest X-ray images from [68]. **Column one** shows the diffeomorphically transformed input images. In **columns two to four**, we show the results in the different steps of DiffeoNN (our approach): First, the canonicalised input images (**column two**), followed by their segmentation (**column three**), and the final output of DiffeoNN (**column four**). The output segmentation of the inner U-Net (naïve approach) is shown in **column five** and the ground-truth segmentation in **column six**. The canonicalisation step tends to align the shoulders to the horizontal axis of the images, while also shrinking the image slightly. The DiffeoNN segmentation results match or outperform the segmentation results of the inner U-Net.

- $\lambda_{\text{grad}} = 1$,
- $\lambda_{\text{jac}} = 10$,
- and $\lambda_{\text{adv}} = 1 \cdot 10^{-6}$.

The parameters were selected manually. Furthermore, we change the number of update steps in the gradient-based optimisation method for finding a canonicalising element to 1 000 steps.

Results. We apply DiffeoNN and the inner U-Net (naïve approach) to the 60 diffeomorphically transformed images from the test dataset (Figure 5.7). More examples can be found in Appendix A.5. In the canonicalised images, the shoulders tend to be more aligned to the horizontal axis. Furthermore, the canonicalisation step seems to shrink the images slightly. In most images, DiffeoNN outputs a segmentation that is comparable to or better than the naïve approach. This happens in approximately 80% of the evaluated images.

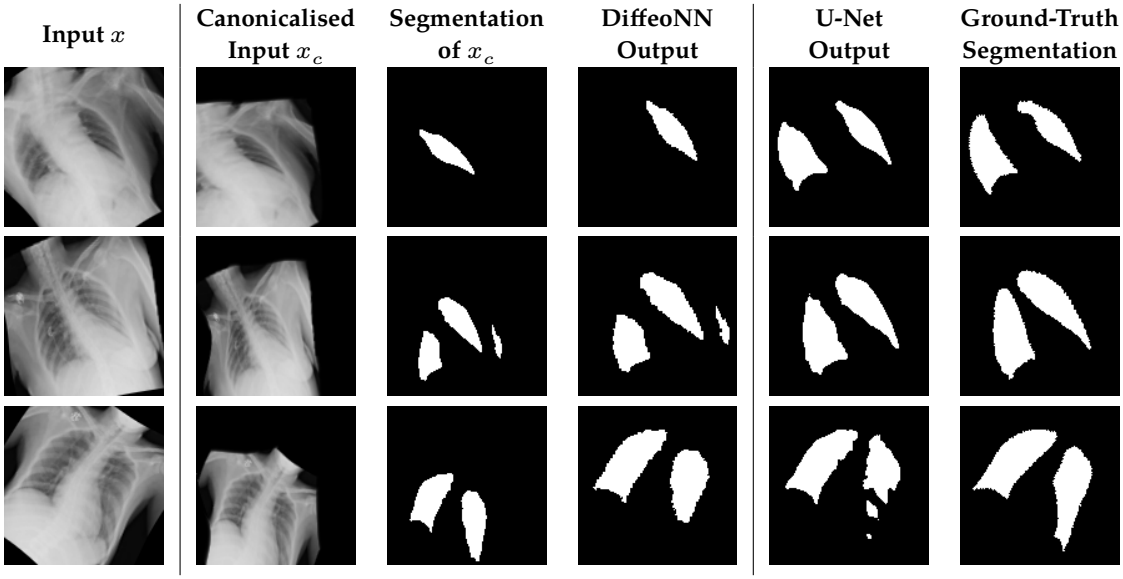


Figure 5.8: Examples of lung segmentation of diffeomorphically transformed chest X-ray images from [68], where the canonicalisation did not work well. The canonicalisation step tends to move the chest partly out of the image domain (**row one and three**) or shrink the image incoherently (**row two**). The former leads to incomplete segmentations, while the latter results in artifacts in the segmentation.

Model	IoU	Dice Coefficient	Accuracy
inner U-Net (naïve)	0.8293	0.8961	0.9638
DiffeoNN	0.7874	0.8668	0.9540

Figure 5.9: Performance of DiffeoNN (our approach) and the inner U-Net (naïve approach) on real-world data. We give the Intersection-over-Union (IoU), Dice coefficient, and pixel-wise accuracy, averaged over 60 images from the test dataset.

However, in some cases, the canonicalisation does not work properly, as visualised in Figure 5.8. In the examples in row one and three, the lung scan is partly moved out of the image domain in the canonicalisation step, resulting in incomplete segmentations. Another way for the canonicalisation to fail is that the image is shrunk incoherently, which creates artifacts in the following segmentation. The bad performance of DiffeoNN on some images is also reflected in the Intersection-over-Union (IoU) (Definition 5.1), Dice coefficient (Definition 5.2), and pixel-wise accuracy (Definition 5.3), which are lower for the DiffeoNN output than the U-Net output in most cases. The results are summarised in Figure 5.9.

We suspect that the canonicalisation did not work every time due to the manually chosen weights in the canonicalisation energy, which might be resolved by systematic hyperparameter tuning. In addition, we work on a relatively small dataset of 234 images. This might be the reason why the inner U-Net could not learn the full complexity of the lung segmentation in the first place. Examples in which the U-Net already struggles with segmentation on untransformed images are shown in Figure 5.10.

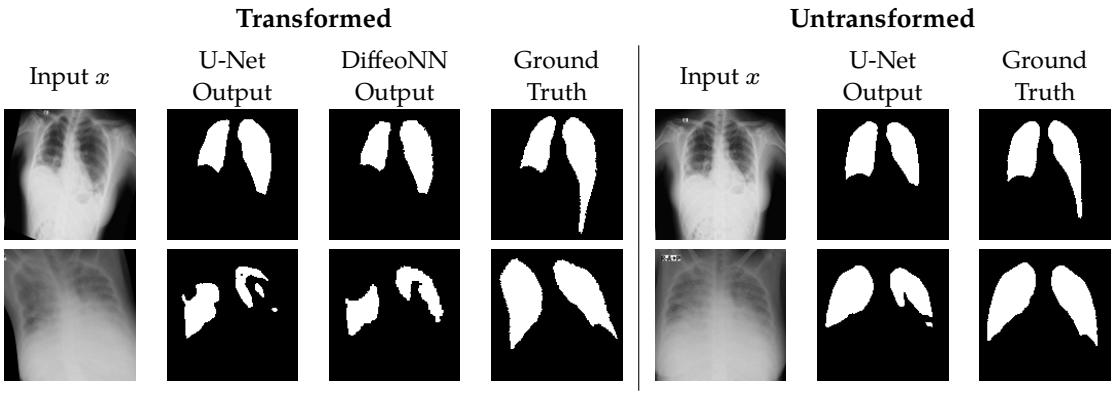


Figure 5.10: Examples of lung segmentation of diffeomorphically transformed chest X-ray images from [68], where the segmentation on the untransformed images is lacking. We present the results on the transformed image (**left**) and the results on the untransformed image (**right**). **Column one** shows the transformed image and **column four** their ground-truth segmentation. In the example in **row one**, none of the methods are able to segment the extension on the right side of the lung that can be seen in both ground-truth segmentations. The right side is also problematic in the example in **row two**. Here, the segmentation output of the U-Net is already imperfect for the untransformed image. The results on the transformed image are worse; however, DiffeoNN performs better than the inner U-Net without canonicalisation there. We suspect that the already imperfect segmentation on the untransformed images is the reason for the imperfect segmentation on the transformed image.

6

Conclusion and Discussion

In this work, we have developed and analysed a strategy for constructing diffeomorphism-equivariant neural networks, using an energy-based canonicalisation strategy inspired by the LieLAC framework [72]. We combined ideas from Lie group theory, differentiable image registration, and energy-based modelling. Our diffeomorphism-equivariant network needs to be trained only on a simple and relatively small dataset X_{TE} (without augmentation), while still achieving approximate equivariance. Furthermore, the proposed setup allows us to turn pretrained networks diffeomorphism-equivariant, as the inner task-performing network remains unchanged.

To underline the advantages of energy-based canonicalisation, in Chapter 3, we provided a theoretical analysis of its generalisation and derived a bound on the expected loss under mild assumptions (Theorem 3.2.3). A crucial assumption for this analysis is the existence of an orbit measure on the underlying data manifold. In future work, a natural next step would be to relax this assumption.

We translated the theoretical insights into a practical neural network architecture designed to achieve diffeomorphism-equivariance for an exemplary segmentation task. In the experimental evaluation, we verified the effectiveness of the proposed approach on a synthetic dataset containing nested squares. We created a synthetic training dataset and a diffeomorphically transformed test dataset. After hyperparameter tuning, we compared our method DiffeoNN to an augmented U-Net and the inner U-Net of DiffeoNN (the naïve approach) on the test dataset (Section 5.3). Our method achieved a significantly better segmentation than the naïve approach and a nearly comparable segmentation to the augmented U-Net. In addition, DiffeoNN showed fewer drastic outliers than the other two methods, indicating higher robustness. We also verified the approximate invariance of the canonicalisation step (Section 5.4).

Furthermore, we applied DiffeoNN to the task of lung segmentation on real-world chest X-ray images (Section 5.5). In many cases, our network matched or outperformed the naïve U-Net approach, demonstrating good generalisation beyond synthetic data.

While the proposed approach already achieves promising results, several open challenges remain and offer interesting opportunities for future research. Firstly, the canonicalisation step is still relatively computationally expensive, and future work could focus on making this process more efficient, either by adjusting the canonicalisation energy or by improving the optimisation strategy. Secondly, our current formulation focuses on

SVF-based diffeomorphisms, which do not include discrete transformations such as flips. Moreover, in the implementation, we focused solely on segmentation as an example application, and our real-world training dataset for lung segmentation was relatively small, leading to a suboptimal inner U-Net performance. We also assume that the weighting in the canonicalisation energy could be further optimised for the real-world setting.

A natural next step is to conduct systematic hyperparameter tuning and to explore alternative weighting strategies in the canonicalisation energy. Repeating the lung segmentation experiment with a larger training dataset could provide further insights. In addition, investigating alternative parametrisations of diffeomorphisms that include discrete transformations could make the canonicalisation more expressive. Finally, future work could explore more complex segmentation tasks, stronger transformations, or other real-world datasets, and extend the approach to a broader range of computer vision applications: A core strength of the proposed approach is that such extensions can be implemented in a flexible and straightforward way by simply replacing the inner task-specific network with an off-the-shelf component.

Bibliography

[1] Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A., Bambrick, J., et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. In: *Nature* 630:493–500, May 2024. DOI: 10.1038/s41586-024-07487-w.

[2] Adhikari, A. and Adhikari, M. *Basic Topology 1: Metric Spaces and General Topology*. Jan. 2022. ISBN: 978-981-16-6508-0. DOI: 10.1007/978-981-16-6509-7.

[3] Adhikari, A. and Adhikari, M. Basic Topology 2, Topological Groups, Topology of Manifolds and Lie Groups. In: Jan. 2022. DOI: 10.1007/978-981-16-6577-6.

[4] Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook*. Cham, Switzerland: Springer, 2018. ISBN: 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0.

[5] Arguillere, S., Trélat, E., Trouvé, A., and Younes, L. *Shape deformation analysis from the optimal control viewpoint*. 2014. arXiv: 1401.0661 [math.OC]. URL: https://arxiv.org/abs/1401.0661.

[6] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein Generative Adversarial Networks. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html.

[7] Arsigny, V., Commowick, O., Pennec, X., and Ayache, N. A Log-Euclidean Framework for Statistics on Diffeomorphisms. In: vol. 9. Feb. 2006, pp. 924–31. ISBN: 978-3-540-44707-8. DOI: 10.1007/11866565_113.

[8] Ashburner, J. A fast diffeomorphic image registration algorithm. In: *NeuroImage* 38(1):95–113, 2007. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2007.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1053811907005848.

[9] Azad, R., Aghdam, E. K., Rauland, A., Jia, Y., Avval, A. H., Bozorgpour, A., Karimijafarbigloo, S., Cohen, J. P., Adeli, E., and Merhof, D. Medical Image Segmentation Review: The Success of U-Net. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46(12):10076–10095, 2024. DOI: 10.1109/TPAMI.2024.3435571.

[10] Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., and Dalca, A. V. VoxelMorph: A Learning Framework for Deformable Medical Image Registration. In: *IEEE Transactions on Medical Imaging* 38(8):1788–1800, 2019. DOI: 10.1109/TMI.2019.2897538.

[11] Beekman, C., Beek, S. van, Stam, J., Sonke, J., and Remeijer, P. Improving predictive CTV segmentation on CT and CBCT for cervical cancer by diffeomorphic registration of a prior. In: *Medical Physics* 49, Feb. 2022. DOI: 10.1002/mp.15421.

[12] Beg, M. F., Miller, M., Trouvé, A., and Younes, L. Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms. In: *International Journal of Computer Vision* 61:139–157, Feb. 2005. DOI: 10.1023/B:VISI.0000043755.93987.aa.

[13] Bekkers, E. J. *B-Spline CNNs on Lie Groups*. 2021. arXiv: 1909.12057 [cs.LG]. URL: https://arxiv.org/abs/1909.12057.

[14] Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A., Pluim, J. P., and Duits, R. *Roto-Translation Covariant Convolutional Networks for Medical Image Analysis*. 2018. arXiv: 1804.03393 [cs.CV]. URL: https://arxiv.org/abs/1804.03393.

[15] Bietti, A., Venturi, L., and Bruna, J. *On the Sample Complexity of Learning under Invariance and Geometric Stability*. 2021. arXiv: 2106.07148 [stat.ML]. URL: https://arxiv.org/abs/2106.07148.

[16] Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press, 1995. ISBN: 978-0198538646.

[17] Bogachev, V. *Measure Theory*. Vol. 1. Jan. 2007, pp. 1–575. ISBN: 978-3-540-34513-8. DOI: 10.1007/978-3-540-34514-5.

[18] Bogachev, V. I. *Measure Theory, Volume II*. Springer Monographs in Mathematics. See Theorem 10.4.3 (Disintegration Theorem). Berlin, Heidelberg: Springer, 2007. ISBN: 978-3-540-34515-9.

[19] Bostelmann, J., Gildemeister, O., and Lellmann, J. *Stationary Velocity Fields on Matrix Groups for Deformable Image Registration*. 2024. arXiv: 2410.10997 [cs.CV]. URL: https://arxiv.org/abs/2410.10997.

[20] Brehmer, J., Behrends, S., Haan, P. de, and Cohen, T. *Does equivariance matter at scale?* 2025. arXiv: 2410.23179 [cs.LG]. URL: https://arxiv.org/abs/2410.23179.

[21] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG]. URL: https://arxiv.org/abs/2104.13478.

[22] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., and Kalinin, A. A. Albumentations: Fast and Flexible Image Augmentations. In: *Information* 11(2):125, Feb. 2020. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: http://dx.doi.org/10.3390/info11020125.

[23] Chen, X., Wang, X., Zhang, K., Fung, K.-M., Thai, T. C., Moore, K., Mannel, R. S., Liu, H., Zheng, B., and Qiu, Y. Recent advances and clinical applications of deep learning in medical image analysis. In: *Medical Image Analysis* 79, 2022. ISSN: 1361-8415. DOI: 10.1016/j.media.2022.102444. URL: http://dx.doi.org/10.1016/j.media.2022.102444.

[24] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[25] Cohen, T., Geiger, M., Koehler, J., and Welling, M. Spherical CNNs. In: Jan. 2018. DOI: 10.48550/arXiv.1801.10130.

[26] Cohen, T. S. and Welling, M. *Group Equivariant Convolutional Networks*. 2016. arXiv: 1602.07576 [cs.LG]. URL: https://arxiv.org/abs/1602.07576.

[27] Costa, N. D., Pförtner, M., and Cockayne, J. *Constructive Disintegration and Conditional Modes*. 2025. arXiv: 2508.00617 [math.ST]. URL: https://arxiv.org/abs/2508.00617.

[28] Dalca, A. V., Balakrishnan, G., Guttag, J., and Sabuncu, M. R. Unsupervised learning of probabilistic diffeomorphic registration for images and surfaces. In: *Medical image analysis* 57:226–236, 2019.

[29] Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., Casas, D., et al. Magnetic control of tokamak plasmas through deep reinforcement learning. In: *Nature* 602:414–419, Feb. 2022. DOI: 10.1038/s41586-021-04301-9.

[30] DeVries, T. and Taylor, G. W. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017. arXiv: 1708.04552 [cs.CV]. URL: https://arxiv.org/abs/1708.04552.

[31] Dieleman, S., Willett, K. W., and Dambre, J. Rotation-invariant convolutional neural networks for galaxy morphology prediction. In: *Monthly Notices of the Royal Astronomical Society* 450(2):1441–1459, Apr. 2015. ISSN: 0035-8711. DOI: 10.1093/mnras/stv632. URL: http://dx.doi.org/10.1093/mnras/stv632.

[32] Dym, N., Lawrence, H., and Siegel, J. W. *Equivariant Frames and the Impossibility of Continuous Canonicalization*. 2024. arXiv: 2402.16077 [cs.LG]. URL: https://arxiv.org/abs/2402.16077.

[33] Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. *Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data*. 2020. arXiv: 2002.12880 [stat.ML]. URL: https://arxiv.org/abs/2002.12880.

[34] Fukushima, K. and Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. In: *Pattern Recognit.* 15:455–469, 1982. URL: https://api.semanticscholar.org/CorpusID:2357880.

[35] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[36] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. Cambridge, MA: MIT Press, 2016. ISBN: 978-0262035613. URL: https://www.deeplearningbook.org/.

[37] Hauberg, S., Freifeld, O., Larsen, A. B. L., III, J. W. F., and Hansen, L. K. *Dreaming More Data: Class-dependent Distributions over Diffeomorphisms for Learned Data Augmentation*. 2016. arXiv: 1510.02795 [cs.CV]. URL: https://arxiv.org/abs/1510.02795.

[38] Haykin, S. *Neural Networks: A Comprehensive Foundation*. 2nd. Upper Saddle River, NJ: Prentice Hall, 1998. ISBN: 978-0132733502.

[39] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[40] Kaba, S.-O., Mondal, A. K., Zhang, Y., Bengio, Y., and Ravanbakhsh, S. *Equivariance with Learned Canonicalization Functions*. 2023. arXiv: 2211.06489 [cs.LG]. URL: https://arxiv.org/abs/2211.06489.

[41] Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[42] Kingma, D. P. and Welling, M. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[43] Kirillov Jr, A. Bibliography. In: *An Introduction to Lie Groups and Lie Algebras*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008, pp. 216–219.

[44] Kondor, R. and Trivedi, S. *On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups*. 2018. arXiv: 1802.03690 [stat.ML]. URL: https://arxiv.org/abs/1802.03690.

[45] Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. In: *Neural Information Processing Systems* 25, Jan. 2012. DOI: 10.1145/3065386.

[46] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[47] Kuang, D. *On Reducing Negative Jacobian Determinant of the Deformation Predicted by Deep Registration Networks*. 2019. arXiv: 1907.00068 [cs.CV]. URL: https://arxiv.org/abs/1907.00068.

[48] Kumar, H., Parada-Mayorga, A., and Ribeiro, A. *Lie Group Algebra Convolutional Filters*. 2024. arXiv: 2305.04431 [eess.SP]. URL: https://arxiv.org/abs/2305.04431.

[49] Lang, S. Manifolds. In: *Differential Manifolds*. New York, NY: Springer US, 1985, pp. 21–40. ISBN: 978-1-4684-0265-0. DOI: 10.1007/978-1-4684-0265-0_2. URL: https://doi.org/10.1007/978-1-4684-0265-0_2.

[50] Lawrence, H., Portilheiro, V., Zhang, Y., and Kaba, S.-O. *Improving Equivariant Networks with Probabilistic Symmetry Breaking*. 2025. arXiv: 2503.21985 [cs.LG]. URL: https://arxiv.org/abs/2503.21985.

[51] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. In: *Neural Computation* 1(4):541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[52] Lee, J. M. Riemannian Manifolds: An Introduction to Curvature. In: 1997. URL: https://api.semanticscholar.org/CorpusID:119659969.

[53] Lee, J. M. Smooth Manifolds. In: *Introduction to Smooth Manifolds*. New York, NY: Springer New York, 2003, pp. 1–29. ISBN: 978-0-387-21752-9. DOI: 10.1007/978-0-387-21752-9_1. URL: https://doi.org/10.1007/978-0-387-21752-9_1.

[54] Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., and Benediktsson, J. A. Deep Learning for Hyperspectral Image Classification: An Overview. In: *IEEE Transactions on Geoscience and Remote Sensing* 57(9):6690–6709, 2019. DOI: 10.1109/TGRS.2019.2907932.

[55] Liu, Y., Chen, J., Wei, S., Carass, A., and Prince, J. *On Finite Difference Jacobian Computation in Deformable Image Registration*. 2023. arXiv: 2212.06060 [eess.IV]. URL: https://arxiv.org/abs/2212.06060.

[56] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[57] Lunz, S., Öktem, O., and Schönlieb, C.-B. *Adversarial Regularizers in Inverse Problems*. 2019. arXiv: 1805.11572 [cs.CV]. URL: https://arxiv.org/abs/1805.11572.

[58] Marcus, G. *Deep Learning: A Critical Appraisal*. 2018. arXiv: 1801.00631 [cs.AI]. URL: https://arxiv.org/abs/1801.00631.

[59] Merchant, A., Batzner, S., Schoenholz, S., Aykol, M., Cheon, G., and Cubuk, E. Scaling deep learning for materials discovery. In: *Nature* 624:1–6, Nov. 2023. DOI: 10.1038/s41586-023-06735-9.

[60] Miller, M. Computational anatomy: Shape, growth, and atrophy comparison via diffeomorphisms. In: *NeuroImage* 23 Suppl 1:S19–33, Feb. 2004. DOI: 10.1016/j.neuroimage.2004.07.021.

[61] Milnor, J. W. Remarks on infinite dimensional Lie groups. In: *Relativity, Groups and Topology II.*, 1984.

[62] Moreno-Barea, F., Strazzera, F., Jerez, J., Urda, D., and Franco, L. Forward Noise Adjustment Scheme for Data Augmentation. In: Nov. 2018. DOI: 10.1109/SSCI.2018.8628917.

[63] Mukherjee, S., Dittmer, S., Shumaylov, Z., Lunz, S., Öktem, O., and Schönlieb, C.-B Data-Driven Convex Regularizers for Inverse Problems. In: Apr. 2024, pp. 13386–13390. DOI: 10.1109/ICASSP48485.2024.10447719.

[64] Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. *Relational Pooling for Graph Representations*. 2019. arXiv: 1903.02541 [cs.LG]. URL: https://arxiv.org/abs/1903.02541.

[65] Omori, H. *Infinite-Dimensional Lie Groups*. Vol. 158. Translations of Mathematical Monographs. Providence, RI: American Mathematical Society, 1997. ISBN: 978-0-8218-0289-9. DOI: 10.1090/mmono/158. URL: https://www.ams.org/books/mmono/158/.

[66] Puny, O., Atzmon, M., Ben-Hamu, H., Misra, I., Grover, A., Smith, E. J., and Lipman, Y. *Frame Averaging for Invariant and Equivariant Network Design*. 2022. arXiv: 2110.03336 [cs.LG]. URL: https://arxiv.org/abs/2110.03336.

[67] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[68] RSUA, R. *RSUA Chest X-Ray Dataset*. Version V1. 2023. DOI: 10.17632/2jg8vfdmpm.1. URL: https://doi.org/10.17632/2jg8vfdmpm.1.

[69] Rudin, W. Principles of mathematical analysis. In: *3rd ed.*, 1976.

[70] Sheikhjafari, A., Krishnaswamy, D., Noga, M., Ray, N., and Punithakumar, K. Deep Learning Based Parametrization of Diffeomorphic Image Registration for the Application of Cardiac Image Segmentation. In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2022, pp. 1164–1169. DOI: 10.1109/BIBM55620.2022.9994849.

[71] Shorten, C. and Khoshgoftaar, T. A survey on Image Data Augmentation for Deep Learning. In: *Journal of Big Data* 6, July 2019. DOI: 10.1186/s40537-019-0197-0.

[72] Shumaylov, Z., Zaika, P., Rowbottom, J., Sherry, F., Weber, M., and Schönlieb, C.-B. *Lie Algebra Canonicalization: Equivariant Neural Operators under arbitrary Lie Groups*. Oct. 2024. DOI: 10.48550/arXiv.2410.02698.

[73] Simard, P., Steinkraus, D., and Platt, J. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: Jan. 2003, pp. 958–962. DOI: 10.1109/ICDAR.2003.1227801.

[74] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. In: *Advances in neural information processing systems* 33:7462–7473, 2020.

[75] Sotiras, A., Davatzikos, C., and Paragios, N. Deformable Medical Image Registration: A Survey. In: *IEEE Transactions on Medical Imaging* 32(7):1153–1190, 2013. DOI: 10.1109/TMI.2013.2265603.

[76] Souza, J. C., Bandeira Diniz, J. O., Ferreira, J. L., França da Silva, G. L., Corrêa Silva, A., and de Paiva, A. C. An automatic method for lung segmentation and reconstruction in chest X-ray using deep neural networks. In: *Computer Methods and Programs in Biomedicine* 177:285–296, 2019. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2019.06.005. URL: https://www.sciencedirect.com/science/article/pii/S0169260719303517.

[77] Sun, C., Shrivastava, A., Singh, S., and Gupta, A. *Revisiting Unreasonable Effectiveness of Data in Deep Learning Era*. 2017. arXiv: 1707.02968 [cs.CV]. URL: https://arxiv.org/abs/1707.02968.

[78] Tahmasebi, B. and Jegelka, S. Generalization Bounds for Canonicalization: A Comparative Study with Group Averaging. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=n0lXaskyk5.

[79] Tahmasebi, B. and Jegelka, S. Generalization Bounds for Canonicalization: A Comparative Study with Group Averaging. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=n0lXaskyk5.

[80] Tahmasebi, B. and Jegelka, S. Regularity in Canonicalized Models: A Theoretical Perspective. In: *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*. Ed. by Y. Li, S. Mandt, S. Agrawal, and E. Khan. Vol. 258. Proceedings of Machine Learning Research. PMLR, 2025, pp. 4789–4797. URL: https://proceedings.mlr.press/v258/tahmasebi25a.html.

[81] Trouvé, A. Diffeomorphisms groups and pattern matching in image analysis. In: *International journal of computer vision* 28(3):213–221, 1998.

[82] Varadarajan, V. S. Lie Groups and Lie Algebras. In: *Lie Groups, Lie Algebras, and Their Representations*. New York, NY: Springer New York, 1984, pp. 41–148. ISBN: 978-1-4612-1126-6. DOI: 10.1007/978-1-4612-1126-6_2. URL: https://doi.org/10.1007/978-1-4612-1126-6_2.

[83] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. In: *Computational Intelligence and Neuroscience* 2018:1–13, Feb. 2018. DOI: 10.1155/2018/7068349.

[84] Wang, T., Lei, Y., Fu, Y., Curran, W. J., Liu, T., and Yang, X. *Medical Imaging Synthesis using Deep Learning and its Clinical Applications: A Review*. 2020. arXiv: 2004.10322 [physics.med-ph]. URL: https://arxiv.org/abs/2004.10322.

[85] Wirgin, A. *The inverse crime*. 2004. arXiv: math-ph/0401050 [math-ph]. URL: https://arxiv.org/abs/math-ph/0401050.

[86] Younes, L. *Shapes and Diffeomorphisms*. Vol. 171. Jan. 2010. ISBN: 978-3-642-12054-1. DOI: 10.1007/978-3-642-12055-8.

[87] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. *Understanding deep learning requires rethinking generalization*. 2017. arXiv: 1611.03530 [cs.LG]. URL: https://arxiv.org/abs/1611.03530.

[88] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. Object detection with deep learning: A review. In: *IEEE transactions on neural networks and learning systems* 30(11):3212–3232, 2019.

[89] Zimmer, V. Bildregistrierung unter Verwendung von Lie-Gruppen. In: 2011. URL: https://www.mic.uni-luebeck.de/fileadmin/mic/publications/StudentProjects/MasterTheses/2011-Zimmer-Masterarbeit.pdf.

A

Appendix

A.1 Artificial Neural Networks

An artificial neural network (ANN), also known as a neural network (NN), is a machine learning method that is inspired by the structure and function of the human brain. Its architecture is based on interconnected nodes called (artificial) neurons, which are organised into layers. A signal x propagates from the *input layer* through *hidden layers* to the final *output layer*. Typically, each neuron from one layer is connected to all neurons in the following layer. Such an ANN is called *fully connected*. The connections have associated weights, which are optimised during training. In this section, we introduce some of the main principles and components of ANNs following [36, 16, 38, 4]. We focus on *convolutional neural networks* (CNNs; Section A.1.1), typical layers (Section A.1.2), and activation functions (Section A.1.3).

A basic type of ANN is a *feedforward neural network* (FNN). The information flows only in one direction – forward from input to output – and the connections between nodes do not form cycles.

Example A.1. Let $x \in \mathbb{R}^d$ be an input vector, $y \in \mathbb{R}^m$ the corresponding network output, $A_{\theta_1} \in \mathbb{R}^{k\times d}$ and $A_{\theta_2} \in \mathbb{R}^{m\times k}$ the weight matrices of the first and second layer, and $b_{\theta_1} \in \mathbb{R}^k$, $b_{\theta_2} \in \mathbb{R}^m$ the corresponding bias vectors. Let $f_1 : \mathbb{R}^k \to \mathbb{R}^k$ and $f_2 : \mathbb{R}^m \to \mathbb{R}^m$ denote activation functions (for a detailed introduction, see Section A.1.3). A feedforward neural network with one hidden layer can then be written as

$$y = f_2(A_{\theta_2}(f_1(A_{\theta_1}x + b_{\theta_1})) + b_{\theta_2}). \tag{A.1}$$

As a concrete example, we consider an input $x = (x^{(1)}, x^{(2)}, x^{(3)})^\top \in \mathbb{R}^3$ and a network with two weight matrices $A_{\theta_1} = \left(a_{ij}^{(1)}\right)_{i=1,2;j=1,2,3} \in \mathbb{R}^{2\times 3}$ and $A_{\theta_2} = \left(a_{ij}^{(2)}\right)_{i,j=1}^{2} \in \mathbb{R}^{2\times 2}$ as well as two bias vectors $b_{\theta_1} \in \mathbb{R}^3$ and $b_{\theta_2} \in \mathbb{R}^2$. The hidden layer is given by

$$h = f_1(A_{\theta_1}x + b_{\theta_1}), \tag{A.2}$$

and the final network output by

$$y = f_2(A_{\theta_2}h + b_{\theta_2}). \tag{A.3}$$

This is visualised in Figure A.1.

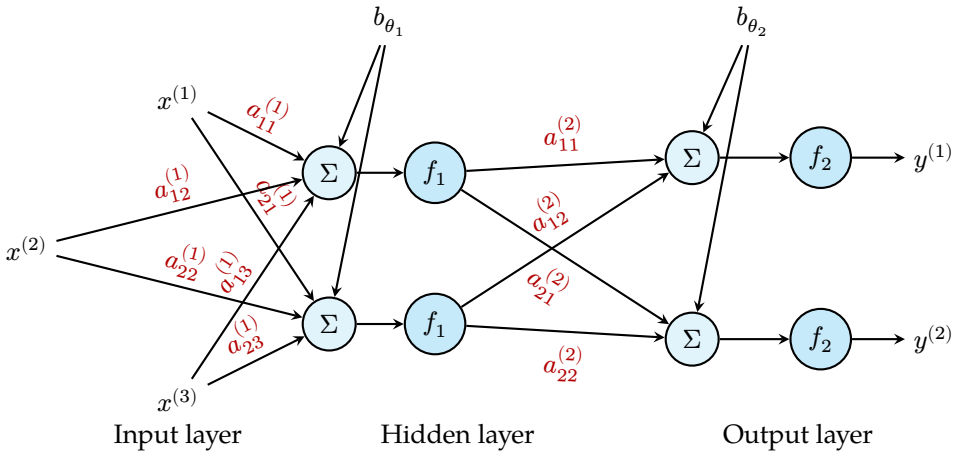


Figure A.1: Feedforward neural network with one hidden layer. Each connection carries a weight $a_{ij}^{(1)}$ or $a_{ij}^{(2)}$. Linear combinations (Σ) are followed by bias addition (b_{θ_1} and b_{θ_2}) and activation functions f_1, f_2 to produce the outputs $y^{(1)}, y^{(2)}$.

This example can be extended to more hidden layers by recursively carrying on with the structure above. In general, artificial neural networks belong to the class of *learning-based methods*. Given training samples $\{x_1, \dots, x_n\}$ with $x_i \in \mathbb{R}^k$ and corresponding ground-truth labels $\{y_1^{gt}, \dots, y_n^{gt}\}$ with $y_i^{gt} \in \mathbb{R}^l$, a neural network defines a mapping

$$F_\theta(x) = f_m(b_{\theta_m} + A_{\theta_m} f_{m-1}(b_{\theta_{m-1}} + A_{\theta_{m-1}} f_{m-2}(\dots f_1(b_{\theta_1} + A_{\theta_1} x) \dots))) \tag{A.4}$$

where A_{θ_i} and b_{θ_i} denote the weight matrices and bias vectors of the i-th layer, and the f_i are activation functions (see Section A.1.3). Depending on the network type, A_{θ_i} can represent fully connected or convolutional transformations, among others.

The parameters $\theta = (\theta_i)_{i=1}^m$ are optimised during training by minimising a *loss function* L, which, for example, measures the difference between the predicted and target outputs:

$$\theta^* \in \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} L(F_\theta(x_i), y_i^{gt}) \tag{A.5}$$

Typical loss functions include the mean squared error and the cross-entropy loss, depending on the application. The optimisation is usually performed using gradient-based methods such as stochastic gradient descent, where gradients are computed via *backpropagation*.

After training, the model parameters θ^* are fixed, and new, unseen inputs x_{new} are processed by a simple forward pass:

$$y_{\text{new}} = F_{\theta^*}(x_{\text{new}}). \tag{A.6}$$

A.1.1 Convolutional Neural Networks

A commonly used type of feedforward neural network, especially in image computing, is the convolutional neural network (CNN). There, features are learned through kernel

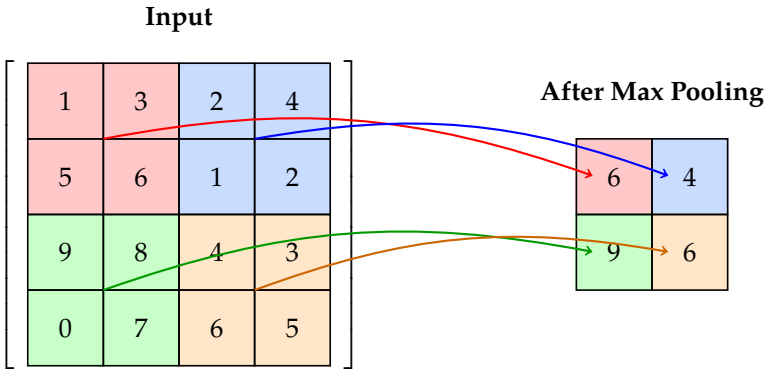


Figure A.2: Example of a 2×2 max pooling on a 4×4 input. Each coloured region is pooled into a single maximum value.

optimisation. Simpler features like edges are usually learned in earlier layers, while more abstract patterns are derived in deeper layers. Instead of having fully connected layers, CNNs have *convolutional layers* to learn spatial features. The layers are typically followed by *activation functions* and *pooling layers* to reduce spatial information and achieve invariance.

A convolutional layer is used to extract local features, such as edges or shapes. This is done by applying filters (kernels) to the input data, producing a feature map. Kernels (filters) are small weight matrices that are learned during training. The 2D convolution operation is a simple discrete convolution:

Definition A.2 (Discrete convolution). Let $x \in \mathbb{R}^{h_1 \times h_2}$ be an input and $w \in \mathbb{R}^{k_1 \times k_2}$ be a kernel. Then, the discrete convolution between x and w is defined as

$$y_{i,j} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} x_{i+m,j+n} \cdot w_{m,n}. \tag{A.7}$$

The ranges of the indices i and j vary depending on the *stride* and the *padding* strategy.

A convolution operation is similar to sliding a small window (the kernel) over the input and multiplying the values of the kernel and input at every position they overlap. The padding strategy and the stride specify how the edges of the input are handled, and how the kernel moves across the input. They are used to control the output size and computational cost.

A.1.2 Pooling Layers

Another way to control output size and computational cost in CNNs is to use pooling layers. They reduce the spatial dimensions (downsample) by summarising local regions. Commonly used pooling strategies are max pooling and average pooling.

Example A.3 (Max pooling). Let $x \in \mathbb{R}^{2h_1 \times 2h_2}$ be the input on which we apply 2×2 max pooling. The output $y \in \mathbb{R}^{h_1 \times h_2}$ is computed as follows:

$$y_{i,j} = \max\{x_{2i,2j}, x_{2i+1,2j}, x_{2i,2j+1}, x_{2i+1,2j+1}\} \tag{A.8}$$

with $(i, j) \in \{0, \dots, h_1 - 1\} \times \{0, \dots, h_2 - 1\}$.

A visualisation of max pooling is shown in Figure A.2.

A.1.3 Activation Functions

Activation functions are typically non-linear functions applied within the network. They allow the network to learn more complex patterns. Commonly used activation functions include the *rectified linear unit* (*ReLU*), the *logistic* function, and the *softmax* function.

Definition A.4 (Rectified Linear Unit (ReLU)). The *Rectified Linear Unit* (*ReLU*) is defined as the function $\mathrm{ReLU} : \mathbb{R}^n \to \mathbb{R}^n_{\geq 0}$, which maps an input $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ to its non-negative part:

$$(\mathrm{ReLU}(x))_i := \max\{0, x_i\}, \quad \text{for all } i = 1, \dots, n. \tag{A.9}$$

ReLU is widely used due to its simplicity and efficiency, though it is not differentiable at 0, and is unbounded. To enable backpropagation, implementations define the gradient at $x = 0$; in PyTorch, it is set to 0.

In contrast, the logistic and softmax functions are bounded and in $\mathcal{C}^\infty$.

Definition A.5 (Logistic activation function). The logistic activation function (also called the *sigmoid* function) maps a vector $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ to $(0, 1)^n$:

$$f(x)_i := \frac{1}{1 + \mathrm{e}^{-x_i}}. \tag{A.10}$$

for all $i \in \{1, \dots, n\}$. It is especially useful for binary classification tasks.

For multi-class classification tasks, the softmax function is more suitable.

Definition A.6 (Softmax function). The softmax function $\sigma : \mathbb{R}^n \to (0, 1)^n$ maps a vector $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ to a probability distribution. It is defined as

$$\sigma(x)_i := \frac{\mathrm{e}^{x_i}}{\sum_{j=1}^n \mathrm{e}^{x_j}} \tag{A.11}$$

for all $i \in \{1, \dots, n\}$.

A less common activation function is the *sinusoidal activation function*:

Definition A.7. The sinusoidal activation function $\mathrm{sine} : \mathbb{R} \to [-1, 1]$ is defined as

$$\mathrm{sine}(x) = \sin(\omega_0 x), \tag{A.12}$$

where $\omega_0 \in \mathbb{R}$ denotes the frequency of the sine wave.

A.2 Synthetic Dataset

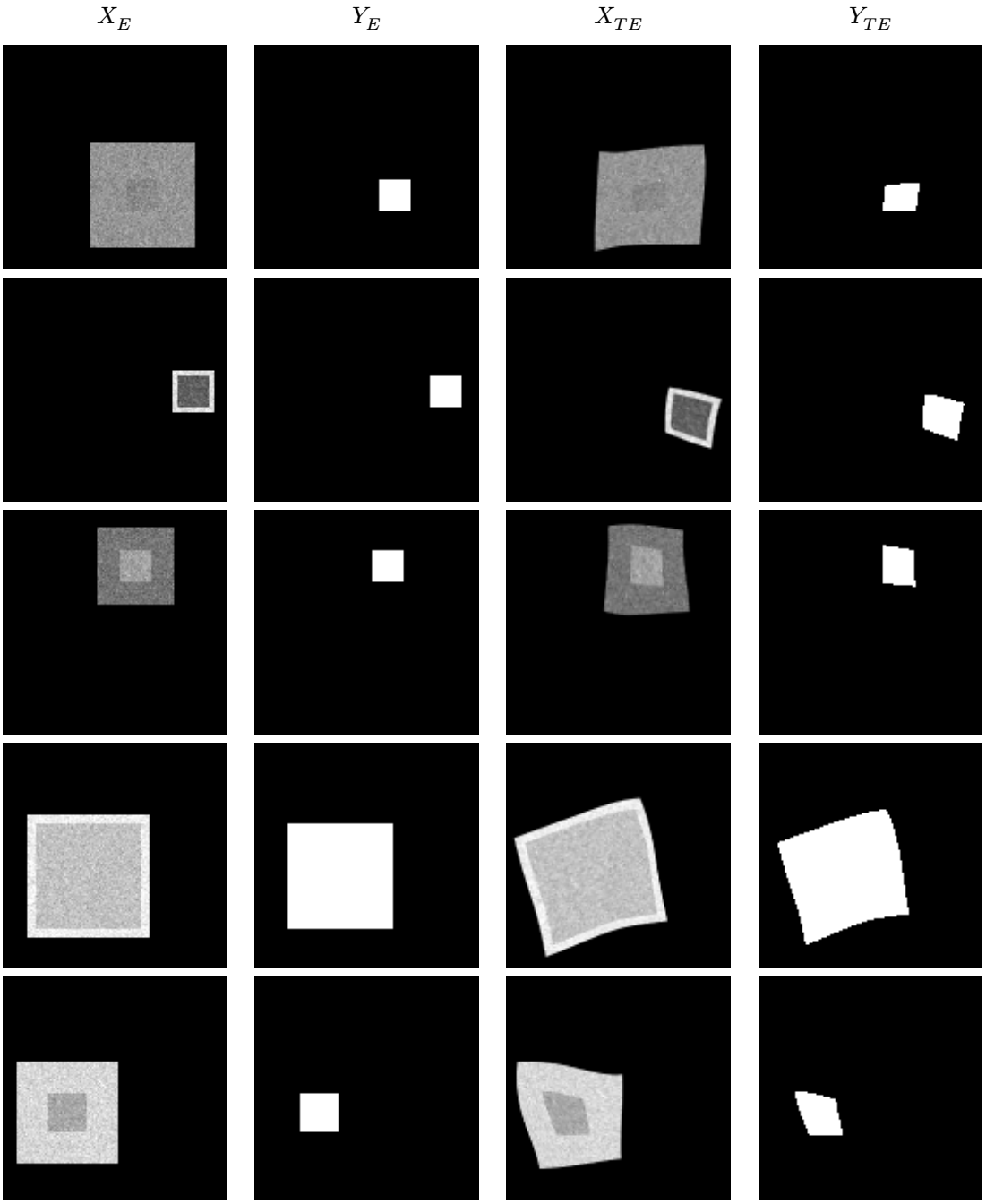


Figure A.3: Examples of the generated square images X_E (**column one**) and their binary segmentation Y_E (**column two**), as well as a random diffeomorphic transformations of those images ,X_{TE}, (**column three**) and the corresponding output Y_{TE} (**column four**). The images in X_E contain two nested squares of varying size and colour, where the squares' edges are parallel to the image edges with added Gaussian noise. The binary segmentation Y_E indicates the inner square. The images X_{TE} and their corresponding segmentations Y_{TE} are obtained by transforming the images X_E and their binary segmentations Y_E with the same randomly chosen SVF-based diffeomorphism.

A.3 DiffeoNN on the Synthetic Dataset

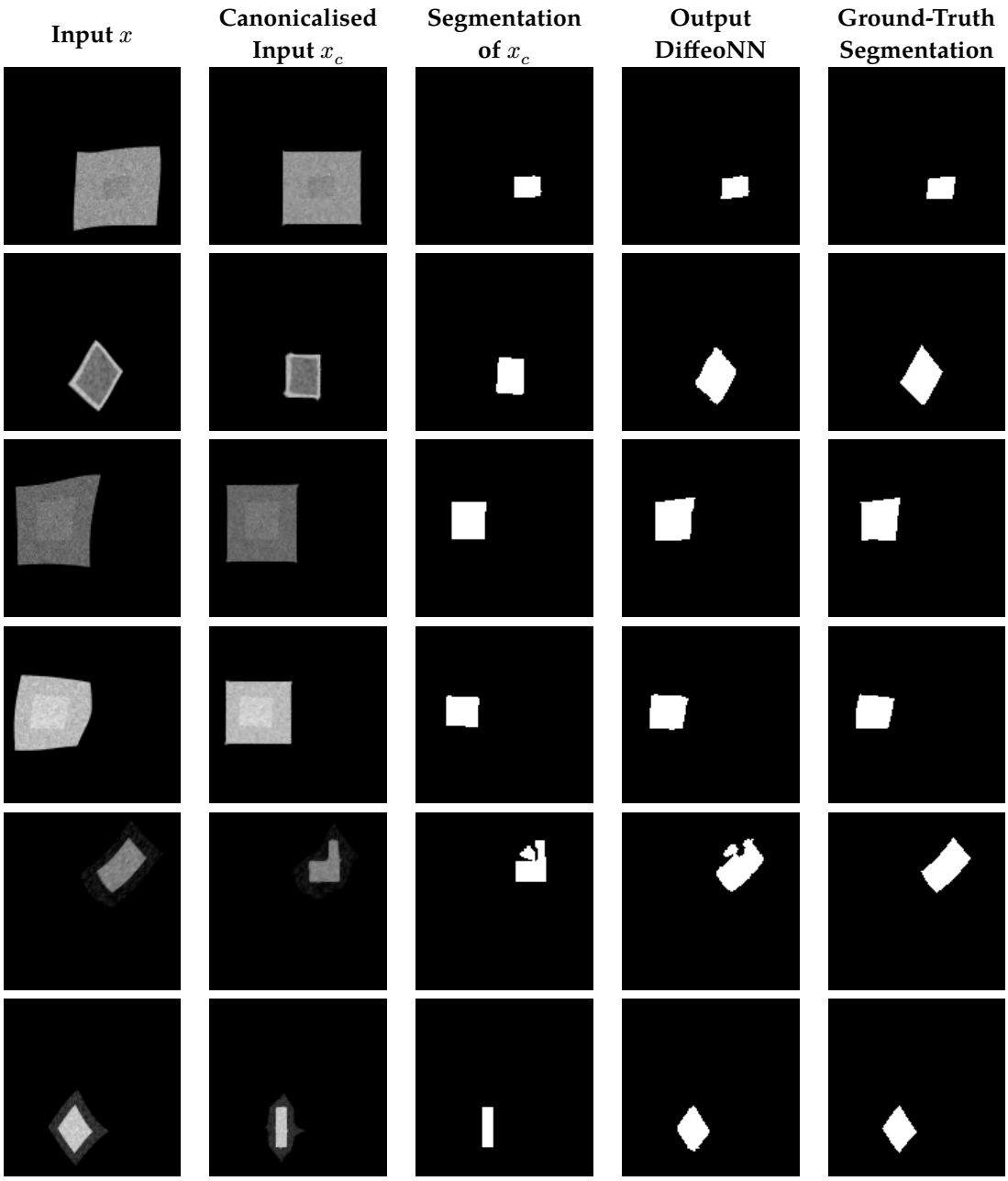


Figure A.4: Examples of results of DiffeoNN step by step on the synthetic test dataset X_{TE}^{test}. The **first four rows** show a nearly perfect segmentation through DiffeoNN. The canonicalised input images look very similar to the training dataset of squares X_E, and the output segmentation is very similar to the ground-truth segmentation. **Row five** shows an example where the canonicalisation step failed. As a result, the segmentation of the canonicalised image x_c contains artifacts. The artifacts are transferred to the output segmentation of the input x, which differs noticeably from the ground-truth segmentation. In **row six**, we show an example where the canonicalisation step leads to an image x_c that contains rectangles rather than squares. Even though this image does not look exactly like the training images with squares X_E, the segmentation output for x_c is still very accurate, which leads to a segmentation output for x that is nearly identical to the ground truth.

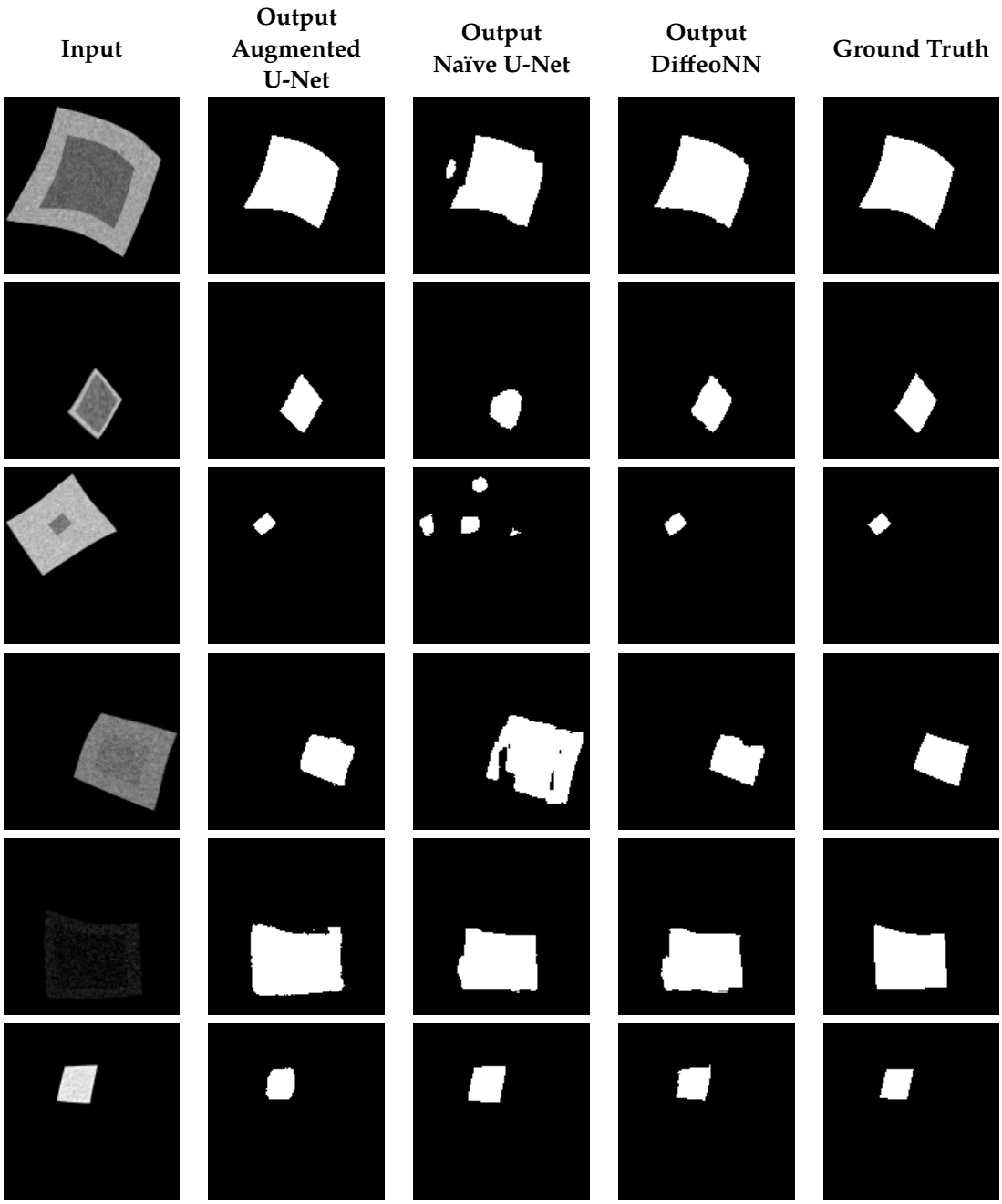


Figure A.5: Examples comparing the results of DiffeoNN, the naïve U-Net, and the augmented U-Net on X_{TE}^{test} (synthetic dataset). **Column one** shows the input image from $x \in X_{TE}^{\text{test}}$, and **column five** shows the corresponding ground-truth segmentation. In **column two**, we present the segmentation output of x by the augmented U-Net. The segmentation output of the naïve U-Net, i.e., the inner U-Net of DiffeoNN without canonicalisation and augmentation, is shown in **column three**. The output of DiffeoNN is shown in **column four**. In the **first four rows**, the segmentation of the naïve U-Net is noticeably different from the ground truth, while the segmentations of DiffeoNN and the augmented U-Net are similar. The **last two rows** show examples where the augmented U-Net performs worse than the other two methods.

A Appendix

A.4 Experiments on Invariance of the Canonicalisation

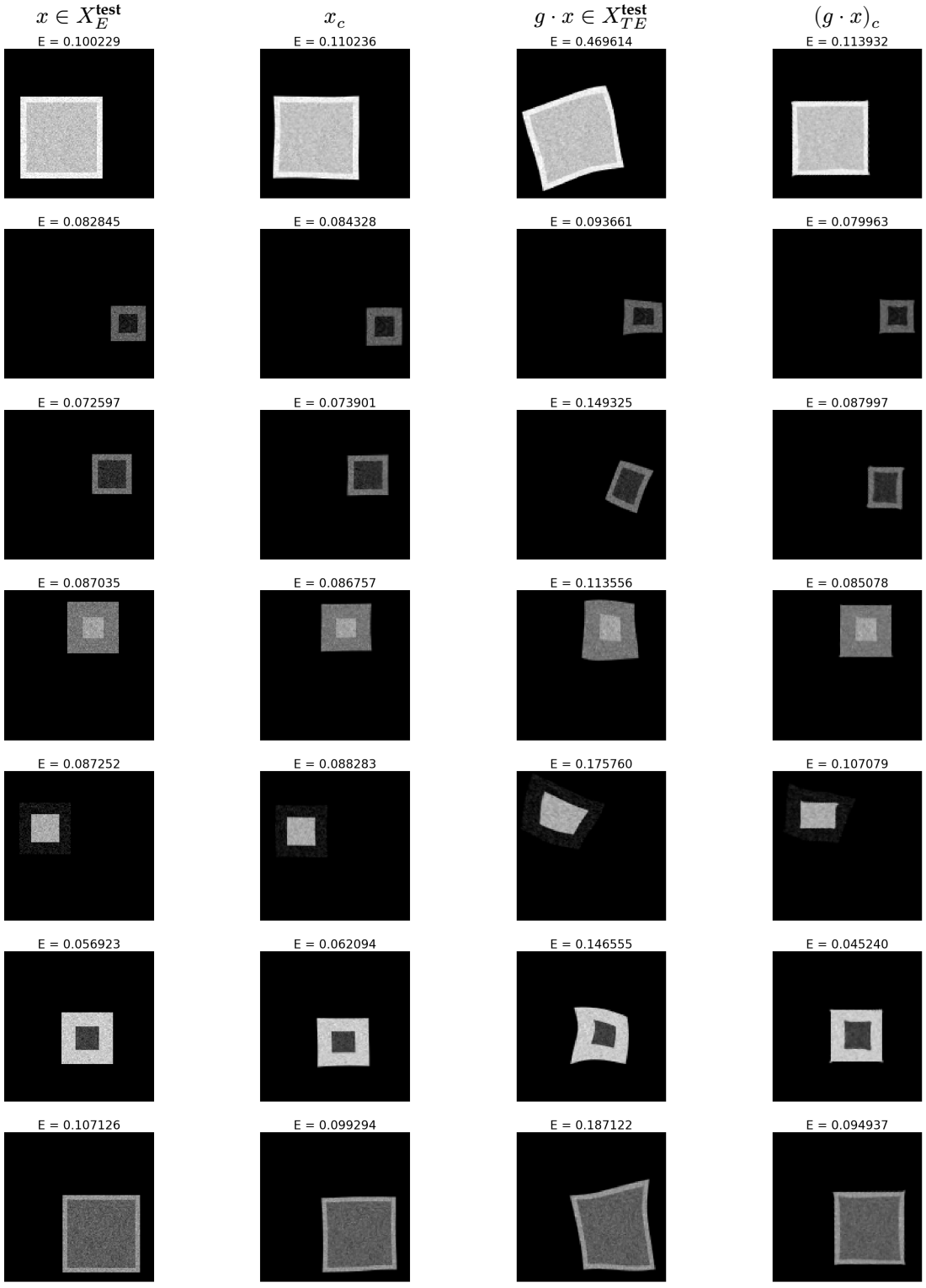


Figure A.6: Examples of images $x \in X_E$ (**column one**), their canonicalised form x_c (**column two**), their corresponding transformed images $g \cdot x \in X_{TE}$ (**column three**), and their canonicalised form $(g \cdot x)_c$ (**column four**) with their canonicalisation energies. The experiment is described in Section 5.4. The energies before the canonicalisation steps are very different. In contrast, the energies after canonicalising the inputs are approximately the same. Furthermore, the canonicalised images look very similar to images from X_E. This verifies the effectiveness of the canonicalisation step and its invariance empirically.

A.5 DiffeoNN for Lung Segmentation

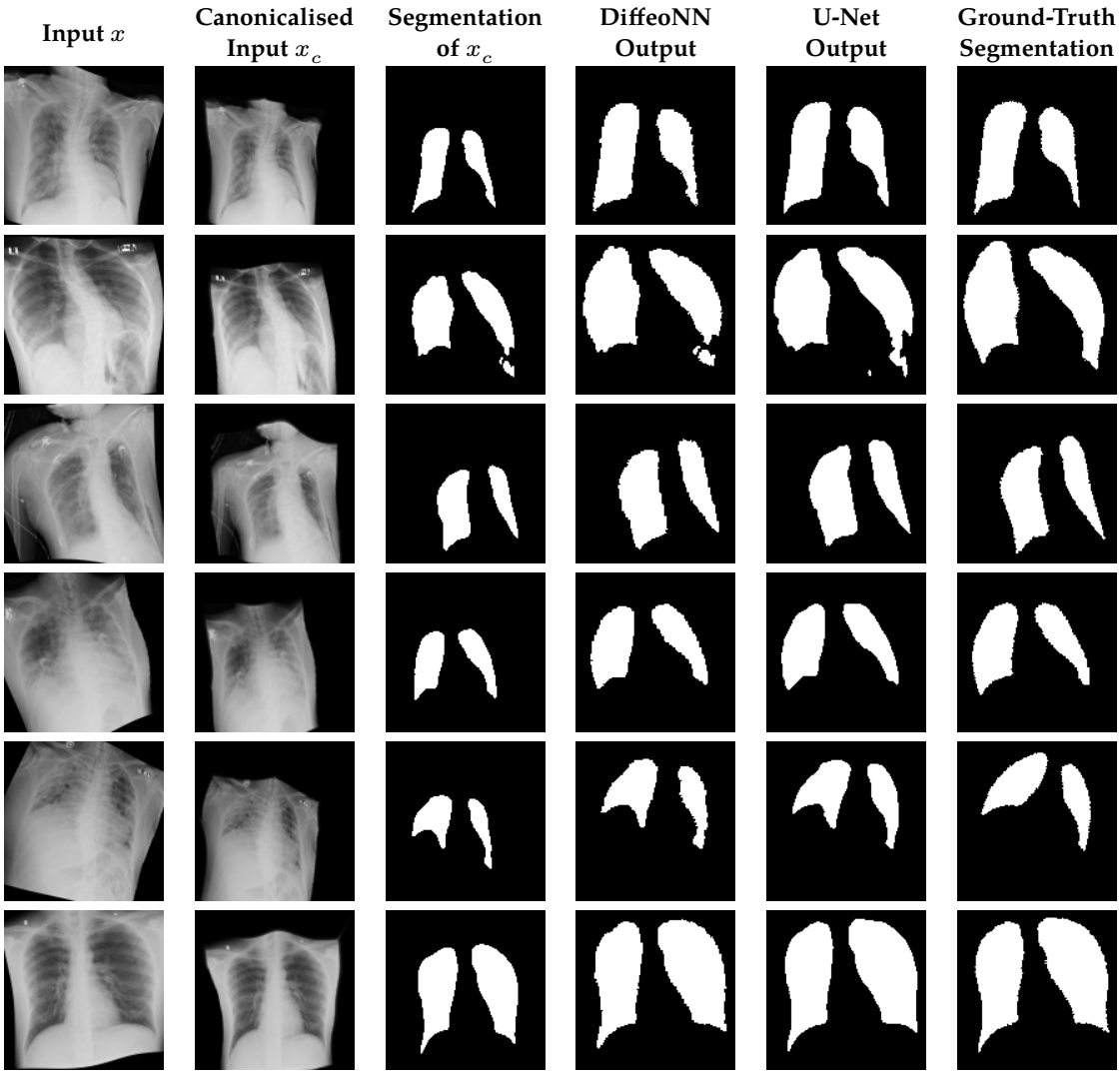


Figure A.7: Examples of lung segmentation of diffeomorphically transformed chest X-ray images from [68]. We show the diffeomorphically transformed input images (**column one**), the canonicalised input images (**column two**), the segmentation of the canonicalised input images (**column three**), the output segmentation of DiffeoNN (**column four**), the output of the inner U-Net (**column five**), and the ground-truth segmentation (**column six**). The canonicalisation step shrinks the images slightly. Furthermore, it aligns the shoulders with the horizontal axis of the images in most cases. The DiffeoNN segmentation performance is better than or at least equally good as the performance of the inner U-Net.