Equivariant Deep Learning Based on Scale-Spaces and Moving Frames

Mateus Sangalli

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL
Préparée à Mines Paris-PSL

Equivariant Deep Learning Based on Scale-Spaces and Moving Frames

Apprentissage Profond Équivariant Basé sur les Espaces d'Échelle et les Repères Mobiles

Soutenue par
Mateus SANGALLI
Le 16 Décembre 2022

Ecole doctorale n° 621
Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité
Morphologie Mathématique

Composition du jury :

Marcos VALLE Professeur associé, UNICAMP	*Président*
Tony LINDEBERG Professeur, KTH Royal Institute	*Rapporteur*
Petros MARAGOS Professeur, National Techinitcal University of Athens	Rapporteur
Nina MIOLANE Professeure assistante, UC Santa Barbara	*Examinatrice*
Samy BLUSSEAU Chargé de recherche, Mines Paris	*Examinateur*
Santiago VELASCO-FORERO Chargé de recherche, Mines Paris	*Examinateur*
Jesús ANGULO Directeur de recherche, Mines Paris	*Directeur de thèse*

MINES PARIS | PSL

Contents

Acknowledgements

I am very grateful to my thesis supervisors, Jesús Angulo, Santiago Velasco-Forero and Samy Blusseau. This thesis was only possible due to their valuable suggestions, feedback and to the discussions we had.

I want to also thank the members of the thesis committee, Marcos Valle, Nina Miolane, Tony Lindeberg and Petros Maragos for their insightful remarks and suggestions during my defence.

I would also like to thank the nice people of the CMM for creating a friendly and enjoyable work environment. I would like to thank the permanent members, Anne-Marie, Michel, Etienne, François, Beatriz, Bruno and José Marcio as well as the PhD students and interns I met during my time here, Martin, Valentin, Tarek, David, Leonardo, Eric, Thomas, Stéfan, João, Romain, Berjo, Tin, Konstantin, Victor, Ayoub, Mateusz, Lucas and Guilherme.

I want to express my gratefulness for having friends I can count on during the hard times that writing a thesis can be. Either when playing board games, traveling or just talking about random stuff I am happy you are in my life Bianca, Marcelo, Bob, Dan, Lucas, Davi, Aline, Zampieri and Manzatto.

Finally, it is thanks to the encouragement and love from my family that I was able to pursue a PhD, so I want to send a heartfelt thanks to my parents Orivaldo and Aparecida and my siblings Andrea, Rinaldo and Rafael for being there for me.

Introduction

The past decade has seen an insurgence of deep learning in the literature and in a large variety of applications. Particularly in computer vision tasks: image classification, segmentation, labeling, etc. much of this progress was due to Convolutional Neural Networks (CNNs). One of the reasons CNNs are particularly interesting for computer vision tasks is their *translation-equivariance*, which is the property of an operator commuting with translations, i.e., translating an input and applying the operator gives the same results as applying the operator to the input and then translating its output. Translation-invariant operators (i.e. their output does not change if you translate the input) are also interesting and can be obtained from a translation equivariant operator by applying some sort of pooling, e.g. taking the supremum of the values of the equivariant output. Convolutions are intrinsically translation-equivariant, and because fully convolutional neural networks (i.e. CNNs without fully connected layers) are built with only convolutions and (in general) point-wise operators, they are either translation-equivariant or translation-invariant.

Equivariance is not a property limited to translations. Indeed, equivariance to other kinds of transformations is frequently sought in computer vision, particularly when the transformations in question form a group or semigroup. It has been a topic in the literature of image processing for quite some time: Scale-spaces (Heijmans and van den Boomgaard, 2002; Heijmans, 2002; Lowe, 1999; Witkin, 1984; Alvarez et al., 1993; Pauwels et al., 1995) are essentially operators equivariant to a semigroup of scalings, the approach of group morphology (Roerdink, 2000) builds general morphological operators that are equivariant to some group of transformations, and diffusion schemes (Weickert, 1998) are often equivariant to rotations.

Recent advances in the literature of deep learning go beyond translation equivariance and seek equivariance or invariance to other domains and other types of transformations: permutation-invariant networks for sets (Zaheer et al., 2017), equivariance to 90° rotations, translations and reflections for images (Cohen and Welling, 2016a), rotation and translation equivariance in 2D (Worrall et al., 2017; Cohen and Welling, 2016b; Shen et al., 2020) and 3D (Weiler et al., 2018a; Shen et al., 2022; Worrall and Brostow, 2018; Thomas et al., 2018; Thomas, 2020), scale and translations equivariance (Worrall and Welling, 2019; Lindeberg, 2022) and even equivariance to Lorentz transformations for physics (Bogatskiy et al., 2020). Equivariance to transformations helps improving generalization error and reducing the number of the training samples needed as soon as the data is symmetric with respect to that family of transformations.

Classical mathematical literature also has much to contribute with group equivariance and invariance. In particular, the method of *moving frames* proposed by Élie Cartan (Cartan, 1935) is a method for finding differential invariants in smooth manifolds. Differential invariants can

be associated to equivariant operators on a space of functions. More recently, the method of moving frames has been thoroughly explored (Fels and Olver, 1999; Hubert and Kogan, 2007) and also applied to problems in image processing such as feature detection (Tuznik et al., 2018), object recognition (Calabi et al., 1998) and curve evolution (Faugeras, 1993).

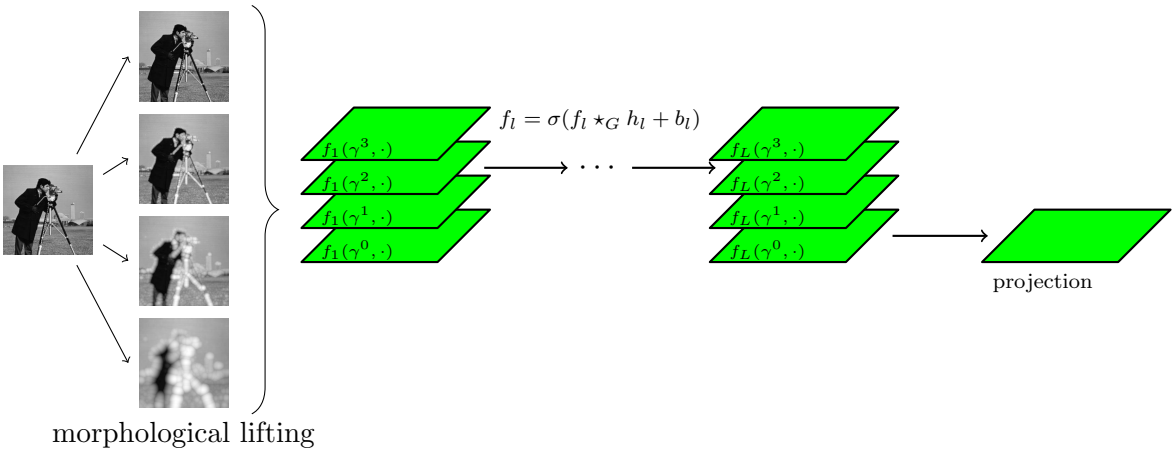
The object of this thesis is the derivation of group and semigroup equivariant neural networks. There are two main frameworks which were used to study this problem of equivariance in the thesis: The first one is an extension of the scale-equivariant networks proposed in Worrall and Welling (2019), which is the main topic of Chapters 2 and 3. The second one is based on the method of moving frames, which we use to define a new class of group equivariant layers for CNNs and is the main topic of Chapters 4, 5 and 6. What follows is a brief summary of each chapter:

Chapter 1 introduces some important notions that will be used throughout the thesis, namely groups, group equivariance and invariance. We proceed to review the notion of geometric deep learning and discuss the group convolution and steerable networks, which are the most used forms of group equivariant layers for neural networks.

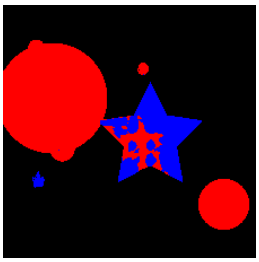
Chapter 2 explores our work with scale-equivariant architectures. Particularly we extend the scale semigroup equivariant network proposed by Worrall and Welling (2019). That network uses the Gaussian scale-space to first lift the input image to a suitable multi-scale representation and apply linear scale-equivariant operator to obtain an equivariant network. We find sufficient conditions for other scale-spaces to be used as the first layer and apply those scale-spaces, especially morphological scale-spaces, to image classification and segmentation. A scale-equivariant network is illustrated in Figure 1. We show that morphological scale-spaces can have an advantage over CNNs and Gaussian scale-space based networks when shape is the only available information, as illustrated in Figure 1 (b) and (c). The approaches presented in this chapter were published in Sangalli et al. (2021).

Chapter 3 continues to work on scale-equivariant networks for image segmentation. We build upon the U-Net segmentation model, which attains state-of-the-art results in a variety of segmentation tasks, but is not a priori scale-equivariant, and experiments show empirically that it is not scale-equivariant in practice as shown in Figure 3. We propose the Scale Equivariant U-Net (SEU-Net), illustrated in Figure 2, in order to have a scale-equivariant network similar to U-Net in terms of global architecture. The blocks of downsampling and upsampling of the U-Net are carefully analyzed in order to define their counterparts in the SEU-Net. The SEU-Net is tested in segmentation tasks and is shown to perform well compared to the U-Net and to Scale-Equivariant architectures that do not perform operations such as upsampling in the equivariant part of the network. The contents of this chapter were published in Sangalli et al. (2022b).

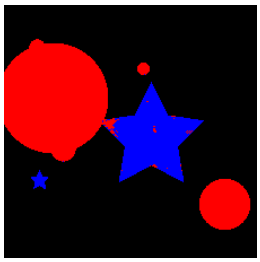
Chapter 4 introduces neural networks based on moving frames. It begins by a review of the method of moving frames and how it can be used to obtain differential invariants. The method is applied to obtain differential invariants of images under the action of the group SE(2) i.e. the group of rotations and translations in the plane. In essence, the method finds an orientation at each point of the image based on the local geometry, and the orientation is used to rotate each neighborhood being processed as illustrated in Figure 4. Because of their ability of describing the local geometry at a given scale we use Gaussian derivatives to compute the differential invariants and implement a SE(2)-equivariant architecture, called SE2DINNet. We



(a) Scale-equivariant network with morphological lifting.



(b) Prediction with Gaussian lifting



(c) Prediction with dilation lifting

Figure 1: An illustration of the morphological lifting(a) (especifically quadratic dilation lifting) proposed in Chapter 2 as an alternative to the Gaussian lifting. We shown the result of a shape classification experiment where the distribution of scales in the training set is not representative of the distribution of scales in the test set. The model with a dilation scale-space lifting (c) manages to improve the results of the model with a Gaussian scale-space lifting(b) in this case, as the morphological operators are better suited at preserving the shape of objects.

perform a series of ablation studies in the MNIST dataset and obtain competitive results in the MNIST-Rot dataset. This chapter explores the works published in Sangalli et al. (2022a)

Chapter 5 extends the approach of the previous chapter to SE(3)-equivariant networks on volumes. However, because of numerical issues we cannot extend those approaches directly. We note that the computation of the differential invariants can be decomposed in two steps: computation of a moving frame followed by the application of the moving frame to partial derivatives of the image (referred to as n-jets), as illustrated in Figure 5(a). In this chapter we show that keeping only the first moving frame also defines an equivariant network. This leads to the definition of an alternative neural network based on moving frames that, instead of computing the moving frames at each feature map as the previous approach we compute the moving frame at a single image, assumed to be the input image, and apply that moving frame to subsequent layers. A network constructed from this principle is illustrated in Figure 5(b). In that way, before the activation function the equivariant layers based on the moving frame become linear with respect to the output of the previous layer. The approach is tested on

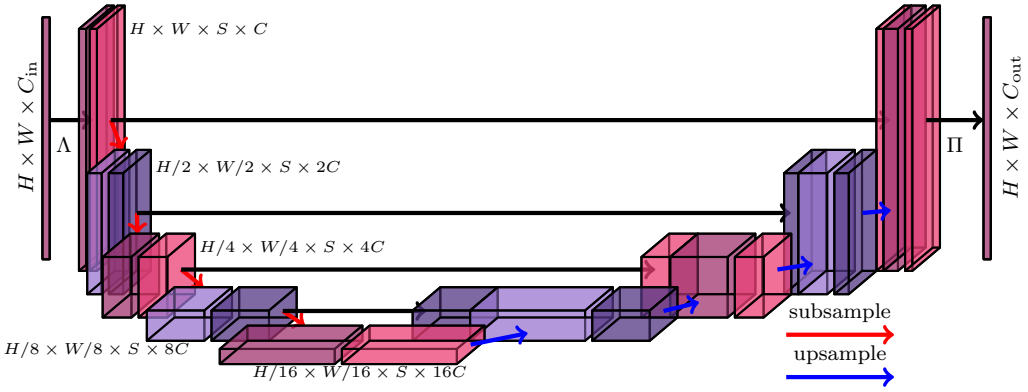
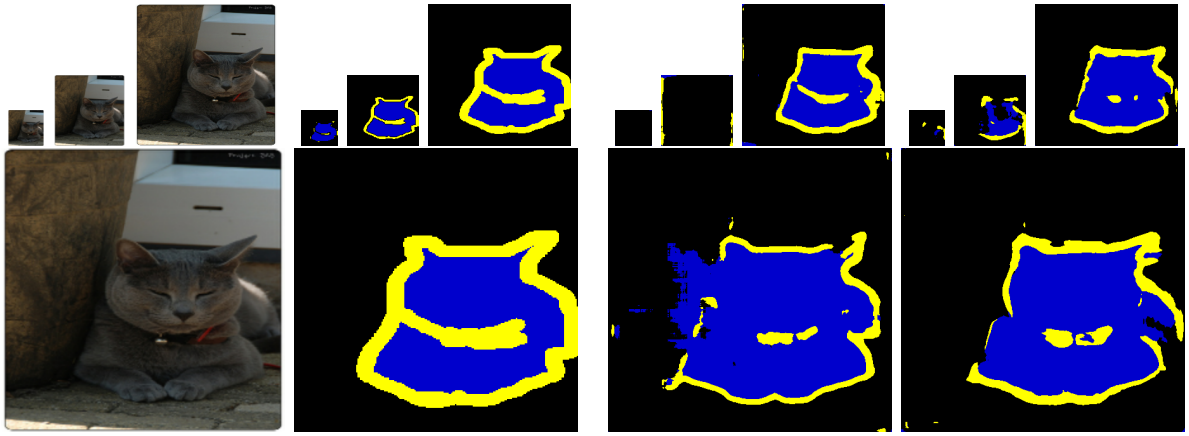


Figure 2: Illustration of the SEU-Net architecture proposed in Chapter 3. Images are lifted to a scale-space representation with a lifting Λ, processed with a U-Net like architecture using scale-cross-correlations and other equivariant or approximately equivariant operators and projected back into the shape.

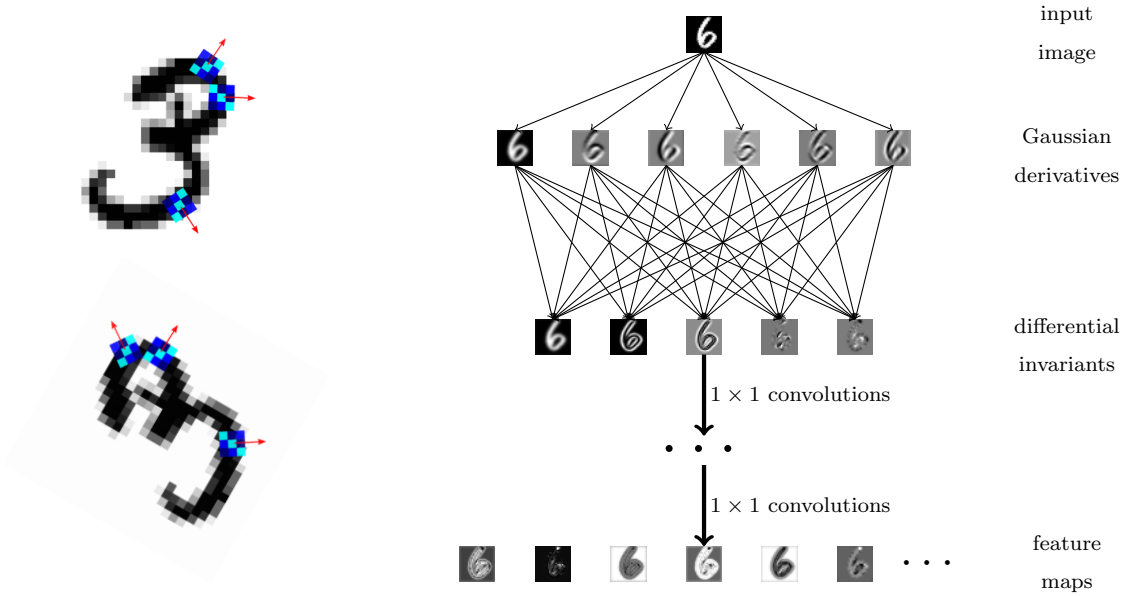


(a) Image (b) Ground Truth (c) U-Net (d) SEU-Net

Figure 3: Results from Chapter 3 showing that the U-Net is not scale-equivariant empirically and that the SEU-Net greatly improves its equivariance. Both models were trained on the scale of the second biggest image.

databases of low-resolution medical images for classification and improves most of the benchmarks on those datasets. This chapter correspond to the results published in Sangalli et al. (2023).

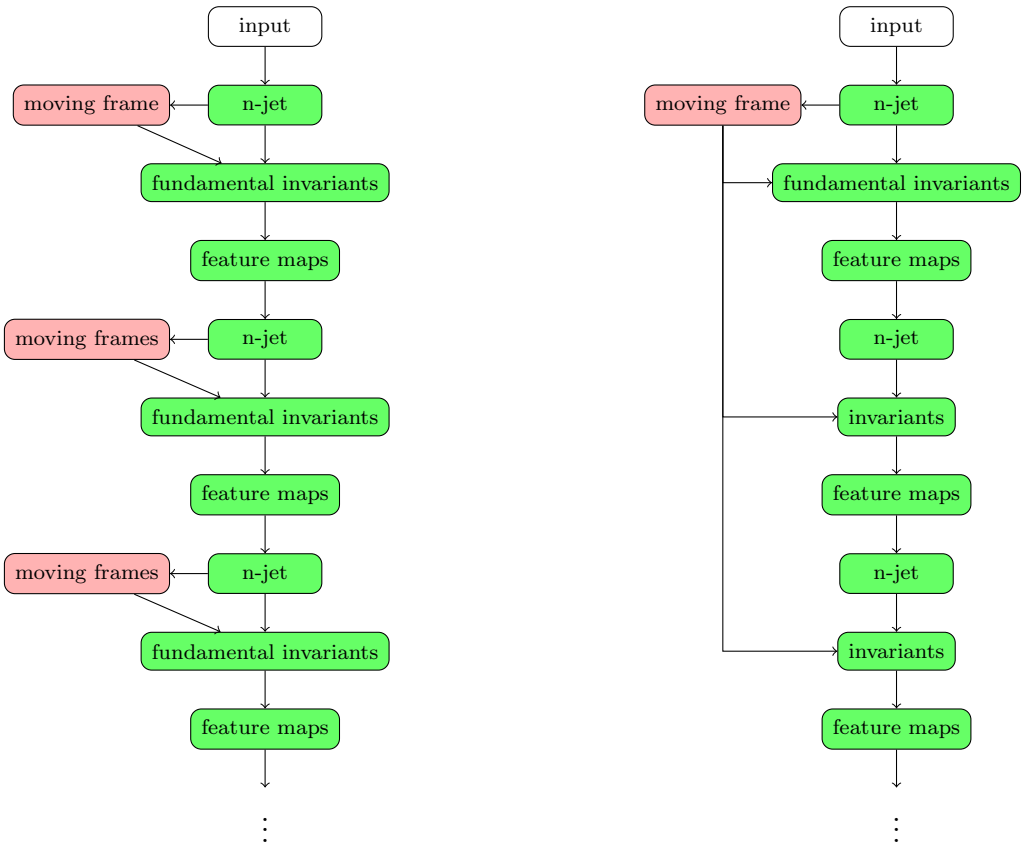
Chapter 6 discusses some potential applications of neural networks based on moving frames that were not discussed in other chapters because they were not as developed as the previous ones. Particularly we explore (i) a contrast-invariant neural network based on a layer invariant to an affine transformation on the gray-levels of the input image, (ii) a scale, rotation and translation invariant network based on applying a multi-scale Gaussian derivative network to the differential invariants from Chapter 4 and (iii) a SE(3)-equivariant neural network for point clouds based on the method of moving frames.



(a) SE2DIN block intuition

(b) SE2DIN block

Figure 4: Illustration of the idea behind our moving frame approach for equivariant networks (a) as well as the illustration of our implementation used in Chapter 4. The idea is to use the local structure of the image to obtain a rotation equivariant quantity (e.g. the direction of the gradient, shown in red) and apply filters (shown in blue) like in a convolution but turned with respect to that quantity. When the image is rotated, the red arrows are equal and so the filters. By computing a linear combination of differential invariants, as the first 1×1 convolution in (b) we can achieve an effect similar to the illustration in example (a).



(a) SE2DINNet

(b) Invariants from a fixed moving frame

Figure 5: Illustration of the difference of the approach of Chapter 4, where the moving frame is computed at every SE2DIN block, and the approach used in Chapter 5, where the moving frame is computed in the first layer and used subsequent layers. Similar diagrams can be shown in the case the network uses residual connections.

Relevant publications. The research work of this thesis led to the following papers:

1. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2021). Scale equivariant neural networks with morphological scale-spaces. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 483–495. Springer

2. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022a). Differential invariants for SE(2)-equivariant networks. In *29th IEEE International Conference on Image Processing (IEEE ICIP)*, Bordeaux, France. (Oral Presentation)

3. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022b). Scale Equivariant U-Net. In *33rd British Machine Vision Conference*, London, United Kingdom

4. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2023). Moving frame net: SE(3)-equivariant network for volumes. In Sanborn, S., Shewmake, C., Azeglio, S., Di Bernardo, A., and Miolane, N., editors, *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 197 of *Proceedings of Machine Learning Research*, pages 81–97. PMLR

Introduction en Français

La dernière décennie a été marquée par une montée en popularité de l'apprentissage profond dans la littérature, dans une grande variété d'applications. Par exemple, dans certaines tâches de vision par ordinateur : classification, segmentation, détection d'objets etc. Une grande partie de ce progrès est due aux Réseaux de Neurones Convolutifs (CNNs). L'une des raisons pour lesquelles les CNNs sont particulièrement intéressants est leur équivariance par rapport aux translations, la propriété de commuter avec les translations, c'est-à-dire que la translation de l'entrée suivie de l'application d'un opérateur donne la même sortie qu'appliquer l'opérateur suivi de la translation. Les opérateurs invariants par translation (leur sortie ne change pas si leur entrée est transformée par une translation) sont également intéressants et peuvent être obtenus par un opérateur équivariant par translation suivi d'une fonction de réduction, comme par exemple le supremum de tous les pixels de l'image. Les convolutions sont intrinsèquement équivariantes par translation, et il en va de même pour les réseaux entièrement convolutifs (CNNs sans couches totalement connectés) cars ils sont construits uniquement à partir de convolutions et d'opérateurs ponctuels.

En général, l'équivariance n'est pas limitée aux translations. En effet, des opérateurs équivariants pour d'autres transformations sont souvent recherchés en vision par ordinateur, notamment dans le cas où ces transformations constituent un groupe ou un semi-groupe. L'équivariance est un sujet d'intérêt dans la littérature du traitement d'images depuis quelques années: Les espaces d'échelle (Heijmans and van den Boomgaard, 2002; Heijmans, 2002; Lowe, 1999; Alvarez et al., 1993) sont des opérateurs équivariants par l'action d'un semi-groupe d'échelles, la morphologie de groupes (Roerdink, 2000) construit des opérateurs morphologiques équivariants par une transformation de groupe, et les schémas de diffusion (Weickert, 1998) sont fréquemment équivariants par rotation.

Les avancées récentes dans la littérature de l'apprentissage profond vont au-delà de l'équivariance par translation et cherchent l'équivariance - ou l'invariance - par rapport à d'autres transformations et sur d'autres domaines: les réseaux équivariants par permutation pour les ensembles (Zaheer et al., 2017), l'équivariance par rotation par 90°, translations et réflections dans le domaine des images (Cohen and Welling, 2016a), l'équivariance par rapport aux rotations et aux translations en 2D (Worrall et al., 2017; Cohen and Welling, 2016b; Shen et al., 2020) et 3D (Weiler et al., 2018a; Shen et al., 2022; Worrall and Brostow, 2018; Thomas et al., 2018; Thomas, 2020), l'équivariance par changement d'échelle (Worrall and Welling, 2019; Lindeberg, 2022) et l'équivariance par transformation de Lorentz pour la physique (Bogatskiy et al., 2020). Équivariance par rapport aux transformations peut améliorer un modèle et réduire le nombre d'échantillons d'entraînement nécessaires pour l'apprentissage si la symétrie par rapport à la famille de transformations est présente dans la base de données.

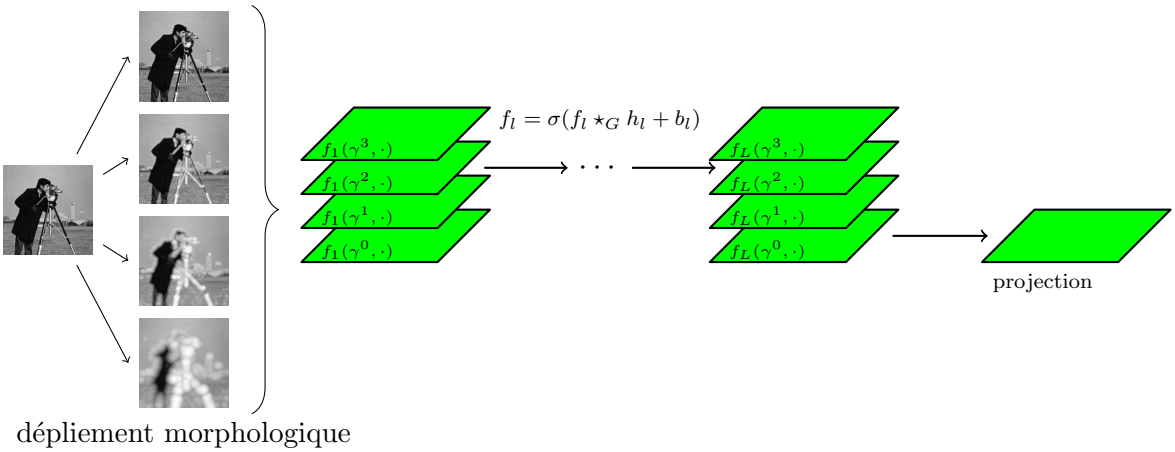
Dans la littérature mathématique on trouve de nombreuses contributions sur l'équivariance et l'invariance. En particulier, la méthode du *repère mobile* proposée par Élie Cartan (Cartan, 1935) est une méthode pour trouver des invariants différentiels dans une variété. Chaque invariant différentiel peut être associé à un opérateur équivariant dans l'espace de fonctions. Plus récemment, la méthode du repère mobile a été soigneusement explorée (Fels and Olver, 1999; Hubert and Kogan, 2007) et appliquée à certains problèmes de traitement d'images, comme la détection de caractéristiques (Tuznik et al., 2018), la reconnaissance d'objets (Calabi et al., 1998) et l'évolution de courbes (Faugeras, 1993).

Cette thèse porte sur la conception de réseaux de neurones équivariants par rapport à un groupe ou semi-groupe. L'équivariance a été étudiée dans de deux cadres. Le premier est une extension des réseaux équivariants par changement d'échelle proposés par Worrall and Welling (2019), et il est le principal sujet des chapitres 2 et 3. Le deuxième utilise la méthode du repère mobile pour définir des réseaux équivariants par l'action d'un groupe de Lie. Le deuxième cadre est abordé dans les Chapitres 4, 5 et 6. Ce qui suit est un résumé de chaque chapitre.

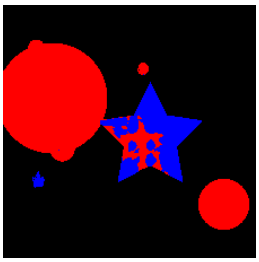
Dans le **Chapitre 1** nous introduisons des notions importantes qui nous seront utiles tout au long de la thèse : groupes, équivariance et invariance. Ensuite nous effectuons une revue des notions d'apprentissage profond géométrique et nous discutons des convolutions de groupes et des réseaux orientables, qui sont les couches équivariantes par l'action d'un groupe les plus utilisées dans la littérature.

Le **Chapitre 2** explore notre travail sur les architectures équivariantes par changement d'échelle. En particulier nous généralisons le réseau équivariant par un semi-groupe d'échelles proposé par Worrall and Welling (2019). Cette approche utilise l'espace d'échelle Gaussien pour obtenir une représentation multi-échelle adéquate à partir de l'image d'entrée, ce qui est appelé dépliement ici. Puis, des opérateurs linéaires et équivariants par changement d'échelle sont appliqués pour obtenir un réseau équivariant. Nous trouvons des conditions suffisantes pour utiliser d'autres espaces d'échelles comme dépliement, et nous en utilisons certains, en particulier des espaces morphologiques, dans des tâches de classification et segmentation d'images. Un réseau équivariant par changement d'échelle est illustré sur la Figure 6. Nous montrons que les espaces d'échelle morphologiques peuvent avoir des avantages sur les CNNs et les réseaux basés sur les espaces d'échelle Gaussiens lorsque la forme est la seule information disponible, comme illustré dans les Figures 6 (b) et (c). Les approches présentées dans ce chapitre ont été publiées dans Sangalli et al. (2021).

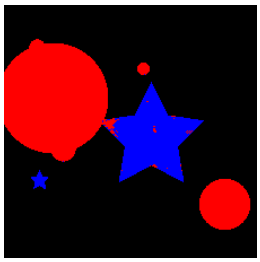
Le **Chapitre 3** poursuit le travail sur les réseaux équivariants par changement d'échelle pour la segmentation d'images. Nous nous appuyons sur le modèle de segmentation U-Net, qui atteint l'état-d'art dans une variété de tâches de segmentation, mais n'est pas équivariant par changement d'échelle ni à priori, ni en pratique, ce qui est confirmé par des expériences, comme montré dans la Figure 8. Nous proposons un U-Net Équivariant par Changement d'échelle (SEU-Net), illustré dans la Figure 7, un réseau équivariant par changement d'échelle qui est comparable à U-Net pour son architecture globale. Les blocs de sous-échantillonnage et suréchantillonnage de U-Net sont méticuleusement analysés pour définir ses analogues dans le SEU-Net. Le SEU-Net est testé dans des tâches de segmentation et il est montré qu'il obtient une bonne performance en comparaison à U-Net et à des architectures qui ne font pas de suréchantillonnage dans la partie équivariante du réseau. Le contenu de ce chapitre été publié dans Sangalli et al. (2022b).



(a) Réseau équivariant par changement d'échelle avec le dépliement Gaussien.



(b) Prédiction avec le dépliement Gaussien



(c) Prédiction avec le dépliement par dilatation

Figure 6: Une illustration des dépliements morphologiques (a) (particulièrement les dépliements basés sur les dilatations quadratiques) proposés dans le Chapitre 2 comme une alternative au dépliement Gaussien. Nous montrons les résultats de l'expérience de classification de formes, où la distribution des échelles dans les données d'entraînement n'est pas la même que celle dans les données de teste. Le modèle avec un dépliement basé sur l'espace d'échelle des dilatations quadratiques (c) obtient de meilleurs résultats que les modèles avec le dépliement Gaussien (b). Dans ce cas, les opérateurs morphologiques sont plus efficaces pour préserver la forme des objets.

Le **Chapitre 4** introduit des réseaux de neurones basés sur la méthode du repère mobile. Nous appliquons la méthode pour trouver des invariants différentiels de l'action de SE(2), le groupe des rotations et translations en deux dimensions. En essence, pour chaque point de l'image, la méthode trouve une orientation basée sur la géométrie locale, puis l'orientation est utilisée pour faire tourner chaque voisinage en train d'être traité, comme illustré par la Figure 9. Nous utilisons les dérivées Gaussiennes pour calculer les invariants différentiels grâce à leur capacité de décrire la géométrie locale, et nous implémentons une architecture (appelée SE2DINNet) équivariante par rapport à l'action de SE(2). Nous faisons une série d'études d'ablation et nous obtenons des résultats compétitifs dans la base de données MNIST-Rot. Ce chapitre explore les travaux publiés dans Sangalli et al. (2022a)

Le **Chapitre 5** étend l'approche du chapitre précèdent aux réseaux équivariants par l'action de SE(3) pour les volumes. Toutefois, en raisons des problèmes numériques causés par la forme

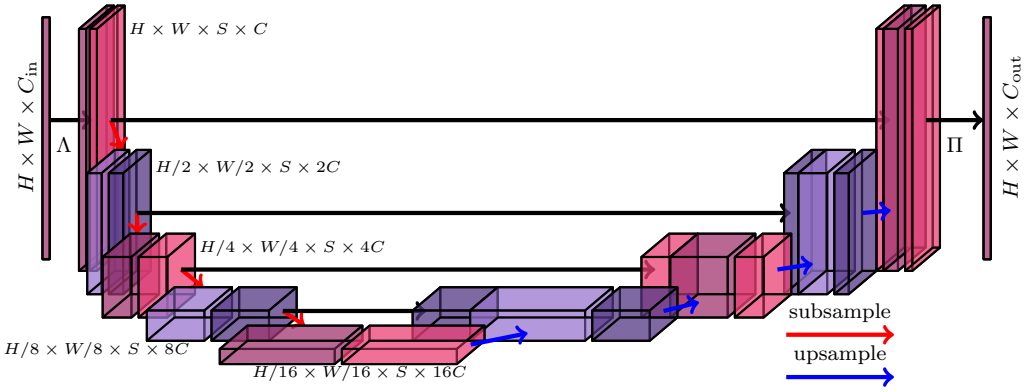
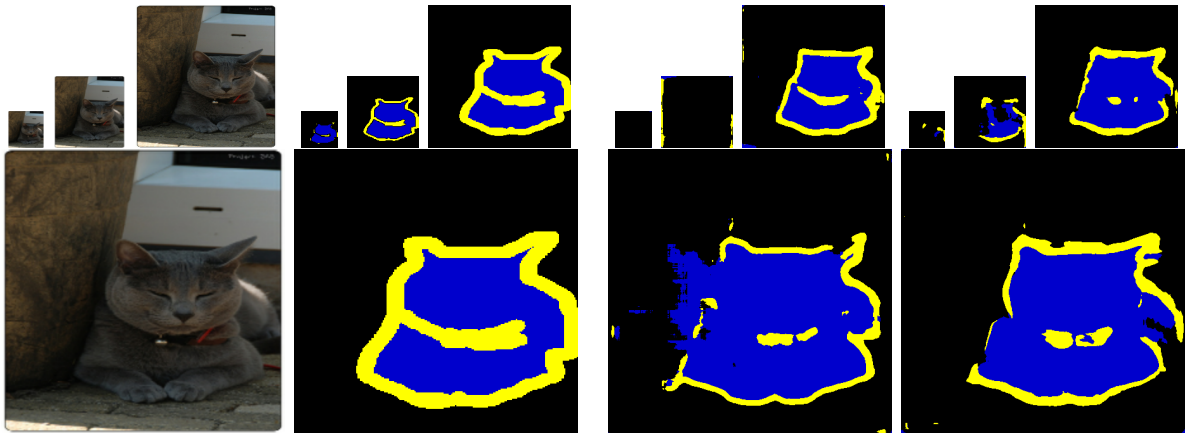


Figure 7: Illustration de l'architecture SEU-Net proposée dans le Chapitre 3.

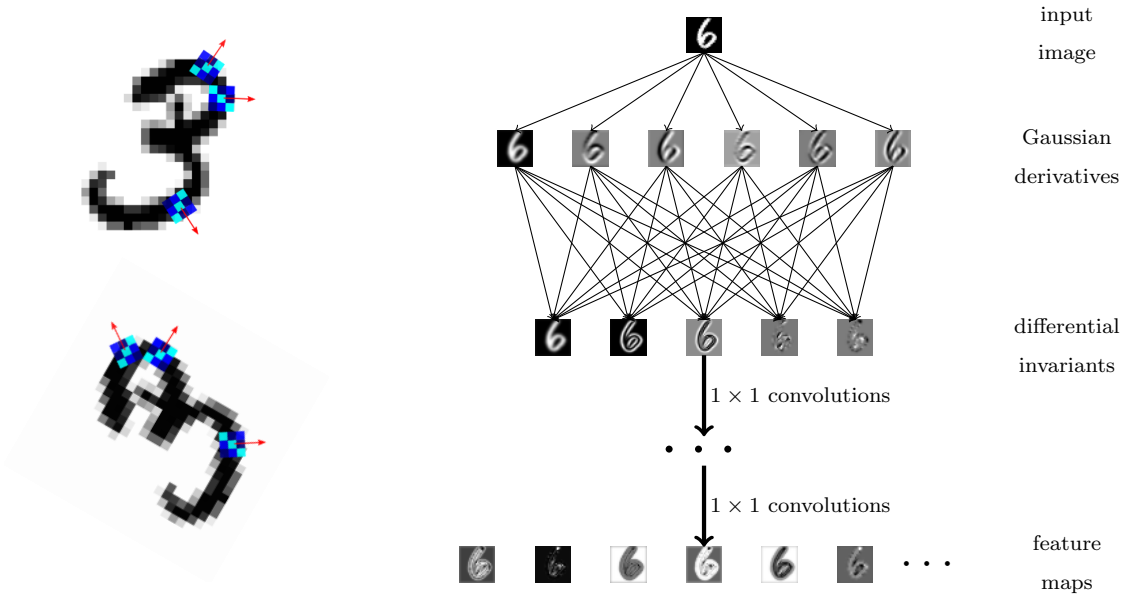


(a) Image (b) Verité Terrain (c) U-Net (d) SEU-Net

Figure 8: Résultats du Chapitre 3 qui montrent que U-Net n'est pas empiriquement équivariant par changement d'échelle et que le SEU-Net améliore la performance sur la tâche. Tous les deux modèles étaient entraînés sur une base de données à l'échelle de la deuxième image la plus grande montrée.

polynomiale des invariants, nous ne pouvons pas étendre ces approches directement. Nous précisons que le calcul des invariants peut être décomposé en deux parties: le calcul du repère mobile suivi par l'application de ce repère mobile aux dérivées (appelées n-jets), comme illustré par la Figure 10(a). Dans ce chapitre nous montrons que nous pouvons garder seulement le repère mobile calculé dans la première couche d'un réseau et l'utiliser pour définir un réseau équivariant. Cela nous ramène à une définition d'un réseau équivariant alternatif basé sur les repères mobiles. Un réseau construit à partir de ce principe est illustré dans la Figure 10(b). De cette façon, avant l'application de la fonction d'activation les couches de ce réseau sont linéaires par rapport à la sortie de la couche précédente. L'approche est testée pour la classification de bases d'images médicales en faible résolution, et il améliore les résultats précédents sur ces bases de données. Ce chapitre correspond aux résultats publiés dans Sangalli et al. (2023).

Le **Chapitre 6** examine des applications potentielles des réseaux de neurones basés sur les repères mobiles. En particulier, dans le Chapitre 6 nous explorons : (i) un réseau invariant par



(a) Intuition

(b) Bloc SE2DIN

Figure 9: Illustration de l'idée principale derrière les réseaux équivariants basés sur les repères mobiles (a) ainsi qu'une illustration de l'implémentation de la méthode utilisée dans le Chapitre 4. L'idée est d'utiliser la structure locale de l'image pour obtenir une transformation (par exemple la rotation par la direction du gradient, en rouge) de façon équivariante et puis appliquer un filtre (en bleu) transformé par cette transformation. Si l'image est tournée, alors les flèches rouges et les filtres sont également tournés. En calculant une combinaison linéaire des invariants différentiels, comme la convolution 1×1 en (b), nous obtenons un effet similaire à l'illustration dans l'exemple (a).

changement de contraste basé sur une couche invariante par transformation affine des niveaux de gris de l'image d'entrée, (ii) un réseau invariant par changement d'échelle, rotation et translation basé sur l'application des dérivées Gaussiennes multi-échelles aux invariants différentiels du Chapitre 4 et (iii) un réseau de neurones équivariant par l'action de SE(3) pour les nuages de points basé sur la méthode du repère mobile.

(a) SE2DINNet (b) Invariants à partir d'un repère mobile fixé

Figure 10: Illustration de la différence entre l'approche du Chapitre 4, où les repères mobiles sont calculés à chaque bloc SE2DIN, et celle du Chapitre 5, où le repère mobile est calculé dans la première couche et utilisé pour toutes les couches suivantes.

Publications Pertinentes.

1. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2021). Scale equivariant neural networks with morphological scale-spaces. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 483–495. Springer

2. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022a). Differential invariants for SE(2)-equivariant networks. In *29th IEEE International Conference on Image Processing (IEEE ICIP)*, Bordeaux, France. (Oral Presentation)

3. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022b). Scale Equivariant U-Net. In *33rd British Machine Vision Conference*, London, United Kingdom

4. Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2023). Moving frame net: SE(3)-equivariant network for volumes. In Sanborn, S., Shewmake, C., Azeglio, S., Di Bernardo, A., and Miolane, N., editors, *Proceedings of the 1st NeurIPS Workshop*

on Symmetry and Geometry in Neural Representations, volume 197 of *Proceedings of Machine Learning Research*, pages 81–97. PMLR

Chapter 1

Group Equivariant Networks

1.1 Introduction

The success of convolutional neural networks is partially determined by its ability to consider local information and to exploit the translational symmetry intrinsic to many computer vision tasks. Translational symmetry is not, however, the only interesting symmetry present in computer vision tasks. Consider the case of a classification problem like in Figure 1.1, it would be desirable that the image be classified as "cat" independently of perspective changes.

"Cat" "Cat" "Dog"

Figure 1.1: An image and its transformations in a classification task. In a fully convolutional network, a correct classification of an image implies the correct classification of all its translated versions (disregarding border effects). Since CNNs are not equivariant to rotations and scalings, for example, a correct classification of an image does not imply correct classification of its rotated and scaled counterpart.

More general than invariance is the property of *equivariance*. Bluntly speaking an operator ϕ is equivariant to a family of transformations if it commutes with each transformation of the family. A case of particular interest is when ϕ is equivariant with respect to a family of group actions (or semigroup actions, as will be seen in Chapters 2 and 3). Figure 1.2 shows an example of equivariance. Segmentation is a task where equivariance can be particularly important, as transforming an input image geometrically should have the same effect on its segmentation maps. In the case of image classification, for example, invariance can be interesting if the data contains the relevant symmetries, as illustrated in Figure 1.1. Invariant networks can be obtained from equivariant layers followed by pooling functions.

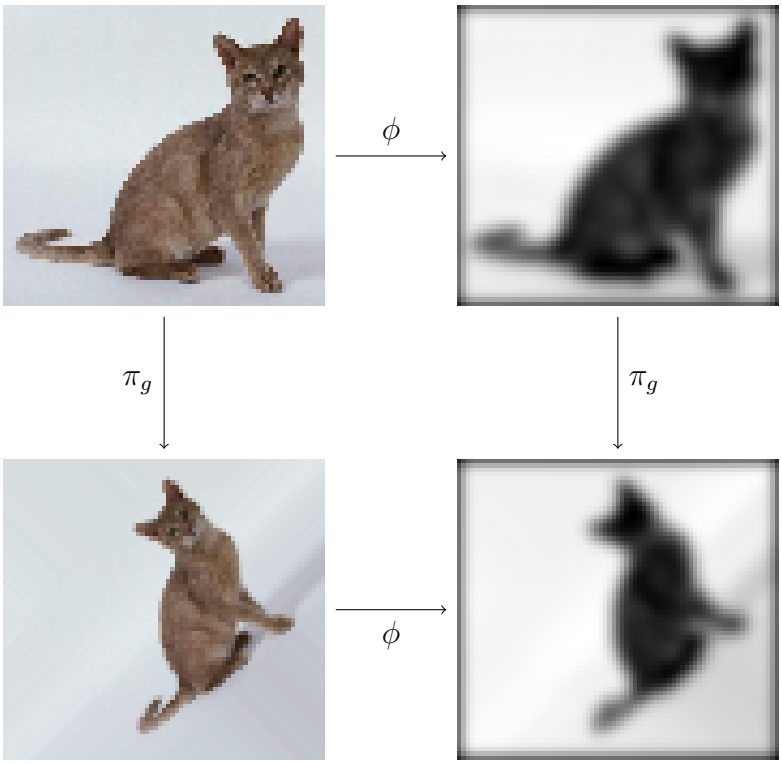


Figure 1.2: Example of equivariance to transformations (similarity transformations in this case). The application of ϕ, which can be thought of as an equivariant neural network layer commutes with the transformation π_g.

The field of Geometric Deep Learning (Bronstein et al., 2021) is a field of deep learning that incorporates symmetry in the neural networks. Group-Equivariant Neural Networks, in particular, are neural networks that are equivariant to certain group transformations. Despite being quite recent, many approaches to group equivariant neural networks exist for different groups: For images we have networks equivariant to 2D rotations (Cohen and Welling, 2016a; Worrall et al., 2017; Weiler et al., 2018b) and scale (Zhu et al., 2019; Lindeberg, 2022; Worrall and Welling, 2019); for point clouds there are 3D rotations equivariant networks (Thomas et al., 2018), for sets there are permutation equivariant networks (Guttenberg et al., 2016), physics-informed networks equivariant to Lorentz transformations (Bogatskiy et al., 2020) to name a few.

This chapter serves as an introduction to many of the topics that will be viewed during this thesis. It is organized as follows: A brief review of the definition and some properties of smooth manifolds and their tangent spaces is given in Section 1.2. Groups and group equivariance, one of the main topics in this thesis, are introduced and discussed in Section 1.3. Geometric deep learning, the area that concerns, among other things, equivariant neural networks, is discussed in Section 1.4. Fiber bundles are a concept from topology and differential geometry that will be useful in some parts of this thesis and is therefore introduced in Section 1.5. The theoretical framework behind many of the group equivariant neural networks, the group convolution, is introduced in Section 1.6, including the framework of classical convolution, the

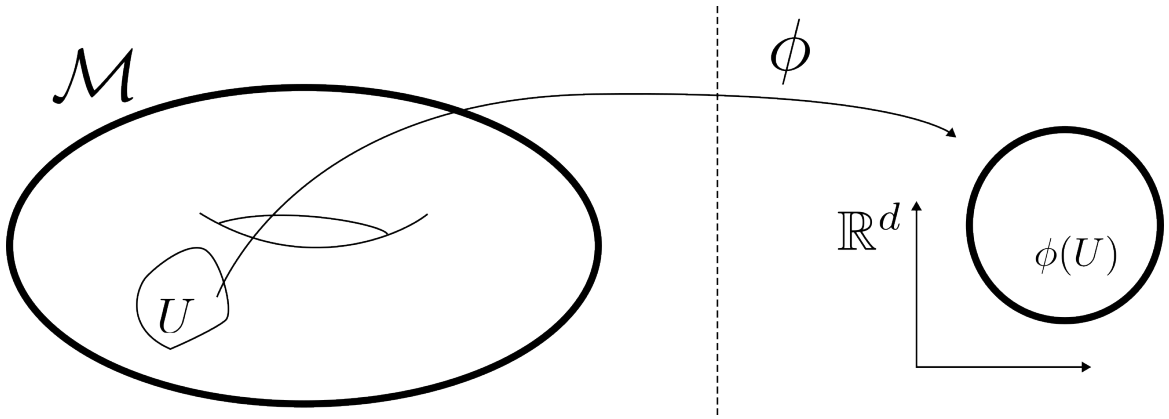


Figure 1.3: Illustration of a manifold $\mathcal{M}$ and a chart (U, ϕ) mapping to Euclidean space $\mathbb{R}^d$.

group convolution for signals on the group and convolutions by steerable filters.

1.2 Smooth Manifolds

Later in this chapter and in following chapters many objects are described in terms of *smooth manifolds* (Lee, 2013). This section serves merely to recall the definition and introduce some of the notations used when discussing manifolds, specially those topics which will be useful in the rest of the thesis. Essentially *manifolds* are objects that locally look like an Euclidean space. A *topological manifold* is a space which has the local topology of a Euclidean space:

Definition 1 (Topological manifold)**.** *A topological manifold $\mathcal{M}$ of dimension d is a topological space which is locally homeomorphic to $\mathbb{R}^d$, i.e. for every $p \in \mathcal{M}$ there exists an open set $U \subseteq \mathcal{M}$ containing p and an open set $V \subseteq \mathbb{R}^d$ such that U and V are homeomorphic.*

A topological manifold alone is not sufficient to allow for things such as differentiation. For that, additional structure will be necessary. A *smooth manifold* $\mathcal{M}$ is a topological manifold with the extra structure that allows the use of differential operations on it. In order to define smooth manifolds we start by defining charts and atlases.

Definition 2 (Chart, Atlas)**.** *Let $\mathcal{M}$ be a topological manifold of dimension d. A* chart *is a tuple (U, ϕ) where $U \subseteq \mathcal{M}$ is an open set and $\phi : U \to \mathbb{R}^d$ is a homeomorphism i.e. a bijection such that ϕ and ϕ^{-1} are continuous. An* atlas *is a collection of charts $(U_\alpha, \phi_\alpha)_{\alpha \in A}$ that covers $\mathcal{M}$, i.e. $\bigcup_{\alpha \in A} U_\alpha = \mathcal{M}$.*

Definition 3 (Smooth manifold)**.** *If $\mathcal{M}$ is a topological manifold of dimension n, $(U_\alpha, \phi_\alpha)_{\alpha \in A}$ is and atlas and for all $\alpha, \beta \in A$ the transition maps $\phi_\alpha \circ \phi_\beta^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ are smooth, then $\mathcal{M}$ is a smooth manifold.*

A smooth manifold may sometimes be described in terms of *local coordinates*. Given a chart (U, ϕ), a point $p \in U$ is written in local coordinates as $\mathbf{x} = \phi(p) = (x^1(p), x^2(p), \ldots, x^d(p))$.

Let $f : \mathcal{M} \to \mathbb{R}$ be a function. We say that f is *smooth* at point $p \in \mathcal{M}$ if there is a chart (U, ϕ) with $p \in U$, such that $f \circ \phi^{-1}$ is smooth, i.e. infinitely continuously differentiable, at

$\phi(p)$, as a function from $\mathbb{R}^d$ to $\mathbb{R}$. If f is smooth at all $p \in \mathcal{M}$ then f is smooth, also referred to as a smooth map. The set of smooth functions is denoted $C^\infty(\mathcal{M}, \mathbb{R})$.

Given that differentiability is defined, a way to compute derivatives in $\mathcal{M}$ is possible. Indeed, there are several definitions of derivative that apply to smooth manifolds, but here we only look at the tangent space sense of derivative, which generalizes the notion of directional derivative of scalar functions in $\mathbb{R}^d$ to the manifold $\mathcal{M}$. Given a smooth manifold $\mathcal{M}$ and a point $p \in \mathcal{M}$, a linear map $v : C^\infty(\mathcal{M}, \mathbb{R}) \to \mathbb{R}$ satisfying the Leibniz rule,

$$\forall f_1, f_2 \in C^\infty(\mathcal{M}, \mathbb{R}) \quad v(f_1 f_2) = f_1(p)v(f_2) + v(f_1)f_2(p) \tag{1.1}$$

is called a derivation at p. For all $p \in \mathcal{M}$, the set of derivations at p forms an d-dimensional real vector space which is denoted $T_p\mathcal{M}$ and is called the *tangent space* at p and its elements can also be called *tangent vectors*. In the Euclidean space, an operator satisfying (1.1) is the derivative along an specific direction, and this definition generalizes this to derivatives in general smooth manifolds.

Given a chart (U, ϕ) describing local coordinates $\mathbf{x} = \phi(p) = (x^1(p), \dots, x^n(p))$, we define $\frac{\partial}{\partial x^i}\Big|_p$ for $p \in \mathcal{M}$ for all $i \in \{1, \dots, d\}$ and $f \in C^\infty(\mathcal{M}, \mathbb{R})$, as

$$\frac{\partial}{\partial x^i}\bigg|_p f = \frac{\partial}{\partial x^i}\bigg|_{\phi(p)} (f \circ \phi^{-1}) \tag{1.2}$$

where the derivative right-hand side of the equation denotes partial differentiation in $\mathbb{R}^d$. The tangent vectors $\frac{\partial}{\partial x^1}\Big|_p, \dots, \frac{\partial}{\partial x^d}\Big|_p$ form a basis of $T_p\mathcal{M}$.

A *vector field* is a mapping that maps each point $p \in \mathcal{M}$ to a vector in $T_p\mathcal{M}$. A more precise definition of the vector field will be given when talking about fiber bundles. In that case, we can define $\frac{\partial}{\partial x^i}$, for $i \in \{1, \dots, d\}$ by associating each $p \in \mathcal{M}$ to $\frac{\partial}{\partial x^i}\Big|_p$.

1.3 Group Equivariance

Groups are objects that arise with the study of symmetry. One can define groups as follows:

Definition 4 (Group)**.** *A tuple $(G, \cdot)$ where G is a set and $" \cdot " : G \times G \to G$ a function is a group if*

- $\forall a, b, c \in G,\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$ *(Associativity)*
- $\exists e \in G, \forall a \in G\ e \cdot a = a \cdot e = a$ *(Neutral Element)*
- $\forall a \in G, \exists a^{-1} \in G,\ a \cdot a^{-1} = a^{-1} \cdot a = e$ *(Inverse)*

Groups are often used to model transformations on objects, because the composition of transformations defines an associative product, and they may be defined in terms of *symmetries* of objects, for example given the four vertices of a square $p_0 = (-1, -1)$, $p_1 = (1, -1)$, $p_2 =$

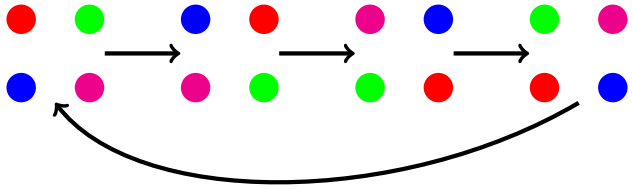


Figure 1.4: Illustration of the group $p4$. It can be seen as rotations by multiples of 90° on these points. As such it has a cyclic structure: if r is a rotation by 90°, its elements are $p4 = \{e, r, r^2, r^3\}$ where e is the identity.

$(-1, 1)$, $p_3 = (1, 1)$, the transformations that map the set $\{p_0, p_1, p_2, p_3\}$ to itself and preserve orientation (i.e. in this example orientation is the relative ordering when passing through the vertices in counter-clockwise direction) are described at Figure 1.4 by the group $p4$, which is also isomorphic to $\mathbb{Z}_4 = \mathbb{Z}/4\mathbb{Z}$, the integers modulo four.

Let us also discuss groups with additional structure that will be seen during the thesis. A group G which is also a topological space, for with the operators of product $(a, b) \mapsto a \cdot b$ and inverse $a \mapsto a^{-1}$ are continuous is a *topological group*. If additionally G is a smooth manifold (Lee, 2013) and the product and inverse are smooth maps, then G is a *Lie Group*.

Definition 5 (Lie Group)**.** *A* Lie group *is a group $(\mathcal{G}, \cdot)$ such that $\mathcal{G}$ is a smooth manifold and the product $(g, h) \mapsto g \cdot h$ and $g \mapsto g^{-1}$ are both smooth functions.*

A Lie group can model a continuous and smooth transformation between equivalent objects, an example could be obtained by the rotations of the circle, as they can be obtained by continuously rotating the circle, in contrast to the rotations $p4$ that need to be discretized.

The main way which groups are used, specially within the context of computer vision, it through group actions.

Definition 6 (Group Action)**.** *Given a set X and a group G, a* group action *action is a map $\pi : G \times X \to X$ such that $\pi(e, x) = x$ for all $x \in X$ where e is the identity in G, and satisfies one of the following properties:*

- $\forall g, h \in G, \forall x \in X \ \pi(g, \pi(h, x)) = \pi(g \cdot h, x)$ *in which case it is called a* left group action*;*
- $\forall g, h \in G, \forall x \in X \ \pi(g, \pi(h, x)) = \pi(h \cdot g, x)$ *in which case it is called a* right group action*.*

We often denote a group action π using subscripts, $\pi_g(x) \coloneqq \pi(g, x)$, $\forall g \in G, x \in X$, and π_g is also used to denote the function $x \mapsto \pi_g(x)$. In that sense, π is a group action if and only if the operators π_g, $g \in G$ together with the composition $\circ$ form a group such that the map $g \mapsto \pi_g$ is a group homomorphism. If G is a Lie Group and X a smooth manifold, π is a *Lie group action* if it is a group action and a smooth map. If the group action being used is clear from the context we denote simply $\pi_g(x) = g \cdot x, \ \forall x \in X, g \in G$.

Group actions model a wide range of geometric transformations e.g. translations, rotations, or the operators in Chapter 2 called homothecies or scalings, etc. For example, a very interesting Lie group is the group of invertible $d \times d$ real matrices $\mathrm{GL}(d)$ equipped with the matrix multiplication rule. Any subset of invertible matrices which is closed under multiplication and inversion is also a group and if it has a smooth manifold structure it is also a Lie group. This gives rise to many interesting Lie groups, for example:

- the orthogonal group (i.e. rotations and reflections in the plane) $\mathrm{O}(d) = \{A \in \mathrm{GL}(d) | A^T A = AA^T = I_n\}$ where I_d is the $d \times d$ identity matrix;
- the special linear group $\mathrm{SL}(d) = \{A \in \mathrm{GL}(d) | \det(A) = 1\}$;
- and the special orthogonal group $\mathrm{SO}(d) = \{A \in \mathrm{O}(d) | \det(A) = 1\}$ i.e. rotations in the plane.

The elements of SO(2) for example may be written as matrices of the form

$$\forall \theta \in \mathbb{R}, R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{1.3}$$

which are referred to as rotations matrices. The group $p4$ is isomorphic to a subgroup of $SO(2)$, which we identify $p4 = \{R_0, R_{\frac{\pi}{2}}, R_\pi, R_{\frac{3\pi}{2}}\}$ and we can verify it maps the vertices of the square to the to the vertices of the square and preserves orientations, but it is not a Lie subgroup.

Groups which contain translations as well as invertible linear transformations can also be written as a subgroup of $\mathrm{GL}(d)$ suppose that $G \ltimes \mathbb{R}^d$ is the product of a group of matrices $G \subseteq \mathrm{GL}(d)$ and translations, then its composition has the form

$$(A, \mathbf{v}) \cdot (B, \mathbf{w}) = (AB, A\mathbf{w} + \mathbf{v}). \tag{1.4}$$

However, if we identify its elements with the $d+1 \times d+1$ matrices

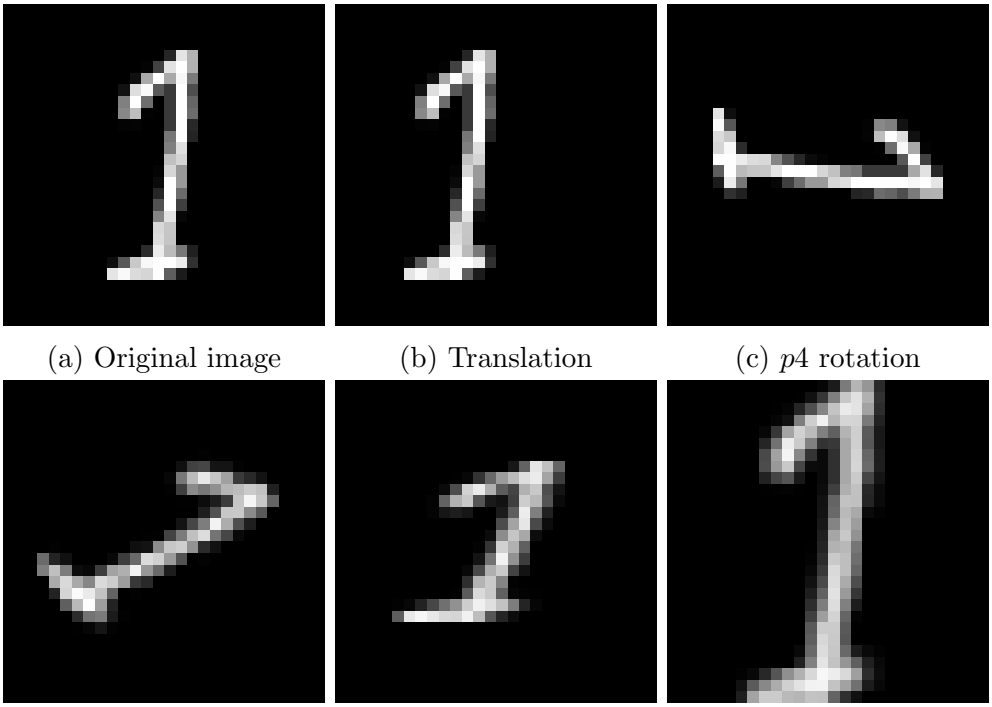
$$(A, \mathbf{v}) = \begin{bmatrix} A & \mathbf{v} \\ 0 & 1 \end{bmatrix}, \tag{1.5}$$

we end with the same composition group, meaning that $G \ltimes \mathbb{R}^d$ can be viewed as a subgroup of $\mathrm{GL}(d+1)$. Interesting groups that can be represented this way are

- affine transformations $\mathrm{Aff}(d) = \mathrm{GL}(d) \ltimes \mathbb{R}^d$;
- equi-affine transformations $\text{Equi-Aff}(d) = \mathrm{SL}(d) \ltimes \mathbb{R}^d$ of area-preserving transformations;
- Euclidean group $\mathrm{E}(d) = \mathrm{O}(d) \ltimes \mathbb{R}^d$ of transformations that preserve Euclidean distance i.e. rotations, translations and reflections;
- and the special Euclidean group $\mathrm{SE}(d) = \mathrm{SO}(d) \ltimes \mathbb{R}^d$ of operators that preserve Euclidean distance and orientation i.e. rotations and translations.

Another matrix group is a group of scalings which can be written as $S = \mathbb{R}_{>0} = (0, +\infty)$ and is already given matrix form for 1×1 matrices, but can be a matrix of arbitrary dimension by writing $S \cong \{s \cdot I_d | s > 0\}$. Furthermore S can be composed with translations $\mathbb{R}^d$ to form a group of scales and translations $S \ltimes \mathbb{R}^d$.

In many computer vision tasks, symmetry to transformations can be found in the data e.g. in segmentation tasks, the prediction of an image spatially shifted by $\mathbf{v} \in \mathbb{Z}^2$ at the pixel position $\mathbf{x}$ should result in the same class as the pixel in $\mathbf{x} - \mathbf{v}$ in the original image. In particular, the action of a group on $\mathbb{Z}^d$ also defines an action on the space of functions mapping



(a) Original image (b) Translation (c) $p4$ rotation

(d) $SO(2)$ rotation (e) Affine transformation (f) Scaling

Figure 1.5: Examples of group actions on images.

$X = \mathbb{Z}^d$ to $Y = \mathbb{R}^C$, which can represent a 1D signal, an image ($d = 2$) or a volume ($d = 3$). This action is defined for all $f \in Y^X$ by

$$(g \cdot f)(\mathbf{x}) = \pi_g^{Y^X}[f](\mathbf{x}) = f(\pi_{g^{-1}}^X[\mathbf{x}]) = f(g^{-1} \cdot \mathbf{x}). \tag{1.6}$$

For example, the translations on X induce translations on Y^X, as rotations by $p4$ define rotations by multiples of 90° in images. We can also view f as a discretization of a signal on $X' = \mathbb{R}^d$. In that way, we can apply the action of Lie groups such as $\mathrm{Aff}(d)$, $SO(d)$ or S on the points in X, mapping the grid X to another grid in $\mathbb{R}^d$ and use interpolation to approximate a transformed image in X. Examples of the aforementioned actions are shown in Figure 1.5.

By ensuring that a neural network treats transformed samples in an equivalent manner we can effectively reduce the search space and improve generalization. To include this constraint in an operator we require that it is *G-equivariant*.

Definition 7 (Equivariance)**.** *Let G be a group acting on sets X and Y with actions denoted by[1] $\pi^X : G \times X \to X$ and $\pi^Y : G \times Y \to Y$, respectively. We say that an operator $\phi : X \to Y$ is G-equivariant with respect to π^X and π^Y if for all $g \in G$*

$$\forall g \in G, \quad \phi \circ \pi_g^X = \pi_g^Y \circ \phi. \tag{1.7}$$

In most cases the actions π^X and π^Y are clear from the context and we say simply that ϕ is G-equivariant.

[1]To avoid triviality, we assume that π_g^X is not the identity for all $g \in G$.

The example in Figure 1.2 illustrates the concept of equivariant operators. Note that it is not necessary for the actions on the input and output domains to match.

Invariance is a particular case of equivariance and can be obtained when $\forall g \in G, \pi_g = \mathrm{id}$, i.e. an operator ϕ is *G-invariant* with respect to π if

$$\phi \circ \pi_g = \mathrm{id} \circ \phi = \phi. \tag{1.8}$$

Again if π is clear from the context we say that ϕ is G-invariant.

The main concern in this thesis is about constructing, training and evaluating neural networks that are equivariant with respect to some transformation.

1.4 Geometric Deep Learning

Group equivariant neural networks generally fall under the broader field of *Geometric Deep Learning*. We assume that inputs are signals $f : \Omega \to X$ defined on some domain Ω. The domain Ω can be several different things, which are assumed to fall into one of the so-called 5 G's (Bronstein et al., 2021): Grids, Graphs, Groups, Geodesics and Gauges. A neural network in this framework will be a function of the $\phi : X^\Omega \to Y$, where X and Y are finite-dimensional vector spaces.

The models presented in this thesis follow the *Geometric Deep Learning Blueprint* (Bronstein et al., 2021) (GDLBP), which characterizes some of the fundamental building blocks of geometric deep learning.

It is assumed that we aim for the network ϕ to be equivariant to the action of a group G. The group G acts on the spaces X, Y and Ω with actions denoted by π. On spaces of functions of these sets the action is given by (1.6).

The blocks characterized by GDLBP are:

- **Linear G-equivariant layer**: A linear operator $B : X^\Omega \to Y^\Omega$ which satisfies $B \circ \pi_g = \pi_g \circ B$
- **Equivariant Non-Linearity**: A pointwise function $\sigma : X \to Y$ satisfying $\sigma \circ \pi_g = \pi_g \circ \sigma$
- **Local Pooling**: $P : X^\Omega \to X^{\Omega'}$ such that $\Omega' \subseteq \Omega$
- **G-invariant layer**: (also referred to as projection) An operator $A : Y^\Omega \to Y$ such that $A \circ \pi_g = A$

Assuming that besides the projection all of the parts above are G-equivariant, a G-equivariant function can then be constructed by composition of these blocks, particularly

$$\phi = \sigma_l \circ B_l \circ P_{l-1} \circ \cdots \circ P_1 \circ \sigma_1 \circ B_1. \tag{1.9}$$

and a G-invariant function is given by $A \circ \phi$.

1.5 Fiber Bundles

The general theory behind group convolutional neural networks can be based on considering spaces as *fiber bundles*(Cohen et al., 2019). Fiber bundles are objects that locally look like the Cartesian product of two topological spaces, but globally their topology is not necessarily that of the Cartesian product. Formally a fiber bundle is given by Definition 8.

Definition 8 (Fiber bundle)**.** *A fiber bundle is a tuple* $(E, B, \mathbf{p}, F)$ *where* E*,* B *and* F *are topological spaces and* $\mathbf{p} : E \to B$ *is a continuous surjection satisfying the condition of* local triviality*: for every* $x \in B$ *there is an open neighborhood* $U \subseteq B$ *containing* x *and a homeomorphism* $\varphi : \mathbf{p}^{-1}(U) \to U \times F$*, referred to as* trivialization*, such that for every* $y \in \mathbf{p}^{-1}(U)$

$$proj_1 \circ \varphi(y) = \mathbf{p}(y) \tag{1.10}$$

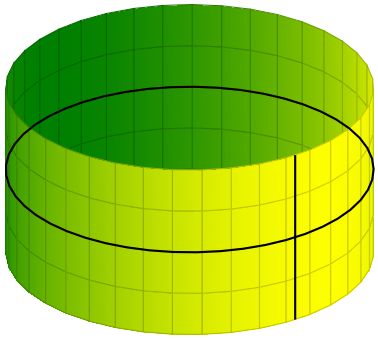
where $proj_1$ *is the projection in the first coordinate.*

The spaces E*,* B *and* F *are referred to, respectively, as the total space, the base space and the canonical fiber.*

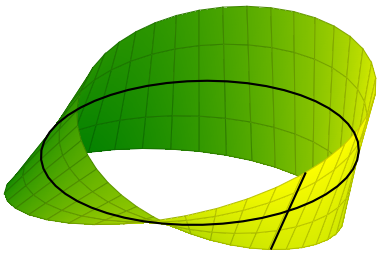
The fiber bundle is such that locally, E "looks like" a Cartesian product of B and F. As examples let us consider the cylinder and the Möbius band. The cylinder can be written as the product $S^1 \times [-1, 1]$ where $S^1 = \{x \in \mathbb{R}^2 | \|x\| = 1\}$ is the circle. The base space and canonical fiber are respectively S^1 and $[-1, 1]$ and as a projection we can use $\mathbf{p} = \text{proj}_1$ and in that case the trivialization is just the identity. The cylinder is a trivial fiber bundle, because it coincides with a Cartesian product. Now for the Möbius band, let the total space be $E = [0, 1] \times [-1, 1] / \sim$ where the equivalence relation $\sim$ is given by $(x, y) \sim (x', y')$ if and only if $(x, y) = (x', y')$, or $x = 0, x' = 1, y = -y'$, or $x = 1, x' = 0, y = -y'$ for all $x \in \mathbb{R}$. The quotient space E can be interpreted as giving a half-twist to one vertical end of $[0, 1] \times [-1, 1]$ and gluing them together, giving rise to the example in Figure 1.6(b). We can define the projection $\mathbf{p} : E \to S^1$, as $q(x, y) \mapsto e^{2i\pi x}$ where q maps $(x, y) \in [0, 1] \times [-1, 1]$ to its equivalence class. This projection is well defined because it maps 0 and 1 to the same point on the circle. To find a local trivialization notice that given a point $(x, y) \in [0, 1] \times [-1, 1]$ one can find a small enough open set $U \subset [0, 1] \times [-1, 1]$ such that q is a homeomorphism restricted to U and then the inverse of q can be used to construct a trivialization.

Definition 9 (Section of a fiber bundle)**.** *A* section *on a fiber bundle is a function assigning to each* $x \in B$ *an element in its fiber own* $\mathbf{p}^{-1}(x)$*. Formally, a section is a continuous function with the property* $\mathbf{p} \circ s = \text{id}_B$*.*

A prominent example of a fiber bundle is the tangent bundle. In a manifold $\mathcal{M}$, associated to each point $\mathbf{x} \in \mathcal{M}$ is the tangent space $T_{\mathbf{x}}\mathcal{M}$. Keeping in mind that the tangent spaces are isomorphic between themselves, their disjoint union $T\mathcal{M} = \coprod_{\mathbf{x} \in \mathcal{M}} T_{\mathbf{x}}\mathcal{M}$ forms the total space of a fiber bundle. The projection function $\mathbf{p}$ is simply defined as $\mathbf{p}(\mathbf{z}) = \mathbf{x}$ for all $\mathbf{z} \in T_{\mathbf{x}}\mathcal{M}$, $\mathbf{x} \in \mathcal{M}$, the base is $\mathcal{M}$ and the fiber is any vector space $V \cong T_{\mathbf{x}}\mathcal{M}$ isomorphic to $T_{\mathbf{x}}\mathcal{M}$ for any $\mathbf{x} \in \mathcal{M}$. More generally we can define a vector bundle if we use an arbitrary vector space V as the fiber. This will play an interesting role in group convolutions as the application of a group representation on V defines a group representation of the sections of the total space E. A vector field can now be more precisely defined as a section in $T\mathcal{M}$.



(a) Cylinder



(b) Möbius band

Figure 1.6: A cylinder and a Möbius band are both examples of fiber bundles, with the same base spaces and fibers, however while the cylinder has the same topology as the Cartesian product, the band has a different topology.

Fiber bundles will be used later in this chapter to talk about the group convolution in homogeneous spaces. In Chapter 4 we use the jet-bundle, which is a fiber bundle that contains differential information of a manifold, but the study of fiber bundles is not necessary in that case because in our cases of interest the jet-bundle reduces to a Cartesian product.

1.6 Group Convolution

As outlined in the GDLBP, a linear-equivariant operator is used as a building block in many equivariant neural networks approaches. In the case of regular CNN this operator is the convolution, but more generally the group convolution can be defined for group-equivariant neural networks. We begin this section by analyzing some aspects of the convolution in Euclidean spaces before we examine the general case of group-convolution.

1.6.1 Special Case: Classical Convolution

Linear operators and have a range of interesting properties for a neural network. Moreover, in our case we are particularly interested in having equivariant neural networks. Therefore let us start our analysis by examining the most-used example of a group-equivariant operator in deep learning: the convolution.

A convolution is an operator defined for signals on a domain E, in which we assume either $E = \mathbb{R}^d$ or $E = \mathbb{Z}^d$. Let us analyze, for instance the convolution of a multivariate signal $f : E \to \mathbb{R}^C$ by a filter $w : E \to \mathbb{R}^{C \times C'}$, which is defined as

$$(f * w)(\mathbf{x}) = \int_{\mathbb{R}^d} w(\mathbf{x} - \mathbf{y}) \cdot f(\mathbf{y}) d\mathbf{y}, \tag{1.11}$$

if $E = \mathbb{R}^d$ and

$$(f * w)(\mathbf{x}) = \sum_{\mathbf{y} \in E} w(\mathbf{x} - \mathbf{y}) \cdot f(\mathbf{y}), \tag{1.12}$$

if $E = \mathbb{Z}^d$, where in both cases $\cdot$ stands for matrix product and the output is a multivariate signal $f * w : E \to \mathbb{R}^{C'}$. Alternatively, for each output coordinate o, $1 \leq o \leq C'$

$$(f * w)(\mathbf{x})_o = \sum_{i=1}^{C} \int_{\mathbb{R}^d} w(\mathbf{x} - \mathbf{y})_{i,o} \cdot f(\mathbf{y})_i d\mathbf{y}, \tag{1.13}$$

for $E = \mathbb{R}^d$,

$$(f * w)(\mathbf{x})_o = \sum_{i=1}^{C} \sum_{\mathbf{y} \in E} w(\mathbf{x} - \mathbf{y})_{i,o} \cdot f(\mathbf{y})_i, \tag{1.14}$$

for $E = \mathbb{Z}^d$. Note that these definitions hold provided the functions $\mathbf{y} \mapsto w(\mathbf{x} - \mathbf{y}) \cdot f(\mathbf{y})$ and $\mathbf{y} \mapsto w(\mathbf{x} - \mathbf{y})_{i,o} \cdot f(\mathbf{y})_i$ are integrable for almost every $\mathbf{x} \in \mathbb{R}^d$, and the sums $\sum_{\mathbf{y} \in E} w(\mathbf{x} - \mathbf{y}) \cdot f(\mathbf{y})$ and $\sum_{\mathbf{y} \in E} w(\mathbf{x} - \mathbf{y})_{i,o} \cdot f(\mathbf{y})_i$ converge for all $\mathbf{x} \in \mathbb{R}^d$. A convolution can be viewed as a the filter w being placed through a translation at every point in the domain, followed by the application of the filter through a scalar product.

Let us focus on the discrete case $E = \mathbb{Z}^d$. In the continuous case, computations are similar, albeit requiring a bit more formalism. Let us define the p-norms, for $p > 1$ as $\|f\|_p = \left(\sum_{\mathbf{y} \in E} |f(\mathbf{y})|^p \right)^{\frac{1}{p}}$ and $\|f\|_\infty = \sup_{\mathbf{y} \in E} |f(\mathbf{y})|$ and $\ell_p = \{f : E \to \mathbb{R} | \|f\|_p < \infty\}$. From now on we assume that $1 \leq i \leq C$ and $1 \leq o \leq C'$, $f_i \in \ell_p$ and $w_{i,o} \in \ell_q$ for some $p > 1$ or $p = \infty$ and q such that $\frac{1}{p} + \frac{1}{q} = 1$. By Hölder's Inequality, the sums in (1.14) converge for all $\mathbf{y} \in \mathbb{Z}^d$.

We define translation of signals on E by vectors $\mathbf{v} \in E$ as the action (1.6), i.e.

$$\forall \mathbf{v}, \mathbf{x} \in E, \quad \pi_{\mathbf{v}}[f](\mathbf{x}) = f(\mathbf{x} - \mathbf{v}), \tag{1.15}$$

where we use the same symbol to denote the actions on both $(\mathbb{R}^C)^E$ and $(\mathbb{R}^{C'})^E$.

It is straightforward to see that for a fixed w, the operator defined by (1.12) is linear and translation equivariant, moreover the converse is also true: if $\varphi : (\mathbb{R}^C)^E \to (\mathbb{R}^{C'})^E$ is such that

- φ is linear
- $\pi_{\mathbf{v}} \circ \varphi = \varphi \circ \pi_{\mathbf{v}}$

then φ is a convolution by some filter w. What follows is a proof.

Proof. Assume that φ is linear and translation-equivariant. In the case where $E = \mathbb{Z}^d$, we can characterize a signal f by computing its impulse responses, i.e. linear functionals given as convolutions by the signals $\delta_{\mathbf{y}} e_i$ where $\delta_{\mathbf{y}}$ is given by

$$\delta_{\mathbf{y}}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{y} \\ \vec{0}, & \text{if } \mathbf{x} \neq \mathbf{y} \end{cases} \tag{1.16}$$

and e_i is the i-th vector of the canonical bases on $\mathbb{R}^m$ i.e. equal to 1 at the i-th coordinate and 0 at the others. Applying this to f gives us $(f * \delta_{\mathbf{y}} e_i)(\mathbf{x}) = f(\mathbf{x} - \mathbf{y})_i$, therefore for all $\mathbf{x} \in E$

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{C} (f * \delta_0)(\mathbf{x}) e_i \\ &= \sum_{i=1}^{C} \sum_{\mathbf{y} \in E} \delta_0(\mathbf{x} - \mathbf{y}) e_i \cdot f(\mathbf{y}) \\ &= \sum_{i=1}^{C} \sum_{\mathbf{y} \in E} \delta_{\mathbf{y}}(\mathbf{x}) e_i \cdot f(\mathbf{y}). \end{aligned} \tag{1.17}$$

For all $\mathbf{x} \in E$, $1 \leq i \leq C$, $1 \leq j \leq C'$ let $h_{ij}(\mathbf{x}, \mathbf{y}) = \varphi(e_i \delta_{\mathbf{y}})_j(\mathbf{x}) = \varphi(\delta_{\mathbf{y}})$ by linearity we obtain the following

$$\begin{aligned} \varphi(f)(\mathbf{x}) &= \varphi\left(\sum_{i=1}^{C} \sum_{\mathbf{y} \in E} (e_i \delta_{\mathbf{y}}) \cdot f(\mathbf{y}) \right)(\mathbf{x}) \\ &= \left(\sum_{i=1}^{C} \sum_{\mathbf{y} \in E} \varphi(e_i \delta_{\mathbf{y}}) f(\mathbf{y}) \right)(\mathbf{x}) \\ &= \sum_{i=1}^{C} \sum_{\mathbf{y} \in E} h(\mathbf{x}, \mathbf{y})_i \cdot f(\mathbf{y}) \\ &= \sum_{\mathbf{y} \in E} h(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{y}), \end{aligned} \tag{1.18}$$

where $h(\mathbf{x}, \mathbf{y})_i$ denotes the i-th row of $h(\mathbf{x}, \mathbf{y})$. Since φ is translation-equivariant, we have

$$h_i(\mathbf{x}, \mathbf{y}) = \varphi(e_i, \delta_{\mathbf{y}}(\mathbf{x}))_j = \varphi(e_i \pi_{\mathbf{y}}[\delta_0](\mathbf{x}))_j = \varphi(e_i, \delta_0)(\mathbf{x} - \mathbf{y})_j = h_{ij}(0, \mathbf{x} - \mathbf{y}), \tag{1.19}$$

defining $w(\mathbf{y}) \coloneqq h(0, \mathbf{y})$ we obtain

$$\begin{aligned} \varphi(f)(\mathbf{x}) &= \sum_{\mathbf{y} \in E} h(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{y}) \\ &= \sum_{\mathbf{y} \in E} w(\mathbf{x} - \mathbf{y}) \cdot f(\mathbf{y}) = (f * w)(\mathbf{x}). \end{aligned} \tag{1.20}$$

□

1.6.2 Haar Measure

The group convolution will be defined by means of an integral for signals on the group, like the convolution is defined as an integral of signals on E. For such we define the Haar Measure.

Given a group G, a set $S \subseteq G$ and $g \in G$ let us define the left-translation of S by g as

$$g \cdot S = \{g \cdot s | s \in S\}. \tag{1.21}$$

Let G be a topological group and σ a σ-algebra generated by the Borel sets of G. A measure $\mu : \sigma \to \mathbb{R}^+$ is called left-invariant if $\forall S \in \sigma$ we have[2]

$$\mu(g \cdot S) = \mu(S). \tag{1.22}$$

[2]Note that $S \in \sigma$ implies $g \cdot S \in \sigma$, as the group multiplication is continuous and therefore the translation of a Borel set is a Borel set.

Haar's theorem states that, up to a multiplicative constant, there is a unique left-invariant measure that is finite on compact sets, outer regular on Borel sets and inner regular on open sets. Such a measure is called Haar measure. In the following subsection, integration in the group G is performed with respect to a Haar measure.

The Lebesgue measure is an example of Haar measure of $\mathbb{R}^d$, invariant to translation. When $G = GL(n)$ is a matrix group, the Haar measure can be obtained by $d\mu(A) = \frac{1}{|\det A|^d} dA$, where dA here denotes the Lebesgue measure of $GL(n)$. Note that it must also be the case for subgroups of $GL(n)$.

1.6.3 Group Convolution

The group convolution generalizes the convolution in the case $G = E$ is a more general group. Particularly let us assume that G is a locally compact topological group. Indeed, we begin with a signal $f : G \to \mathbb{R}^C$ and wish to compute an operator that is both linear and G-equivariant, where the group action of G on signals like f is given by

$$\forall g, h \in G, \pi_g[f](h) = (g \cdot f)(h) \coloneqq f(g^{-1}h). \tag{1.23}$$

The group convolution operators can be deduced from the Haar measure using a definition similar to the convolution. Assuming that $f : E \to \mathbb{R}$ and $w : E \to \mathbb{R}$ are signals on the group, by looking at (1.11) substitute the expression $\mathbf{x} - \mathbf{y}$ for the translation group by the expression $\mathbf{x} \cdot \mathbf{y}^{-1}$ in a more general group.[3] A convolution-like operator for a topological group G can be written in terms of the Haar measure

$$(f * w)(g) = \int_G w(h^{-1}g) f(h) dh = \int_G w(h^{-1}) f(gh) dh, \tag{1.24}$$

and indeed we can show that it is equivariant with respect to transformations $[g' \cdot f](g) = f(g'^{-1}g)$

$$\begin{aligned} ([g' \cdot f] * w)(g) &= \int_G w(h^{-1}g)[g \cdot f](h) dh \\ &= \int_G w(h^{-1}g) f(g'^{-1}h) dh \\ &= \int_G w(h^{-1}g'^{-1}g) f(h) dh \\ &= (f * w)(g'^{-1}g) = [g' \cdot (f * w)](g). \end{aligned} \tag{1.25}$$

In the case where G is a discrete group the Haar measure is the counting measure and the integral in (1.24) becomes a sum

$$(w * f)(g) = \sum_{g' \in G} w(g'^{-1}g) f(g'). \tag{1.26}$$

Circling back to where we started, if we replace G by $\mathbb{Z}^d$ and $\mathbf{x} = g$, $\mathbf{y} = g'$ in (1.26) then $g'^{-1}g$ becomes $\mathbf{x} - \mathbf{y}$ and we obtain (1.12). Analogously if we assume $G = \mathbb{R}^d$ in the continuous

[3] using the group operation $\cdot = +$ we have that $\mathbf{x} - \mathbf{y} = \mathbf{x} \cdot \mathbf{y}^{-1} = \mathbf{y}^{-1} \cdot \mathbf{x}$ because the operation is commutative, we use the first one in the definition of group convolution

definition (1.11), it becomes the classical continuous convolution. While the convolution can be viewed as a filter being translated over the Euclidean domain and applied via a scalar product, the group convolution can be viewed as a filter being transformed according to the group composition law before begin applied via a scalar product.

As an example of the group convolution consider one of the first group-equivariant convolutional networks that is equivariant to something besides translations in the literature, the discrete rotations and translations equivariant neural network[4] proposed by Cohen and Welling (2016a). The group $p4$ is identified by the group of 90° rotations on the plane, making the group $G = p4 \times \mathbb{Z}^2$ a group of translations and rotations on the (discrete) plane $\mathbb{Z}^2$. The group multiplication on G is given, for $(R_1, \mathbf{v}_1), (R_2, \mathbf{v}_2) \in p4 \times \mathbb{Z}^2 = G$

$$(R_1, \mathbf{v}_1) \cdot (R_2, \mathbf{v}_2) = (R_1 \cdot R_2, R_1\mathbf{v}_2 + \mathbf{v}_1), \tag{1.27}$$

where R_1, R_2 are identified by rotation matrices of angles multiples of 90° and thus can be applied to vectors $\mathbf{v} \in \mathbb{Z}^2$.

Since $p4 \times \mathbb{Z}^2$ is discrete we can apply the definition (1.26) and obtain

$$(w * f)(R, \mathbf{x}) = \sum_{P \in p_4} \sum_{\mathbf{y} \in \mathbb{Z}^2} w(P^{-1}R, \mathbf{x} - P^{-1}\mathbf{y}) f(P, \mathbf{y}). \tag{1.28}$$

So in comparison with the $\mathbb{Z}^2$ convolution, which translates a filter by every possible vector in $\mathbb{Z}^2$, the $p4 \times \mathbb{Z}^2$, convolutions translate and rotate by multiples of 90° the filters and because filters are in $p4 \times \mathbb{Z}^2$ the rotation is accompanied by a cyclic translation in the first coordinate.

An issue is that input and output signals of neural networks are not often in a domain G when $G \neq \mathbb{Z}^d$, but usually they have the form $f : \mathbb{Z}^d \to \mathbb{R}^C$ or $f : \mathbb{R}^d \to \mathbb{R}^C$. In the previous example, inputs are supposed to be images on $\mathbb{Z}^2$ and because $G = p4 \times \mathbb{Z}^2$ we can apply the so-called *lifting* to these operators, which we will talk about in more detail in the next chapter, but in this case a valid way to compute the network would be to use a signal $\bar{f} : G \to \mathbb{R}^C$ such that $\forall P \in p4$, $\mathbf{x} \in \mathbb{Z}^2$, $\bar{f}(P, \mathbf{x}) = f(\mathbf{x})$ as input. In this way we can apply a group convolutional neural network to an input $f : \mathbb{Z}^2 \to \mathbb{R}^C$. To show that this architecture is equivariant first we must know how the group acts on f, which is $((P, \mathbf{y}) \cdot f)(\mathbf{x}) = (P\mathbf{x} + \mathbf{y})$. We can verify that the lifting is equivariant $g \bar{\cdot} f = g \cdot \bar{f}$, and moreover if the network is constructed only from group convolutions and point-wise functions it becomes evident that the result is equivariant as the composition of equivariant operators is equivariant. The visualization of the group action on feature maps of a network like this is seen in Figure 1 in Cohen and Welling (2016a).

1.6.4 Steerable Filters

When discussing steerable networks, group actions generally have additional structure. Specifically we have a *group representation* acting on the function spaces. A group representation is a mapping of group elements into linear maps i.e. $\tau : G \to \mathrm{GL}(V)$, where $\mathrm{GL}(V)$ is the set of linear automorphisms of a vector space V, such that $\tau(g) \cdot \tau(h) = \tau(g \cdot h)$. We can check that a group representation defines a group action $(g, y) \mapsto \tau(g)y$ for $g \in G$ and $y \in \mathbb{R}^C$. In the following we assume that either $E = \mathbb{Z}^d$ or $E = \mathbb{R}^d$.

[4] In the paper the network is also equivariant to reflections, but we examine a simpler case here.

A direct sum of group representations $\tau_1 : G \to \mathrm{GL}(V_1)$, $\tau_2 : G \to \mathrm{GL}(V_2)$ is a new representation in $\mathrm{GL}(V_1 \oplus V_2)$ given by a block diagonal matrix as follows

$$\tau(g) = \tau_1(g) \oplus \tau_2(g) = \begin{bmatrix} \tau_1(g) & 0 \\ 0 & \tau_2(g) \end{bmatrix}. \tag{1.29}$$

Steerable CNNs work by defining equivariant filter banks. More specifically, they are usually defined by means of a classical convolution by an equivariant filter bank. This leads to equivariance to the restrictions to groups G that are the product of $(E, +)$ with a group $(H, \cdot)$ that fixes the origin 0. In this sense, we view a filter bank as a convolutional filter $w : E \to \mathbb{R}^{C \times C'}$. Equivariant filter banks refer to filter banks with the following property: for all $h \in H$, $f : E \to \mathbb{R}^C$, $\mathbf{x} \in E$

$$\tau(h)(f * w)(\mathbf{x}) = (\pi_h f * w)(\mathbf{x}) \tag{1.30}$$

where τ is a representation of H and π is the action (1.6), which is also a representation. More specifically τ is a representation on $\mathbb{R}^{q'}$ and π is a representation for the space of functions $(\mathbb{R}^C)^E$.

A subspace $W \subseteq V$ is called invariant if the representation $\tau(g)$ maps W to W i.e. $\tau(g) \cdot W = W$ for all $g \in G$. A representation τ is called *irreducible* if it only has no non-trivial invariant subspaces, i.e. if the only invariant subspaces are V and $\{0\}$. The term *irrep* is used for short to refer to an irreducible representation.

If G is finite or if G is a semisimple Lie group we have that any representation can be broken into a direct sum of irreps and a change of basis, that is, for some $A \in \mathrm{GL}(V)$, and for all $g \in G$

$$A \cdot \tau(g) \cdot A^{-1} = \varphi_1(g) \oplus \varphi_2(g) \oplus \cdots \oplus \varphi_k(g) = \begin{bmatrix} \varphi_1(g) & & & \\ & \varphi_2(g) & & \\ & & \ddots & \\ & & & \varphi_k(g) \end{bmatrix} \tag{1.31}$$

where φ_i, $i \in \{1, \ldots, k\}$ are irreps.

Steerable CNNs profit from (1.31) to create equivariant operators based on multiple operators filter banks that are equivariant with respect to multiple irreps, i.e. the representations in the left-hand-side of (1.30) are given by different irreps.

Let us consider a simplified example to illustrate how a steerable network may be computed from the definitions above. Let us work again to construct a $p4 \times \mathbb{Z}^2$-equivariant linear operator. In that way we have $H = p4$. The input actions of $p4$ are rotations on E and the action of $G = p4 \times \mathbb{Z}^2$ is the action π_g defined in the last subsection. Let us define a representation of $p4$ in $\mathbb{R}^3$ given by $\tau(R) = \varphi_1(R) \oplus \varphi_2(R)$, where $\varphi_1(R) = I_1 = 1$ is the identity in $\mathbb{R}$ and $\varphi_2(R) = R$ used R as a rotation matrix by multiples of $90°$ in $\mathbb{R}^2$. We can make a steerable filter basis by making a filter $w^i = (w_1^i, w_2^i, w_3^i) : \mathbb{Z}^2 \to \mathbb{R}^3$ $i = 1, 2, 3, 4, 5, 6$ where the filters are given by functions with support in a 3×3 cross in $\mathbb{Z}^2$ given by.

$$\begin{array}{lll|lll}
w_1^1 = \begin{smallmatrix} 0&1&0 \\ 1&1&1 \\ 0&1&0 \end{smallmatrix} & w_2^1 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_3^1 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_1^2 = \begin{smallmatrix} 0&0&0 \\ 0&1&0 \\ 0&0&0 \end{smallmatrix} & w_2^2 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_3^2 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} \\
w_1^3 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_2^3 = \begin{smallmatrix} 0&0&0 \\ 0&0&1 \\ 0&0&0 \end{smallmatrix} & w_3^3 = \begin{smallmatrix} 0&1&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_1^4 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_2^4 = \begin{smallmatrix} 0&1&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_3^4 = \begin{smallmatrix} 0&0&0 \\ 1&0&0 \\ 0&0&0 \end{smallmatrix} \\
w_1^5 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_2^5 = \begin{smallmatrix} 0&0&0 \\ 1&0&0 \\ 0&0&0 \end{smallmatrix} & w_3^5 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&1&0 \end{smallmatrix} & w_1^6 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{smallmatrix} & w_2^6 = \begin{smallmatrix} 0&0&0 \\ 0&0&0 \\ 0&1&0 \end{smallmatrix} & w_3^6 = \begin{smallmatrix} 0&0&0 \\ 0&0&1 \\ 0&0&0 \end{smallmatrix}
\end{array} \tag{1.32}$$

Given some $w = \sum_{i=1}^{6} \alpha_i w^i$, the convolution $f * w$ satisfies

$$(\pi_{R,\mathbf{v}} f * w)(\mathbf{x}) = \tau(R)(f * w)(\mathbf{x} - \mathbf{v}). \tag{1.33}$$

This is illustrated in Figure 1.7. Some more details are needed in order to define a full steerable CNN, as pointwise non-linearties may not be equivariant to the representation τ, but we do not go into detail here.

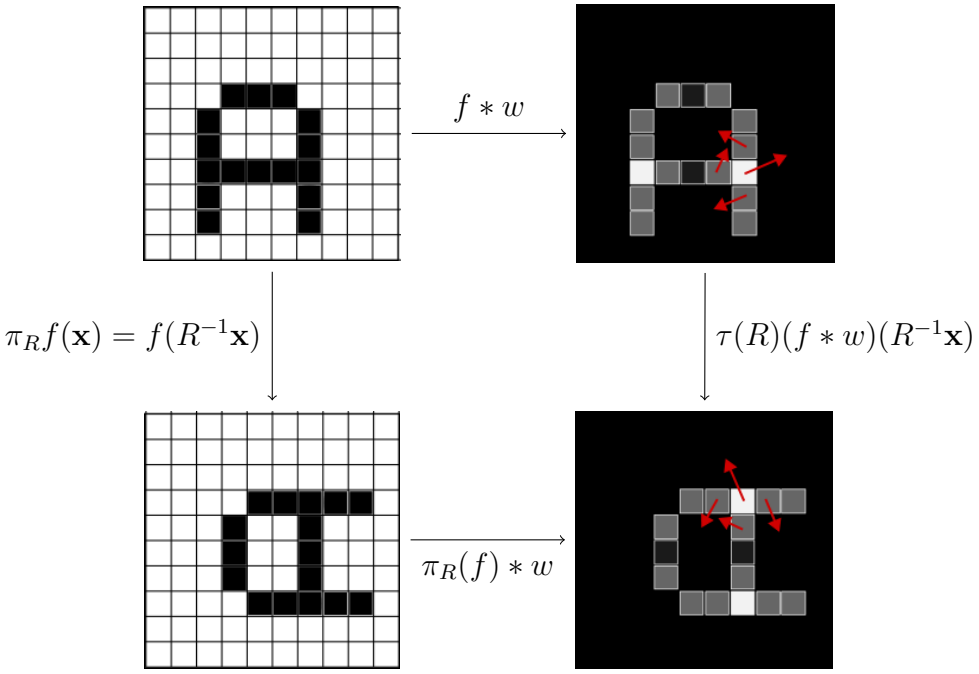


Figure 1.7: Illustration of the equivariance of steerable convolutions. The feature maps $f * w$ in $\mathbb{R}^3$ can be seen as a scalar, represented by the gray level of the images on the right, and a vector, represented by a red arrow (only shown for some pixels). Rotating the image f by a multiple of 90° will also rotate the feature maps, but leaves the scalar values unchanged, and the vector part is rotated as well.

1.6.5 Group Convolution in Homogeneous Spaces

General theories formalizing group-equivariant neural networks (Cohen et al., 2019; Kondor and Trivedi, 2018) rely on the definition of group convolution on *homogeneous spaces* and can unify the concepts discussed above. A set X, acted upon by G is called an homogeneous space if for every $x, y \in X$ there exists $g \in G$ such that $g \cdot x = y$ (in this case we also say that the action of G is transitive). A particular example of an homogeneous space is the set of cosets $G/H = \{gH | g \in H\}$ for a given normal subgroup $H \leq G$, i.e. a subgroup H such that $\forall h \in H, g \in G$, $g \cdot h \cdot g^{-1} \in H$. In this section we review the approach of Cohen et al. (2019) to define an equivariant linear map on functions of homogeneous spaces.

We can represent a group as a fiber bundle using H and G/H as fiber and base, respectively. Let $H \subseteq G$ be a subgroup, the map $\mathbf{p} : G \to G/H$ given by $\mathbf{p}(g) = gH$ is a projection and each fiber is homeomorphic to H (Cohen et al., 2019). This is called the principal H-bundle.

An associated vector bundle is defined by choosing a representation (V, τ) of H, we take $G \times V$, such that $h \cdot (g, v) = (hg, \tau(h^{-1}) \cdot v)$ and computing the quotient $(G \times V)/H = A$. This is equivalent to a fiber bundle with base G/H and a fiber V, and thus sections $s : G/H \to A$ can be viewed as a stack of feature maps. The last ingredient necessary to define equivariant operators is to represent sections as Mackey functions. A Mackey function can be written as a function $f : G \to V$ that is subject to the condition $f(gh) = \tau(h^{-1})f(g)$. Mackey functions encode sections of A. In (Cohen et al., 2019) it is shown that every bounded, linear and G-equivariant operator on a homogeneous space can be written as a group convolution like in (1.24) applied to a Mackey function.

Now to put things into perspective, let us assume for instance that $G = p4 \times \mathbb{Z}^2$. The group $\mathbb{Z}^2$ is an homogeneous space with respect to G, because its elements are related by translations. Let us define by H the stabilizer of the origin $o = (0, 0)$ at $\mathbb{Z}^2$, which is defined by $H = \{g \in G | g \cdot o = o\}$. We have that the stabilizer H is exactly $p4$ and thus $G/H \cong \mathbb{Z}^2$. If we choose $V = \mathbb{R}^3$ with the representation from last section we end with the same kinds of equivariant operators on Mackey functions that are obtained extending $f : \mathbb{Z}^2 \to V$ to a function $f : G \to V$ that is constant along the orbits.

If, on the other hand, we choose $H = \{e\}$ and $V = \mathbb{R}^n$ with $\tau(g) = \mathrm{id}$, we end with the group convolution defined in Section 1.6.3.

1.7 Conclusions

This chapter serves as an introduction to some themes central to the thesis, namely groups, group actions, group-equivariant and group-invariant operators. We relate those concepts to the field of geometric deep learning, showing how one can obtain an equivariant neural network from blocks of linear equivariant operators, pointwise non-linearities and so on. In this chapter the group convolution is also reviewed. Group convolutions, including steerable networks, are the main operators which are used to define equivariant networks in the literature. In Chapters 2 and 3 we use the approach of Worrall and Welling (2019) which is a generalization of the group convolution, the semi-group convolution, for scale-equivariant networks. In Chapters 4, 5 and 6, however, we steer away from group convolutions and we propose a new approach based on the method of moving frames.

1.8 Résumé en Français

Ce chapitre est une introduction à quelques sujets centraux de la thèse, en particulier les groupes, les actions de groupe et les opérateurs équivariants ou invariants par l'action d'un groupe. Nous montrons les liens entre ces sujets et l'apprentissage profond géométrique, en montrant qu'il est possible de définir un réseau équivariant à partir de blocs d'opérateurs équivariants linéaires, de non-linéarités appliquées ponctuellement et d'autres opérateurs standard de l'apprentissage profond comme la *Batch Normalization*. Dans ce chapitre la convolution de groupe est également revue. La convolution de groupe, y compris les convolutions orientables, est le principal opérateur utilisé pour obtenir des réseaux équivariants dans la littérature. Dans les chapitres 2 et 3 nous utilisons l'approche de Worrall and Welling (2019), les convo-

lutions de semi-groupe, une généralisation des convolutions de groupe. Dans les chapitres 4, 5 et 6, nous n'utilisons pas les convolutions de groupe mais nous proposons une méthode pour obtenir de couches équivariantes basée sur la méthode du repère mobile.

Chapter 2

Scale-Equivariant Networks

2.1 Introduction

Scale transformations are naturally found in many computer vision tasks, since distance from the camera causes a scale variation in the captured images. This makes scale-equivariance a particularly interesting property for some networks. Generally speaking, convolutional neural networks are not equivariant either in theory or in practice, as illustrated in Figure 3.1 for the case of the U-Net. Scale-spaces are methods from image processing that allow for multi-scale representation of images. This chapter focuses on studying scale-equivariant networks based on scale-spaces.

In contrast to the group convolution discussed in Chapter 1, Deep Scale-Spaces (Worrall and Welling, 2019) introduce neural networks equivariant to the action of semigroups, instead of groups. Semigroup actions can model non-invertible transformations and in Worrall and Welling (2019) the authors focused on equivariance to downsampling in discrete domains. Focusing on downsampling is a way to address equivariance to scalings without creating spurious information through interpolation. In their architecture, the first layer is based on a Gaussian scale-space operator, and subsequent layers of this network are equivariant to the action of a semigroup of scalings and translations. Effectively, these operators are equivariant to rescaling of a discrete image. There are other scale-spaces with similar mathematical properties to the Gaussian scale-space, in particular the morphological scale-spaces.

In this chapter we generalize the approach of semigroup scale-equivariant neural networks by finding a sufficient condition in which scale-spaces, in the sense of Heijmans and van den Boomgaard (2002), can be used as the first layer, or the so-called *lifting*, in an equivariant network. Furthermore we investigate architectures built using morphological scale-spaces liftings. We observe that the morphological scale-spaces networks compare favorably to the Gaussian one in tasks of classification and segmentation of images at scales previously unseen by the network, in contrast to Worrall and Welling (2019), in which the experiments test the overall performance of the network, but where the train and test sets objects follow the same scale distribution.

The chapter is organized as follows: We begin by introducing the groups and semigroups of scalings, particularly the semigroup which we seek equivariance to, in Section 2.2. In Section 2.3 we apply the group convolution (1.24) to the case of the scalings group and in Section 2.4

we review the approach of Worrall and Welling (2019) to define scale-cross-correlations and we also discuss their advantages, such as their increased receptive fields compared to CNNs. We talk about the lifting and projection layers in Sections 2.5 and 2.5.2, respectively, which are layers that are used to map images to and from signals on semigroup in an equivariant (or approximately equivariant) way. In Section 2.5.2 we also introduce the scale-dropout applied before the projection layer to increase robustness and encourage the use of multi-scale information. In Section 2.5.3 we discuss the conditions necessary for a lifting layer and we propose a class of liftings based on scale-spaces. Particularly this includes the Gaussian and morphological scale-spaces. We validate our methods experimentally in Section 2.6, where we test scale-equivariant methods in tasks of classification and segmentation. For the classification task we use the Large Scale MNIST dataset (Jansson and Lindeberg, 2020) and for the segmentation we use a synthetic dataset of shapes. In both cases the goal is to measure how well a model performs when trained for images in specific scales and tested in images with objects in unseen scales. In both cases the equivariant models outperform CNN baselines, including the U-Net (Ronneberger et al., 2015) for the segmentation task and the models with morphological lifting, particularly with the quadratic dilations lifting, seem to have some advantages when shape matters more than texture. We end the chapter with some concluding remarks in Section 2.7.

2.2 Scalings Group and Semigroup

Throughout the rest of the chapter, the transformations of interest are representations of a group or semigroup of scalings and translations. The group of scalings can be identified by the positive real numbers using real multiplication, denoted $(\mathcal{S}, \cdot)$, where $\mathcal{S} = \mathbb{R}_{>0} = \{s \in \mathbb{R} | s > 0\}$ and $\cdot$ is the real multiplication. Adding translations operators amounts to computing the direct product of a group of scalings and a group of translations, $\mathcal{S} \ltimes \mathbb{R}^2$. An element of that group is denoted by (s, x), $s \in \mathcal{S}, x \in \mathbb{R}^2$, and the product $\cdot$ is defined by

$$(s, x) \cdot (t, y) = (st, sy + x). \tag{2.1}$$

This group acts on images $f : \mathbb{R}^2 \to \mathbb{R}^n$ by means of a re-scaling followed by a translation, i.e. we define the action $R'_{s,x}$, $(s, x) \in \mathcal{S} \times \mathbb{R}^2$ as, for all $(s, x) \in \mathcal{S} \times \mathbb{R}^2$, $y \in \mathbb{R}^2$

$$R'_{s,x}[f](y) = f(sy + x), \tag{2.2}$$

and similarly, for functions $f : \mathcal{S} \times \mathbb{R}^2 \to \mathbb{R}^n$ we define the action $R_{s,x}$, $(s, x) \in \mathcal{S} \times \mathbb{R}^2$ as

$$R_{s,x}[f](t, y) = f(st, sy + x). \tag{2.3}$$

In this chapter many of the approaches are based on representing scales as semigroups. A semigroup is a tuple $(G, \cdot)$ of a set G and a function $\cdot : G \times G \to G$ with the property of associativity. In other words it is an object obtained by removing the properties of inverse and neutral element from the definition of group. Similarly to groups, we can define semigroup actions on a set X as $\pi : G \times X \to X$ which can be written as families of functions $\pi_g(x) \coloneqq \pi(g, x)$ $\forall g \in G, x \in X$ which are homorphic to the semigroup structure i.e. $\forall g, h \in G$ $\pi_g \circ \pi_h = \pi_{g \cdot h}$ (left semigroup action) or $\forall g, h \in G$ $\pi_g \circ \pi_h = \pi_{h \cdot g}$ (right semigroup action).

To model the semigroup approach, we consider some semigroups that are sub-semigroups of $\mathcal{S} \ltimes \mathbb{R}^2$, i.e. we consider a semigroup $S \times E \subseteq \mathcal{S} \times \mathbb{R}^2$ that is closed under the product "$\cdot$". A first approach would be to take $\mathcal{S}_{\geq 1} \ltimes \mathbb{R}^2$, where $\mathcal{S}_{\geq 1} = \mathbb{R}_{\geq 1} = \{s \in \mathbb{R} | s \geq 1\}$. A discrete semigroup of scalings can also be obtained by $\mathcal{S}_\gamma = \{\gamma^n | n \in \mathbb{N}\}$ for some $\gamma > 1$. In the case where γ is an integer, both $\mathcal{S}_\gamma \times \mathbb{R}^2$ and $\mathcal{S}_\gamma \times \mathbb{Z}^2$ form sub-semigroups of $\mathcal{S} \ltimes \mathbb{R}^2$, but for a general $\gamma > 1$, only $\mathcal{S}_\gamma \times \mathbb{R}^2$ is closed under the product (2.1). In all of these cases, the operators $R_{s,x}$ and $R'_{s,x}$, when restricted to values in $S \times E$, form semigroup actions on functions defined on $S \times E$ and E, respectively.

2.3 Scale Group Convolution

Before we investigate the case of semigroup equivariant networks for which most applications in this chapter were developed, let us first look at the group convolution of the scalings group in order to gain some intuition into the form of the semigroup convolution. With the goal of defining a linear, scale-equivariant operator, such as outlined in Section 1.4 by the GDLBP, we first examine a special case of networks equivariant to the group $\mathcal{S} \ltimes \mathbb{R}^2$. Moreover, we assume the normal subgroup H to simply be the identity $\{(1, 0)\}$, meaning that the group representation acts only on the domain of the input functions $f : \mathcal{S} \ltimes \mathbb{R}^2$.

Assume that we have a function $f : \mathcal{S} \ltimes \mathbb{R}^2 \to \mathbb{R}$ and a filter $h : \mathcal{S} \ltimes \mathbb{R}^2 \to \mathbb{R}$. We compute the group convolution of a single-channel image by a single channel filter in order to simplify the computations, as in the multi-channel case it is only a question of applying multiple single-channel convolutions. We find that the left-invariant Haar measure in $\mathcal{S} \times \mathbb{R}^2$ can be obtained by

$$d\mu((s, \mathbf{x})) = \left(\det \begin{bmatrix} s & 0 & x_1 \\ 0 & s & x_2 \\ 0 & 0 & 1 \end{bmatrix} \right)^{-3} dsdx_1dx_2 = s^{-6}dsdx_1dx_2 \tag{2.4}$$

and the convolution (1.24) becomes

$$\begin{aligned}
(f * h)(s, x) &= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(s', x') h\big((s', x')^{-1} \cdot (s, x)\big) ds'dx' \qquad (2.5)\\
&= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(s', x') h(s'^{-1}s, s'^{-1}x - s'^{-1}x') ds'dx' \\
&= \int_{\mathcal{S}} \int_E \frac{1}{s'^6} f(s', x') h(s'^{-1}s, s'^{-1}x - s'^{-1}x') dx'ds' \\
&= \int_{\mathcal{S}} \int_E \frac{1}{s'^6} f(ss', sx' + x) h(s'^{-1}, -s'^{-1}x') dx'ds' \\
&= \int_{\mathcal{S}} \int_E \frac{1}{s'^6} f\big((s, x) \cdot (s', x')\big) h\big((s', x')^{-1}\big) dx'ds'.
\end{aligned}$$

It is equivariant to the action (2.3), which we can verify:

$$\begin{aligned}
(R_{t,y}f * h)(s,x) &= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(ts', tx' + y) h(s'^{-1}s, s'^{-1}x - s'^{-1}x') ds'dx' \\
&= \int_{\mathcal{S}\times E} \frac{t^2}{s'^6} f(s', tx' + y) h(s'^{-1}ts, s'^{-1}tx - s'^{-1}tx') ds'dx' \\
&= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(s', x' + y) h(s'^{-1}ts, s'^{-1}tx - s'^{-1}x') ds'dx' \\
&= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(s', x') h(s'^{-1}ts, s'^{-1}tx - s'^{-1}x' + s'^{-1}y) ds'dx' \\
&= \int_{\mathcal{S}\times E} \frac{1}{s'^6} f(s', x') h((s', x')^{-1}(t, y)(s, x)) ds'dx' \\
&= R_{t,y}(f * h)(s, x).
\end{aligned}$$

2.4 Semigroup Convolution

Let's first stress the interest of extending the equivariance setting to semigroups. In image processing, important examples of semigroup actions are scale-spaces. As we will present in more details in Section 2.5.3, semigroup actions on images may be the convolution with a Gaussian kernel (Gaussian scale-space) or the application of a morphological operator such as erosion, dilation, opening or closing (morphological scale-spaces). Scale-spaces highlight the multi-scale nature of images and have shown great efficiency as image representations (Lowe, 1999). Besides, they are naturally complementary to the scaling operation. For example, the Gaussian blurring acts as a low-pass filter and allows the subsampling (or downscaling on a discrete domain) of an image to avoid aliasing artifacts.

Hence, equivariance of linear operators to semigroups seems highly desirable, as it is natural to expect that the same information at different scales produce the same responses up to some shift due to scale change. Since we cannot reproduce the results of the group convolution, a new expression has to be derived. In this case, however, we constrain the problem to only discrete semigroups, as that is the object that is most interesting at this stage. Taking inspiration from the group convolution, we notice that (1.26) can also be written, for $u \in G$, $H(f)(u) = \sum_{g\in G} h(g)f(u \cdot g)$ if we change the function w for its conjugate h, that is $h(g) = w(g^{-1})$, and thanks to a change in variables. Now this expression can be applied to semigroups, considering the semigroup right action $R_u(f)(g) = f(u \cdot g)$. We get that operators H defined by

$$\forall u \in G, \quad H(f)(u) = \sum_{g\in G} R_u(f)(g)h(g) \tag{2.6}$$

are indeed equivariant to the semigroup action $R_t, t \in G$, since

$$H(R_t(f))(u) = \sum_{g\in G} R_u(R_t(f))(g)h(g) = \sum_{g\in G} R_{tu}(f)(g)h(g) = R_t(H(f))(u). \tag{2.7}$$

This class of semigroup equivariant operators is the semigroup cross-correlation proposed in Worrall and Welling (2019). We also write $f \star_G h \coloneqq H$, remarking however that, contrary to the

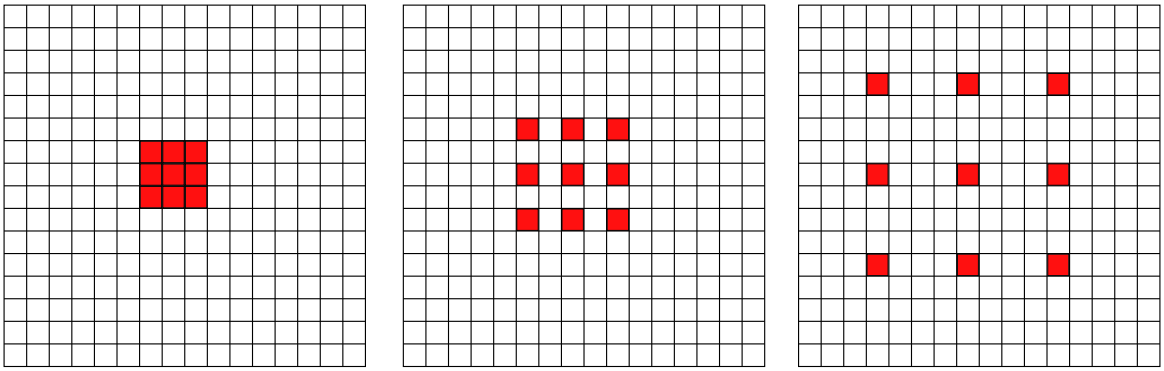


Figure 2.1: The Scale-cross-correlation performs dilated convolutions at each scale, where the dilation rate is equal to the current scale. Each grid represents how filters are applied to the same feature map in $S_\gamma \times \mathbb{Z}^2$ at a different scale level. In the present example we assume that $\gamma = 2$. We show a filter being applied to feature maps at scales γ^0, γ^1 and γ^2.

group case, this operation is not symmetrical in f and h even when the law $\cdot$ on G is commutative.

2.4.1 Implementation for Integer Scalings

Following Worrall and Welling (2019), assuming that the base of the scale-semigroup is an integer γ, the operator (2.6) applied to a signal f at a point (γ^k, x) becomes

$$H(f)(\gamma^k, x) = (f \star_G h)(\gamma^k, x) = \sum_{l \geq 0} \sum_{y \in \mathbb{Z}^2} f(\gamma^{k+l}, \gamma^k y + x) h(\gamma^l, y). \tag{2.8}$$

We can also try to obtain a discrete operator from (2.5). That operator is given by

$$H(f)(\gamma^k, x) = (f \star_G h)(\gamma^k, x) = \sum_{l \geq 0} \sum_{y \in \mathbb{Z}^2} \gamma^{-6l} f(\gamma^{k+l}, \gamma^k y + x) h(\gamma^l, y). \tag{2.9}$$

and it differs from (2.8) by the term γ^{-6l} multiplying the arguments of the sum. In fact both approaches as equivalent as we can compute the cross-correlations by $\bar{h}(\gamma^l, y) = \gamma^{-6l} h(\gamma^l, y)$, therefore, we keep the first approach (2.8) because it is simpler.

In Figure 2.1, the shape of the filters for different scales of the same image is shown. Note that it is the same filter being applied with different dilation rates at each scale.

The operations here were defined for single channel images on G, but they can easily be applied to multichannel images. Let the input $f = (f_1, \ldots, f_n) \in (\mathbb{R}^C)^G$ be a signal with n channels. Assuming the output has C' channels, the filter is of the form $h : G \to \mathbb{R}^{C \times C'}$. In this case we can compute a multi-channel scale-cross-correlation operator at channel $o \in \{1, \ldots, mb\}$ as

$$(f \star_G h)_o(\gamma^k, x) \coloneqq \sum_{c=1}^{C} (f_c \star_G h_{c,o})(\gamma^k, x). \tag{2.10}$$

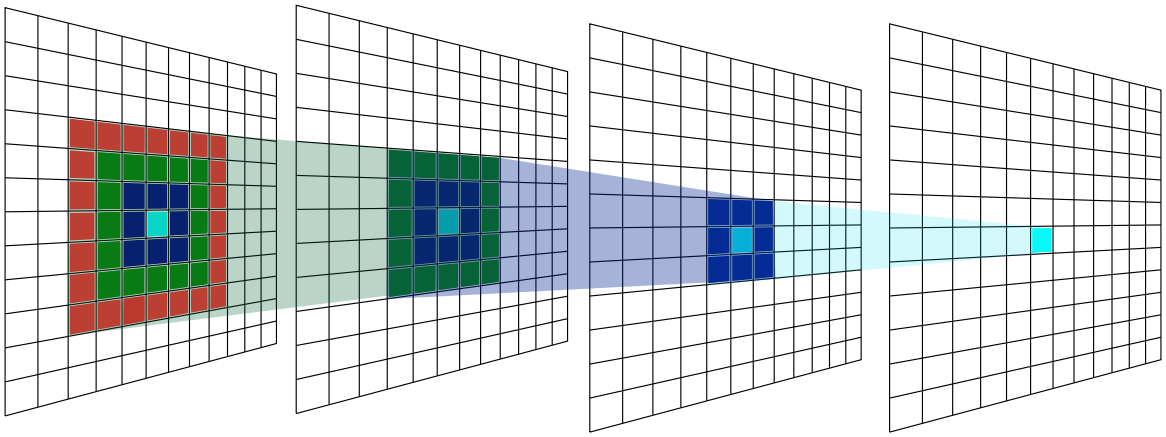


Figure 2.2: Illustration of the receptive field of a 4-layer CNN with 3×3 filters in each layer. Each convolution by a 3×3 filter increases the size of the receptive field by 2. In general each convolution by a $K \times K$ filter increases the receptive field size by $K - 1$.

We can verify that it is equivariant

$$\begin{aligned}
(R_g f \star_G h)_o(\gamma^k, x) &= \sum_{c=1}^{C} \big(R_g(f)_c \star_G h_{c,o}\big)(\gamma^k, x) \\
&= \sum_{c=1}^{C} \big(R_g(f_c) \star_G h_{c,o}\big)(\gamma^k, x) \\
&= \sum_{c=1}^{C} R_g(f_c \star_G h_{c,o})(\gamma^k, x) \\
&= R_g \left(\sum_{c=1}^{C} (f_c \star_G h_{c,o}) \right)(\gamma^k, x) \\
&= R_g\big((f \star_G h)_o\big)(\gamma^k, x).
\end{aligned} \tag{2.11}$$

2.4.2 Improved Receptive Field

In a convolutional, or semigroup-convolutional neural network, the receptive field is the region which is effectively used to compute an output point. More precisely, if f_{out} is the output of the network, then the pixel $f_{\text{out}}(\mathbf{x})$ is computed only from the pixels of the input f falling inside the receptive field. For example, in a CNN consisting of 4 layers of 3×3 convolutions, the last layer will use a 3×3 square from the previous layer in the convolutions, and to compute each pixels in that 3×3 square another 3×3 square from the previous layer will be used, but since the squares overlap that results in a 5×5 square from layer 2 and a similar argument shows that it uses a 7×7 from layer 1. This network is illustrated in Figure 2.2.

More generally, a CNN of L layers, with each of these layers computing a convolution by a $K \times K$ filter has as a receptive field a square of size $L(K-1)+1$. Other operations such as subsampling also influence the receptive field.

The advantage of having a large receptive field is that more of the information on the image is used at a time and more information can be extracted for objects which are far from each other in the image. Note that in (2.8), because of the multiplicative constant γ^k in the spatial component, the receptive fields of networks consisting of scale semigroup correlations are much

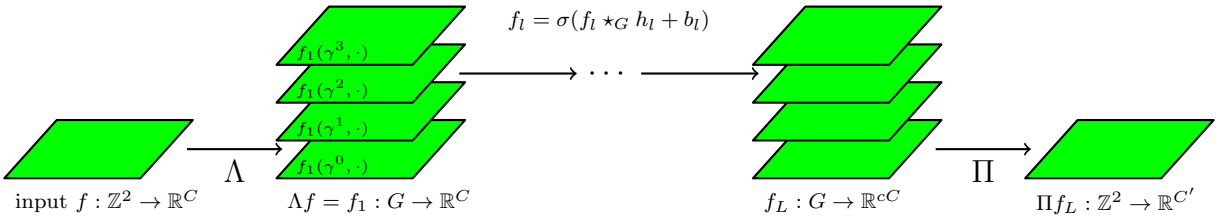


Figure 2.3: Illustration of a scale-crosscorrelation architecture using a lifting and a projection layer. The lifting, denoted Λ will map a 2D signal to a signal on G and the projection will map a signal on G to a 2D signal. Scales truncated at four.

larger than those of a CNN of corresponding size. Indeed, a network consisting of L scale cross-correlations with filters with dimensions $P \times K \times K$ (i.e. the support of the filter is a grid $\{(p, k_1, k_2) | p \in \{\gamma^0, \gamma^1, \ldots, \gamma^{P-1}\}, 1 \leq k_1, k_\gamma \leq K\}$) at a scale coordinate s has a receptive field of size greater than $\gamma^s K + \gamma^{s+P-1}(L-1)(K-1)$, which is significantly larger than the size of the receptive field of a CNN with the same number of layers and the same kernel size even for $s = 1$ or $s = 2$. At $s = 0$, $P = 1$ it is equivalent to a CNN. In other words, an architecture built on scale cross-correlations attains the same receptive field as a deep CNN with much smaller depth and number of parameters.

2.5 Lifting and Projection

The operators of the previous section are defined on the set of functions with the semigroup as a domain, $\mathcal{F} = (\mathbb{R}^C)^G$, but images input to networks are functions $f : \mathbb{Z}^2 \to \mathbb{R}^C$. In this section we review the lifting and projections layers - operators which map images to functions on the semigroup and vice-versa. They operate by equivariantly mapping the inputs from $\mathbb{Z}^2$ to the semigroup and vice-versa, as illustrated in Figure 2.3.

2.5.1 Lifting

So far, in the general case, we have considered functions $f : G \to \mathbb{R}^C$ defined in the semi-group G, but the input to CNNs are images defined on a discrete set X, which may be different from the semigroup we seek equivariance to. In particular in this chapter we consider the semigroup product of translations *and* scalings.

In general, if the semigroup $G = S \ltimes X$ is a semidirect product of two semigroups S and X then in theory the issue is easily overcome because applying (2.6) to a signal $f : X \to \mathbb{R}$ instead of a signal defined on G does not change the equivariance property, provided we have an appropriate semigroup action R'_u on f, but the output will be a signal on G. So in order to apply equivariant operators a signal f on X on we must *lift* it to a signal G at some point. Accordingly, the task is divided into two steps. First, we define (2.6) which holds as it is, like in Section 2.4.1. Second, we introduce a *lifting* operator Λ to map a function f defined on X into a function Λf defined on G, as it is done in Section 2.5.3. The operator H becomes then:

$$\forall u \in G, \quad H(f)(u) = \sum_{g \in G} R_u(\Lambda f)(g) h(g).$$

Now we easily check that it is sufficient for the lifting operator to be equivariant to G, that is, $R_u \circ \Lambda = \Lambda \circ R'_u$, to have equivariance of H. Indeed, in that case by omitting parentheses for readability,

$$H(R'_t f)(u) = \sum_{g \in G} R_u \Lambda R'_t f(g) h(g) = \sum_{g \in G} R_u R_t \Lambda f(g) h(g) = R_t(Hf)(u). \tag{2.12}$$

The advantage of this two-step approach is the richness of operators induced by the variety of possible liftings, as exposed in the rest of this section.

2.5.2 Projection

Now we have taken a signal $f : X \to \mathbb{R}^C$ and mapped it to a signal on G where we can apply equivariant operators, obtaining a signal $\phi(f) : G \to \mathbb{R}^{C'}$ such that $\phi(R'_u f) = R_u(\phi(f))$. However, an output of this form is not suitable for most applications and applying densely connected neural network layer to change the output form would destroy the equivariance. As the last part of the GDLBP, we must apply a projection layer, its form depends on the application: the output should lie in X like in a segmentation task, so the projection $\Pi : (\mathbb{R}^C)^G \to (\mathbb{R}^C)^X$ must be an equivariant operator, i.e. $\Pi \circ R_u = R'_u \circ \Pi$; the output should be a vector, like in a classification task, so the projection $\Pi' : (\mathbb{R}^C)^G \to \mathbb{R}^C$ must be invariant, i.e. $\Pi' \circ R_u = \Pi'$.

It turns out that in practice the projections are only approximately equivariant when G is not a group, but in later sections this approximation will be verified empirically.

In general a permutation invariant operator (e.g. sum, average or supremum) can be applied to the scale-dimension to define the projection layer, i.e. if $\alpha : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}$ is permutation invariant we can define

$$\Pi(f)(x) = \alpha(f(\gamma^0, x, \gamma^1 x, \gamma^2 x, \dots)) \tag{2.13}$$

or

$$\Pi(f)(x) = \alpha((f(\gamma^k, x)_{k \geq 0, x \in \mathbb{Z}^2}) \tag{2.14}$$

where we assume that f has a finite support in order for the maximum and sum to be well defined.

In practice to perform the projection we apply a *max-projection* along the scale dimension, defined by $\forall z \in \mathbb{Z}^2,\ \Pi[f](z) = \sup_{k \in \mathbb{N}} \{f(\gamma^k, z)\}$. To be consistent with the lifting, we would like to have

$$R'_{\gamma^k, z} \circ \Pi = \Pi \circ R_{\gamma^k, z}.$$

Instead, we have

$$R'_{\gamma^k, z} \Pi f(y) = \sup_{l \in \mathbb{N}} f(\gamma^l, \gamma^k y + z),$$

and

$$\Pi R_{\gamma^k, z} f(y) = \sup_{l \in \mathbb{N}} f(\gamma^{l+k}, \gamma^k y + z) = \sup_{l \geq k} f(\gamma^l, \gamma^k y + z)$$

so that $R'_{\gamma^k, z} \Pi f(y) = \max\{\Pi R_{\gamma^k, z} f(y), \max_{0 \leq l < k} f(\gamma^l, \gamma^k y + z)\}$. The previous expression will be equivariant if the scale where the maximum is attained is smaller than k, but in general we can only hope for approximate equivariance for small enough k. The approximate equivariance

will be empirically verified in experiments in Section 2.6. Note that other projections (e.g. sum or average) have the same flaw, as this is intrinsic to the semigroup-equivariant approach, even though it was omitted previously in the literature.

Scale Dropout

The projection layer produces an approximately equivariant/invariant feature map, but it is not guaranteed to use the multiscale information at the last layer. In order to incentive the use of multiple scales by the classification layers and to produce more robust results with respect to scale changes we proposed a special version of dropout (Srivastava et al., 2014), which we call *Scale Dropout*, applied before the projection layer.

The Scale Dropout works by dropping (i.e. setting to 0) all values of a feature map at randomly chosen scales. Formally, given a feature map $f : S_\gamma \times \mathbb{Z}^2 \to \mathbb{R}^C$, the scale dropout with rate $r \in [0,\ 1]$ is computed as

$$\text{ScaleDropout}_r(f)(s, x) = X(s)f(s, x) \tag{2.15}$$

where $X(s)$, for every $s \in S_\gamma$ is a Bernoulli variable of parameter r, i.e. $P\big(X(s) = 1\big) = 1 - r$ and $P\big(X(s) = 0\big) = r$.

2.5.3 Scale-Spaces as Lifting Operators

We focus on the semigroup product of scalings and translations $G = \mathcal{S} \rtimes \mathbb{Z}^2 = (\mathcal{S} \times \mathbb{Z}^2, \cdot)$.

A lifting Λ in the space of scalings is characterized by the property

$$\Lambda \circ R'_{t,z} = R_{t,z} \circ \Lambda, \tag{2.16}$$

With the goal of defining a suitable and general lifting, scale-spaces will serve as a theoretical framework.

Many studies have been made on defining scale-spaces (Heijmans and van den Boomgaard, 2002; Witkin, 1984; Pauwels et al., 1995; Alvarez et al., 1993). In here, following the definitions of Heijmans and van den Boomgaard (2002), a scale-space can be viewed as an operator that commutes with a *scaling*. A scaling is a family $\{S_t : V^{\mathbb{R}^2} \to V^{\mathbb{R}^2} | t > 0\}$, where $V = \mathbb{R}^C$, of operators on images such that:

$$S_1 = \text{id}, \qquad \forall t, s > 0 \;\; S_t S_s = S_{ts} \tag{2.17}$$

where id is the identity transform. Therefore a scaling is a group action of $(\mathbb{R}^+_*, \times)$. An example is the family S^p, $p \geq 0$, given by $(S^p_t f)(\mathbf{x}) = f\left(\frac{1}{t^p}\mathbf{x}\right)$, where p controls the rate of scaling. We have that $R'_{t,0} = S^p_{t^{-\frac{1}{p}}}$.

Let S be a scaling and $\dot{+}$ a commutative operation such that $(\mathbb{R}^+_*, \dot{+})$ is a semigroup. Then a $(S, \dot{+})$ *scale-space* is a family $\{T(t) | t > 0\}$ of operators such that, for all $t, s > 0$ (Heijmans and van den Boomgaard, 2002):

$$T(t)T(s) = T(t\dot{+}s), \qquad T(t)S(t) = S(t)T(1). \tag{2.18}$$

The property $T(t)S(s) = S(s)T(t/s)$, for all $t, s > 0$, is a direct consequence of the second property. Here, in addition to (2.18), we assume that the scale-space $T(t)$ is translation-equivariant for all $t > 0$ (i.e. $T(t)(L_z f) = L_z(T(t)f)$ where $L_z(f)(x) = f(x + z)$). Thanks to the second property in (2.18), a $(S^p, \dot{+})$ scale-space T defines an operator on images $f : \mathbb{R}^2 \to \mathbb{R}^C$

$$\forall (s, x) \in \mathcal{S} \times E \quad (\Lambda f)(s, x) = (T(s^{-\frac{1}{p}})f)(x) \tag{2.19}$$

such that, for all $(t, z) \in \mathcal{S} \times \mathbb{Z}^2$, $R_{(t,z)} \circ \Lambda = \Lambda \circ R'_{(t,z)}$, where $R'_{(s,x)}$ is applied to an image on a continuous domain. So, in order for Λ to be our lifting operator we assume that the input f is a function on a continuous domain. In practice, we discretize Λ and the input images. With this, the morphological scale-spaces, as well as the Gaussian scale-space, being $(S^{\frac{1}{2}, 0}, \dot{+})$ scale-spaces, can be used as the lifting operators.

2.5.4 Gaussian Scale-Space

The Gaussian scale-space is a $(S^{\frac{1}{2}}, +)$ scale-space defined by the family $T_{\mathcal{G}}(t)[f](x, y) = u(x, y, t)$, where

$$\begin{cases} \partial_t u = \Delta u, \\ u(x, y, 0) = f(x, y), \ \forall (x, y) \in \mathbb{R}^2 \end{cases} \tag{2.20}$$

For all images $f \in \mathbb{R}^{\mathbb{R}^2}$ and points $x \in \mathbb{R}^2$. The solution to the previous PDE, which is the scale-space $T_{\mathcal{G}}$ can be computed as the convolution

$$\begin{aligned} (T_{\mathcal{G}}(t)f)(x) &= (f * \mathcal{G}_t)(x) \\ &= \int_{\mathbb{R}^2} f(y)\mathcal{G}_t(x - y)dy \\ &= \frac{1}{2\pi t} \int_{\mathbb{R}^2} f(y) \exp\left(\frac{\|x - y\|^2}{2t}\right) dy, \end{aligned} \tag{2.21}$$

where $\mathcal{G}_t(x) = (2\pi t)^{-1} \exp\left(-\frac{\|x\|^2}{2t}\right)$. This was the scale-space considered in Worrall and Welling (2019). There, it was assumed that image f has a maximum spatial frequency content. They model this by assuming that there exists a signal f_0 and a constant $s_0 > 0$ such that $f = (f_0 * \mathcal{G}_{s_0})$. An example of the Gaussian scale-space representation of an image is given in Figure 2.4.

2.5.5 Morphological Scale-Spaces

Similarly to the Gaussian scale-space, we can obtain morphological scale-spaces using partial differential equations (Brockett and Maragos, 1994; Van Den Boomgaard and Smeulders, 1994). The quadratic morphological erosions and dilations, for example, can be regarded as morphological counterparts to the Gaussian scale-space (Van Den Boomgaard and Smeulders, 1994) and are obtained by the PDE

$$\begin{cases} \partial_t u = \pm|\nabla u|^2 \\ u(x, y, 0) = f(x, y) \ \forall (x, y) \in \mathbb{R}^2 \end{cases} \tag{2.22}$$

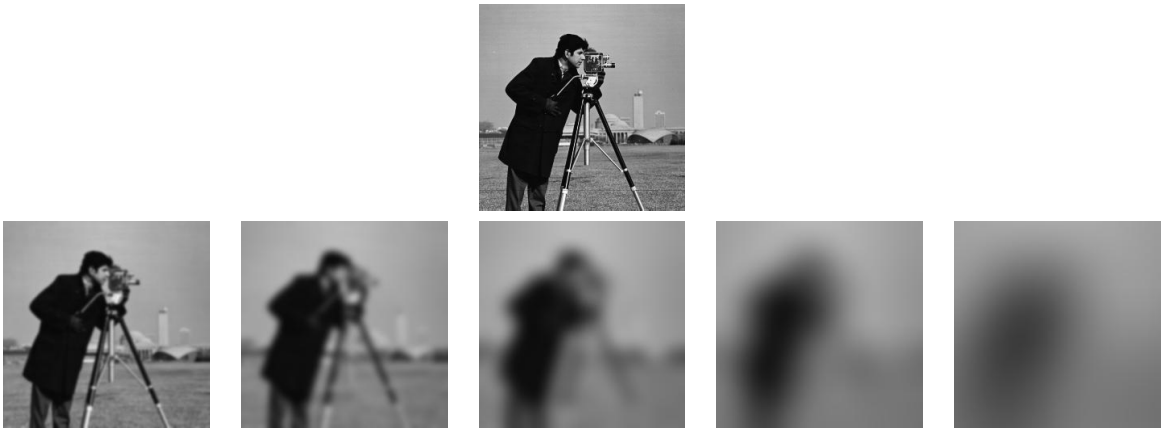


Figure 2.4: An image f and the first five scales of its Gaussian scale-space representation, $\Lambda_{\mathcal{G}} f(\gamma^0, \cdot), \dots \Lambda_{\mathcal{G}} f(\gamma^4, \cdot)$ using a base $\gamma = 2$.

The solution to the equation above generate the families of quadratic erosions and dilations. They can be written as dilations and erosions by the structuring functions $q_t(x) = -\frac{\|x\|^2}{4ct}$ $t > 0$, i.e.

$$\varepsilon_{q_t}(f)(x) = \varepsilon_t(f)(x) = \inf_{y \in \mathbb{R}^2} \left(f(x+y) - q_t(y) \right), \quad \text{and,} \tag{2.23}$$

$$\delta_{q_t}(f)(x) = \delta_t(f)(x) = \sup_{y \in \mathbb{R}^2} \left(f(x-y) + q_t(y) \right), \tag{2.24}$$

form $(S^{\frac{1}{2}}, +)$ scale-spaces. Here, to increase flexibility in the context of deep learning, we consider a parameter $c > 0$ learned by gradient descent with the rest of the parameters of the network. Examples of quadratic dilations and quadratic erosions scale-spaces applied to an image are shown in Figure 2.5.

The choice of the quadratic structuring functions as the natural counterpart of the Gaussian kernels can be generalized to the notion of poweroid structuring function (Jackway, 1994). The corresponding multiscale structuring function (Diop and Angulo, 2015; Schmidt and Weickert, 2016)

$$b_{p,t}(x) = -\frac{p-1}{p^{p/(p-1)}} \frac{\|x\|^{p/(p-1)}}{t^{1/(p-1)}} \tag{2.25}$$

is a scale-space and a solution of the Hamilton–Jacobi equation

$$\begin{cases} \partial_t u = \pm |\nabla u|^p \\ u(x, y, 0) = f(x, y) \ \forall (x, y) \in \mathbb{R}^2 \end{cases} \tag{2.26}$$

Besides the previous case of parabolic structuring function for $p = 2$, note that for a fixed $t > 0$, when $p \to 1$,then $b_{p,t} \to 0$, i.e., flat structuring element of radius t, and when $p \to \infty$ one gets $b_{p,t} \to \|x\|/t$, i.e., conical structuring function.

The families of dilations and erosions $\delta_{b_{p,t}}$ and $\varepsilon_{b_{p,t}}$ for some $p \geq 1$ form $(S^{\frac{1}{p}}, +)$ scale-spaces and thus constitute liftings.

(a) Dilation scale-space.

(b) Erosion scale-space.

Figure 2.5: The first five scales of quadratic dilations and erosions scale-spaces representation of an image f, $\Lambda f(\gamma^0, \cdot), \dots \Lambda f(\gamma^4, \cdot)$ using base $\gamma = 2$.

Their compositions $\delta_{b_{p,t}} \circ \varepsilon_{b_{b,t}}$, $\varepsilon_{b_{p,t}} \circ \delta_{b_{b,t}}$ yield families of morphological openings and closings, which are $(S^{\frac{1}{p}}, \vee)$ scale-spaces (Heijmans and van den Boomgaard, 2002), where $a \vee b = \max\{a, b\}$.

2.5.6 Combining Liftings

Linear Combination

From two liftings Λ_1 and Λ_2 we have that the linear combination $\alpha\Lambda_1 + \beta\Lambda_2$ also constitutes a lifting, as we can see by the linearity of $R_{t,z}$ and $R'_{t,z}$

$$\begin{aligned}(\alpha\Lambda_1 + \beta\Lambda_2)R'_{t,z} &= (\alpha\Lambda_1 R'_{t,z} + \beta\Lambda_2 R'_{t,z}) \\ &= (\alpha R_{t,z}\Lambda_1 + \beta R_{t,z}\Lambda_2) \\ &= R_{t,z}(\alpha\Lambda_1 + \beta\Lambda_2).\end{aligned}$$

In particular this opens the way to convex combinations of two liftings. To motivate the interest in these kinds of combinations, consider the scale-space given by

$$L_t^\alpha = \alpha\delta_t + (1 - \alpha)\varepsilon_t \tag{2.27}$$

where δ_t and ε_t are flat dilations by a disk of radius t and $\alpha \in [0, 1]$. In the case where $\alpha = 0$ we have an erosion scale-space and in the case $\alpha = 1$ we have a dilation scale-space. In the case $\alpha = 0.5$ we have a self-dual scale-space, that is $L_t^{0.5}[f] = -L_t^{0.5}[-f]$. If it is not known a priori what kind of scale-space would be better suited for the data and neural network architecture, α can be optimized by gradient descent along with the weights of the network, interpolating between those scale-spaces.

Composition

The composition of different liftings also yields a lifting. Consider T_s and U_s two scale-spaces and $\Lambda[f](s,x) = (T_{s^{-\frac{1}{p}}} \circ U_{s^{-\frac{1}{p}}} f)(x)$ then

$$\begin{aligned}
\Lambda[R'_{t,z}f](s,x) &= (T_{s^{-\frac{1}{p}}} U_{s^{-\frac{1}{p}}} R'_{t,z} f)(x) \\
&= (T_{s^{-\frac{1}{p}}} R'_{t,z} U_{(st)^{-\frac{1}{p}}} f)(x) \\
&= R'_{t,z}(T_{(st)^{-\frac{1}{p}}} U_{(st)^{-\frac{1}{p}}} f)(x) \\
&= (T_{(st)^{-\frac{1}{p}}} U_{(st)^{-\frac{1}{p}}} f)(t^{-1}x + z) \\
&= \Lambda[f](st, s^{-1}x + z) \\
&= R_{t,z}(\Lambda[f])(s,x).
\end{aligned}$$

The Lasry-Lions operators (Angulo, 2014) can be obtained as compositions of the form $\delta_{q_{ct}} \circ \varepsilon_{q_t}$ and $\varepsilon_{q_{ct}} \delta_{q_t}$, with $c \in [0,1]$. The operators can vary between an erosion, or dilation, for $c = 0$ to an opening, or closing, for $c = 1$. For intermediate values, they possess interesting regularization properties (Angulo, 2014). This parameter could as well be learned by the network.

Now consider the operator

$$\psi_t^{(k,\alpha,c,p)} = \alpha \delta_{b_{p,ckt}} \varepsilon_{b_{p,kt}} + (1-\alpha) \varepsilon_{b_{p,ckt}} \delta_{b_{p,kt}}. \tag{2.28}$$

By changing the parameters (k, α, c, p), we can control the base size of the structuring element, the closeness to an extensive or anti-extensive morphological operator, its closeness to a morphological filter or to an additive operator and if the structuring elements are flat, smooth or conical. In other words, it can interpolate between morphological dilations, erosions, openings and closings and self-dual operators, with structuring elements of different sizes that can be flat or smooth.

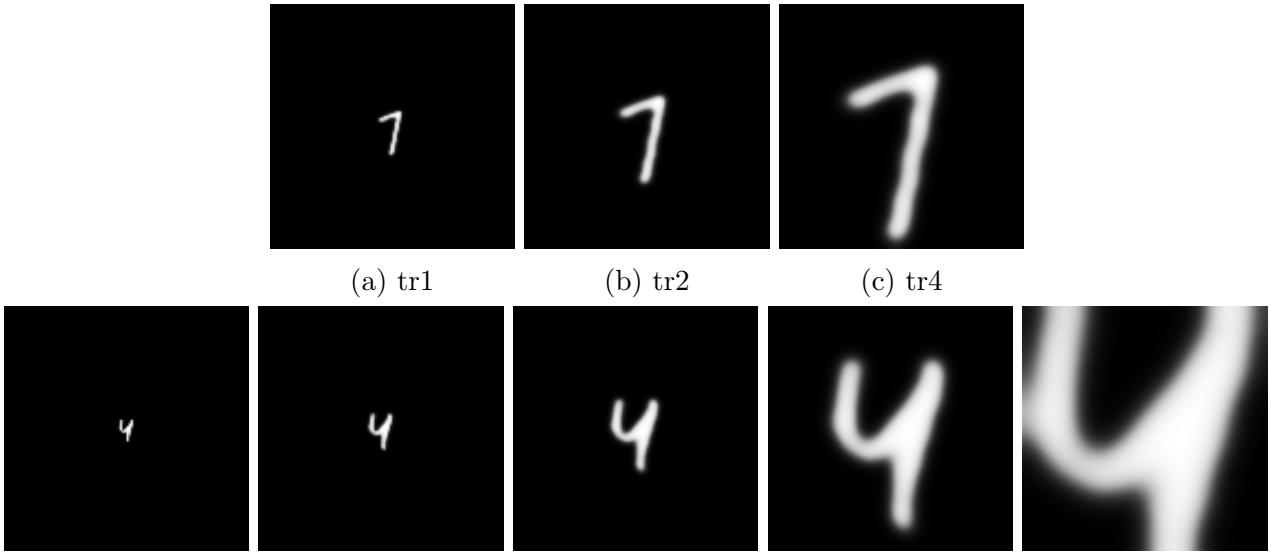
2.6 Experiments

2.6.1 Datasets

MNIST Large Scale

The MNIST Large Scale (Jansson and Lindeberg, 2020) dataset is built upon the MNIST dataset (LeCun et al., 2010) and was introduced to evaluate the ability of CNNs to generalize to scales not seen in the training set. The dataset contains three training sets, tr1, tr2 and tr4, which consist of 50000 samples from the MNIST dataset upscaled by factors one, two and four respectively. Each training set is accompanied by 10000 samples in the same scale to be used as validation sets. Figure 2.6(a) - (c) shows examples of images on each of the training sets corresponding to the same MNIST image.

As for the scales of the test set we have one for each of the scales $2^{\frac{i}{4}}, i = -4, \ldots, 12$, amounting to 17 test sets. Figure 2.6 shows examples of the same image on different test sets.



(a) tr1 (b) tr2 (c) tr4

(d) Test image on several different scales.

Figure 2.6: Examples of training and testing MNIST Large Scale images.

The rescalings are obtained from the MNIST images which have size 28×28. In all scales, MNIST Large Scale images have size 112×112, and images are either padded or cropped to match that shape.

Synthetic Shapes Dataset

In order to first assess the classification performance of scale-equivariant networks with different types of lifting operators we perform a segmentation experiment on a synthetic dataset. We are specifically interested in the case where the networks are trained in a dataset to segment objects divided into fixed scales, but in the test set objects can be scaled independently of one another, with the re-scaling factor following some probability distribution. Unlike in the classification case, it would be difficult to define a data augmentation technique that when applied to the training set mimics the behavior of the test set.

The dataset used for that consists of 224×224 synthetic binary images of shapes such as Figure 2.9 which are divided in three classes: disks, stars and the background. In the training set, only one scale is present, like in Figure 2.9(a). We construct test sets where each shape is re-scaled by a factor uniformly sampled from the interval $[2^{-i}, 2^{i}]$, in which we use $i = 1, 2$. Figure 2.9(b) shows an example of a test image. The train set contains 10000 images and the test sets contain 500 images each. The experiment is repeated ten times, each time generating a different training/test set pair.

2.6.2 Image Classification

Firstly we begin by examining properties of the scale-equivariant networks when applied to a task of image classification. Specifically we want to see what effect changing each module of the networks produces in the result based on unseen scales.

In order to separately evaluate different aspects of scale-equivariant neural networks, we begin by first defining a base architecture consisting of $L = 5$ layers consisting of scale-cross correlation layers, or convolution layers as a baseline. The layers have 12, 16, 24, 32 and 64 filters, based on the architecture used by Lindeberg (2022). We always use scales varying in a range of size 4. We also fix the use of max projection at the projection layer. We change other aspects of the network in order to test its components:

- **Lifting**: We train several different liftings in the model: Identity lifting, quadratic dilation scale-space, Gaussian scale-space and a convex combination of quadratic dilations and erosions (where the paremeter α of the convex combination is optimized as a network weight).
- **Varying scale dimension**: Filters can have a different size in the scale dimension, we test sizes 1 and 2. As was done by Worrall and Welling (2019), only some layers can have a scale greater than one. So we train a layer with a scale dimension of 1 in every layer and one using a scale dimension of 2 the second and fourth layers and 1 at the other layers.
- **Scale Dropout**: Tested models appear both with and without the usage of scale dropout. When present, its drop rate is set to 25%.

We compare our results with a CNN with the same number of layers and filters per layer, 3×3 filters and strided convolutions at layers two and four. We train the CNN baseline with and without scale jittering. Scale jittering refers to rescaling an image according to a random value sampled uniformly from the interval $[s^{-1}, s]$, for some $s > 0$, and padding or cropping the image to have the same size as before. When cropped, the location of the cropping box is random. We use a value of s equal to 4.

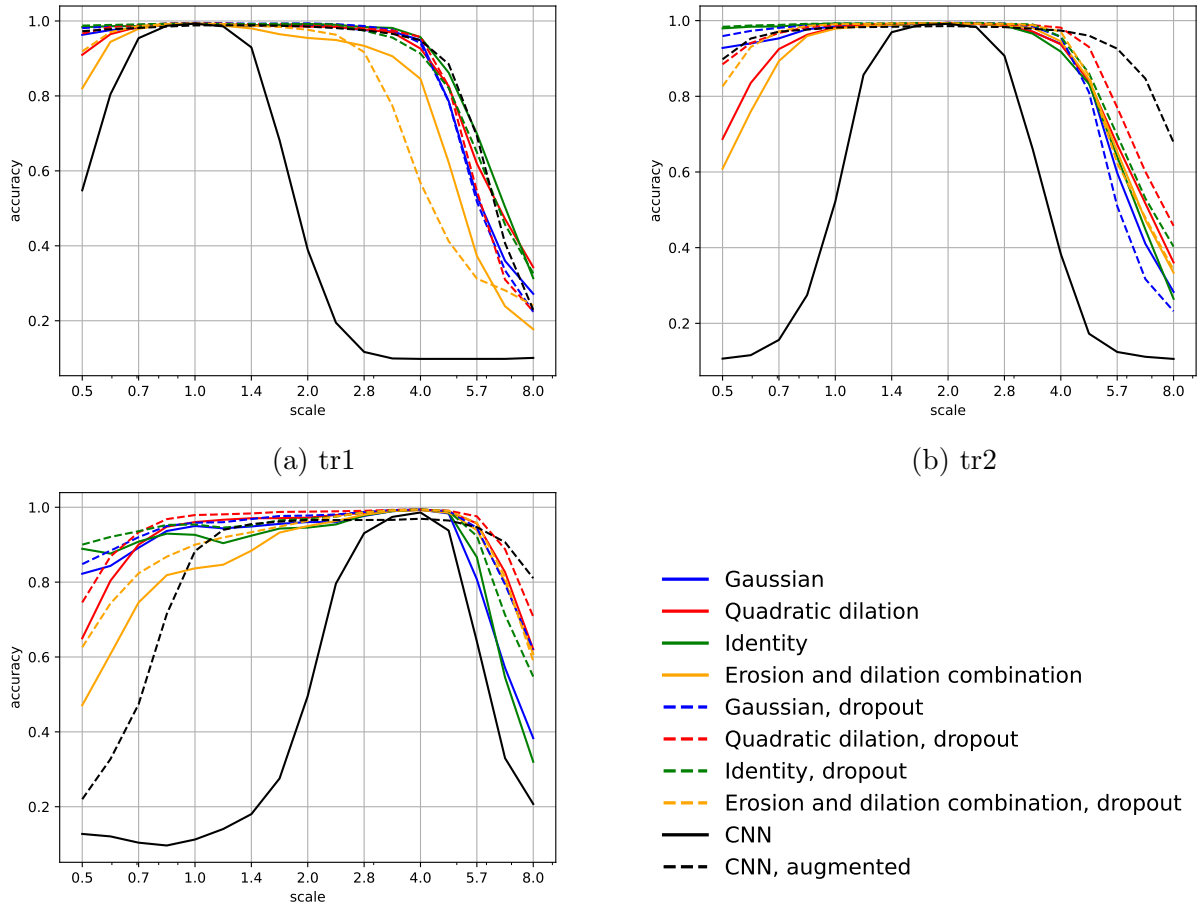
The accuracies obtained from different liftings are shown in Figure 2.7. First off we note that there does not seem to be a single lifting that performs better than the rest, instead it seems that the quadratic dilation and the identity liftings perform better in different situations. More specifically when training in tr1, the identity lifting has a better accuracy, in tr4 it is the dilation lifting, and in tr2 it depends on the test scale. Moreover, it seems that dropout consistently improved the performance of equivariant models.

We can see that the augmented CNN trained at scale 2 has a better generalization than the rest of the models, but when trained at scale 4 it underperforms other models for most scales, indicating that the equivariant models are more robust to the training set scale than the augmented CNN.

Accuracies obtained from models of different scale dimensions are shown in Figure 2.8. From the figure it becomes apparent that increasing the scale dimension to two does not necessarily improve performance or generalization. This result is consistent with the one in Worrall and Welling (2019) in the Cityscapes experiment, but in that case scale-equivariance was not explicitly tested.

2.6.3 Image segmentation

The architecture chosen for the equivariant models consists simply of six layers of semigroup cross-correlations. Because the scale cross-correlation has a naturally large receptive field,

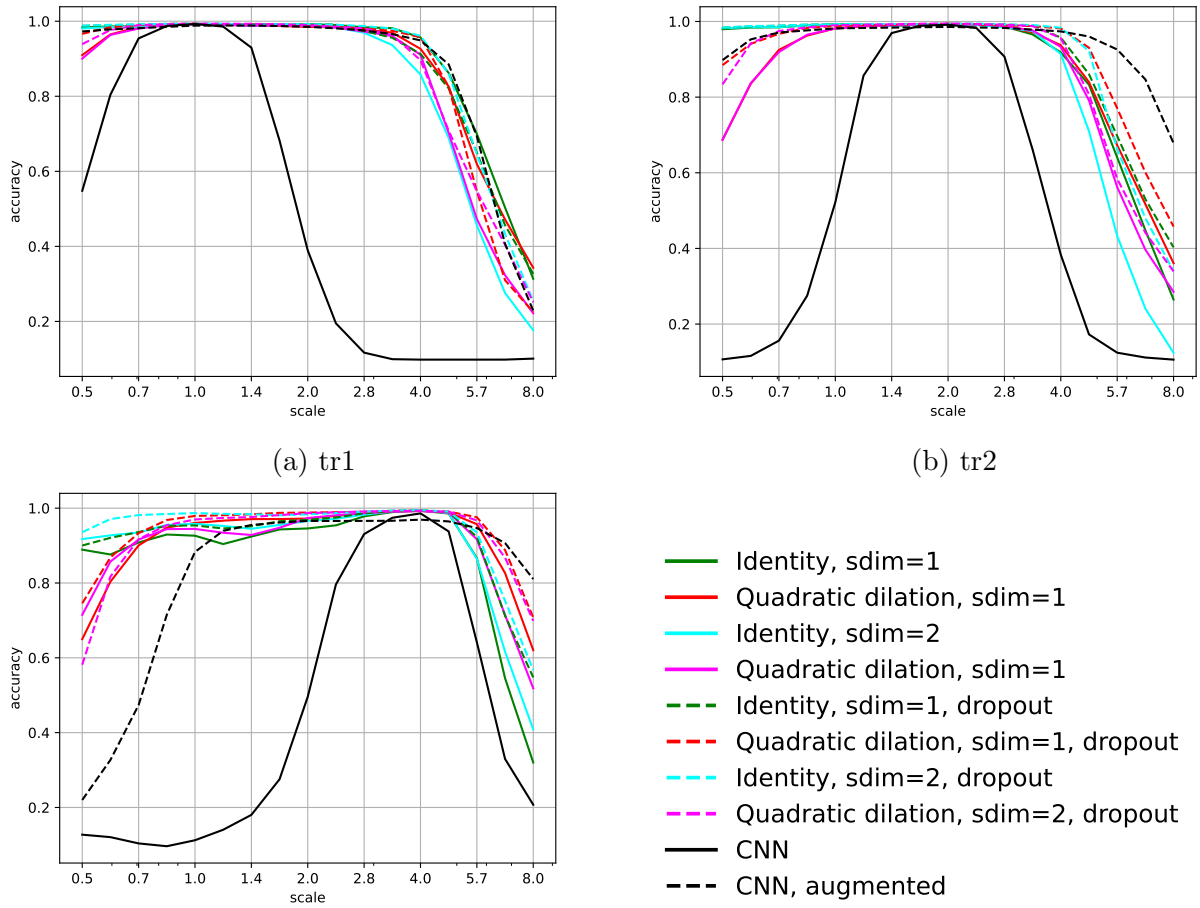


(a) tr1

(b) tr2

(c) tr4

Figure 2.7: Accuracy of scale equivariant models in the Large Scale MNIST dataset, trained on each of the training sets tr1, tr2 and tr4 and tested on all test sets. Four different liftings were used: Identity, Gaussian, quadratic dilations and a convex combination of dilation and erosion.



(a) tr1

(b) tr2

(c) tr4

Figure 2.8: Accuracies with models training with scale dimensions sdim = 1 and sdim = 2. Both identity and quadratic dilations liftings were compared, with scale dropout values of 0% and 25%. Accuracies of scale equivariant models in the Large Scale MNIST dataset, trained on each of the training sets tr1, tr2 and tr4 and tested on all test sets.

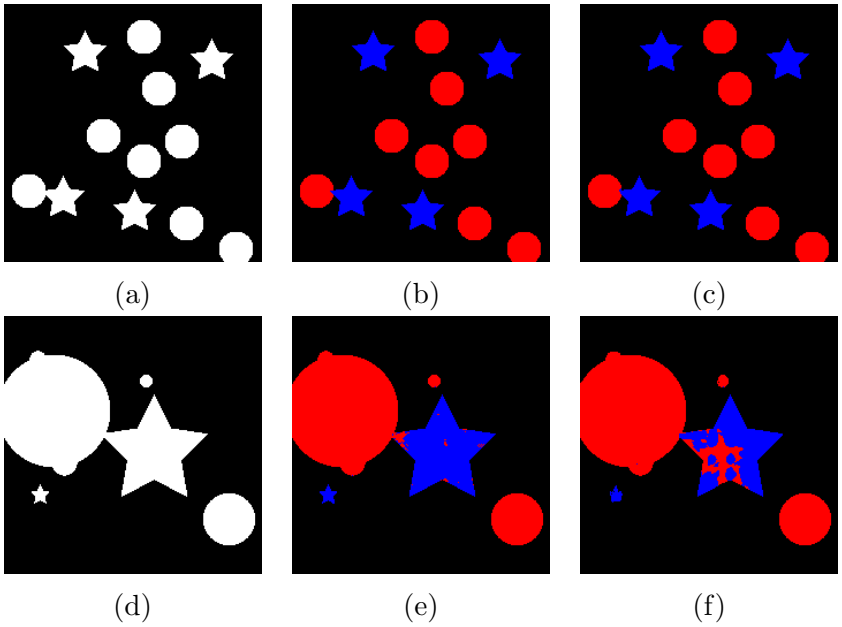


Figure 2.9: Example images from the (a) training and (d) test sets of the segmentation experiments, and segmentation results using equivariant models with the (b)(e) proposed dilation and (c)(f) Gaussian scale-spaces. Pixels in red are classified as *disk*, those in blue as *star*.

subsampling is not necessary. The output of the network is a three-channel image with the scores for each class, and the class is chosen as the coordinate with the greatest score. To quantitatively evaluate the models we use the Intersection over Union(IoU), or Jaccard index, between the prediction and the ground truth which can be written as

$$\mathrm{IoU}(\mathrm{GT}, \mathrm{Pred}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{2.29}$$

where GT is the ground truth image, Pred is the prediction by the network, TP is the number of true positives, FP of false positives and FN of false negatives. When more than two classes are present we compute the IoU for each class individually and then compute the mean.

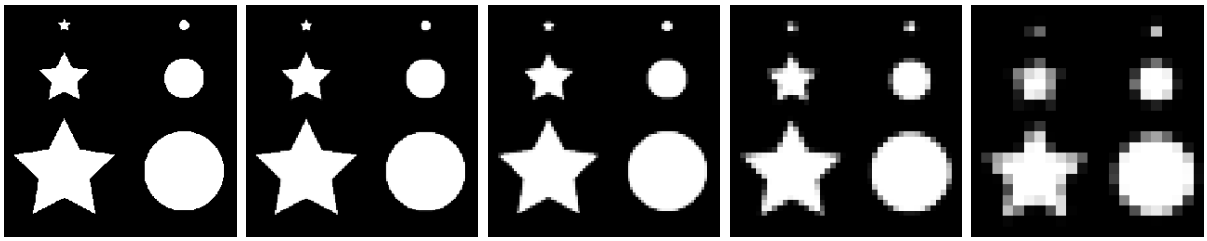
As baselines, we compare the models to a CNN with the same number of layers and a similar size and number of parameters, and also to a U-Net (Ronneberger et al., 2015) architecture. In Chapter 3 the comparison between segmentation by a U-Net and by scale-equivariant architectures will be studied more in-depth.

In Table 2.1 we compare the IoU obtained from different models. We see that CNN performs badly, compared to the equivariant models, even in the training set scale. This is partially attributed to the fact that the receptive field of the CNN is not as large, although having the same number of layers and a similar number of parameters. As expected, the equivariant models outperformed the CNN architectures and the U-Net architecture in the generalization to other scales.

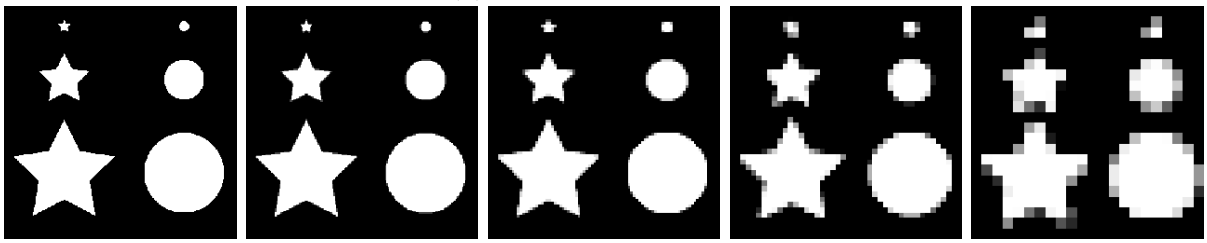
To analyze why the dilation is suited to this particular dataset, we can analyze the effect of applying a discrete re-scaling, i.e. a subsampling to the objects processed by the scale-spaces. In Fig. 2.10 we see the difference between a Gaussian and dilation lifting followed by a

Scales	Gaussian Lifting	Dilation Lifting	Closing Lifting	Id Lifting	U-Net	CNN
1.	0.9929 ± 0.0006	0.9929 ± 0.0006	0.9929 ± 0.0005	0.9927 ± 0.0008	0.9927 ± 0.0006	0.9083 ± 0.0006
$[\frac{1}{2}, 2]$	0.92 ± 0.06	0.97 ± 0.01	0.89 ± 0.06	0.91 ± 0.03	0.86 ± 0.02	0.68 ± 0.01
$[\frac{1}{4}, 4]$	0.88 ± 0.07	0.93 ± 0.02	0.86 ± 0.05	0.88 ± 0.03	0.70 ± 0.04	0.627 ± 0.008

Table 2.1: IoU between the ground truth images and the predictions obtained from equivariant models with different liftings, trained on images where objects only appear at scale one and tested on test sets where scales vary in the range $[2^{-i}, 2^i]$ for $i = 0, 1, 2$.



(a) Gaussian scale-space



(b) Quadratic Dilation scale-space

Figure 2.10: The same image after being processed by the Gaussian (a) and quadratic dilations (b) scale-spaces and being subsampled by factors 2^i $i = 0, 1, 2, 3, 4$.

subsampling operator. Indeed, the persistence of concavities of the star shapes makes it easier to distinguish the objects in the last images.

2.7 Conclusions

In this chapter the semigroup cross-correlation approach to scale-equivariant networks was reviewed, in particular the scale-cross-correlation, which is the special case of the semigroup cross-correlation when applied to a semigroup of scales and translations. Some interesting properties were discussed, for example the large receptive field that these models have compared to matched CNNs. We also talked about the lifting and projection layers.

Because input images to neural networks are normally defined as signals on $\mathbb{Z}^2$ and the scale-cross-correlation is defined for signals on a semigroup which is the Cartesian product of $\mathbb{Z}^2$ with a semigroup of scales, it is necessary to map images to the semigroup, and because the end results should be equivariant, this has to be done in an equivariant manner. The lifting layer does that. We extend the approach to lifting given by Worrall and Welling (2019), where only the Gaussian scale-space was used, to a general class of scale spaces, particularly it allows for both morphological and the Gaussian scale-spaces. We also show that linear combinations

and compositions of liftings are also liftings, allowing for example for the Lasry-Lions lifting.

The projection layer maps signals on the semigroup back to signals on $\mathbb{Z}^2$ in a way that is approximately equivariant. We also introduce the scale dropout layer before the projection layer. The scale dropout improves robustness of the model by randomly dropping scales during training and encouraging the use of multi-scale information.

In the experimental section, two tasks were studied: classification and segmentation. Classification is done in the Large Scale MNIST dataset, a dataset obtained by rescaling images of MNIST, and a synthetic dataset based on the segmentation of different shapes and the background. In both tasks the goal was to train the model with images at a certain scale range and test it in scales unseen during training. In both tasks the images have a clearly defined polarity in their pixels, for which we expected the morphological scale-spaces to be particularly adapted. Our experiments confirmed this hypothesis, in particular for the quadratic dilation scale-space, as it can help preserve the shapes of objects after a re-scaling. Even when compared to data augmentation in the MNIST Large Scale experiment, the equivariant models were shown to have a higher degree of robustness with respect to the training scale.

2.8 Résumé en Français

Dans ce chapitre, la théorie des réseaux équivariants basés sur la corrélation-croisée sur les semi-groupes a été revue. En particulier, nous utilisons la corrélation-croisée d'échelle, c'est-à-dire la corrélation-croisée sur un semi-groupe appliquée à un semi-groupe d'échelles et translations. Quelques propriétés intéressantes sont discutées, comme le champ réceptif de ces modèles, qui est significativement plus grand que celui des CNNs similaires.

Les images d'entrée aux réseaux de neurones sont normalement définies comme des signaux sur $\mathbb{Z}^2$ et la corrélation croisée d'échelle est définie par des signaux sur un semi-groupe qui est le produit Cartésien de $\mathbb{Z}^2$ et un semi-group d'échelle. Cela nous oblige à mapper les images à ce produit Cartésien. Comme les résultats finaux doivent être équivariants, cela doit être fait de façon équivariante. C'est à cela que sert la couche de *dépliement*. Nous généralisons l'approche de dépliement proposée par Worrall and Welling (2019), où seulement l'espace d'échelle Gaussien a été utilisée, à une classe générale d'espaces d'échelle. En particulier cela nous permet d'utiliser les espaces d'échelle morphologiques, comme par exemple les espaces d'échelle de dilatations, ainsi comme l'espace d'échelle Gaussien. Nous montrons aussi que les combinaisons linéaires et compositions de dépliements sont aussi des dépliements, ce qui nous permet d'utiliser, par exemple, un dépliement basé sur l'opérateur de Lasry-Lions (Angulo, 2014).

La couche de projection mappe un signal sur le semi-groupe à un signal sur $\mathbb{Z}^2$ de façon approximativement équivariante. Nous présentons aussi la couche de *dropout* d'échelle avant la couche de projection. Le dropout d'échelle améliore la robustesse du modèle en mettant à zéro les sorties correspondantes à certaines échelles, choisies aléatoirement, de la dernière couche avant la projection pendant l'apprentissage du réseau, ce qui encourage l'utilisation d'information multi-échelle.

Dans la section expérimentale, deux tâches ont été étudiées : la classification et la segmentation. La classification est faite dans la base de données MNIST en grande échelle (Jansson and Lindeberg, 2020), qui est obtenue en changent l'échelle des images de MNIST (LeCun

et al., 2010), et la segmentation sur une base de données synthétique. Dans les deux tâches les images ont une polarité clairement définie, pour laquelle nous nous attendions que les espaces d'échelle morphologiques soient particulièrement adaptés. Nos expériences confirment cette hypothèse, en particulier pour l'espace d'échelle des dilatations quadratiques, car il peut aider à mieux préserver les formes des objets après un changement d'échelle. Même comparé à l'augmentation de données dans la base MNIST en grande échelle, nous avons montré des résultats qui indiquent que les modèles équivariants ont un plus grand taux de robustesse au changement de l'échelle de l'ensemble d'apprentissage.

Chapter 3

Scale Equivariant U-Net

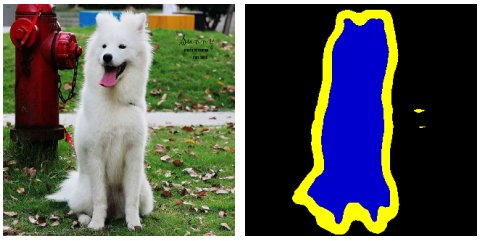
3.1 Introduction

In the previous chapter a strong argument was made for scale-equivariant networks when symmetry to changes in scale are present in the network, however, a larger architecture was not fully investigated. Moreover, the behaviour of common layers such as pooling and upsampling in a scale-equivariant setting were ignored. In this chapter a larger architecture that uses both aforementioned operators is studied. The SResNet was proposed in Worrall and Welling (2019) in order to perform scale-equivariant segmentation and obtained good results in the cityscapes dataset, but it was not tested in re-scaled images, meaning that it is difficult to determine the contribution of scale-equivariance. Moreover it restricts the upsampling operator to be applied outside of semigroup domain, i.e. after projection, meaning that it is restricted to apply an upsampling by a large factor at the last layer to compensate for its downsamplings.

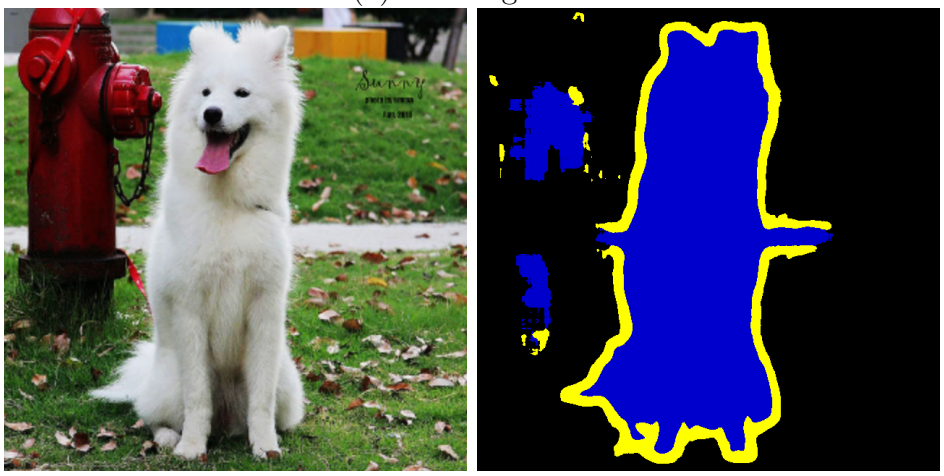
The U-Net architecture (Ronneberger et al., 2015) has become famous for its great performance in semantic segmentation. It is a fully convolutional neural network, *i.e.* a CNN without any dense layer. Thanks to this, its output is equivariant to a certain subgroup of translations. However, architectures like U-Net are not scale equivariant *a priori*, and experiments show they are not in practice (Sangalli et al., 2021) as illustrated by Figure 3.1. Having a scale-equivariant counterpart of such an architecture is desirable as scale symmetry is frequently present in semantic segmentation data. For this reason in this chapter the Scale-Equivariant U-Net (SEU-Net) is introduced.

The SEU-Net is obtained by substituting the building blocks of the U-Net by their scale-equivariant counterparts. The counterpart to the convolution, a linear scale-equivariant operator, are the scale-cross correlations. Counterparts of subsampling and upsampling operators, both which are important components of the U-Net, are studied in this chapter.

We begin the chapter by disscussing related work in Section 3.2 followed by a review the architecture of U-Net in Section 3.3. In Section 3.4 the SEU-Net is proposed and reviewed and its building blocks are analysed in regards to scale-equivariance. Pooling and upsampling are given special attention. Later The SEU-Net is tested and compared against the U-Net and the SResNet in Section 3.5. More specifically the SEU-Net is tested in segmentation tasks, where the training images are at a fixed scale and the test images are re-scaled to scales unseen during training. The datasets used for the experiments were a dataset of strand images for plane



(a) Training scale



(b) Unseen scale

Figure 3.1: Example where a U-Net trained on one scale distribution and is applied to predict the output on the training(a) and on an unseen(b) scale i.e. a scale not well represented in the training scale distribution. The image with the unseen scale represents the same object but the U-Net no longer segments it correctly.

manufacture, the Oxford-IIIT dataset for natural image segmentation and the DIC-C2DH-HeLa dataset for cell segmentation. We also compare the SEU-Net with a scale-augmented U-Net in both the Oxford-IIIT Pet and the DIC-C2DH-HeLa datasets and obtain positive results as the SEU-Net can obtain a higher degree of scale generalization without losing performance at scale 1, compared to the U-Net with augmentation. In the last experiment the SEU-Net is also shown to have a good interation with scale augmentation. We end the chapter in Section 3.6

3.2 Related Work

As discussed in the last chapter, scale-equivariance and scale-invariance are topics already discussed in the deep learning literature (Zhu et al., 2019; Ghosh and Gupta, 2019; Jansson and Lindeberg, 2020; Lindeberg, 2022; Sosnovik et al., 2019). The experimental benchmarks found in those papers are interesting as a first way to measure equivariance, but tend to be based on very simple tasks, such as the classification of re-scaled digits from the MNIST dataset or low resolution images of clothes from the Fashion-MNIST dataset. In Sosnovik et al. (2021), combinations of base filters are optimized to minimize the equivariance error of discrete scale convolutions. This is applied to classification, tracking and geometry estimation, but not

segmentation.

In Worrall and Welling (2019) the semigroup equivariant models were applied to classification and semantic segmentation of datasets of large images, achieving better results compared to matched non-equivariant architectures. Yet, the role of scale-equivariance was not isolated, as the performance of the models was not measured for inputs on scales unseen in the training set. Later on, this approach was revisited by Sangalli et al. (2021) (Chapter 2), where the Gaussian scale-space originally used was generalized to other scale-spaces and the models were tested in experiments where the networks are trained in one fixed scale and tested on unseen scales, albeit on synthetic or simple datasets. In all these approaches, the authors either avoided pooling and upsampling in their architectures, or used them but did not discuss their impact on scale equivariance.

While scale-equivariance has been a topic in the literature for some time, as far as we know a scale-equivariant U-Net has not yet been proposed, contrary to the rotation-equivariance case (Chidester et al., 2019). Moreover, the current benchmarks for scale-equivariance were either based on simple datasets like MNIST or did not explicitly measure the equivariance in their segmentation or classification experiments, by training the networks on one fixed scale and testing on unseen scales. Here we propose semantic segmentation experiments based on natural data which measure the equivariance of the predictions.

3.3 U-Net

The U-Net (Ronneberger et al., 2015), illustrated in Figure 3.2(a), is a CNN architecture for semantic segmentation. It consists of of an auto-encoder structure and as such can be divided into two main parts: the *encoder* and the *decoder*. The encoder and decoder are linked by blocks referred to as *skip connections*. The encoder and decoder can be further decomposed into four main blocks: convolution blocks, pooling, upsampling and skip connections.

Here we given a description of the U-Net. First off, to parametrize it we use to integer parameters, the *depth* D and the number of channels at the first layer C. The encoder consists of L blocks $B_l^{\downarrow}$ $l = 1, \dots, L$, of two convolutions followed by pooling or more precisely

$$B_l^{\downarrow}[f] \coloneqq \text{Pool} \circ \sigma \circ \text{BN}_l^2 \left(h_l^2 * \sigma \circ \text{BN}_l^1 (h_l^1 * B_{l-1}^{\downarrow}[f]) \right) \tag{3.1}$$

where σ denotes a non-linearity such as ReLU or leaky ReLU, BN_l^i denotes a batch normalization layer (Ioffe and Szegedy, 2015) and h_l^i denote convolutional filters for $i \in \{1, 2\}$ $l \in \{1, \dots, D\}$ and Pool denotes a subsampling functions. The filters h_l^i both have output dimension given by $2^l C$. In the original paper they used 3×3 convolutional filters and a 2×2 max-pooling with stride two as subsampling the first encoder block is the identity $B_0^{\downarrow}[f] \coloneqq f$.

The decoder blocks are applied in reverse order. The first decoder block is just the last encoder block $B_L^{\uparrow}[f] \coloneqq B_L^{\downarrow}[f]$ and subsequent decoder blocks are the concatenation of the upsampling of the previous decoder block with the encoder block at the same level, followed by two convolutions, i.e. for $l \in \{0, \dots, L-1\}$

$$B_l^{\uparrow}[f] \coloneqq \sigma \circ \text{BN}_l^2 \left(w_l^2 * \sigma \circ \text{BN}_l^1 (w_l^1 * \text{conc}[B_l^{\downarrow}[f], \text{Upsamp}(B_{l+1}^{\uparrow})]) \right) \tag{3.2}$$

where conc : $(\mathbb{R}^{C_1})^\Omega \times (\mathbb{R}^{C_2})^\Omega \to (\mathbb{R}^{C_1+C_2})^\Omega$ denotes concatenation, or equivalently the skip connections, i.e.

$$\text{conc}[f, g](x) \coloneqq (f_1(x), \dots, f_{C_1}(x), g_1(x), \cdot, g_{C_2}(x)), \tag{3.3}$$

and Upsamp denotes some kind of upsampling (2×2 transposed convolution with stride two in the original paper). The convolution filters w_l^i, $i \in \{1, 2\}$, $l \in \{0, \dots, L-1\}$ have output dimension $2^l C$.

Finally the output of the network is given by the application of a convolutional layer at $B_0^\uparrow$ to compute the logits of the segmentation layer.

Because of its skip connections, the U-Net effectively removes the spatial dimension bottleneck of classical auto-encoders, where a high resolution segmentation map must be entirely recovered from a low resolution feature map at the output of the encoder.

Keep in mind that the description we gave in this section refers to a specific type of U-Net architecture. In practice the encoder architecture can be changed to be different kinds of architectures, including modern and powerful classification architectures such as a ResNet (Abedalla et al., 2020). This practice is commonly referred to as changing the *backbone* of the U-Net. In this chapter we use the formulation we provided.

3.4 Scale-Equivariant U-Net

In this section we aim to propose the Scale-Equivariant U-Net (SEU-Net), i.e. a U-Net architecture made with scale-equivariant components. The resulting model is approximately scale-equivariant and should have an increased generalization capacity compared to the U-Net. The architecture proposed for the SEU-Net is described in Figure 3.2(b).

In the framework of the previous chapter, a network can be written as $\Gamma = \Pi \circ \Sigma \circ \Lambda$, where Λ and Π are the lifting and projection respectively, and Σ is the core part of the network mapping the lifted space to itself. We already saw that $\Lambda \circ R'_{\gamma^k,z} = R_{\gamma^k,z} \circ \Lambda$ and we assume $R'_{\gamma^k,z} \circ \Pi \approx \Pi \circ R_{\gamma^k,z}$. Hence, to build a (approximately) scale-equivariant network, it is sufficient to have $\Sigma \circ R_{\gamma^k,z} = R_{\gamma^k,z} \circ \Sigma$. In order to remove some choice of hyperparameters, lifting and projections layers are fixed being a Gaussian lifting and a max-projection.

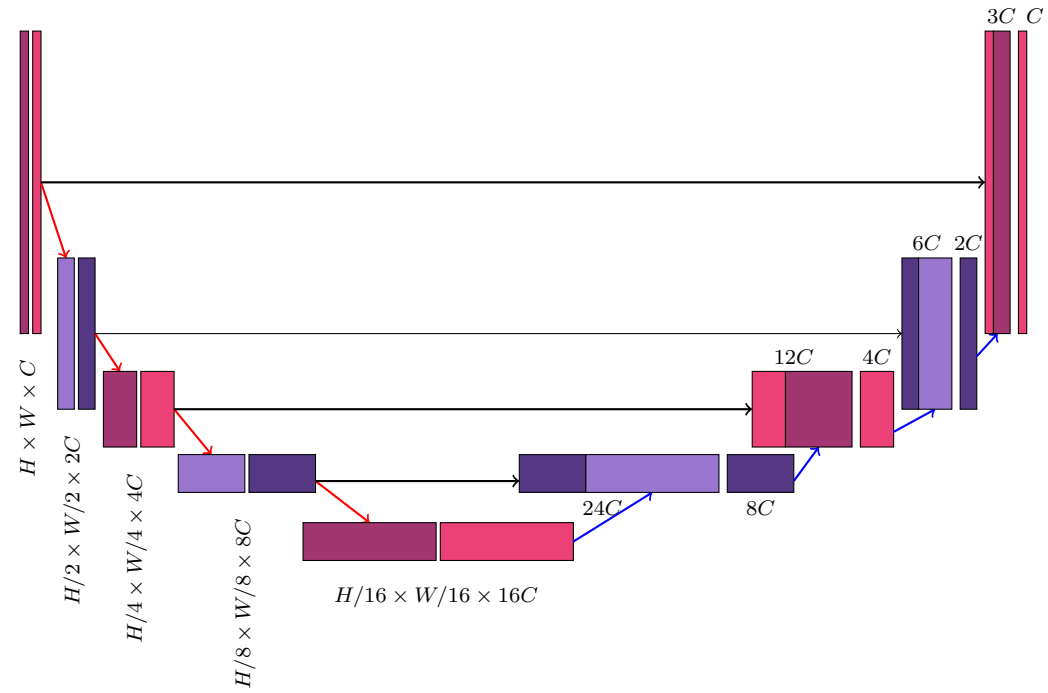
In particular, a way to render the U-Net scale-equivariant is to design scale-equivariant versions of its components in Σ. Convolutions are already rendered equivariant by scale-cross-correlations, and since batch normalization and pixelwise activations preserve scale-equivariance, we can easily render the convolution blocks equivariant. Skip connections are also equivariant as is.

The rest of this section is dedicated to the remaining components: pooling and upsampling.

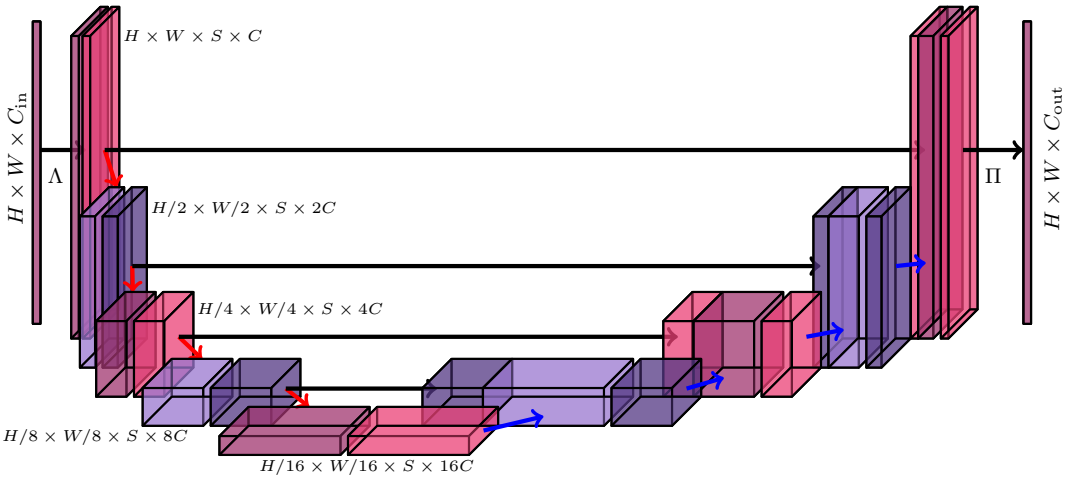
3.4.1 Pooling

Classical pooling operators naively applied scale by scale do not result in scale-equivariant poolings in the lifting space. For example, the max-pooling $\text{MP}[f](x) = \max_{y \in N} f(rx + y)$ with strides $r \in \mathbb{N}$ and neighborhood $N \subseteq \mathbb{Z}^2$ (usually a $r \times r$ square). Its naive extension to the lifted space $\text{MP}'[f](\gamma^k, x) = \text{MP}[f(\gamma^k, \cdot)](x)$ $\forall k \in \mathbb{N}$ does not commute with $R_{\gamma^k,x}$.

Strided convolutions however, generalize well to this scenario, written as the subsampling operator $D_t[f](\gamma^k, x) = f(\gamma^k, tx)$ following a scale-cross-correlation. We can verify that it



(a) U-Net



(b) SEU-Net

Figure 3.2: Diagrams representing a U-Net and a SEU-Net with depth 4 and C channels at the first layer given inputs of spatial dimension $H \times W$. The SEU-Net blocks have the same dimension of their corresponding U-Net blocks, but with the additional scale dimension i.e. the first block has dimensions $H \times W \times S \times C$, the second one $H/2 \times W/2 \times S \times 2C$ and so on. The lifting and projection layers are denoted respectively Λ and Π. In both diagrams, red arrows signify subsampling layers and blue arrows signify upsampling layers. Features maps on the same level (i.e. y coordinate in (a) and z coordinate in (b)) have the same spatial dimension. Subsequent feature maps not followed by an arrow are obtained by convolution / scale-cross-correlation layers followed by nonlinearities and batch normalization.

is scale-equivariant: $D_t[R_{\gamma^k,x}f](\gamma^l, y) = (R_{\gamma^k,x}f)(\gamma^l, ty) = f(\gamma^{l+k}, \gamma^k ty) = D_t[f](\gamma^{l+k}, \gamma^k y) = R_{\gamma^k,x}[D_t f](\gamma^l, y)$. We use strides as the subsampling in our networks, with a stride of $t = 2$.

Besides the strided scale-crosscorrelations we used, we can define another class of pooling operators, inspired by classical max-pooling. Let us place ourselves in a slightly different case of pooling a function in a continuous domain $f : \mathcal{S} \times \mathbb{R}^2 \to \mathbb{R}$, with $\mathcal{S} \times \mathbb{R}^2$ acting on it by $R_{\gamma^l,z}f(\gamma^k, x) = f(\gamma^{k+l}, \gamma^l x + z)$, $k, l \in \mathbb{N}$, $x \in \mathbb{R}^2$, $z \in \mathbb{Z}^2$. We define the pooling of f as an operator F followed by a downsampling $D_{\gamma^l}[f](\gamma^k, z) = f(\gamma^k, \gamma^l z)$

$$P[f] = D_{\gamma^l} F f. \tag{3.4}$$

If F commutes with $R_{\gamma^k,x}$, then so does P. We consider three pooling functions: $F_{\text{id}} = \text{id}$ (strides) and two dilation scale-spaces (Heijmans and van den Boomgaard, 2002):

- The max-pooling of scale-semigroup-valued images is given by a re-scaled max-pooling

$$F_{\max}[f](\gamma^k, z) = \sup_{y \in N_k \times N_k} \{f(z - y)\} \tag{3.5}$$

  where $N_k = \{\gamma^k x | x \in N\}$ and N is for example a γ^l-sided square in $\mathbb{R}^2$.

- The quadratic dilation (quadpool) scale-space is a morphological counterpart to the Gaussian scale-space (Van Den Boomgaard and Smeulders, 1994) defined by

$$F_{\text{quad}}[f](\gamma^k, z) = \sup_{y \in \mathbb{R}^2} \left\{ f(z - y) - \frac{\|y\|^2}{c\gamma^{2k}} \right\}, \tag{3.6}$$

  where $c > 0$ is some constant.

In contrast to the strided scale-cross-correlations given by F_{id}, the functions $F_{\max}$ and F_{quad} are scale-equivariant only in this continuous setting, their discretized versions are not actually equivariant. Nonetheless, a network employing scale-cross-correlations and these poolings would be equivariant when applied to signals in the domain $\mathcal{S} \times \mathbb{R}^2$.

3.4.2 Upsampling

Upsampling blocks are a well established part of modern neural network architectures for segmentation and other tasks. In order to extend upsampling to a scale-equivariant setting, we look at the case where f is defined on a continuous domain. In that case, the downsampling D_{γ^l} has an inverse U_{γ^l} which is the natural upsampling.

In the discrete case the problem becomes more complicated as downscaling is not invertible, but for $k, l \in \mathbb{N}$ we can define an upsampling U_{γ^l} as an operator satisfying $\forall x \in \mathbb{Z}^2$

$$U_{\gamma^k}[f](\gamma^l, \gamma^k x) = f(\gamma^l, x) \quad \text{and} \quad U_{\gamma^{lk}} = U_{\gamma^k} \circ U_{\gamma^l}. \tag{3.7}$$

With this, we have $D_{\gamma^k} \circ U_{\gamma^k} = \text{id}$. For all k, $U_{\gamma^k}(f)$ values are only restricted in the points $y \in k\mathbb{Z}^2 = \{kx | x \in \mathbb{Z}^2\}$, and the values on the other pixels can be defined in several ways (*e.g.* copies, interpolation) as long as it satisfies (3.7). Now, assume

$$U_{\gamma^l} R_{\gamma^k,x} f = R_{\gamma^k,\gamma^l x} U_{\gamma^l} f, \tag{3.8}$$

for any f, and disregarding the approximation in pooling, then $\Sigma \circ R_{\gamma^k,x} = R_{\gamma^k,x} \circ \Sigma$. Indeed let $\psi_i = D_{\gamma^l} L_i \cdots D_{\gamma^l} L_1$, $i = 1, \ldots, m$ denote the part of a SEU-Net of height m from the layer after lifting until the i-th downsampling block, where L_j, $j = 1, \ldots, m$, are blocks that commute with $R_{\gamma^k,x}$(constructed by scale-cross-correlations, pointwise activations and batch normalization). Denote $\phi_m = L_{m+1}\psi_m$ and $\phi_i = L_i C(U_{\gamma^l}\phi_{i+1}, \psi_i)$, $i = m, \ldots, 1$ where C denotes concatenation. With the above hypothesis, we have $\phi_i R_{\gamma^k,x} f = R_{\gamma^k,\gamma^{li}x}\phi_i f$. In particular, $R_{\gamma^k,x}\phi_0 f = \phi_0 R_{\gamma^k,x} f$, and we notice that ϕ_0 is precisely Σ.

The sufficient condition $U_{\gamma^l} R_{\gamma^k,x} f = R_{\gamma^k,\gamma^l x} U_{\gamma^l} f$ is not verified in general:

We can show that $U_{\gamma^l} R_{\gamma^k,x} f \neq R_{\gamma^k,\gamma^l x} U_{\gamma^l} f$ for at least one lifted image f, one couple of integers (k, l) and a point $x \in \mathbb{Z}^2$. Note that U_{γ^l} is an upsampling defined in the associated paper.

Given any $k \in \mathbb{N}$, take $l = k$, $x = (0, 0)$ and any two lifted images f_1 and f_2 that coincide on certain points,

$$f_1(s, \gamma^k y) = f_2(s, \gamma^k y) \quad \forall s \in \mathcal{S}, y \in \mathbb{Z}^2,$$

and are different elsewhere, as illustrated in Figure 3.3. Let us show that $U_{\gamma^l} R_{\gamma^k,x} f_i \neq R_{\gamma^k,\gamma^l x} U_{\gamma^l} f_i$ either for $i = 1$ or $i = 2$ or both. The set of points where f_1 and f_2 coincide implies in particular that $R_{\gamma^k,0} f_1 = R_{\gamma^k,0} f_2$. Then we have

$$R_{\gamma^k,0} U_{\gamma^k} f_1 \neq R_{\gamma^k,0} U_{\gamma^k} f_2,$$

as $R_{\gamma^k,0} U_{\gamma^k} f_i : (s, y) \mapsto f_i(\gamma^k s, y)$, and $f_1(\gamma^k s, y) \neq f_2(\gamma^k s, y)$ for $y \notin k\mathbb{Z}^2$. Note that $R_{\gamma^k,0} U_{\gamma^k}$ is nothing else than an upsampling followed by a downsampling, as in Figure 3.3.
Since, on the other hand, $R_{\gamma^k,0} f_1 = R_{\gamma^k,0} f_2$, we get

$$U_{\gamma^k} R_{\gamma^k,0} f_1 = U_{\gamma^k} R_{\gamma^k,0} f_2.$$

Hence, either $R_{k,0} U_k f_1 \neq U_k R_{k,0} f_1$ or $R_{k,0} U_k f_2 \neq U_k R_{k,0} f_2$ or both, proving our point.

The sufficient condition (3.8) is not verified in general, but Proposition 1 shows a case where it is valid.

Proposition 1. *For $N \in \mathbb{N}^*$ and $i \in \{1, \ldots, N\}$, let $\mathcal{U}_i = \{U_{\gamma^{n_i+l}} f_i | l \in \mathbb{N}\}$, where each $f_i : G \to \mathbb{R}^n$ is a function on G and each n_i an integer. Let $n_0 \leq \min\{n_i | i = 1, \ldots, N\}$ and $\mathcal{U} = \bigcup_{i=1}^{N} \mathcal{U}_i$. Then for all $f \in \mathcal{U}$, and $k, l \in \mathbb{N}$ such that $k - l \leq n_0$,*

$$U_{\gamma^l} R_{\gamma^k,x} f = R_{\gamma^k,\gamma^l x} U_{\gamma^l} f. \tag{3.9}$$

Proof. First, consider $k < m$

$$\begin{aligned} R_{\gamma^k,0} \circ U_{\gamma^k} f(\gamma^p, y) &= (U_{\gamma^k} f)(\gamma^k \gamma^p, \gamma^k y) \\ &= f(\gamma^k \gamma^p, y) \end{aligned}$$

so $R_{\gamma^k,0} U_{\gamma^m} f(\gamma^p, y) = U_{\gamma^{m-k}} f(\gamma^{p+m}, y)$.

Now, let $f = U_{\gamma^m} f_i \in F$, $k \leq \min\{n_i | i = 1, \ldots, N\} \leq m$, we have

$$\begin{aligned} U_\gamma R_{\gamma^k,x} f(\gamma^p, y) &= U_\gamma R_{\gamma^k,x} U_{\gamma^m} f_i(\gamma^p, y) \\ &= U_\gamma R_{1,x} R_{\gamma^k,0} U_{\gamma^m} f_i(\gamma^p, y) \\ &= R_{1,\gamma x} U_\gamma U_{\gamma^{m-k}} f_i(\gamma^{p+m} r, y) \\ &= U_{\gamma^{m-k+1}} f_i(\gamma^{m+p}, y + \gamma x) \end{aligned}$$

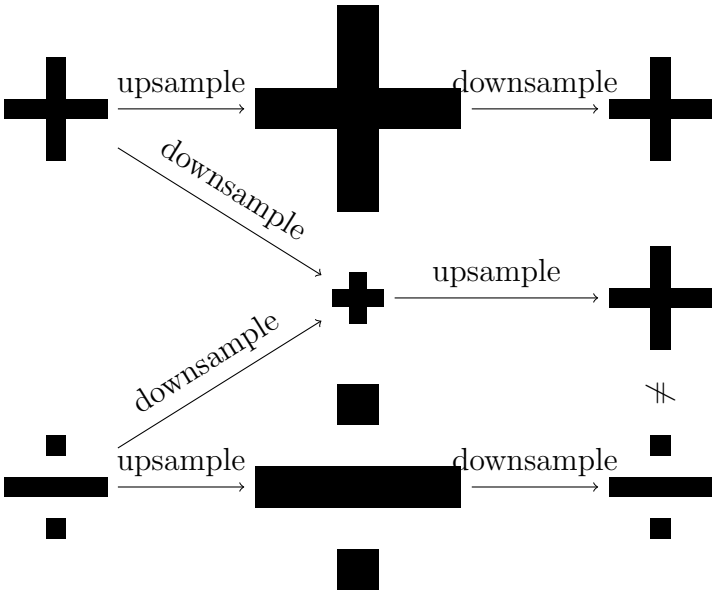


Figure 3.3: Example illustrating the problem with upsampling in a scale-equivariant architecture. We have images f_1 and f_2 such that when both are downsampled and then upsampled they yield the same result, but if both are upsampled and then downsampled they yield different results.

and, on the other hand

$$\begin{aligned} R_{\gamma^k,\gamma x} U_\gamma f(\gamma^p, y) &= R_{\gamma^k,\gamma x} U_\gamma U_{\gamma^m} f_i(\gamma^p, y) \\ &= R_{1,\gamma x} R_{\gamma^k,0} U_{\gamma^{m+1}} f_i(\gamma^p, y) \\ &= R_{1,\gamma x} U_{\gamma^{m-k+1}} f_i(\gamma^{p+k}, y) \\ &= U_{\gamma^{m-k+1}} f_i(\gamma^{p+m}, y + \gamma x) \\ &= U_\gamma R_{\gamma^k,x} f(\gamma^p, y), \end{aligned}$$

implying $U_\gamma R_{\gamma^k,x} f = R_{\gamma^k,\gamma x} U_\gamma f$. Repeated application gives us the desired result. □

This property states that upsampling behaves as an equivariant operator as long as the input image is an upsampling of some image in a base scale. It can be interpreted as saying that the downscaling should not destroy information of the images in $\mathcal{U}$. We model this by constraining the scaling factors of the downscaling actions and assuming that the objects of interest in an image are sufficiently big. We would like to point out that this hypothesis is never verified but reasonable for most of the datasets for semantic segmentation.

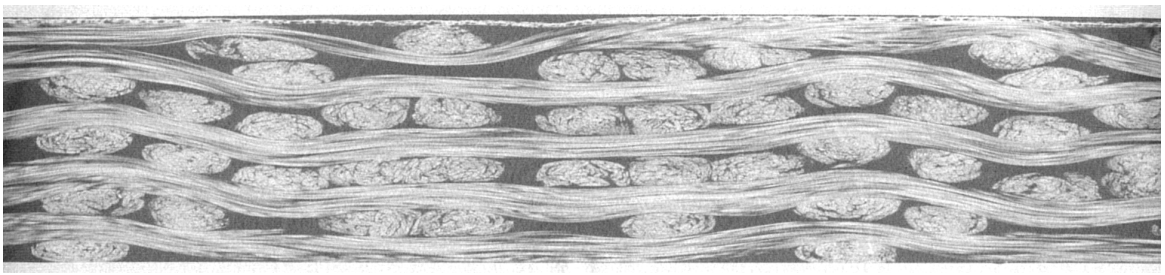
Before moving on to the experimental part, let us sum up the theoretical properties of a SEU-Net $\Gamma = \Pi \circ \Sigma \circ \Lambda$. By our construction we can hope for an approximated scale-equivariance $\Gamma \circ \mathbb{R}'_{\gamma^k,z} \approx \mathbb{R}'_{\gamma^k,z} \circ \Gamma$. Three approximations prevent from exact equivariance. The first one is the approximated equivariance of the projection operator Π, which is intrinsic to the lifting approach. The second one is the discretization of the pooling operators. The last one is the assumption to guarantee an equivariant upsampling, which is never verified in practice. We will see in our experiments that the SEU-Net shows a high degree of scale-equivariance despite

these approximations. Furthermore, note that each of these approximations is intrinsic to the problem. If the problem was formulated in the domain $\mathcal{S}_\gamma \times \mathbb{R}^2$ instead of G, all of these inaccuracies, except for the one coming from the projection layer, would be avoided on paper, but they would all be present at implementation. The same is true for every scale-equivariant network using an upsampling satisfying our definition.

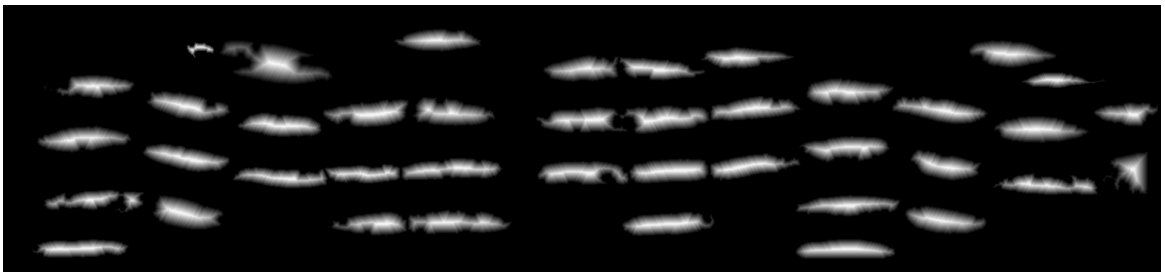
3.5 Experiments

3.5.1 Datasets

Tomographic Images



(a) Image



(b) Ground Truth

Figure 3.4: A slice from the 3D tomography and its corresponding target segmentation.

We test the equivariance of the SEU-Net on a dataset composed of 2D slices Figure 3.4a) from a tomographic scan of a composite fabric, used in airplane manufacture. The ground truth is made of the morphological distance functions of each axial strand (Blusseau et al., 2022), resulting in images like Figure 3.4b. The training, validation and test sets are composed respectively of 1501, 264 and 1178 images from both warp and weft directions.

Oxford IIIT Pet

The Oxford-IIIT Pet [1] dataset (Parkhi et al., 2012) consists of pictures containing cats and dogs. The relevant labeling for this paper, the trimaps, is the segmentation of the images into three classes: the animal, the background and the boundaries of the animal.

[1] `https://www.robots.ox.ac.uk/~vgg/data/pets/`, CC BY-SA 4.0 license

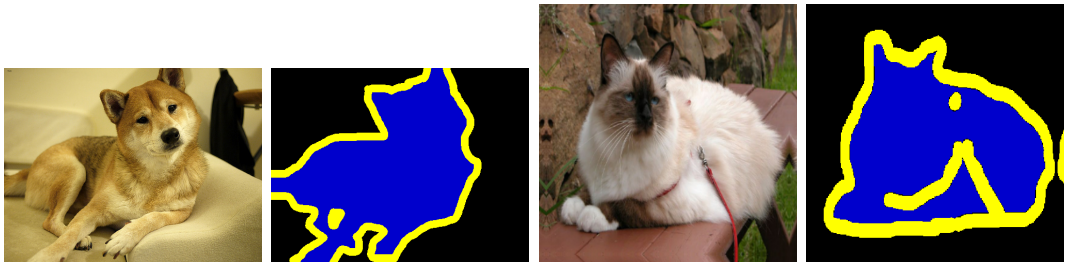


Figure 3.5: Some images and ground truths from the Oxford Pet IIIT dataset.

For some examples from this dataset, look at Figure 3.5. The dataset was loaded from the TensorFlow package[2], where it is divided into 3680 training samples and 3669 test samples. We use 20% of the training samples for validation. During training and testing images are resized to 224×224 pixels. We define multiple test sets by re-scaling the original test set by $s \in \{2^{\frac{i}{2}} | i \in \{-4, -3, \dots, 4\}\}$. We used bilinear interpolation to up-scale images.

DIC-HeLa Cells

Another dataset to test the segmentation networks is a cell segmentation dataset, namely the DIC-C2DH-HeLa dataset (Ulman et al., 2017) of HeLa cells on a flat glass recorded by differential interference contrast (DIC). We used 83 images for train/validation and 83 for testing. Columns (a) and (b) in Figure 3.12 show examples images images from the test set and its label in different scales.

3.5.2 Results

Tomographic Images

During training, we extract random 256×256 patches of each image as input. As before, we resize test images creating a different test set for each $s \in \{2^{\frac{i}{2}} | i \in \{-4, -3, \dots, 3\}\}$. Both U-Net and SEU-Net have height four and four filters in their first layer. The lifting contains six scales and filters have scale dimension equal to one. We use the Mean Squared Error (MSE) to measure the performance and as the loss function and train it with the Adam optimizer.

With the aim of rendering the network more robust to change in scale, we propose a *scale dropout* applied before projection. We show the results with dropout only for the tomographic images experiment as it did not improve the results in the Pets dataset. For every scale coordinate $s \in \mathcal{S}_\gamma$, the dropout layer sets to 0 all pixels $f(s, x), x \in \mathbb{Z}^2$ with some fixed probability p. We assume that the feature map f in this case is the output of a ReLU activation, so its minimum value is 0. Effectively this implies that the max-projection will ignore that scale. In all the equivariant models in the tomographic images we apply a dropout with probability $p = 0.5$.

The overall results are shown in Figure 3.6. Even though it has a drop in performance with respect to U-Net at the training scale, the equivariant model with strides as pooling significantly improves the generalization to other scales. Quadratic pooling also shows an

[2] `https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet`

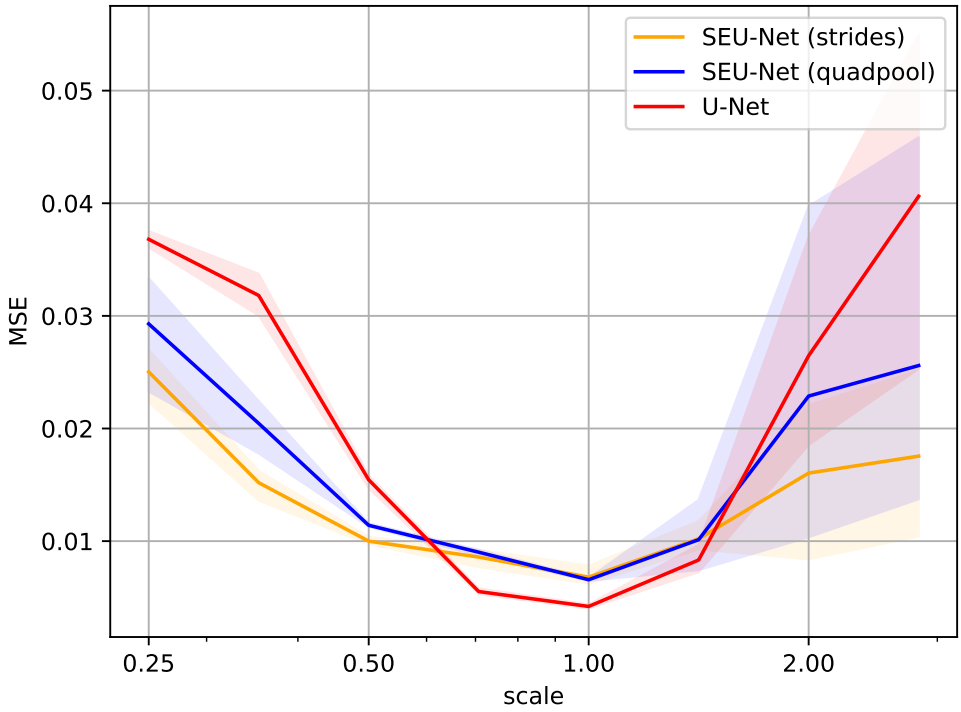


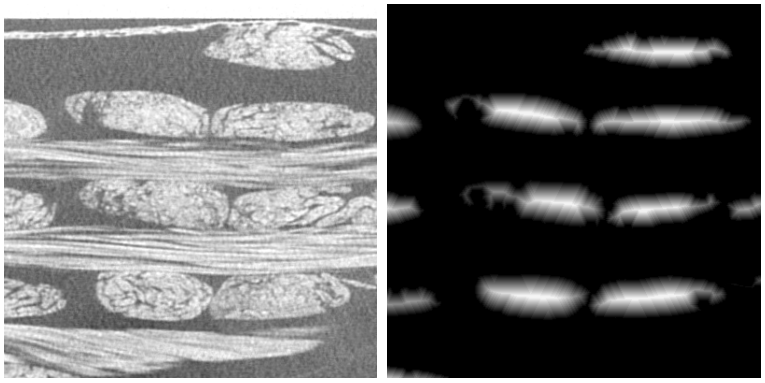
Figure 3.6: Overall results of the tomographic images dataset. The curves show the average MSE from three runs and the shaded regions show are between the minimum and maximum values attained. We omit the max-pooling for better visibility of the other methods. It can be seen in Appendix D. We see that in the cases of quadpool strides, equivariance brings a significant improvement.

improvement compared to the U-Net, but not enough to be better than strided scale-cross-correlation pooling.

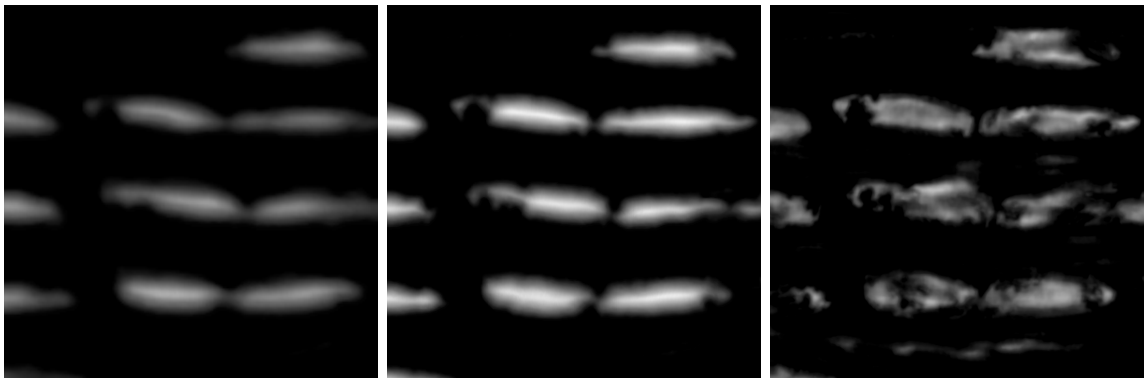
Oxford IIIT Pet

The compared U-Net and SEU-Net are both of height four and contain eight filters in the first layer. Both the U-Net and SEU-Net have height four and contain sixteen filters in the first layer and use 3×3 filters. The SEU-Net truncates at four scales, and filters have depth one in the scales dimension (their values is different from zero in one scale value). The networks are trained using the Adam (Kingma and Ba, 2014) optimizer with categorical cross-entropy loss. Training the U-Net, SResNet and SEU-Net takes approximately 24, 73, and 97 seconds per epoch respectively, on a Tesla P100-SXM2-16Gb GPU.

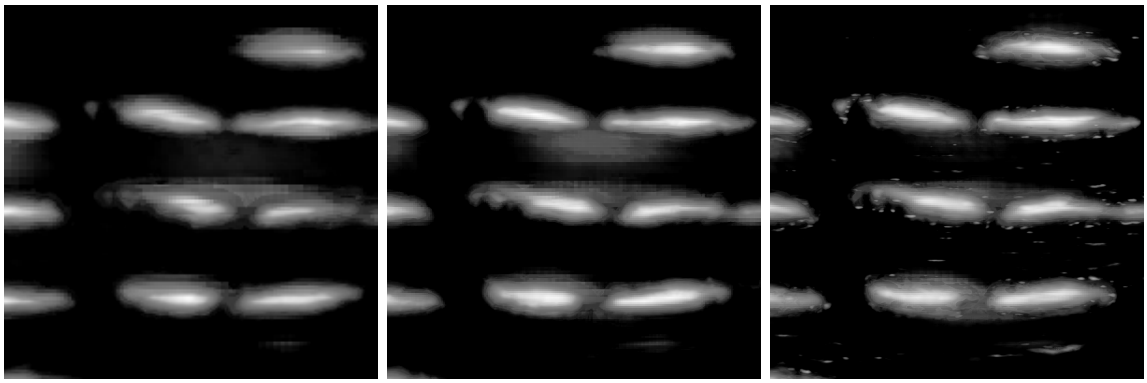
Comparison with data augmentation. We also performed scale jittering in the U-Net to compare the effect of the equivariant network with the effect of data augmentation. Scale jittering is performed by rescaling the image by a randomly chosen scale α and either random cropping or padding to the original image. We trained a U-Net with scale jittering in the



(a) Image and ground truth.



(b) U-Net predictions.

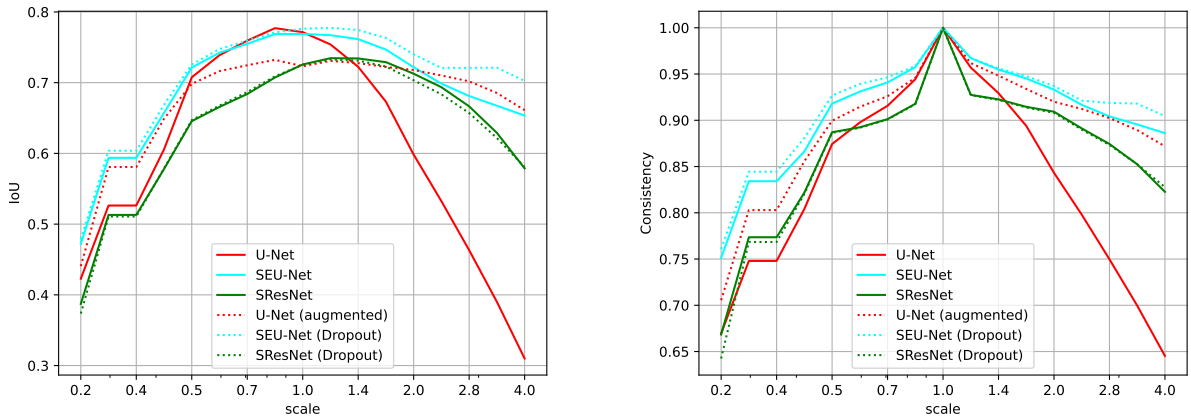


(c) SEU-Net (strides) predictions.

Figure 3.7: Predictions computed from a square crop of a test image in tomographic images dataset. Computed from a small region of a test image 3.7a. The first, second and third columns correspond to scales $0.5, 1$ and 2, respectively.

interval $[\frac{1}{4}, 4]$, equal to the interval of test scales.

Training details. All models, except for the augmented U-Net are trained for 300 epochs. The augmented U-Net is trained for four times as many epochs. To train all models we apply data augmentation consisting of, rotations by a uniformly sampled angles in $[-10°, 10°]$, linear contrast changes by values in the range $[0.9, 1.1]$, random horizontal flipping and random cropping to size 112×112. Learning rate starts at 10^{-3} an is reduced by 10 when the validation



(a) IoU (b) Consistency

Figure 3.8: Overall results in terms of IoU and Consistency for each scale of the Pet dataset.

loss does not improve for 30 epochs. We use a batch size of 8.

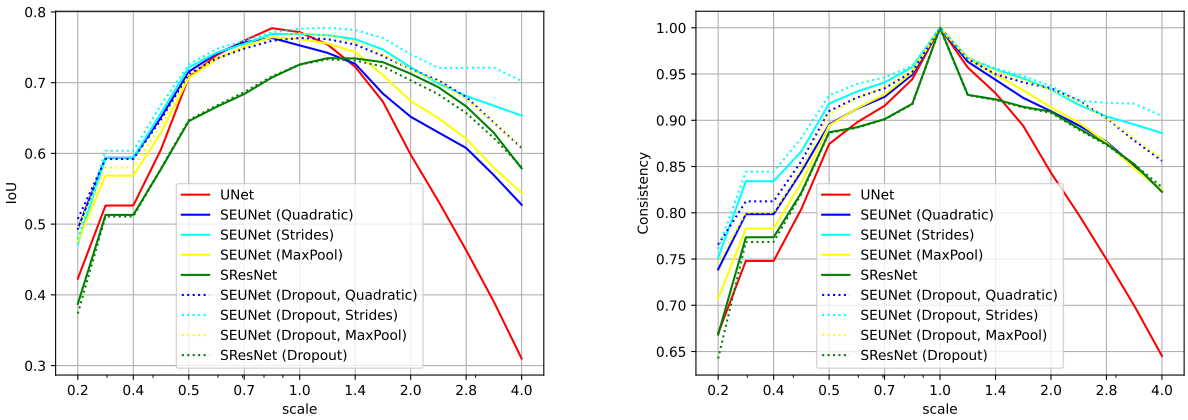
Results. The overall results in terms of the IoU are shown in Figure 3.8. Firstly we notice that the SEU-Net increases performance compared to both SResNet and U-Net. SResNet, however, does not consistently generalize better than the U-Net. Dropout improves the quality of SEU-Net, particularly for more extreme scales, indeed, for larger scales the augmented U-Net has a better IoU than the SEU-Net without scale dropout, but not than the one with scale dropout. The augmented U-Net loses performance scale 1, it would probably need to be larger to retain the same performance. We show examples of the predictions of the U-Net and SEU-Net in Figure 3.11. We also did a comparison of the models with different kinds of scale-space poolings in Figure 3.9. The results of poolings other than strided cross-correlations is generally poor compared with strides.

Cell Segmentation

We also evaluate the models in a medical image segmentation dataset, namely the DIC-C2DH-HeLa dataset (Ulman et al., 2017) of HeLa cells on a flat glass recorded by differential interference contrast (DIC). We used 83 images for train/validation and 83 for testing. Figure 3.12 (a) and (b) shows an example from the test set with its labels at different scales.

Models are trained with the AdamW optimizer (Loshchilov and Hutter, 2017). Like in the previous experiment, we first train the models in the training set with the original scale distrubtion and test in the test set re-scaled by different values.

Training Details. We also perform scale jittering, but now for both U-Net and SEU-Net. For U-Net we trained models with scale jittering with ranges 4 (α is chosen each step from the interval $[\frac{1}{4}, 4]$) and 1.5 (α is chosen from the interval $[\frac{2}{3}, \frac{3}{2}]$) and for SEU-Net we used only the range 1.5 jittering. All models, except for the U-Net with jittering 4 are trained for 200 epochs. The augmented U-Net with jittering 4 is trained for four times as many epochs. To train all models we apply data augmentation consisting of, rotations by a uniformly sampled angles in $[-10°, 10°]$, linear contrast changes by values in the range $[0.9, 1.1]$, random horizontal



(a) IoU (b) Consistency

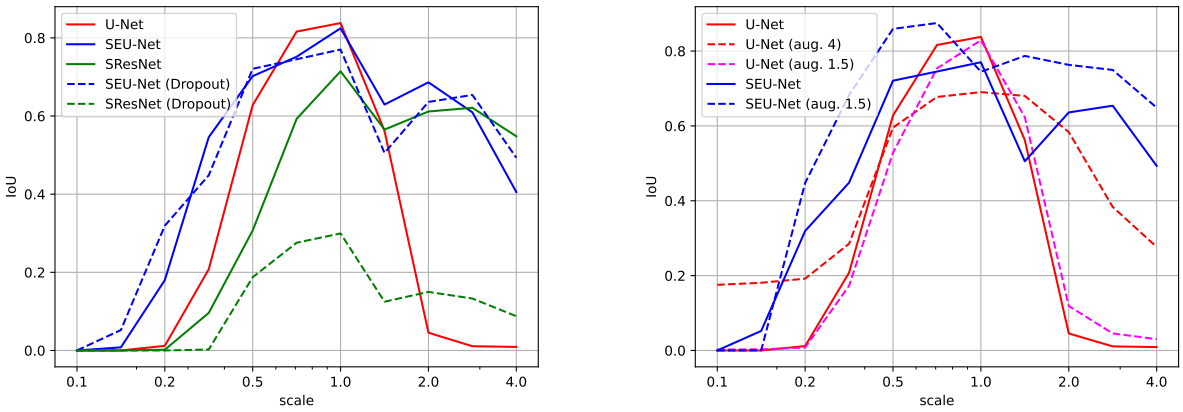
Figure 3.9: Overall results in terms of IoU and Consistency for each scale of the Pet dataset. This time comparing different pooling functions and using the non-augmented U-Net's performance for reference.

and vertical flipping and elastic transformations. Learning rate starts at 10^{-3}, weight decay starts at 10^{-4} and both are reduced by exponential decay such that they are divided by 10 every 100 epochs (the decay stops at epoch 300 for the U-Net with size 4 jittering). We use a batch size of 1.

Results. Figure 3.10 (a) shows the IoU of different models on the re-scaled test sets. Figure 3.12 shows some segmentation examples. Again, the SEU-Net outperforms the U-Net. The poor results of the SResNet for smaller scales is possibly due to the cell images containing more high-frequency information, compared to the pets images. In contrast to the previous experiment, dropout did not seem to significantly increase performance of the SEU-Net, neither in the train scale nor the test scales. Moreover the SResNet results were greatly decreased due to dropout. This is likely a result of the agumented dataset being more difficult to segment than the original and not being representative of the dataset at base scale. The gain in generalization is only better than the SEU-Net for the smallest scales. The jittering with range 1.5 does not have a very noticeable effect. On the other hand the SEU-Net with 1.5 jittering has a noticeable gain in generalization to larger scales.

3.5.3 Discussion

In all three experiments the SEU-Net is shown to have a greater scale generalization capability than the U-Net. Furthermore, when looking at the comparisons in Figures 3.8 and 3.10(b) we see that the U-Net with augmentation attains a greater performance than the U-Net but at the cost of having a worse performance at the scale 1. We can conjecture that in order for the U-Net to perform better at both all scales it would be necessary to increase its size, i.e. the number of parameters, while simultaneously using jittering with range 4. The SEU-Net, on the other hand, has approximately the same number of parameters as the U-Net and does not need to have its size increased to perform well in multiple scales without losing its performance in



(a) IoU per scale.

(b) Augmentation comparison.

Figure 3.10: IoUs of the cell segmentation experiment with comparisons with U-Net, SResNet and data augmentation. U-Net (aug. 4) refers to the U-Net trained with scale jittering with range 4 and U-Net (aug. 1.5) refers to the U-Net trained with jittering with range 1.5. The same for SEU-Net (aug. 1.5).

the scale 1. Besides, both scale dropout and data augmentation by small scale jitterings were shown to increase the generalization and performance of the SEU-Net at virtually no additional cost. The SResNet does not seem to have the same nice properties, and the main difference between the SEU-Net and the SResNet is the presence of upsampling inside the equivariant pipeline.

3.6 Conclusions

In this chapter the non-equivariance to scale of the U-Net was pointed out. Moreover we point some possible problems that could arise from constructing equivariant networks for segmentation without closely examining the downsampling and upsampling layers. For example, the SResNet is a network build on scale-cross-correlations, but instead of performing gradual upsamplings it performs an upsampling by a large factor at the last layer.

Here the SEU-Net was proposed in order to render the U-Net invariant. In order to define the SEU-Net, each building block of the U-Net was studied and a scale-equivariant counterpart scale-equivariant is proposed. The relevant building blocks are: convolutions, pixelwise activations, batch-normalization, subsampling and upsampling. In the SEU-Net, convolutions are substituted by scale-cross-correlations. Pixelwise activations and batch normalization are already equivariant by virtue of being applied in a pixelwise manner. Subsampling and upsampling needed further study to assert their scale-equivariance.

The naive extension of the max-pooling to the signals on the semigroup was shown not to be scale-equivariant, but strided scale-cross-correlations are indeed scale-equivariant, moreover, a class of approximately scale-equivariant operators, which are discretizations of scale-equivariant operators, was proposed as pooling functions.

Upsampling was shown to not be scale-equivariant in general, but sufficient conditions for it to be scale-equivariant were shown. In that way, images were the objects of interest are sufficiently big for a given subsampling can garantee the equivariance of the upsampling.

In the experimental section the SEU-Net was tested in three tasks, namely the segmentation of tissue strands, the segmentation of natural images of the Oxford-IIIT Pet dataset and the segmentation of cells from the DIC-HeLa dataset. The SEU-Net was tested using multiples sets of parameters depending on the pooling function and the use of dropout. Moreover it was compared with two other models, the SResNet and the U-Net. Experimental results show that the SEU-Net can greatly improve the generalization to new scales and even the performance in the training scale. Moreover, the results lead us to conjecture that the U-Net with scale jittering would need to have more parameters to have a good performance in all the range of scales, while the SEU-Net achieves good results without increasing its size. The results suggest that implementing that the improvement comes not only from the scale-equivariant cross-correlations, but also from the SEU-Net global architecture and applying the pooling and upsampling operators inside the equivariant pipeline. The proposed scale dropout was also shown to have the potential to increase scale-equivariant models' performance. In future works it would be interesting to study an equivariant regularization term such as in Sosnovik et al. (2021) in addition to the scale-dropout.

3.7 Résumé en Français

Dans ce chapitre l'absence d'équivariance par rapport au changement d'échelle du réseau de segmentation U-Net a été soulignée. Par ailleurs, nous avons montré des problèmes qui peuvent survenir lorsqu'on propose un réseau équivariant par changement d'échelle pour la segmentation sans examiner les opérations de sous-échantillonnage et suréchantillonnage. Le SResNet (Worrall and Welling, 2019), par exemple, est un réseau basé sur la couche de corrélation-croisée mais au lieu de faire des suréchantillonnages graduels, le suréchantillonnage est fait une seule fois avec un facteur d'échelle très élevé dans la dernière couche.

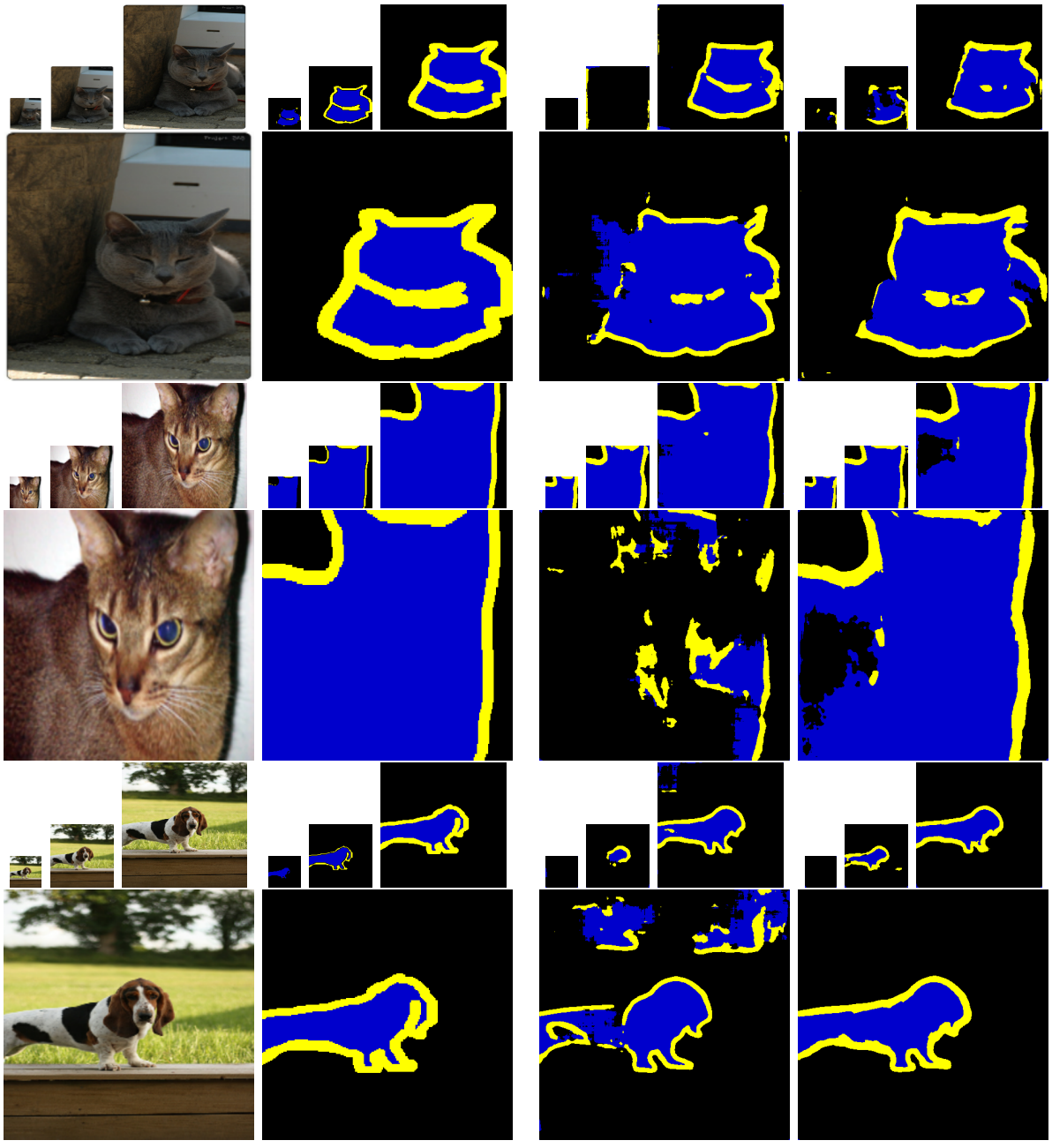
Ici nous avons proposé le U-Net Équivariant par Changement d'Échelle (SEU-Net). Pour obtenir le SEU-Net, chaque bloc de U-Net est étudié et une alternative équivariante est proposée. Les blocs pertinents sont : convolutions, activations ponctuelles, normalisation par lot, sous-échantillonnage et suréchantillonnage. Dans le SEU-Net, les convolutions sont substituées par des corrélations-croisées d'échelle. Les activations et les normalisations par lot sont déjà équivariantes en vertu d'être appliquées ponctuellement. Le sous-échantillonnage et le suréchantillonnage ont eu besoin d'une étude plus approfondie pour pouvoir évaluer leur équivariance.

L'extension naïve du max-pooling aux signaux sur un semi-groupe n'est pas équivariante, mais la corrélation-croisée d'échelle avec strides (i.e. en appliquant les filtres un pixel sûr deux) l'est. Par ailleurs, une classe d'opérateurs approximativement équivariants par changement d'échelle a été proposée pour le sous-échantillonnage, donnée par la discrétisation de certains opérateurs équivariants par changement d'échelle.

Nous avons montré que le suréchantillonnage n'est pas équivariant par changement d'échelle en général, mais il existe des conditions suffisantes pour qu'il soit équivariant. Nous pouvons garantir l'équivariance si les objets d'intérêt son suffisamment grands pour que le sous-

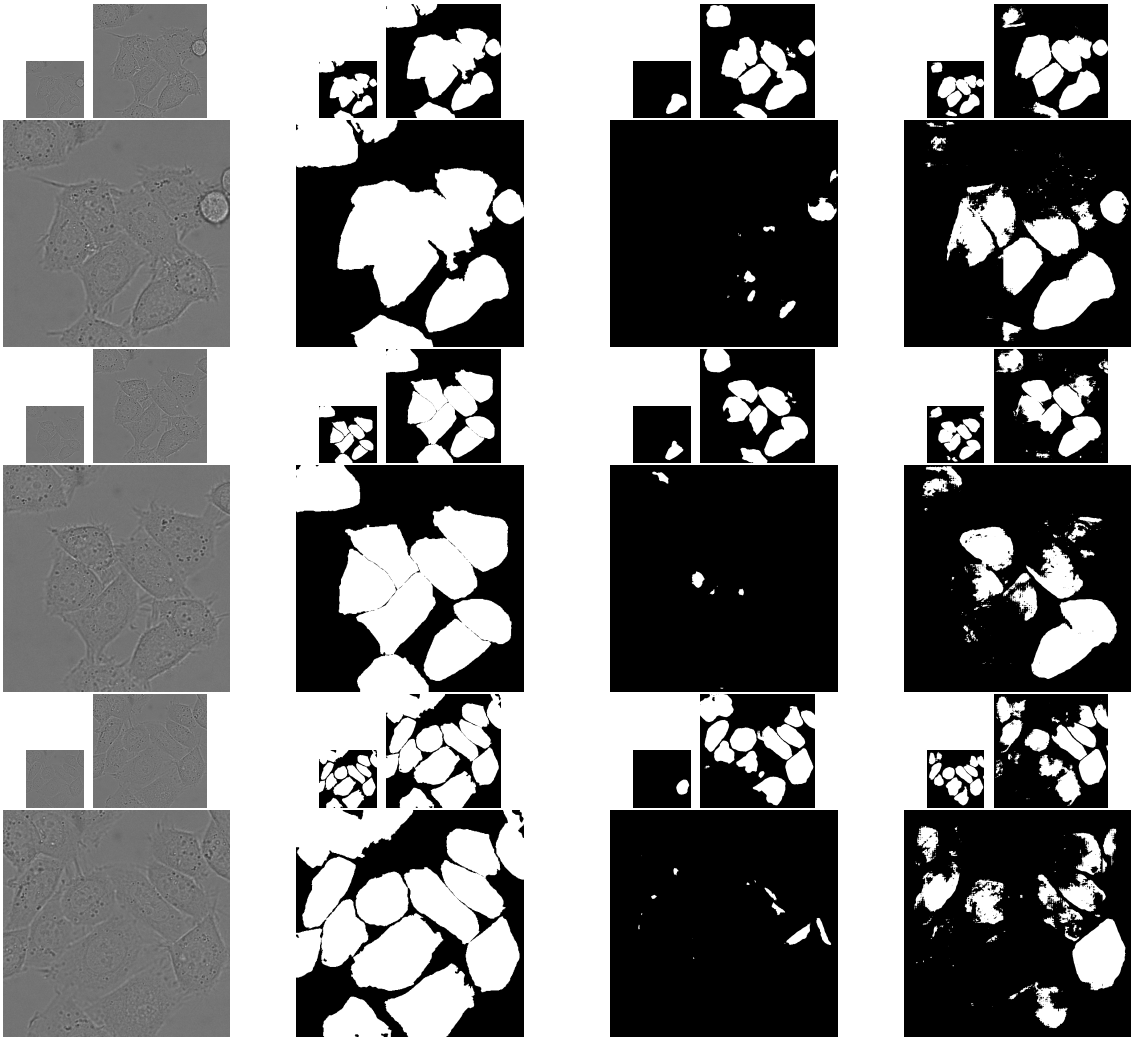
échantillonnage ne détruise pas d'information.

Dans la section expérimentale le SEU-Net a été testé sur trois tâches: la segmentation de torons de tissu, la segmentation d'images naturelles dans la base de données Oxford-IIIT Pet (Parkhi et al., 2012) et la segmentation de cellules de la base de données DIC-Hela (Ulman et al., 2017). Le SEU-Net a été testé en utilisant plusieurs configurations d'hyperparamètres des fonctions de sous-échantillonnage et probabilité de *dropout* d'échelle. Il a été comparé avec deux autres modèles : U-Net et le SResNet. Les résultats expérimentaux montrent que le SEU-Net peut augmenter considérablement la généralisation à des échelles qui ne sont pas représentées dans les données d'entraînement (et même la performance à l'échelle d'apprentissage) par rapport aux résultats donnés par les deux autres modèles. Par ailleurs, les résultats nous amènent à conjecturer que U-Net entraîné avec augmentation par changement d'échelle a besoin d'un nombre plus élevé de paramètres pour obtenir une bonne performance sur différentes échelles, alors que le SEU-Net obtient des bons résultats sur différentes échelles sans augmenter le nombre de paramètres. Les résultats suggèrent que l'amélioration ne vient pas seulement des corrélations-croisées d'échelle, mais aussi de l'architecture globale du SEU-Net qui applique le sous-échantillonnage et le suréchantillonnage avant de la projection. Dans des travaux futurs, il serait intéressant d'étudier une méthode de régularisation équivariante, comme dans Sosnovik et al. (2019), en plus du dropout d'échelle.



(a) Image (b) Ground Truth (c) U-Net (d) SEU-Net

Figure 3.11: Sample test images at different scales and ground truth from the Oxford-IIIT Pet dataset, along with the U-Net and SEU-Net predictions. The scales present are $0.25, 0.5, 1$ and 2 times the training scale.



(a) Image (b) Ground Truth (c) U-Net (d) SEU-Net

Figure 3.12: Predictions from DIC-HeLa at different scales, namely scales 0.5, 1 and 2.

Chapter 4

Differential Invariants Blocks

4.1 Introduction

Transformation equivariance can be introduced to a neural network to make use of the symmetry intrinsic to data features in many deep learning tasks. Convolutional Neural Networks (CNN) for example, reduce the number of parameters and improve the results in computer vision compared to densely connected neural networks thanks to benefiting from the translation symmetry present in computer vision tasks. Since their introduction as a general framework (Cohen and Welling, 2016a), group equivariant networks have been extended to work with other transformation groups, most notably translations combined with rotations (Cohen and Welling, 2016a; Worrall et al., 2017; Shen et al., 2020; Marcos et al., 2017; Weiler et al., 2018b; Weiler and Cesa, 2019) or scalings (Lindeberg, 2022).

Many approaches to group equivariant neural networks involve first lifting the data to a higher dimensional space obtained by sampling elements of the group and performing group convolutions there (Cohen and Welling, 2016a; Kondor and Trivedi, 2018). This effectively introduces equivariance to the network, but it has the downside of increasing the size and computational cost according to the group dimension, as well as requiring the group transformations to be discretized. Steerable CNNs (Cohen and Welling, 2016b; Worrall et al., 2017) constrain filters in a way that the convolution is equivariant, so it does not require to use signals on the group, but they support less parameters than a CNN with the same number of feature maps.

We propose to use adaptive filters that are rotated to match the local geometry of the image. Since they match the local geometry a rotation of the image implies a rotation of the filters, thus the application of the filters will define an equivariant operator, as illustrated in Figure 4.1. The method of *moving frames* (Fels and Olver, 1998; Olver, 2007) formalizes this concept and gives us tools to define such steerable kernels.

The method of moving frames was introduced by (Cartan, 1935) and can be applied to derive differential invariants on smooth manifolds. Given a manifold $\mathcal{M}$ and a Lie group $\mathcal{G}$ acting on it, the method is based on finding, for each point, a transformation from the group, in a way that elements in the same group orbit (i.e. elements related by a group transformation) are mapped to the same point. Given an operator F on $\mathcal{M}$, by first transforming elements according to the moving frame it is possible to obtain an invariant operator $\imath[F]$ corresponding to F. If F happens to be a differential operator, its invariantization is a so-called differential invariant.

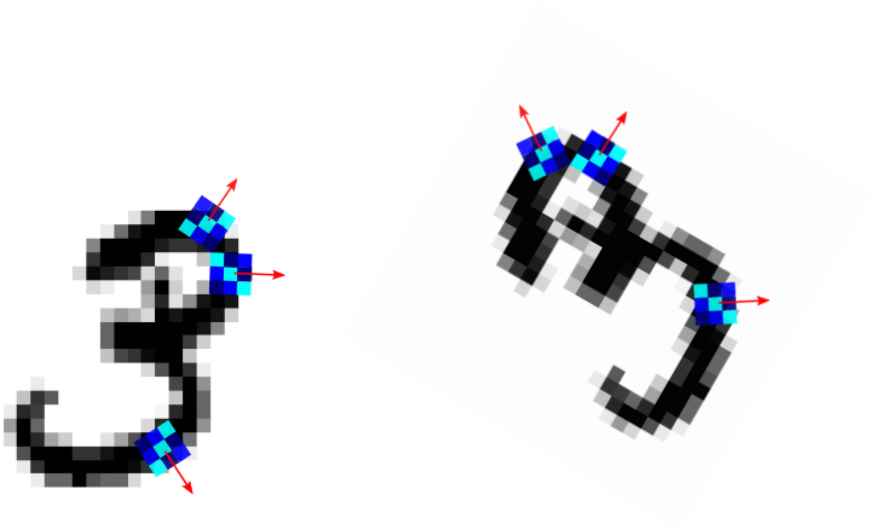


Figure 4.1: Example of the application of a kernel (shown as blue squares) where its orientation (shown as red arrows) depends on the local image geometry. In the previous example and in the method we are going to propose for rotation equivariant operators on images, an orientation for each point can be obtained as the direction of the gradient, for example.

In this chapter we are mainly concerned with applying the method of moving frames to derive networks equivariant to the orientation-preserving isometries of the plane and the space. We apply these ideas to define SE(2)-equivariant networks for images.

The rest of the chapter is organized as follows: In Section 4.2 we discuss related works in the literature. Section 4.3 reviews the method of moving frames and the concepts that will be necessary for the derivation of the SE(2)-equivariant networks. Section 4.4 computes differential invariants of arbitrary order for SE(2) acting on images. In Section 4.5 introduces the main contribution of the chapter, the networks based on Gaussian derivatives, differential invariants and the method of moving frames in SE(2). In Section 4.6 we perform experiments to validate the use of Gaussian derivatives and investigate pooling method, and we subsequently test the method in the MNIST-Rot dataset, obtaining competitive results. Section 4.7 concludes the chapter.

4.2 Related Work

Group equivariant networks have been obtained in many ways, must many involve sampling group elements, lifting elements to a higher dimensional space (Cohen and Welling, 2016a; Shen et al., 2020) using discretized group elements. The disadvantage of that approach is that computational cost is increased according to the number of samples. Our proposed approach does not require lifting the data to another space and equivariant operations are performed on the same space as the input data, instead. Moreover, we achieve equivariance to continuous rotations. In the literature, equivariance to continuous rotations was achieved using steerable filters (Worrall et al., 2017; Weiler et al., 2018b) but in contrast to our method, it supports less parameters than a CNN with the same quantity of filters.

In computer vision, differential invariants form a basis for many of the filtering schemes, specially in the theory of scale-spaces (Weickert, 1998; Heijmans, 2002). They have also been applied to the design of feature descriptors (Tuznik et al., 2018) and for the automatic learning of PDEs for image restoration (Liu et al., 2010). Moreover, convolutional neural networks have been compared with numerical schemes for solving PDEs (Ruthotto and Haber, 2020) and this idea was applied to the construction of equivariant neural networks (Shen et al., 2020) albeit not with the use of differential invariants.

4.3 Moving Frames and Differential Invariants

4.3.1 The Method of Moving Frames

The method of moving frames is a powerful tool that uses the differential properties of Lie groups and manifolds to derive operators invariants under the group action. In the following $\mathcal{G}$ denotes a Lie group, $\mathcal{M}$ a manifold and we denote the action of $\mathcal{G}$ on $\mathcal{M}$ as $\pi_g[z] = g \cdot z$, for $g \in \mathcal{G}$ and $z \in \mathcal{M}$. A moving frame is a $\mathcal{G}$-equivariant operator associating to each point in $\mathcal{M}$ a transformation in $\mathcal{G}$, as formalized by the following definition

In this chapter wee shall implicitly consider *local Lie group actions* when we consider group actions. A local Lie group action of $\mathcal{G}$ on $\mathcal{M}$ is a smooth map $\pi : U \to \mathcal{M}$ where $U \subseteq \mathcal{G} \times \mathcal{M}$ is an open subset of $\mathcal{G} \times \mathcal{M}$ containing $\{e\} \times \mathcal{M}$, it satisfies one of the following

- $\forall g, h \in \mathcal{G}$, $x \in \mathcal{M}$ such that $(g \cdot h, x) \in U$, $\pi[g, \pi[h, x]] = \pi[g \cdot h, x]$ (left action);
- $\forall g, h \in \mathcal{G}$, $x \in \mathcal{M}$ such that $(h \cdot g, x) \in U$, $\pi[h, \pi[g, x]] = \pi[g \cdot h, x]$ (right action).

Similarly to group action we use the notation $\pi_g(x) \coloneqq \pi(g, x)$ for all $(g, x) \in U$.

Definition 10 (Moving frame Fels and Olver (1998))**.** *A* moving frame *is a $\mathcal{G}$-equivariant map $\rho : \mathcal{M} \to \mathcal{G}$, which specifically verifies $\rho(g \cdot z) = \rho(z) \cdot g^{-1}$ for all $g \in \mathcal{G}$, $z \in \mathcal{M}$.*

From the above definition we can deduce

$$\rho(g \cdot z) \cdot g \cdot z = \rho(z) \cdot g^{-1} \cdot g \cdot z = \rho(z) \cdot z, \tag{4.1}$$

which means that the mapping $z \mapsto \rho(z) \cdot z$ is constant over the orbits

$$\mathcal{O}_z = \{g \cdot z | g \in \mathcal{G}\}. \tag{4.2}$$

From a moving frame one can obtain an invariant operator from an arbitrary differentiable operator. We use $\imath$ to denote the *invariantization* of an operator between manifolds $F : \mathcal{M} \to \mathcal{N}$ defined as

$$\imath[F](z) \coloneqq F(\rho(z) \cdot z). \tag{4.3}$$

We have that $\imath[F]$ is invariant with respect to $\mathcal{G}$, *i.e.* $\imath[F](g \cdot z) = \imath[F](z)$ which can be derived directly from (4.1)

$$\imath[F](g \cdot z) = F(\rho(g \cdot z) \cdot g \cdot z) = F(\rho(z) \cdot z) = \imath[F](z). \tag{4.4}$$

To the end of reviewing properties of moving frames we recall some properties of group actions that will be useful for the discussion.

Definition 11 (Free and regular group action)**.** *Let π_g be the action of a Lie Group $\mathcal{G}$ on a manifold $\mathcal{M}$. We say that π_g is*

- *semi-regular if its orbits all have the same dimension;*
- *regular if it is semi-regular and every point $z \in \mathcal{M}$ has arbitrarily small neighborhoods U such that the orbits intersect U in pathwise connected subsets;*
- *free if the only group element that fixes all points $z \in \mathcal{M}$ is the identity, i.e. $\forall g \in \mathcal{G}$, $\forall z \in \mathcal{M}$, if $g \cdot z = z$ then $g = e$ where e is the identity element.*
- *locally free if there exists an open neighborhood U of the identity such that for all $\forall g \in U$ $\forall z \in \mathcal{M}$, $g \cdot z = z \implies g = e$.*

Theorem 1. *Fels and Olver (1998) A moving frame ρ exists in a neighborhood $U \subseteq \mathcal{M}$ containing $z \in \mathcal{M}$ if and only if $\mathcal{G}$ acts locally freely and regularly in U.*

When $\mathcal{G}$ acts freely and semi-regularly near $z \in \mathcal{M}$, a moving frame can be obtained by first defining a *cross-section*.

Definition 12 (Cross section)**.** *Let $\mathcal{G}$ act semi-regularly in $\mathcal{M}$ and have s-dimensional orbits. An embedded submanifold $\mathcal{K} \subseteq \mathcal{M}$ of dimension $m - s$ is a cross-section to the group orbits if it intersects each orbit transversally i.e. the tangent spaces of the intersecting orbit and the tangent space of the cross-section generate the tangent space of the intersection point in $\mathcal{M}$. If this intersection is unique we call $\mathcal{K}$ a regular cross-section.*

Now let $\mathcal{K}$ be a regular cross-section to the orbits on an open set $U \subseteq \mathcal{M}$ containing $z \in \mathcal{M}$ and assume that $\mathcal{G}$ acts locally freely and regularly on U. Then, because $\mathcal{K}$ is regular on U the orbit $\mathcal{O}_z$ intersects $\mathcal{K}$ in a single point, and because π_g is free, there exists a single $g_z \in \mathcal{G}$ such that $g_z \cdot z \in \mathcal{K}$. Now define $\rho : \mathcal{M} \to \mathcal{G}$ as the function mapping each z to the g_z the group element that takes z to the cross section $\rho(z) \cdot z = g \cdot z \in \mathcal{K}$. For some $h \in \mathcal{G}$, we have both $\rho(z) \cdot z \in \mathcal{K}$ and $\rho(h \cdot z) \cdot h \cdot z \in \mathcal{K}$, moreover, $h \cdot z \in \mathcal{O}_z$, therefore, because the intersection is unique, we get $\rho(z) \cdot z = \rho(h \cdot z) \cdot h \cdot z$. As shown in the equation below, this implies that ρ is a moving frame, by Definition 10. Indeed,

$$\begin{aligned} & \rho(z) \cdot z = \rho(h \cdot z) \cdot h \cdot z \\ \iff \; & h^{-1} \cdot \rho(h \cdot z)^{-1} \rho(z) \cdot z = z \\ \iff \; & h^{-1} \cdot \rho(h \cdot z)^{-1} = \rho(z)^{-1} \\ \iff \; & \rho(h \cdot z)^{-1} = h \cdot \rho(z)^{-1} \\ \iff \; & \rho(h \cdot z) = \rho(z) \cdot h^{-1} \end{aligned} \tag{4.5}$$

where the equivalence from the second line to the third is due to freeness.

In order to simplify the calculations of the moving frame, $\mathcal{K}$ is usually a *coordinate cross-section* i.e. $\mathcal{K}$ is a set defined by points $z \in \mathcal{M}$ where r coordinates are set to constants, or, formally the cross-section a set of the form

$$\mathcal{K} = \{z \in \mathcal{M} | z_{k_1} = c_1, z_{k_2} = c_2, \ldots, z_{k_r} = c_r\}, \tag{4.6}$$

where $k_i \neq k_j$ if $i \neq j$, $\{k_1, \ldots, k_r\} \subseteq \{1, \ldots, m\}$ and $c_i \in \mathbb{R}$ for all $i \in \{1, \cdots r\}$. In the case where $\mathcal{K}$ is a coordinate cross-section, then the moving frame can be obtained by finding, for each $z \in \mathcal{M}$, a $g \in \mathcal{G}$ that solves the system of equations

$$\begin{aligned} w_1(g, z) &= c_1 \\ w_2(g, z) &= c_2 \\ &\vdots \\ w_r(g, z) &= c_r \end{aligned} \tag{4.7}$$

where $w_i(g, z) = \text{proj}_{k_i}(g \cdot z)$, $i = 1, \ldots, r$, $g \in \mathcal{G}$ $z \in \mathcal{M}$.

Example 1. *As an example of a moving frame, consider the manifold $\mathcal{M} = \mathbb{R}^2 \setminus \{(0, 0)\}$ with the group of rotations SO(2) acting on it by*

$$(x, y)^T \mapsto R_\theta (x, y)^T = (x \cos\theta - y \sin\theta, x \sin\theta + y \cos\theta)^T \tag{4.8}$$

where $R_\theta \in SO(2)$ is the rotation matrix by an angle $\theta \in [0, 2\pi)$. The group action is free because the only rotation having a fixed point in $\mathbb{R}^2 \setminus \{(0, 0)\}$ is the identity $I = R_0$.

Let us consider the orbits of $\mathcal{G}$ in $\mathcal{M}$. Given a point $z \in \mathcal{M}$, its orbit is given by $\mathcal{O}_z = \{R_\theta \cdot z | \theta \in [0, 2\pi)\}$ which are all the rotations of z around the origin $(0, 0)$. The orbits are, therefore, all the circles centered at the origin. Moreover, since they are all circles they all have dimension $\dim \mathcal{O}_z = 1$ for all $z \in \mathcal{M}$. In Figure 4.9 in grey we have examples of the group orbits.

Consider the set $\mathcal{K} = \{(x, 0) \in \mathcal{M} | x > 0\}$, illustrated in Figure 4.2 in blue, given by the horizontal half-line from the origin to the positive x-axis. $\mathcal{K}$ also has dimension 1 and when a circle centered in the origin passes through $\mathcal{K}$, its tangent is perpendicular to $\mathcal{K}$, meaning that their tangent spaces are perpendicular at the intersection and generate a vector space of dimension two $V \cong \mathbb{R}^2$ i.e. the tangent space of $\mathcal{M}$. Therefore, $\mathcal{K}$ is a cross-section. Moreover, because for each positive real $r > 0$ there is only on value with that distance from the origin in $\mathcal{K}$. Moreover, since for each positive real number $r > 0$ there is only one point in $\mathcal{K}$ at distance r from the origin, then each circle intersects $\mathcal{K}$ once, meaning that $\mathcal{K}$ is a regular cross-section. Given any $z = (x, y) \in \mathcal{M}$, the rotation

$$R_z = \frac{1}{\sqrt{x^2 + y^2}} \begin{bmatrix} x & y \\ -y & x \end{bmatrix} \in SO(2) \tag{4.9}$$

is the only one that brings z to the cross-section, i.e. $R_z \cdot z \in \mathcal{K}$. Therefore, $\rho : z \in \mathcal{M} \mapsto R_z$ is a moving frame.

Equivariant Function Operators from Invariant Manifold Operators

Now before we move on to obtain differential invariant operators, let us explain the interest of having invariant operators when our goal is to obtain equivariant neural network layers.

We assume we are in the context where we compute the invariantization of operators $F : \mathcal{M}_1 \to \mathcal{M}_2$, with $\mathcal{M}_1 = X \times Y$ and $\mathcal{M}_2 = Z$, and the action $\mathcal{G}$ can be decomposed written as

$$g \cdot (\mathbf{x}, u) = (g \cdot \mathbf{x}, u), \tag{4.10}$$

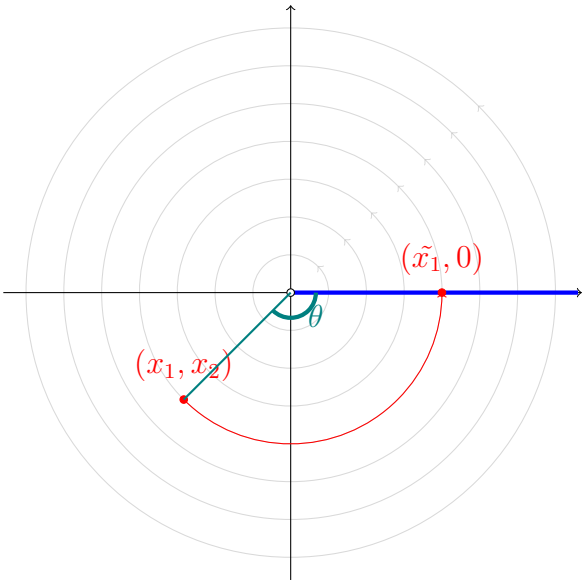


Figure 4.2: Group orbits (grey) and cross-section $\mathcal{K}$ (blue) in $\mathcal{M} = \mathbb{R}^2 \setminus \{(0,0)\}$ under the action of SO(2).

for all $g \in \mathcal{G}$, $(\mathbf{x}, u) \in X \times Y$.

We define the actions on functions $f : X \to Y$ like in Chapter 1 (and analogously for functions $f' : X \to Z$) as

$$(g \cdot f)(\mathbf{x}) = f(g^{-1} \cdot \mathbf{x}). \tag{4.11}$$

Now we want to show that for every $\mathcal{G}$-invariant smooth operator $\psi : X \times Y \to Z$ on $\mathcal{M}$ there is a corresponding operator $\bar{\psi} : C^\infty(X, Y) \to C^\infty(X, Z)$ that is $\mathcal{G}$-equivariant. An equivariant operator in $\mathcal{M}_1$ can be related to an equivariant one in the space of functions Y^X. Suppose $\psi : \mathcal{M}_1 = X \times Y \to Z$ is $\mathcal{G}$-invariant, then take $\bar{\psi} : Y^X \to Z^X$ to be $\bar{\psi}[f](\mathbf{x}) = \psi(\mathbf{x}, f(\mathbf{x}))$, for all $\mathbf{x} \in X$, $f \in Y^X$. Assuming that the action on Y is the identity $g \cdot u = u$ for $u \in Y$ and the same for Z we have

$$\begin{aligned} \bar{\psi}(g \cdot f)(\mathbf{x}) &= \psi(\mathbf{x}, (g \cdot f)(\mathbf{x}))) \\ &= \psi(\mathbf{x}, f(g^{-1} \cdot \mathbf{x})] \\ &= \psi(g^{-1}\mathbf{x}, f(g^{-1} \cdot \mathbf{x}) \\ &= \bar{\psi}(f)(g^{-1} \cdot \mathbf{x}) = [g \cdot \bar{\psi}(f)](\mathbf{x}), \end{aligned} \tag{4.12}$$

therefore $g \cdot \bar{\psi}(f) = \bar{\psi}(g \cdot f)$ for all $g \in \mathcal{G}$, $f \in Y^X$. In other words, an invariant operator in the Cartesian product $X \times Y$ to Z induces an equivariant operator taking functions in Y^X to functions in Z^X.

4.3.2 Jet-Space

In general, the action on $\mathcal{M}$ may not be free and regular. Indeed a necessary condition for this is that the dimension of the orbits $\mathcal{O}_z$ of $\mathcal{G}$ be equal to the dimension of $\mathcal{G}$, i.e. $\dim \mathcal{O}_z = \dim \mathcal{G} = r$. When the orbits are lower-dimensional, a solution might be to compute a moving frame in the higher dimensional jet space $J^n(\mathcal{M})$, for a large enough n, equipped with the prolonged group action (Olver, 1995).

Bluntly speaking, a n-jet is the tuple of partial derivatives of order $\leq n$ of a section on a fibre bundle and the Jet-Bundle is a fibre bundle containing the spaces of n-jets as its fibres. In our case, we have to assume a fibre bundle structure on $\mathcal{M}$.

We are mainly interested in jet-bundles of Euclidean spaces $\mathcal{M} = X \times Y$ where $X = \mathbb{R}^d$, $Y = \mathbb{R}$. Given a multi-index $I = (i_1, \cdots, i_d) \in \mathbb{N}^d$ let us denote by $|I| = \sum_{k=1}^{d} i_k$ its modulus and by $\mathcal{I}_n^d = \{I \in \mathbb{N}^d, |I| \leq n\}$ the set of multi-indices in $\mathbb{N}^d$ of modulus at most $n \in \mathbb{N}$. For $\mathbf{x} \in X$ and $n \in \mathbb{N}$, we introduce $\mathcal{J}_{\mathbf{x}}^{(n)}$ the operator mapping $C^\infty(X, Y)$ to $Y_d^{(n)} = \mathbb{R}^{\binom{n+d}{n}}$, and defined for any $f \in C^\infty(X, Y)$ by

$$\mathcal{J}_{\mathbf{x}}^{(n)} f = \left(\partial^I f(\mathbf{x})\right)_{I \in \mathcal{I}_n^d} = \left(\frac{\partial^{|I|}}{\partial x_1^{i_1} \partial x_2^{i_2} \cdots \partial x_d^{i_d}} f(\mathbf{x}) \right)_{I=(i_1,\ldots i_d) \in \mathcal{I}_n^d}. \tag{4.13}$$

Then given $u^{(n)} = \left(u_I\right)_{I \in \mathcal{I}_n^d} \in Y_d^{(n)}$ and $\mathbf{x} \in X$, the set $\mathcal{C}_{\mathbf{x},u^{(n)}} \coloneqq (\mathcal{J}_{\mathbf{x}}^{(n)})^{-1}(u^{(n)})$ is an equivalence class over $C^\infty(X, Y)$ for the equivalence relation $f_1 \sim f_2 \iff \mathcal{J}_{\mathbf{x}}^{(n)}(f_1) = \mathcal{J}_{\mathbf{x}}^{(n)}(f_2)$. This class $\mathcal{C}_{\mathbf{x},u^{(n)}}$ is represented in particular by the polynomial function defined, for any $\mathbf{t} \in X$, by

$$u(\mathbf{t}) = \sum_{I=(i_1,\ldots,i_d) \in \mathcal{I}_n^d} \frac{u_I}{I!} (t_1 - x_1)^{i_1} \ldots (t_d - x_d)^{i_d}, \tag{4.14}$$

with $I! = i_1! i_2! \ldots i_d!$. It is the Taylor polynomial of order n at $\mathbf{x}$ of any function of the class. The nth-order jet space of $\mathcal{M}$, noted $J^n(\mathcal{M})$, is the union of all such equivalence classes, and can therefore be identified to $X \times Y_d^{(n)}$. According to the above, for an element $(\mathbf{x}, u^{(n)}) = (\mathbf{x}, (u_I)_{I \in \mathcal{I}_n^d})$, each u_I is also a partial derivative of u evaluated in $\mathbf{x}$, namely $u_I = \partial^I u(\mathbf{x})$. For example, if $d = 3$ and $\mathbf{x} = (x, y, z)$, $u_{(0,0,0)} = u(\mathbf{x})$, $u_{(1,0,0)} = \frac{\partial u}{\partial x}(\mathbf{x}) = u_x(\mathbf{x})$, $u_{(1,1,0)} = \frac{\partial^2 u}{\partial x \partial y}(\mathbf{x}) = u_{xy}(\mathbf{x})$ and so on. In practice we will often use these partial derivative notations to identify elements of the jet space, and omit the variable as it is explicit from the first component. For example in the case $d = 2$, $n = 2$, an element $\mathbf{z} \in \mathcal{M} = J^0(\mathcal{M})$ is identified by $\mathbf{z} = (\mathbf{x}, u) = (x, y, z, u)$ and an element $\mathbf{z}^{(2)} \in J^2(\mathcal{M})$ by

$$\mathbf{z}^{(2)} = (\mathbf{x}, u^{(2)}) = (x, y, z, u, u_x, u_y, u_{xx}, u_{xy}, u_{yy})$$

and analogously for higher orders.

By prolonging the elements of $\mathcal{M}$ to the jet-space the goal is to find a new space where the group action acts freely. It is therefore necessary to know how exactly the group acts on the elements of $J^n(\mathcal{M})$. In the following we describe the *prolonged group action* which is the natural extension of the action π_g on $\mathcal{M}$ to $J^n(\mathcal{M})$.

Prolonged Group Action

The goal of looking into the jet-space is to have a higher dimensional space where the action of $\mathcal{G}$ is (locally) free and extends the manifold $\mathcal{M}$. We have defined the jet-space, but it is left to know how $\mathcal{G}$ acts on it. This action is what we refer to *prolongation* of the group action to the jet-space.

An action on $C^\infty(X, Y)$ induces an action on the space of polynomials by restriction, $(g \cdot \mathbf{p})(\mathbf{t}) = \mathbf{p}(g^{-1} \cdot \mathbf{t})$. Let $(\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M})$ and let $u(\mathbf{t})$ be any function such that $\mathcal{J}_{\mathbf{x}}^{(n)} u = u^{(n)}$(for

example the polynomial (4.14) associated to $\mathbf{x}$ and $u^{(n)}$). We define the *prolongation* of the action of $\mathcal{G}$ on $\mathcal{M}$ to the jet-space $J^n(\mathcal{M})$, as follows: for any $g \in \mathcal{G}$ and $\mathbf{z}^{(n)} = (\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M})$,

$$g \cdot \mathbf{z}^{(n)} = g \cdot (\mathbf{x}, u^{(n)}) \coloneqq (g \cdot \mathbf{x}, \mathcal{J}^{(n)}_{g\cdot\mathbf{x}}(g \cdot u)). \tag{4.15}$$

The group transformation does not depend on the choice of the representative of $u^{(n)}$, i.e., that for all $u, v \in C^\infty(X, Y)$, if $\mathcal{J}^{(n)}_{\mathbf{x}} u = \mathcal{J}^{(n)}_{\mathbf{x}} v$ then $\mathcal{J}^{(n)}_{g\cdot\mathbf{x}} g \cdot u = \mathcal{J}^{(n)}_{g\cdot\mathbf{x}} g \cdot v$. This can be verified by checking that the expression above depends only on the values $\mathbf{x}$ and the derivatives $\mathcal{J}^{(n)}_{\mathbf{x}} u$ (Olver, 1993). We can prove that (4.15) is a left group action, i.e. for $g_1, g_2 \in \mathcal{G}$

$$\begin{aligned} g_1 \cdot \big(g_2 \cdot (\mathbf{x}, u^{(n)})\big) &= g_1 \cdot (g_2 \cdot \mathbf{x}, \mathcal{J}^{(n)}_{g_2\cdot\mathbf{x}}(g_2 \cdot u)) \\ &= (g_1 \cdot g_2 \cdot \mathbf{x}, \mathcal{J}^{(n)}_{g_1\cdot g_2\cdot\mathbf{x}}(g_1 \cdot g_2 \cdot u) \\ &= (g_1 \cdot g_2) \cdot (\mathbf{x}, u^{(n)}). \end{aligned} \tag{4.16}$$

With the prolonged group action we can define a differential invariant.

Definition 13 (Differential invariant)**.** *Given manifolds $\mathcal{M} = X \times Y$ and $\mathcal{N}$, a differential invariant of order n is a smooth function $F : U \to \mathcal{N}$, with $U \subseteq J^n(\mathcal{M})$ open, that is invariant under the (local) action of $\mathcal{G}$, i.e.*

$$\exists V \subseteq \mathcal{G},\ \forall g \in V,\ \forall \mathbf{z}^{(n)} \in U,\ F(g \cdot \mathbf{z}^{(n)}) = F(\mathbf{z}^{(n)}). \tag{4.17}$$

Since the jet-space of order n $J^n(\mathcal{M})$ is a manifold acted upon by $\mathcal{G}$, if the action is locally free we can apply the method of moving frames to obtain the differential invariants of order n. It turns out that if n is large enough we can garantee the local freeness of the jet-space (Olver, 2007, 2000). Furthermore, a differential invariant defines an equivariant operator on functions just like an invariant in $\mathcal{M}$ defines an equivariant operator. Let $\psi : J^n(\mathcal{M}_1) = X \times Y^{(n)}$ be a differential invariant and define $\bar{\psi} : C^\infty(X, Y) \to C^\infty(X, Z)$ as $\bar{\psi}[f](\mathbf{x}) = \psi(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f)$. To see that $\bar{\psi}$ is $\mathcal{G}$-equivariant, first notice that $\mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f) = g \cdot \mathcal{J}^{(n)}_{g^{-1}\cdot\mathbf{x}} f$. Now, we have

$$\begin{aligned} \bar{\psi}(g \cdot f)(\mathbf{x}) &= \psi(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f)) \\ &= \psi(\mathbf{x}, g \cdot \mathcal{J}^{(n)}_{g^{-1}\cdot\mathbf{x}} f] \\ &= \psi(g^{-1}\mathbf{x}, \mathcal{J}^{(n)}_{g^{-1}\cdot\mathbf{x}} f) \\ &= \bar{\psi}(f)(g^{-1} \cdot \mathbf{x}) = [g \cdot \bar{\psi}(f)](\mathbf{x}), \end{aligned} \tag{4.18}$$

i.e. $\bar{\psi} \circ \pi_g = \pi_g \circ \bar{\psi}$.

4.3.3 Fundamental Invariants

Given a moving frame on a finite-dimensional manifold $\mathcal{M}$ there exists a set of generating invariants of order n from which all other invariants of order n can be obtained through functional combination.

First, let us consider an operator $F : \mathcal{M} \to \mathcal{N}$ invariant to an action of a Lie group $\mathcal{G}$: $\forall g \in \mathcal{G}, \forall \mathbf{z} \in \mathcal{M},\ F(g \cdot \mathbf{z}) = F(\mathbf{z})$. Then in particular, the existence of a moving frame $\rho : \mathcal{M} \to \mathcal{G}$ yields:

$$\forall \mathbf{z} \in \mathcal{M},\ F(\mathbf{z}) = F(\rho(\mathbf{z}) \cdot \mathbf{z}) = F(\iota[\mathbf{z}]) \tag{4.19}$$

where we use the notation $\imath[\mathbf{z}] = \imath[\mathrm{id}](\mathbf{z}) = \rho(\mathbf{z}) \cdot \mathbf{z}$. Since we did not assume anything of $\mathcal{I}$ other than being an invariant, this applies to all invariant operators on $\mathcal{M}$.

Now recall from Section 4.3.2 that $J^n(\mathcal{M}) \simeq X \times Y_d^{(n)}$ where $X = \mathbb{R}^d$. Therefore an element $\mathbf{z}^{(n)} \in J^{(n)}(\mathcal{M})$ is identified to the vector $\mathbf{z}^{(n)} = (\mathbf{x}, u^{(n)}) = \Big(x_1, \ldots, x_d, (u_I)_{I \in \mathcal{I}_n^d}\Big)$, where $\mathcal{I}_n^d$ is the set of d-dimensional multi-indices of modulus non-larger than n. We note proj_i, $1 \leq i \leq d$, the application that maps each $\mathbf{z}^{(n)} \in J^{(n)}(\mathcal{M})$ to its ith spatial coordinate x_i, $\mathbf{z}^{(n)} \mapsto x_i$, and proj_I, $I \in \mathcal{I}_n^d$, the one mapping $\mathbf{z}^{(n)}$ to its coefficient u_I of multi-index I, $\mathbf{z}^{(n)} \mapsto u_I$. The fundamental invariants are the invariantized versions of these projectors.

Definition 14 (Fundamental invariants)**.** *The fundamental invariants of order n are the invariantizations $(\imath[proj_i])_{1 \leq i \leq d}$ and $(\imath[proj_I])_{I \in \mathcal{I}_n^d}$ of the $d + \binom{n+d}{n}$ coordinates projectors in the jet-space $J^{(n)}(\mathcal{M})$. By definition, $\forall \mathbf{z}^{(n)} \in J^{(n)}(\mathcal{M})$,*

$$\imath[proj_i](\mathbf{z}^{(n)}) = proj_i(\imath[\mathbf{z}^{(n)}]) \quad \textit{and} \quad \imath[proj_I](\mathbf{z}^{(n)}) = proj_I(\imath[\mathbf{z}^{(n)}]) \tag{4.20}$$

where, like earlier, $\imath[\mathbf{z}^{(n)}] = \rho(\mathbf{z}^{(n)}) \cdot \mathbf{z}^{(n)}$. We will also note $\imath[x_i] \coloneqq \imath[proj_i](\mathbf{z}^{(n)})$ and $\imath[u_I] \coloneqq \imath[proj_I](\mathbf{z}^{(n)})$.

From the above, we get that a $\mathcal{G}$-invariant operator F defined on the jet-space verifies

$$\begin{aligned} \forall \mathbf{z}^{(n)} \in J^{(n)}(\mathcal{M}), \quad F(\mathbf{z}^{(n)}) &= F(\imath[\mathbf{z}^{(n)}]) \\ &= F\Big(\mathrm{proj}_1(\imath[\mathbf{z}^{(n)}]), \ldots, \mathrm{proj}_d(\imath[\mathbf{z}^{(n)}]), (\mathrm{proj}_I(\imath[\mathbf{z}^{(n)}]))_{I \in \mathcal{I}_n^d}\Big) \\ &= F\Big(\imath[\mathrm{proj}_1](\mathbf{z}^{(n)}), \ldots, \imath[\mathrm{proj}_d](\mathbf{z}^{(n)}), (\imath[\mathrm{proj}_I](\mathbf{z}^{(n)}))_{I \in \mathcal{I}_n^d}\Big) \\ &= F\left(\imath[x_1], \ldots, \imath[x_d], (\imath[u_I])_{I \in \mathcal{I}_n^d}\right). \end{aligned} \tag{4.21}$$

In other words, any $\mathcal{G}$-invariant operator defined on $J^{(n)}(\mathcal{M})$ is a function of the fundamental invariants of order n.

4.4 Differential Invariants of SE(2) on Images

Let us view a 2D image as a surface in $\mathcal{M} = \mathbb{R}^3$ given by the graph of a function $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ with open Ω. We seek equivariance to the special Euclidean group, SE(2), the group of planar rotations and translations. We denote $g \in \mathrm{SE}(2)$ as $g = (R, \mathbf{v})$ where $R \in \mathrm{SO}(2)$ is a planar rotation and $\mathbf{v} = (v_1, v_2) \in \mathbb{R}^2$ is a translation. Its action π_g, $g \in \mathcal{G}$ on a point $(\mathbf{x}, u) = ((x, y), u) \in \mathcal{M}$ by

$$\begin{aligned} \pi_{R,\mathbf{v}}(\mathbf{x}, u) &= (R\mathbf{x} + \mathbf{v}, u) \\ &= ((\tilde{x}, \tilde{y}), u) \\ &= ((x\cos\theta - y\sin\theta + v_1, x\sin\theta + y\cos\theta + v_2), u) \end{aligned} \tag{4.22}$$

Before we can apply the method of moving frames we must assure that $\mathcal{G}$ acts freely on $\mathcal{M}$. That is not the case, however. Take for example $((0,0), u)$ and transform it by rotating it by any angle and we arrive at the same point, but the transformation is not the identity. Because the angle can be arbitrarily small this action is also not locally free. In order to be able to

obtain a moving frame let us instead consider $\mathcal{G}$ acting on the n-th order jet-space $J^n(\mathcal{M})$ via the prolonged group action.

Given an element $\mathbf{z}^{(n)} = (\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M})$ we want to compute the prolonged action $g \cdot \mathbf{z}^{(n)}$. In light of Section 4.3.2 we identify $u(\mathbf{t})$ as the polynomial associated with $u^{(n)}$. To obtain an expression for the prolonged action of $\mathcal{G}$ in the $J^n(\mathcal{M})$ we must compute the partial derivatives $\mathcal{J}^{(n)}_{g\cdot\mathbf{x}}(g \cdot u)$ of the function $g \cdot u$, according to (4.15).

$$\begin{aligned} \tfrac{\partial^{i+j}}{\partial\tilde{x}^i\partial\tilde{y}^j}(g\cdot u)\big|_{g\cdot(x,y)} &= \tfrac{\partial^{i+j}}{\partial\tilde{x}^i\partial\tilde{y}^j}u(g^{-1}\cdot g\cdot(x,y)) \\ &= \tfrac{\partial^{i+j}}{\partial\tilde{x}^i\partial\tilde{y}^j}u(x,y) \\ = \tfrac{\partial^{i+j}}{\partial\tilde{x}^i\partial\tilde{y}^j}u((\tilde{x}-v_1)\cos\theta + (\tilde{y}-v_2)\sin\theta, &(v_1-\tilde{x})\sin\theta + (\tilde{y}-v_2)\cos\theta). \end{aligned} \tag{4.23}$$

substituting the values $\mathbf{t} = g \cdot \mathbf{x}$

Keeping in mind the identification of the coordinates of the jet-space given by $u_{(0,0)} = u$, $u_{(1,0)} = u_x$ $u_{(0,1)} = u_y$ and so on we can now obtain closed expressions for the prolonged action. In particular, the prolonged action on the first order jet-space $J^1(\mathcal{M})$ is $\pi_{R,\mathbf{v}}(x, y, u, u_x, u_y) = (\tilde{x}, \tilde{y}, \tilde{u}, \tilde{u}_x, \tilde{u}_y)$, where $\tilde{x}$, $\tilde{y}$ are the same as in (4.22), $\tilde{u} = u$ and

$$\begin{aligned} \tilde{u}_x &= u_x\cos\theta - u_y\sin\theta \\ \tilde{u}_y &= u_x\sin\theta + u_y\cos\theta. \end{aligned} \tag{4.24}$$

Furthermore, with the interest of later computing invariants, the action on the second-order jet-space maps u_{xx}, u_{xy} and u_{yy} to $\tilde{u}_{xx}, \tilde{u}_{xy}$ and $\tilde{u}_{yy}$, where

$$\begin{aligned} \tilde{u}_{xx} &= u_{xx}(\cos\theta)^2 + 2u_{xy}\cos\theta\sin\theta + u_{yy}(\sin\theta)^2, \\ \tilde{u}_{xy} &= (u_{yy}-u_{xx})\cos\theta\sin\theta - u_{xy}((\sin\theta)^2 - (\cos\theta)^2), \\ \tilde{u}_{yy} &= u_{xx}(\sin\theta)^2 - 2u_{xy}\cos\theta\sin\theta + u_{yy}(\cos\theta)^2. \end{aligned} \tag{4.25}$$

By choosing an appropriate cross-section, we compute a moving frame ρ on $J^n(\mathcal{M})$ for $n \geq 1$ from these five components only. Let $\mathcal{K}$ be the cross-section in $J^n(\mathcal{M})$ defined by $\mathcal{K} = \{\mathbf{z}^{(n)} \in J^n | x = y = u_y = 0, u_x > 0\}$, near a point $\mathbf{z}^{(n)}$ such that $\nabla u = (u_x, u_y) \neq \vec{0}$. For every open set $U \subset J^n(\mathcal{M})$ such that $\forall \mathbf{z}^{(n)} \in U, (u_x, u_y) \neq (0, 0)$, $\mathcal{K}$ constitutes a regular cross-section, to see that first notice that it has $3 = \dim \mathrm{SE}(2)$ equations. Moreover when observing only the coordinates (u_x, u_y) the orbits and cross-section reduce to those in Example 1 and there is a unique translation that maps (x, y) to $(0, 0)$.

Writing the system of equations (4.7) gives us

$$\begin{aligned} x\cos\theta - y\sin\theta + v_1 &= 0 \\ x\sin\theta + y\cos\theta + v_2 &= 0 \\ u_x\sin\theta + u_y\cos\theta &= 0. \end{aligned} \tag{4.26}$$

From the last constraint we can deduce

$$0 = \tilde{u}_y = u_x\sin\theta + u_y\cos\theta \implies \theta = \tan^{-1}\left(-\frac{u_y}{u_x}\right),$$

hence θ is the opposite of the argument of ∇u. Moreover, the other parameters can be deduced from the other two constraints: $v_1 = -x\cos\theta + y\sin\theta$ and $v_2 = -x\sin\theta - y\cos\theta$. The moving

frame that is obtained from the cross-section $\mathcal{K}$ can thus be written in the form

$$\rho(\mathbf{z}^{(n)}) = \left(\frac{1}{\|\nabla u\|} \begin{bmatrix} u_x & u_y \\ -u_y & u_x \end{bmatrix}, \frac{-1}{\|\nabla u\|} \begin{bmatrix} u_x & u_y \\ -u_y & u_x \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \right). \tag{4.27}$$

This completely determines ρ. Fundamental differential invariants can now be obtained by the invariantization of the partial derivatives of u.

Let us denote the fundamental invariants associated with the multi-index (i, j) as $\mathcal{I}_{ij} \coloneqq \imath[u_{i,j}]$. By the moving frame deduced from the cross-section $\mathcal{K}$, we have $\imath[x] = \imath[y] = 0$. From (4.20), we know that given any differential operator $F(\mathbf{z}^{(n)})$, its invariantization is given by

$$\imath[F](\mathbf{z}^{(n)}) = \imath[F](x, y, (u_{i,j})_{0\leq i+j\leq n}) = F(0, 0, (\mathcal{I}_{i,j})_{0\leq i+j\leq n}).$$

We show the fundamental invariants of the second order jet-space $J^2(\mathcal{M})$ in Table 4.1. They are computed by substituting (4.27) into (4.24) and (4.25).

$\imath[u]$	$\mathcal{I}_{00} = u$
$\imath[u_x]$	$\mathcal{I}_{10} = \sqrt{u_x^2 + u_y^2}$
$\imath[u_{xx}]$	$\mathcal{I}_{20} = \frac{1}{\|\nabla u\|^2}(u_{xx}u_x^2 + u_{xy}u_xu_y + u_{yy}u_y^2)$
$\imath[u_{xy}]$	$\mathcal{I}_{11} = \frac{1}{\|\nabla u\|^2}(u_xu_y(u_{yy} - u_{xx}) - u_{xy}(u_y^2 - u_x^2))$
$\imath[u_{yy}]$	$\mathcal{I}_{02} = \frac{1}{\|\nabla u\|^2}(u_{xx}u_y^2 - u_{xy}u_xu_y + u_{yy}u_x^2)$

Table 4.1: Invariantizations of the partial derivatives of u up to order 2, assuming $\|\nabla u\| \neq 0$. These determine all other differential invariants of order two by functional combination, for example, the Laplacian $\Delta u = u_{xx} + u_{yy} = \mathcal{I}_{20} + \mathcal{I}_{02}$.

Let us obtain a closed expression for the other invariants. First let us consider the polynomial $u(\mathbf{t})$ associated to $u^{(n)}$. If we expand the expression of the prolonged group action (4.23) we obtain

$$\begin{aligned}
\widetilde{\frac{\partial^n u}{\partial x^i \partial y^j}}(x, y) &= \frac{\partial^n u(x,y)}{\partial \tilde{x}^i \partial \tilde{y}^j} \\
&= \frac{\partial^n}{\partial \tilde{x}^i \partial \tilde{y}^j} u(\cos\theta(\tilde{x} - v_1) + \sin\theta(\tilde{y} - v_2), \cos\theta(\tilde{y} - v_2) - \sin\theta(\tilde{x} - v_1)) \\
&= \left(\tfrac{\partial}{\partial \tilde{x}}\right)^i \left(\tfrac{\partial}{\partial \tilde{y}}\right)^j u(\cos\theta(\tilde{x} - v_1) + \sin\theta(\tilde{y} - v_2)), \cos\theta(\tilde{y} - v_2) - \sin\theta(\tilde{x} - v_1))) \\
&= \left(\cos\theta \tfrac{\partial}{\partial x} + \sin\theta \tfrac{\partial}{\partial y}\right)^i \left(\cos\theta \tfrac{\partial}{\partial y} - \sin\theta \tfrac{\partial}{\partial x}\right)^j u(x, y).
\end{aligned} \tag{4.28}$$

Substituting by the values of the moving frame ρ from (4.27), i.e., making the substitution $\cos\theta = u_x/\|\nabla u\|$ and $\sin\theta = u_y/\|\nabla u\|$, and further expanding the expression results in

$$\begin{aligned}
\mathcal{I}_{ij} &= \tfrac{1}{\|\nabla u\|^n} \left(u_x \tfrac{\partial}{\partial x} + u_y \tfrac{\partial}{\partial y}\right)^i \left(u_x \tfrac{\partial}{\partial x} - u_y \tfrac{\partial}{\partial x}\right)^j u \\
&= \tfrac{1}{\|\nabla u\|^n} \left(\sum_{k=0}^{i} \tbinom{i}{k} u_x^k u_y^{i-k} \tfrac{\partial^i}{\partial x^k \partial y^{i-k}} \right) \left(\sum_{l=0}^{j} \tbinom{j}{l} (-1)^l u_x^{j-l} u_y^l \tfrac{\partial^j}{\partial x^l \partial y^{j-l}} \right) u \\
&= \tfrac{1}{\|\nabla u\|^n} \left(\sum_{k=0}^{i} \sum_{l=0}^{j} \tbinom{i}{k} \tbinom{j}{l} (-1)^l u_x^{k+j-l} u_y^{l+i-k} \tfrac{\partial^n}{\partial x^{k+l} \partial y^{i+j-k-l}} \right) u \\
&= \tfrac{1}{\|\nabla u\|^n} \sum_{m=0}^{n} \left(\sum_{l=\max\{0,m-i\}}^{\min\{m,j\}} \tbinom{i}{m-l} \tbinom{j}{l} (-1)^l u_x^{m-2l+j} u_y^{2l-m+i} \right) \tfrac{\partial^n}{\partial x^m \partial y^{n-m}} u.
\end{aligned} \tag{4.29}$$

4.4.1 Invariants From a Recurrence Equation

In Sangalli et al. (2022a) we used a different method of computing the differential invariants of orders higher than two. There we based the computation of the invariants in the recurrence equations in Olver (2007). Those equations show that all differential invariants can be written as a functional combination of I_{00}, I_{02} and its invariant derivatives, which for SE(2) are

$$\begin{aligned}\mathcal{D}_1 = \imath[D_x] &\coloneqq \|\nabla u\|^{-1}(u_x\tfrac{\partial}{\partial x} + u_y\tfrac{\partial}{\partial y}),\\ \mathcal{D}_2 = \imath[D_y] &\coloneqq \|\nabla u\|^{-1}(-u_y\tfrac{\partial}{\partial x}x + u_x\tfrac{\partial}{\partial y}),\end{aligned} \tag{4.30}$$

where D_x and D_y denote the total differentiation operators on the jet-space, which are the operators on the tangent bundle of the jet-space that given by $D_x = \frac{\partial}{\partial x} + \sum_{0\le i+j\le n} u_{i+1,j}\frac{\partial}{\partial u_{i,j}}$, and $D_y = \frac{\partial}{\partial y} + \sum_{0\le i+j\le n} u_{i,j+1}\frac{\partial}{\partial u_{i,j+1}}$. They are related to the partial derivatives in the sense that $D_xF(\mathbf{x}, u^{(n)}) = \frac{\partial}{\partial x}F(\mathbf{x}, u(\mathbf{x}))$ (and similarly for D_y) where $u(\mathbf{x})$ is the polynomial associated to $u^{(n)}$. The invariant derivatives (4.30) are equivalently the directional derivatives in the directions of the gradient and its perpendicular.

More specifically, the recurrence relation applied to SE(2) on $J^n(\mathcal{M})$ has as initial conditions $\mathcal{I}_{00}$ and $\mathcal{I}_{02}$ given in Table 4.1, and $\mathcal{I}_{01} = 0$, $\mathcal{I}_{10} = \mathcal{D}_1\mathcal{I}_{00}$, $\mathcal{I}_{20} = \mathcal{D}_1\mathcal{I}_{10}$, $\mathcal{I}_{11} = \mathcal{D}_2\mathcal{I}_{10}$ and for i, j such that $i + j = n \ge 3$:

$$\mathcal{I}_{ij} = \begin{cases} -\mathcal{D}_1\mathcal{I}_{i-1,j} - P_{ij}(\mathcal{F}_{n-1})\frac{1}{\mathcal{I}_{10}}, & \text{if } i > 0\\ \mathcal{D}_2\mathcal{I}_{0,j-1} - P_{0,j}(\mathcal{F}_{n-1})\frac{1}{\mathcal{I}_{10}}, & \text{if } i = 0.\end{cases} \tag{4.31}$$

where the P_{ij} are polynomials and $\mathcal{F}_n$ are invariantizations of the derivatives up to order n, $\mathcal{F}_n = (\mathcal{I}_{i,j})_{0\le i+j\le n}$. The polynomials P_{ij} can be determined by solving a system of equations, but since their purpose here is to show that the fundamental invariants of superior order are functional combination of the invariant derivatives of the fundamental invariant of order two, we do not need to know their explicit forms.

The invariant derivatives do not commute, i.e. $[\mathcal{D}_1, \mathcal{D}_2] \neq 0$ in general, but the following is valid

$$[\mathcal{D}_1, \mathcal{D}_2] = \mathcal{D}_1 \circ \mathcal{D}_2 - \mathcal{D}_2 \circ \mathcal{D}_1 = J_1\mathcal{D}_1 + J_2\mathcal{D}_2 \tag{4.32}$$

where J_1 and J_2 are some differential invariants or order ≤ 2. In particular this implies that all of the combinations of $\mathcal{D}_{k_1}\cdots\mathcal{D}_{k_K}$, $k_i \in \{1, 2\}$ and $0 \le K \le n$ generate all differential invariants of order n.

In the rest of the chapter we use the expressions (4.29) to compute the fundamental invariants, instead of using these recurrence relations as was done in Sangalli et al. (2022a).

4.4.2 Equivariant Neural Network Using Differential Invariants

Suppose we want to use differential invariants to obtain a SE(2)-equivariant neural network ϕ that takes signals on $\mathbb{R}^2$ and outputs signals on $\mathbb{R}^2$. The following approach would work.

- Let an input be a function $f \in C^\infty(X, Y) = C^\infty(\mathbb{R}^2, \mathbb{R})$, we use a single-channel image for simplicity here. The first step would be the computation of the differential invariants,

$$(\imath[x], \imath[y], (\mathcal{I}^0_{i,j})_{0\le i+j\le n}) = \rho(\mathbf{z}^0_{\mathbf{x}}) \cdot \mathbf{z}^0_{\mathbf{x}} = \rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f) \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f).$$

where $\mathbf{z}^0_{\mathbf{x}} \coloneqq (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f)$. From those invariants we can compute an arbitrary invariant of order n by applying a smooth function $\psi_1 : J^n(\mathcal{M}) \to \mathbb{R}^C$. In a deep learning context such function can be represented/approximated by a Multi-Layer Perceptron (MLP). The output of the first layer is given by

$$f_1(\mathbf{x}) = \phi_1(f)(\mathbf{x}) = \psi_1(\imath[x], \imath[y], (\mathcal{I}^0_{i,j})_{0\leq i+j\leq n})$$

- Now from f_1 we compute the second layer in a similar manner. Let us denote $f_1 = (f_1^1, \dots, f_1^C)$. The fundamental invariants are given by

$$(\imath[x], \imath[y], (\mathcal{I}^{1,k}_{i,j})_{0\leq i+j\leq n}) = \rho(\mathbf{z}^{1,k}_{\mathbf{x}}) \cdot \mathbf{z}^{1,k}_{\mathbf{x}} = \rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f_1^k) \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f_1^k).$$

  where $\mathbf{z}^{1,k}_{\mathbf{x}} \coloneqq (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f_1^k)$, $k = 1, \dots C$. Given a smooth function $\psi_2 : (J^n(\mathcal{M}))^C \to \mathbb{R}^{C'}$ we can compute the second layer by

$$\phi_2'(f_1)(\mathbf{x}) = \psi_2(\imath[x], \imath[y], (\mathcal{I}^{1,1}_{i,j})_{0\leq i+j\leq n}, \dots \imath[x], \imath[y], (\mathcal{I}^{1,C}_{i,j})_{0\leq i+j\leq n}).$$

  The output from the second layer given f is given by $\phi_2 = \phi_2' \circ \phi_1$.

The proccess above can be repeated L times to obtain a L-layer networks $\phi_L = \phi$. Note that at each step the operators ϕ_l' are equivariant by the fact that they are functional combinations of differential invariants, and thus equivariant operators on functions by (4.18).

4.5 SE(2) Differential Invariants Networks

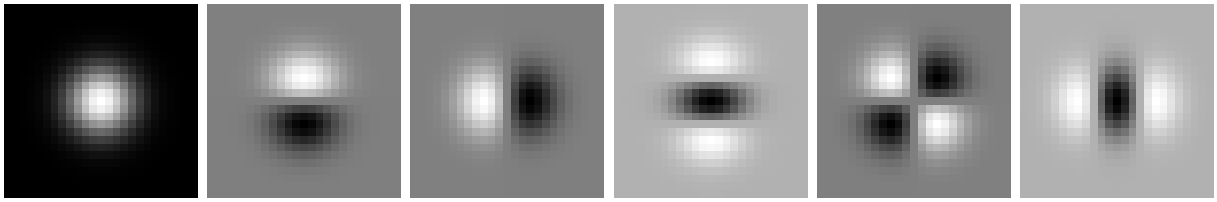
In this section we define the SE(2) Differential Invariants Networks (SE2DINNet). A SE2DINNet is made of equivariant blocks (SE2DIN blocks) consisting in computing the derivatives, followed by computing the differential invariants of SE(2) up to a certain order, followed by a series of 1×1 convolutions, in the style of Network in Network (NiN) (Lin et al., 2013). We illustrate the SE2DIN block in Figure 4.5 and we will explain its components in this section.

4.5.1 Gaussian Derivative Layers

In order to compute the differential invariants we use Gaussian derivatives (Koenderink and van Doorn, 1987). In classical computer vision Gaussian derivatives were combined with differential invariants to perform automatic scale selection (Lindeberg, 1998, 1999). Gaussian derivatives have also been used in neural networks to produce structured receptive fields in CNNs (Jacobsen et al., 2016; Penaud-Polge et al., 2022) and they provide a regularization effect to the CNN. In this work we combine Gaussian derivatives with differential invariants and apply it to build equivariant neural networks. Gaussian derivatives are used to compute the derivatives of a Gaussian filtered image, *i.e.*

$$\frac{\partial^{i+j}}{\partial x^i \partial y^j}(u * G_\sigma) = u * \frac{\partial^{i+j}}{\partial x^i \partial y^j} G_\sigma = u * G_\sigma^{i,j}, \tag{4.33}$$

where $G_\sigma^{i,j} = \frac{\partial^{i+j} G_\sigma}{\partial x^i \partial y^j}$.



$G_\sigma^{0,0}$ $G_\sigma^{1,0}$ $G_\sigma^{0,1}$ $G_\sigma^{2,0}$ $G_\sigma^{1,1}$ $G_\sigma^{0,2}$

Figure 4.3: Gaussian derivative filters of order ≤ 2 computed in a discrete domain.

In Jacobsen et al. (2016) this is used to define a basis of filters such that a CNN filter $h : \mathbb{R}^2 \to \mathbb{R}$ at scale σ can be written as

$$h(\mathbf{x}) = \sum_{0 \leq i+j \leq n} c_{i,j} G_\sigma^{i,j}(\mathbf{x}) \tag{4.34}$$

so that the convolution by h becomes

$$\begin{aligned}(f * h)(x) &= \int_{\mathbb{R}^2} f(\mathbf{y}) h(\mathbf{x} - \mathbf{y}) d\mathbf{y} \\ &= \int_{\mathbb{R}^2} f(\mathbf{y}) \sum_{i+j \leq n} c_{i,j} G_\sigma^{i,j}(\mathbf{x} - \mathbf{y}) d\mathbf{y} \\ &= \sum_{i+j \leq n} c_{i,j} \int_{\mathbb{R}^2} f(\mathbf{y}) G_\sigma^{i,j}(\mathbf{x} - \mathbf{y}) d\mathbf{y} \\ &= \sum_{i+j \leq n} c_{i,j} (f * G_\sigma^{i,j})(\mathbf{x}),\end{aligned} \tag{4.35}$$

that is, the convolution by h can be obtained by linear combinations of the Gaussian derivatives of f. The basis of filters $G_\sigma^{i,j}$ is shown to form a complete description of the local geometry of an image (Koenderink and van Doorn, 1987).

A one dimensional Gaussian derivative can be computed by means of the probabilist's Hermite polynomials. Let $H_m = (-1)^m e^{\frac{x^2}{2}} \frac{d^m}{dx^m} e^{-\frac{x^2}{2}}$, with $m \in \mathbb{N}$ be the Hermite polynomial of order m. The Gaussian derivative of order m in one variable, denoted G_σ^m can be computed as

$$G_\sigma^m(x) = (-1)^m \frac{1}{\sqrt{\sigma^m}} H_m \left(\frac{x}{\sigma\sqrt{2}} \right) G_\sigma(x). \tag{4.36}$$

Furthermore, a 2-dimensional Gaussian derivative is a separable filter, i.e.

$$G_\sigma^{i,j}(x, y) = G_\sigma^i(x) G_\sigma^j(y) \tag{4.37}$$

and can therefore be expressed with the one-dimensional Hermite polynomials. Moreover separability means that the discrete approximation of $f * G_\sigma^{i,j}$ can be obtained by computing a convolution with a horizontal filter, with the values of $G_\sigma^i(x)$ in the line $y = 0$ and a vertical one with the values of $G_\sigma^j(y)$ in the line $x = 0$.

Given a multi-channel function $f : \mathbb{Z}^2 \to \mathbb{R}^C$, a Gaussian derivative block is the concatenation of a layer computing Gaussian derivatives of each channel of f and a layer computing

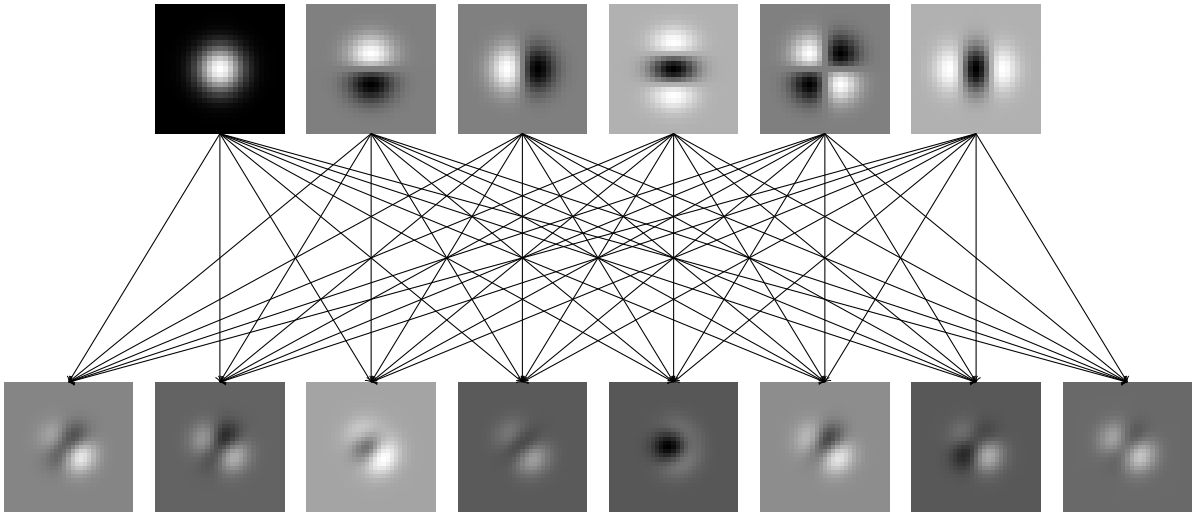


Figure 4.4: Diagram illustrating how one can obtain filters as linear combinations of Gaussian derivatives.

C' linear combinations of those derivatives, which can be computed as 1×1 convolutions. A Gaussian derivative block is illustrated in Figure 4.4. The output of a Gaussian derivative block is a signal $f' : \mathbb{Z}^2 \to \mathbb{R}^{C'}$ given by, for $o \in \{1, \dots, C'\}$

$$f'_o = \gamma \left(\sum_{i=1}^{C} f_i * h_{i,o} \right) \tag{4.38}$$

where for every i, o $h_{i,o}$ is a linear combination of Gaussian derivatives of scale σ like (4.34) and γ is a non-linearity function such as ReLU.

Gaussian filters are already SE(2)-equivariant, so their composition with a differential invariant yields a SE(2)-equivariant operator. Moreover the Gaussian filters have the property of smoothing a discrete signal. These properties motivate the use of Gaussian derivatives as the first part of each SE2DIN block.

4.5.2 SE2DIN Block

From the derivatives, we compute the invariantizations of the derivatives up to order n, $(\mathcal{I}_{i,j})_{0 \leq i+j \leq n}$. As any differential invariant $\mathcal{I}$ of order n is a functional combination $\mathcal{I} = F((\mathcal{I}_{i,j})_{0 \leq i+j \leq n})$ (Olver, 2007), a neural network with one hidden layer and taking as input $(\mathcal{I}_{i,j})_{0 \leq i+j \leq n}$, can approximate any differential invariant of order n, according to the Theorem of Universal Approximation (Leshno et al., 1993). In practice we compute a l-layer densely connected network, with $l \geq 2$, applied at every spatial point $(x, y) \in \Omega$ which takes as input the vector $(\mathcal{I}_{i,j})_{0 \leq i+j \leq n}$. Computationally this is equivalent to a series of l 1×1 convolutions with pointwise activation functions applied to the multi-channel image with values $A(x, y) = (\mathcal{I}_{i,j})_{0 \leq i+j \leq n}$. The output of the first 1×1 convolution is equivalent to the application of an adaptive filter that is rotated according to the gradient at each point of the image, as illustrated in Figure 4.7. We visually compare the effect of a rotation on the input of a CNN vs a rotation on the input of a SE2DIN block in Figure 4.6.

As the moving frame was derived assuming $\nabla u \neq \vec{0}$, $\mathcal{I}_{ij}(x,y)$ is not defined for some i,j at (x,y) when $\nabla u(x,y) = \vec{0}$. Therefore, we introduce the normalized invariants $\bar{\mathcal{I}}_{i,j}(x,y)$ given by

$$\begin{aligned}\bar{\mathcal{I}}_{i,j}(x,y) &= \sum_{m=0}^{n}\left(\sum_{l=\max\{0,m-i\}}^{\min\{m,j\}}\binom{i}{m-l}\binom{j}{l}(-1)^l u_x^{m-2l+j}(x_0,y_0)u_y^{2l-m+i}(x_0,y_0)\right)\frac{\partial^n}{\partial x^m \partial y^{n-m}}u(x_0,y_0)\\ &= \lim_{(x_0,y_0)\to(x,y)}\|\nabla u(x_0,y_0)\|^n \mathcal{I}_{i,j}(x,y).\end{aligned} \tag{4.39}$$

They are still invariants as they are the limit of a function of invariants and are well defined when $\nabla u(x,y) = 0$ as they are a polynomial expression in these values. Moreover the original invariants can be recovered with functional combinations, specifically, because $\bar{\mathcal{I}}_{10} = \|\nabla u\|^2$, then $\mathcal{I}_{ij} = \frac{1}{\sqrt{\bar{\mathcal{I}}_{10}^{i+j}}}\bar{\mathcal{I}}_{ij}$ therefore the normalized invariants form a generating set.

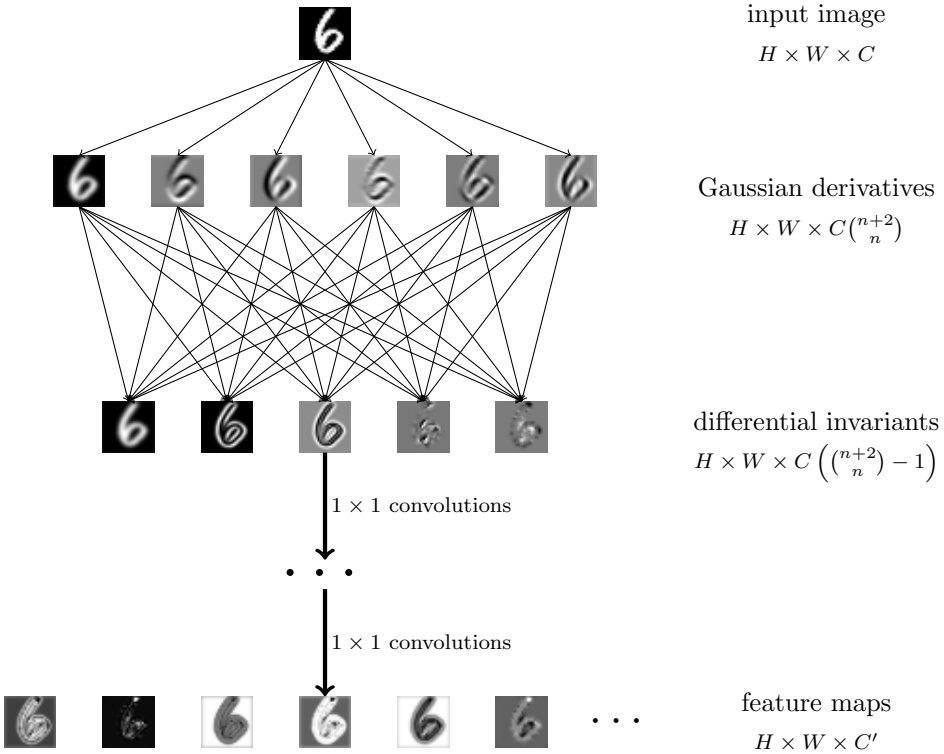
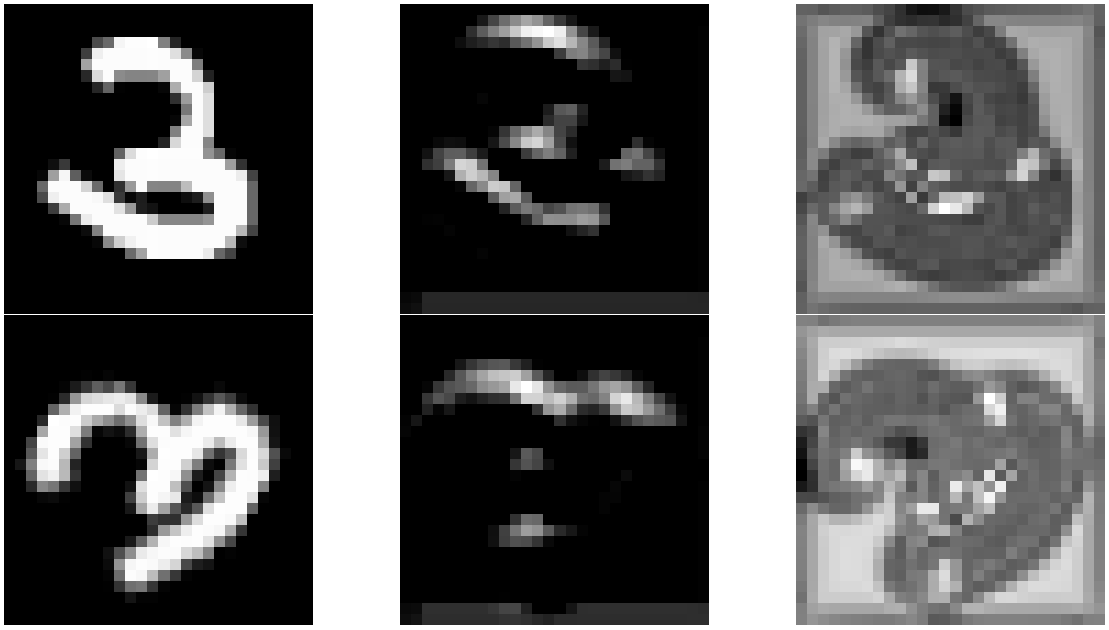


Figure 4.5: Illustration of a SE2DIN block applied to a MNIST-Rot image. Convolutions are followed by batch normalization and activation function. It takes as input an image of height H, width W and C channels and outputs an image with the same spatial dimensions and C' channels.

Complexity Analysis

Assuming that the input feature maps have height H, width W and have C feature maps we can estimate the cost of computing the differential invariants up to order n by this method. Using either Gaussian derivatives or finite differences, each derivative can be computed as a separable filter, let us assume that all filters are of length $K \times K$, then computing all of the Gaussian



(a) Input (b) CNN (c) SE2DIN block

Figure 4.6: Example of the output of a CNN layer and a SE2DIN block given an input and its rotation. We can see that the SE2DIN block output rotates as the input is rotated. Images made using the models from Section 4.6.

derivatives up to order n takes $2HWC\binom{n+2}{n}K$ multiplications and $2HWC\binom{n+2}{n}(K-1)$ sums, resulting in a total of $4HWC\binom{n+2}{n}(K-\frac{1}{2})$ flops. If the derivatives are already computed, computing (4.29) for an invariant of order $m < n$ and on one feature map and one input point takes $O(m)$ flops, hence, for all invariants it can be computed in $O(n^3)$ flops. Doing it for each feature map and each input point gives us $O(HWCn^3)$ flops. The 1×1 convolutions and activations combined take $O\left(\left(\binom{n+2}{n}Ck + lk^2\right)HW\right)$ flops where $k = \max_{i=1\dots l} k_i$ is the maximum number of feature maps between the layers.

The cubic dependency on n may seem like a significant drawback but in practice the value of n is kept small as it was shown that a value as low as 4 of finite differences filters can represent any 3×3 convolutional filter (Ruthotto and Haber, 2020). Overall the computation of the fundamental differential invariants takes $O\left(HWC\left(\binom{n+2}{n}K + n^3\right)\right)$ flops which is akin to a depthwise convolution if we assume n and K are fixed.

4.5.3 Rotation-Equivariant Pooling

Pooling is a standard layer in many network architectures and can be obtained by

$$P = D_t \circ \varphi. \tag{4.40}$$

where an operator φ is composed with a downsampling operator D_t, $t > 1$, defined, for $u : \mathbb{Z}^d \to \mathbb{R}^C$, where is a grid, by

$$\forall \mathbf{x} \in \mathbb{Z}^d, \quad D_t(u)(\mathbf{x}) = u(t\mathbf{x}). \tag{4.41}$$

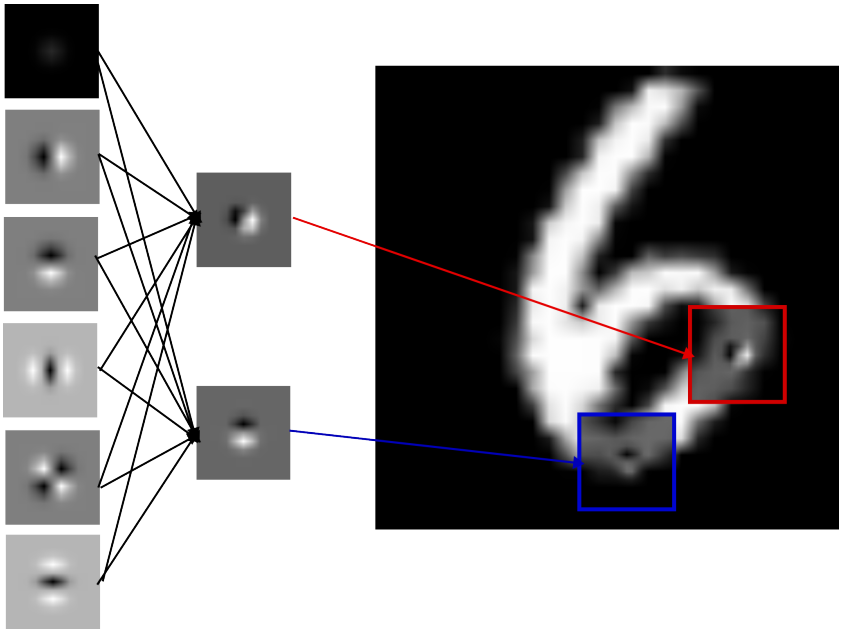


Figure 4.7: Illustration of how a single rotated filter is applied. The rotated filter is the output of the first 1×1 convolution in the SE2DIN block, i.e. a linear combination of the fundamental invariants. A linear combination of the the fundamental invariants can also be identified as the rotation of a linear combination of a local rotation of the Gaussian derivatives (where the rotation angle is determined by the gradient at each point), but instead of computing the rotations by interpolation they are computed by a non-linear expression of the convolutions of the image by the Gaussian derivatives. This examples shows the link between the differential invariants blocks and the initial goal of applying rotated filters illustrated in Figure 4.1.

In order for a network containing a pooling layer to be rotation equivariant, the pooling layer must be rotation equivariant. A straightforward answer is to substitute the maximum in a square neighborhood of the max-pooling by the maximum in a circular neighborhood.

In practice, however, images are defined on a grid $\Omega \subseteq \mathbb{Z}^d$. In that case, the downsampling D_t of (4.41), but with $t \in \mathbb{N}$, can induce a large deviation from perfect equivariance. Indeed, using $t = 2$ and for images with an even dimension, even for 90° rotations the downsampling D_2 is not rotation equivariant due to the grid shape of the images as illustrated in Figure 4.8. One way to achieve better results is to ensure that images have odd sizes in both dimensions, because in that case the downsampling by D_2 commutes with rotations by 90°. An alternative that works for any size is to, given a function u and for every angle multiple of 90° θ, to rotate u by θ, downsample and then rotate by $-\theta$, and then compute an aggregation function of those operators, that is

$$\bar{D}_t(u)(\mathbf{x}) = \mathrm{Aggr}(R_{i\frac{-\pi}{2}} \cdot D_t(R_{i\frac{\pi}{2}} \cdot u)(\mathbf{x})_{i=0,1,2,3}) \tag{4.42}$$

where Aggr is some permutation invariant function, like the average or the max, and R_θ denotes the rotation of angle θ. $\bar{D}_t$ is equivariant to rotations by $k\frac{\pi}{2}$, $k = 0, 1, 2, 3$ because the output of $\bar{D}_t$ given $R_{k\frac{\pi}{2}} \cdot u$ at $R_{k\frac{\pi}{2}} \cdot \mathbf{x}$ is the aggregation of the same four values as the output of $\bar{D}_t$ given u and $\mathbf{x}$.

Alternatively, if we choose to compute a discrete approximation of R_θ for angles not multiple

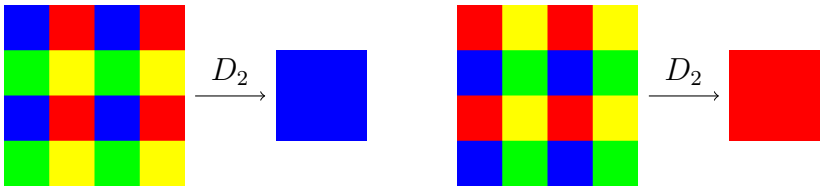


Figure 4.8: Subsampling of a 4×4 image and its rotation by 90° by a factor of 2. The position $(0, 0)$ is assumed to be the pixel at the top-left corner. Of course in real scenarios images have a much bigger dimension and do not necessarily have such high frequency information, but the subsampling will still generate small errors to equivariance that can propagate to the next layers. In the experimental section it will be shown that the choice of pooling does have a significant impact in practice.

of $\frac{\pi}{2}$, then we can use a smaller angle than $\frac{\pi}{2}$ in (4.42). In the following, we fix Aggr = max and we refer to (4.42) as P4Strides. In the case $t = 2$, $\bar{D}_2$ is equivalent to a simple subsampling for images with odd dimensions and to a 2×2 max-pooling for images with even dimension.

4.5.4 Architecture

We define the SE(2) Differential Invariants Network (SE2DINNet) as a network constructed by SE2DIN blocks applied with residual connections like the ResNet (He et al., 2016). In that way we compute a SE2DIN block from current feature map and add it to the feature map. The SE2DINNet is the repeated application of the previous step, illustrated in Figure 4.9.

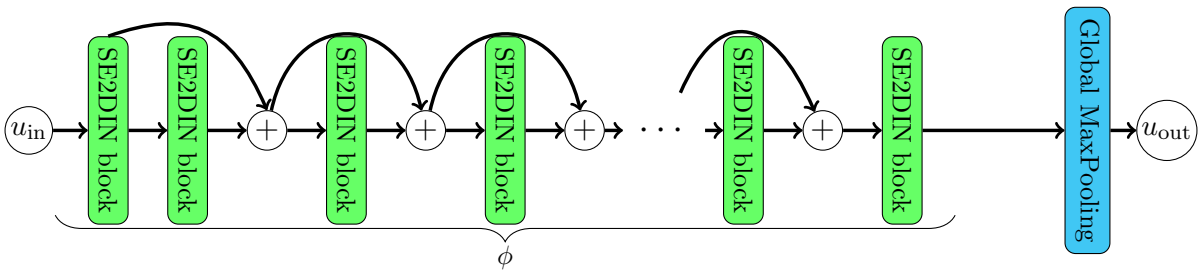


Figure 4.9: Illustration of a basic SE2DINNet. Global maxpooling renders the architecture invariant. All blocks except the first and last have the same number of input channels C and filters C' and follow the structure of a numerical scheme for solving a PDE.

Given the interpretation of SE2DIN blocks as differential invariants, a SE2DINNet is viewed as a numerical scheme to solve a time-varying PDE obtained by applying an analogous to the Finite Difference method to solve

$$u_t = F_t((\mathcal{I}_{i,j})_{0 \leq i+j \leq n}). \tag{4.43}$$

Namely, given the input u_0, we obtain $u_1, u_2, \ldots, u_L$, where u_l is the output of the $l - th$ layer and the solution of (4.43), at the time $l\Delta t$, with Δt the temporal step. That is,

$$u_l = u_{l-1} + \Delta t F_l((\mathcal{I}^{\sigma}_{i,j}(u_{l-1}))_{0 \leq i+j \leq n})), \tag{4.44}$$

where $\mathcal{I}^{\sigma}_{i,j}(u_{l-1})$ are the differential invariants computed from the Gaussian derivatives of u_{l-1}, i.e. the differential invariants of a Gaussian filtered u_{l-1}.

We restrict the temporal step to $\Delta t = 1$. The network in Figure 4.9 before global max-pooling, ϕ is SE(2)-equivariant, *i.e.* $\phi((R_\theta, \mathbf{v}) \cdot u) = (R_\theta, \mathbf{v}) \cdot \phi(u)$ for $u : \mathbb{R}^2 \to \mathbb{R}^{C_0}$.

4.6 Experiments

4.6.1 Ablation Study

We begin by individually examining the effect of different components on the SE2DINNet. There are two components which are going to be examinated: (i) the use of Gaussian derivatives and (ii) the choice of pooling functions. In order to test the parameters we use MNIST (LeCun et al., 2010) and Fashion MNIST (Xiao et al., 2017). Models are trained on the original training set of those datasets, but in order to evaluate the rotation invariance of the models we test at rotated test sets. The test sets are all rotations of the original test sets with angles given by $i\frac{\pi}{12}$ $0 \leq i \leq 23$.

Let ϕ denote the features of the network, i.e. the result of the global max-pooling layer and let ϕ' denote the output of the network, i.e. the prediction into one of the possible labels. Also let $u_1, \ldots, u_N$ be the images of the test set. As evaluation metrics we use

- the overall accuracy, i.e. the proportion of correctly classified images for each angle
- the consistency i.e. the probability that an image and its rotation result in the same class,

$$\text{Consistency}(\phi', \theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{X}(\phi'(u_i), \phi'(R_\theta \cdot u_i)) \tag{4.45}$$

  where $\mathcal{X}(A, B) = 1$ if $A = B$ and 0 otherwise.

- the Mean Square Error (MSE)

$$\text{MSE}(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\phi(u_i) - \phi(R_\theta \cdot u_i)\|^2}{\|\phi(u_i)\|^2} \tag{4.46}$$

- the cosine similarity

$$\text{CosineSimilarity}(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{\langle \phi(u_i), \phi(R_\theta \cdot u_i) \rangle}{\|\phi(u_i)\|^2 \|\phi(R_\theta \cdot u_i)\|^2}. \tag{4.47}$$

Gaussian Derivatives

Different methods of computing the derivative layers are tested in this section: finite difference method, Gaussian derivatives with fixed σ and Gaussian derivatives with increasing σ with respect to layer depth. In both cases where Gaussian derivatives are applied, σ starts at $\sigma = 1$

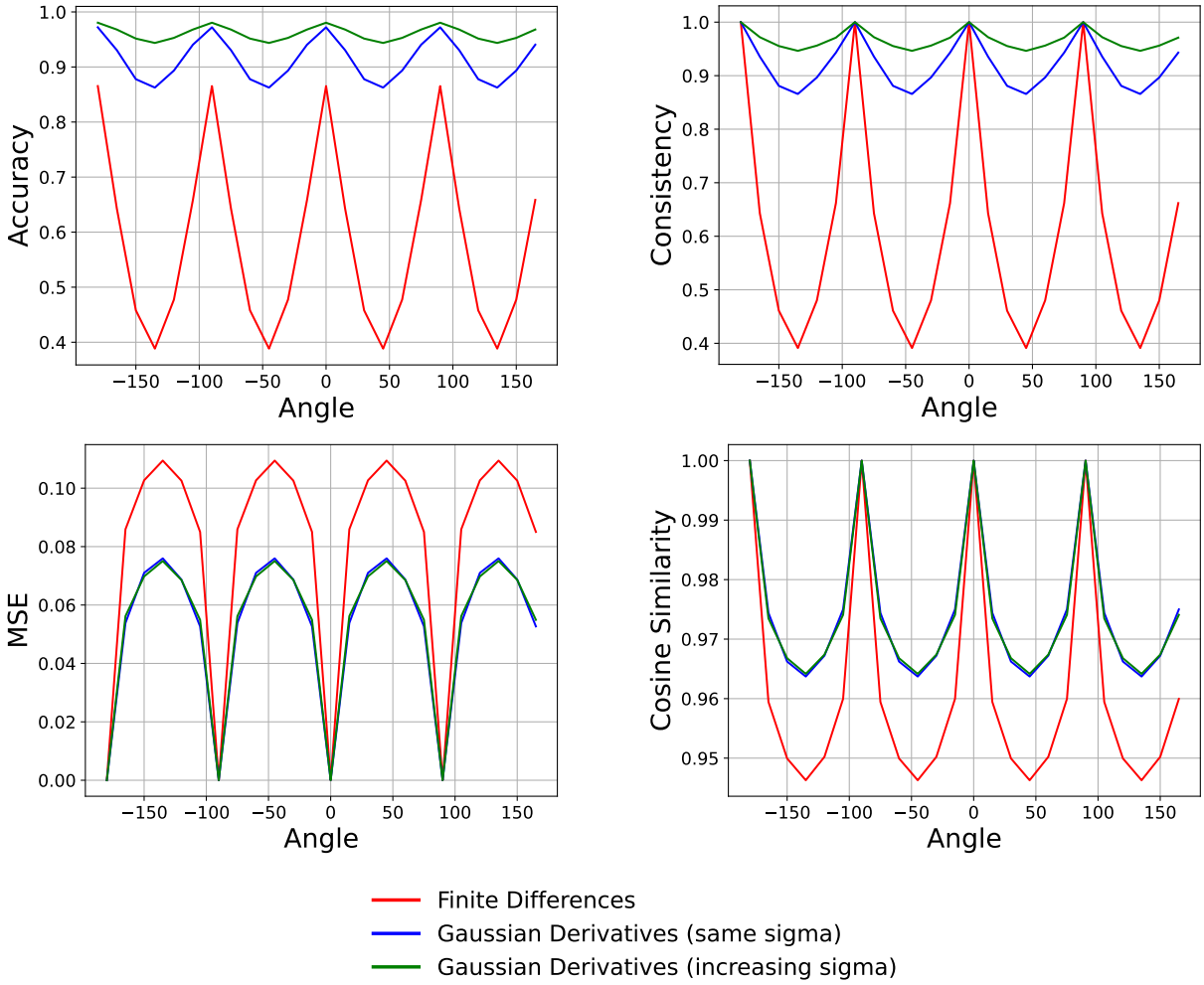


Figure 4.10: Evaluation metrics comparing methods on MNIST using Gaussian derivatives and finite differences to compute the derivatives. Models are trained on the MNIST training set and tested in rotated versions of the test set with angles (in degrees) varying in the interval $[-180, 180)$.

but in the increasing case it is increased by a factor $\sigma_{\text{mult}} = \sqrt{2}$ each layer. Keep in mind that the purpose here is not to find the best value of σ or its proportion, but to find the effect of the method of computing derivatives on the invariance.

Figures 4.10 and 4.11 contain evaluation metrics for each of the tested methods. Firstly we notice that all models have a periodicity of 90° on every metric. This is due to the fact that everything is built for a square grid. In that case, numerical derivatives are equivariant to rotations by 90° of the image. When rotations deviate from that margin, however, finite difference methods are much more degraded compared to the Gaussian derivative. This is evidence of the Gaussian derivatives robustness to rotations.

The results of the finite difference models also show a drastic accuracy difference with respect to the methods using Gaussian derivatives when evaluated in the unrotated test sets. This is partially due to the fact that the Gaussian derivatives have a larger receptive field than

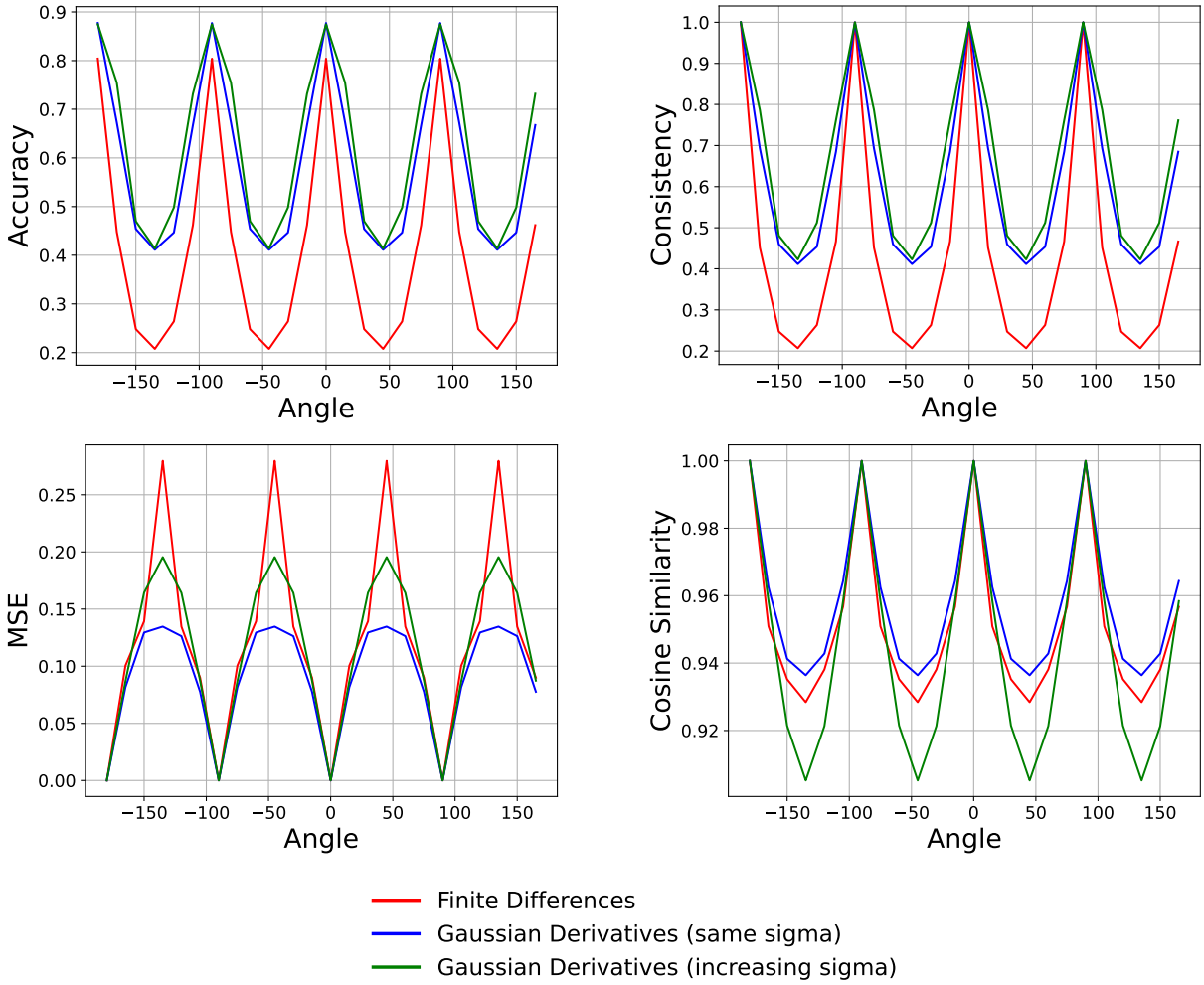


Figure 4.11: Evaluation metrics comparing methods on Fashion-MNIST using Gaussian derivatives and finite differences to compute the derivatives. Similarly to the results in Figure 4.10, models are trained on the Fashion-MNIST training set and tested in rotated versions of the test set.

the 3×3 finite difference stencils which, with only five layers and no pooling in between, is only a square of side 11. Figures 4.12 and 4.13 show the same metrics for networks applying different kinds of subsampling, effectively increasing the receptive field sizes of the networks. There we can see that even with the presence of pooling, the finite difference method falls short of the Gaussian derivatives in terms of equivariance. MSE plots in those figures also suggest numerical instability of the finite difference method.

We can also compare the filters when placed at regions with different gradient angles. Indeed, for each point in the domain, the output value of the composition of the invariants and the first 1×1 convolution in the SE2DIN block is given by the scalar product of the input image and a certain filter. The process is illustrated in Figure 4.7. In Figure 4.14 we perform an example where we take one of the rotated filters of the first layer of a Gaussian-derivative-based SE2DINNet and a finite-difference-based one and we compute the filter that would be

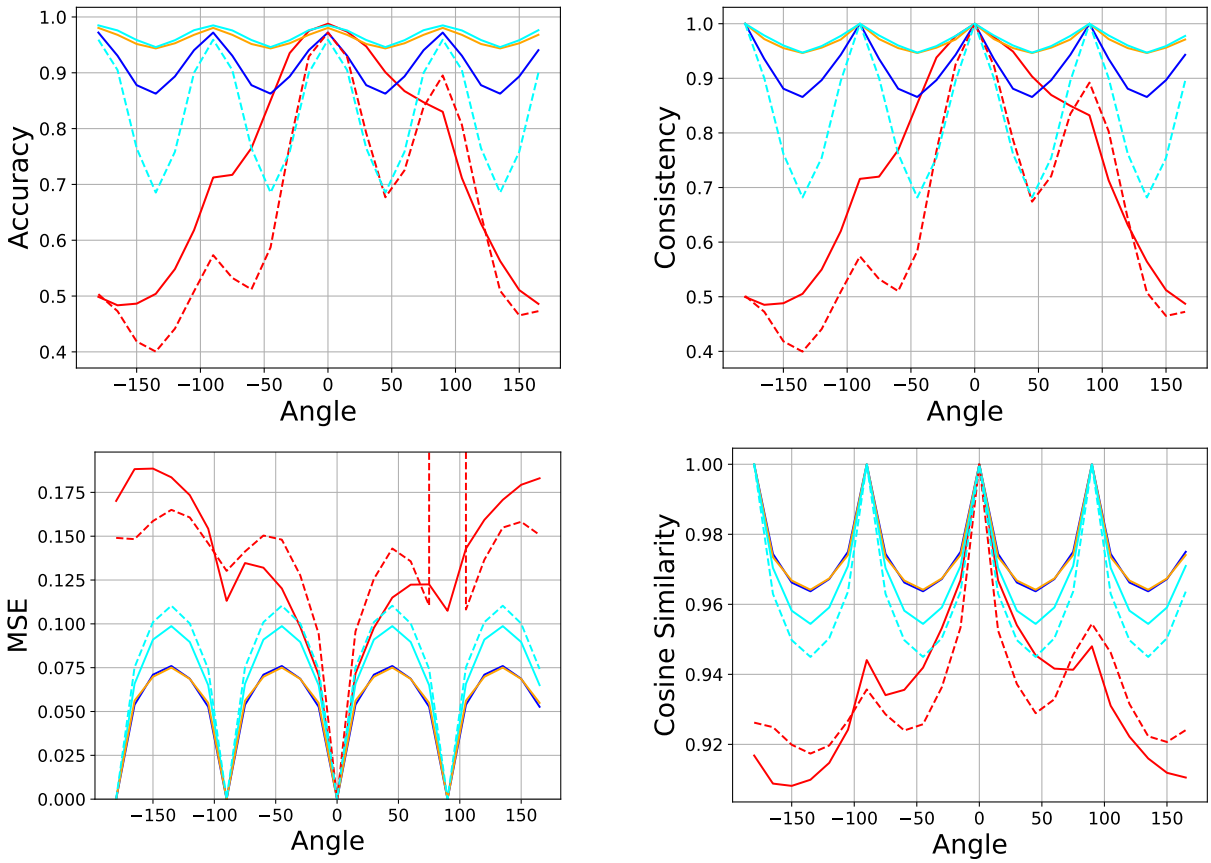


Figure 4.12: Evaluation metrics comparing methods using different kinds of subsampling schemes on MNIST.

applied in a zone where the gradient is a vector in the unit circle at angles $i\frac{\pi}{8}$ $i = 0, 1, 2, 3, 4$. The rotated finite difference filters in Figure 4.14(a) show very abrupt changes between consecutive angles while the Gaussian rotated filters in Figure 4.14(b) have a smooth transition, further justifying the choice of Gaussian derivatives as the method of approximating derivatives.

Pooling

In order to evaluate the contribution of choosing the right pooling functions we test the models with different pooling protocols: no subsampling, strided operators and P4Strides. In the no-subsampling case we also compare Gaussian derivatives with constant or increasing σ with the same values of σ as the previous subsection.

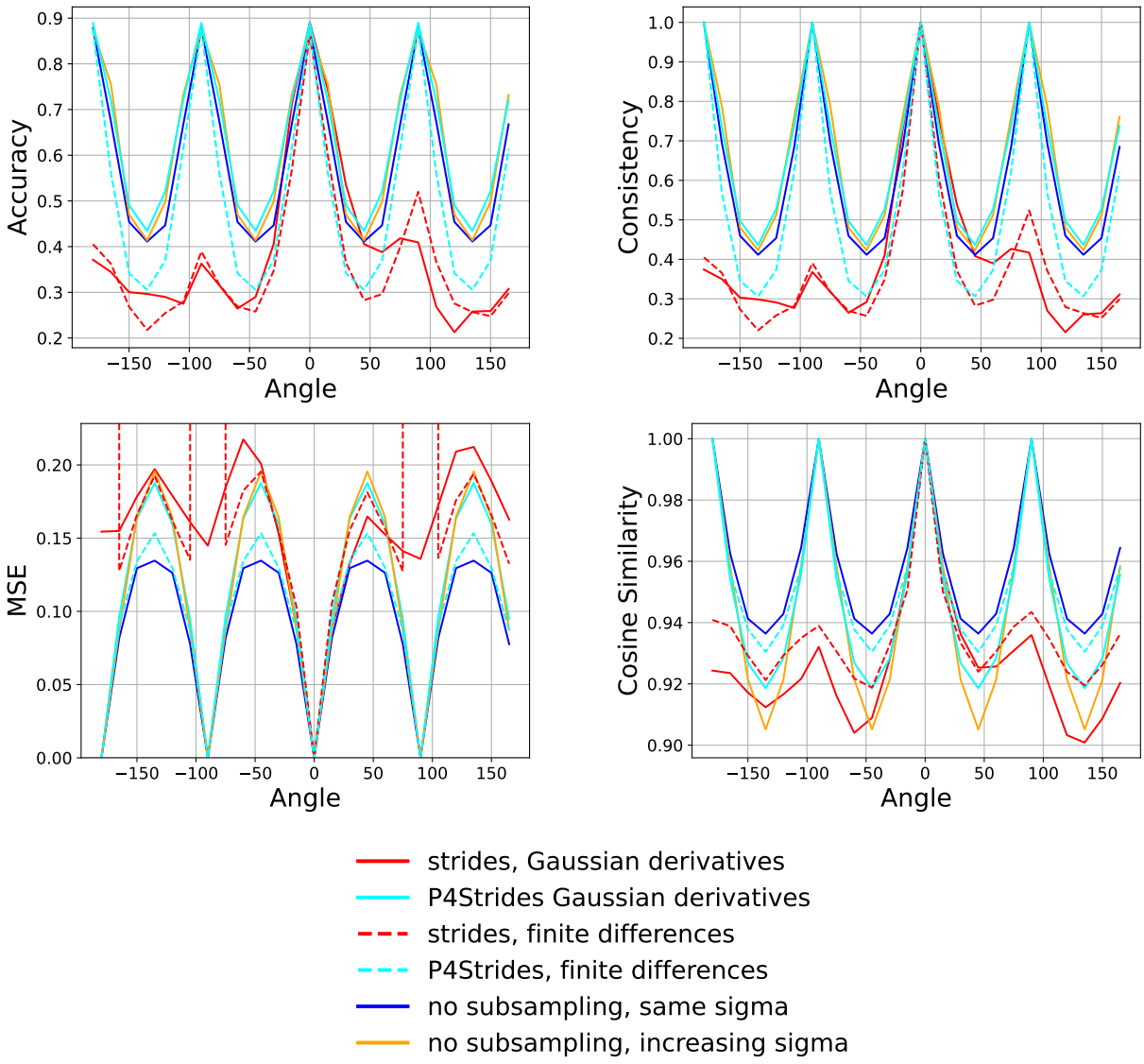
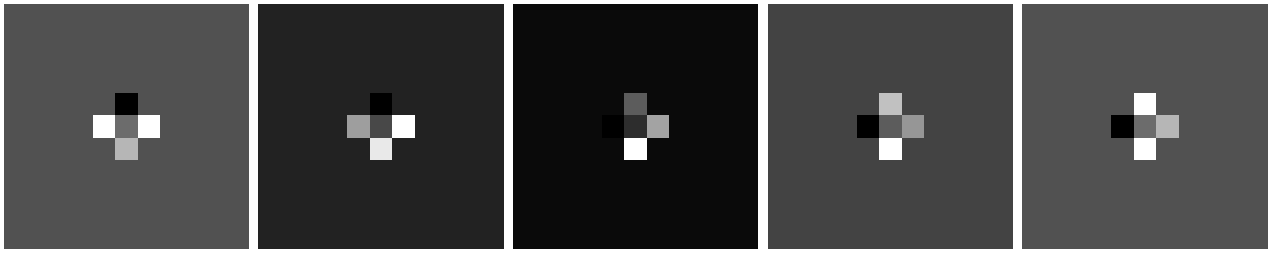
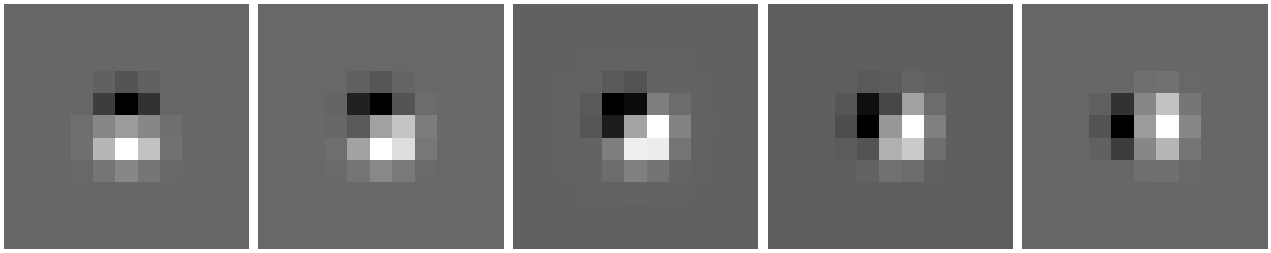


Figure 4.13: Evaluation metrics comparing methods using different kinds of subsampling schemes on MNIST.

Looking at results from Figures 4.12 and 4.13 it is interesting to notice how much the choice of pooling affects equivariance. Even though the difference between strides and P4Strides is small in paper, it completely alters the behaviour of the network on unseen angles, even for datasets obtained by rotating the test set by multiples of 90°. Interestingly, when the distance and similarity metrics are computed for untrained models, we do not obtain the same kinds of results. As can be observed in Figure 4.15, even naive strided SE2DIN blocks have a better invariance according to those metrics than all trained models and corrected strides have an almost perfect equivariance, suggesting that optimization (specifically on a single-orientation dataset) pulls the model farther from equivariance. Looking again at results from Figure 4.12 and 4.13, it is interesting to see that the model using Gaussian derivatives and P4Strides



(a) Filters from finite difference stencils.



(b) Filters from Gaussian derivative stencils.

Figure 4.14: Some examples of filters obtained by linear combination of differential invariants in first SE2DIN layer of a network, by computing the rotation of the derivatives from equations (4.24) and (4.25). Rotation angles are equally spaced in the interval $[0, \frac{\pi}{2}]$.

method compares well with the method without subsampling but with increasing σ, and it has the advantage of reducing computation times for subsequent layers. Based on that observation P4Strides are used when subsampling is needed in SE2DINNets.

4.6.2 MNIST-Rot

The MNIST-Rot dataset (Larochelle et al., 2007) was constructed by randomly rotating by an angle in $[\pi, -\pi)$ each image from the MNIST 12k dataset (Larochelle et al., 2007), which itself is obtained by selecting 12000 images from the MNIST dataset for training/validation and 50000 for testing. The dataset is widely used for benchmarking of rotation invariant classification models. We divide the MNIST-Rot training set into 10000 images for training and 2000 for validation. Differently from the data of the previous section, in the MNIST-Rot dataset every image, training and test, is assigned a new, fixed, orientation.

The architecture used for this task is shown in Figure 4.16. As for the orders of differentiation, we use two, three and four. The architecture is based on ResNet18 (He et al., 2016) but with the inverted bottlenecks from ConvNext (Liu et al., 2022), in other words, the SE2DIN block we use is constituted of a layer computing the differential invariants (which would be analogous to the 3×3 convolution in the ResNet), followed by two 1×1 convolutions, the first having four times the output size of the second, as shown in Figure 4.16(b). In the normal ResNet, 3×3 layer is the middle one, but since we have to compute a functional approximation of differential invariants we put the fundamental invariants in the first layer of the block and the other two are used to compute the function approximation. Moreover, the layer with more feature maps is usually at the end of the block, but we put it in the second place in order to require less computations of the fundamental invariants layer, the one which is more

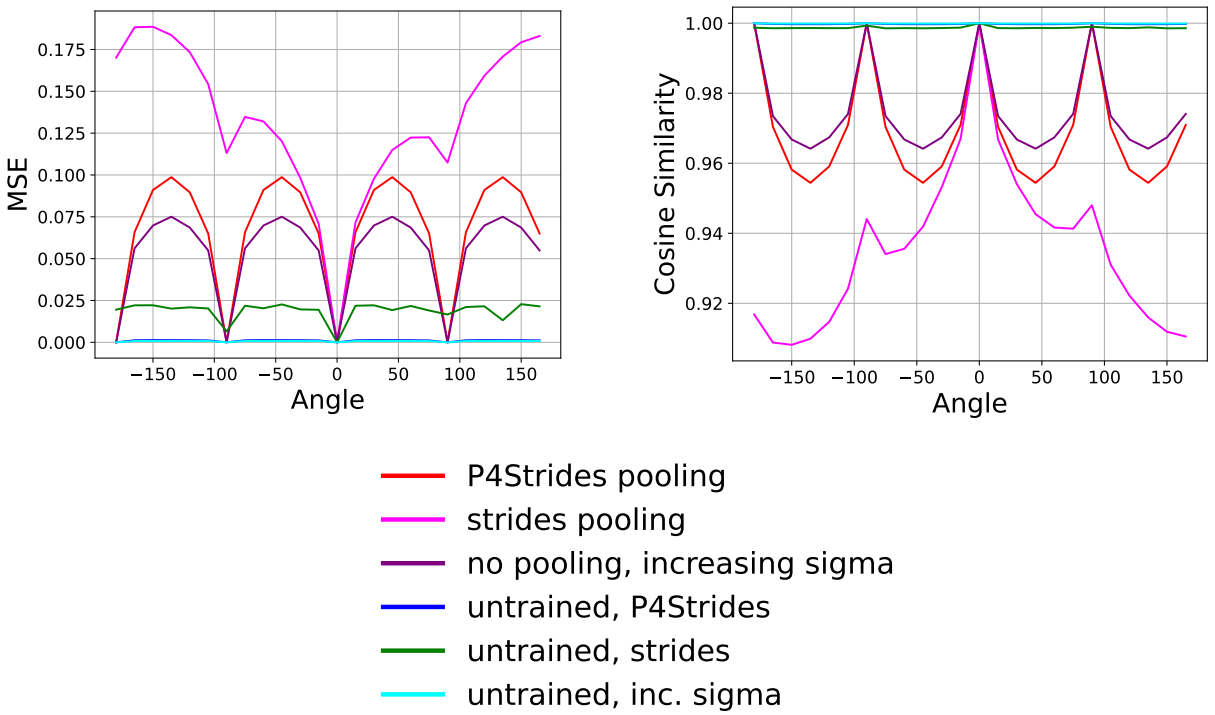


Figure 4.15: Evaluation metrics of untrained and trained models on MNIST and its rotated versions. Because untrained models are being tested, only measure distances between features are computed, and not accuracy and consistency metrics.

computationally expensive.

For training, we used the AdamW optimizer (Loshchilov and Hutter, 2017), i.e. the Adam (Kingma and Ba, 2014) optimizer with decoupled weight decay. We used a starting learning rate of 10^{-3} and weight decay values of 10^{-4} for the order two model and $2.5 \cdot 10^{-4}$ for the orders three and four models. The models were trained for 300 epochs with a batch size of 128 and the learning rate and weight decay are divided by 10 every 100 epochs.

Our results, as well as those of the state-of-the-art methods are shown in Table 4.2. We can see that the proposed method is still below the performance of the state-of-the-art but we can see that the method still achieves a good performance nonetheless, compared with the models that do not use data augmentation. Our model also has the advantage of being lightweight, as the number of floating point operations that it takes to compute the outputs is similar to that of a CNN with the same number of trainable parameters.

In the course of training the SE2DINNets, the main issue was ensuring numerical stability. Because the invariants form non-linear expressions which may cause the variance of the output to differ greatly from that of the input, it can in turn cause the gradients of the neural network computed by backpropagation to explode (Glorot and Bengio, 2010). In this experiment we used activity regularization (using norm l_2 and a value of 10^{-5}) in the computation of the invariants and found empirically that it helps dealing with these problems, but in Chapter 5 we propose a solution that is based on architectural design is much more effective to mitigate these issues.

Method	Test error(%)	#params
H-Net (Worrall et al., 2017)	1.69	33k
P4CNN (Cohen and Welling, 2016a)	2.84	25k
Z2CNN (Cohen and Welling, 2016a)	5.03	22k
PDO-eConv (Shen et al., 2020)	1.87	26k
PDO-eConv † (Shen et al., 2020)	0.709	650k
RotEqNet (Marcos et al., 2017)	1.09	100k
RotEqNet ‡ (Marcos et al., 2017)	1.01	100k
E2CNN † (Weiler and Cesa, 2019)	0.716	-
E2CNN † (Weiler and Cesa, 2019)	**0.682**	-
SFCNN (Weiler et al., 2018b)	0.880	6.5M
SFCNN † (Weiler et al., 2018b)	0.714	6.5M
SE2DINNet, order two (Ours)	1.21	380k
SE2DINNet, order three (Ours)	1.27	630k
SE2DINNet, order four (Ours)	1.45	920k

Table 4.2: Mean accuracies on the MNIST-Rot dataset. Legend: † - train time augmentation, ‡ - test time augmentation.

4.7 Conclusions

In this chapter we reviewed the method of moving frames used to obtain differential invariants, and applied it to derive differential invariants of SE(2) on two-dimensional signals. The computed differential invariants, called fundamental invariants, of a certain order n form a set that can generate all other invariants of order n through functional combination, and conversely every functional combination of these invariants is an invariant. These properties allow for the definition of a neural network layer, called SE2DIN block, that takes as input a stack of feature maps, computes their Gaussian derivatives, which are used to obtain fundamental differential invariants and combine the latter through a pixel-wise multi-layer perceptron , allowing us to approximate arbitrary differential invariant of order n.

The neural network, called SE2DINNet, obtained from the differential invariants was implemented for discrete images using Gaussian derivatives, as they are theoretically and empirically more adequate to this task than finite difference filters. This network is lightweight computation-wise, as computing the invariants has a computational complexity similar to that of a depthwise convolution.

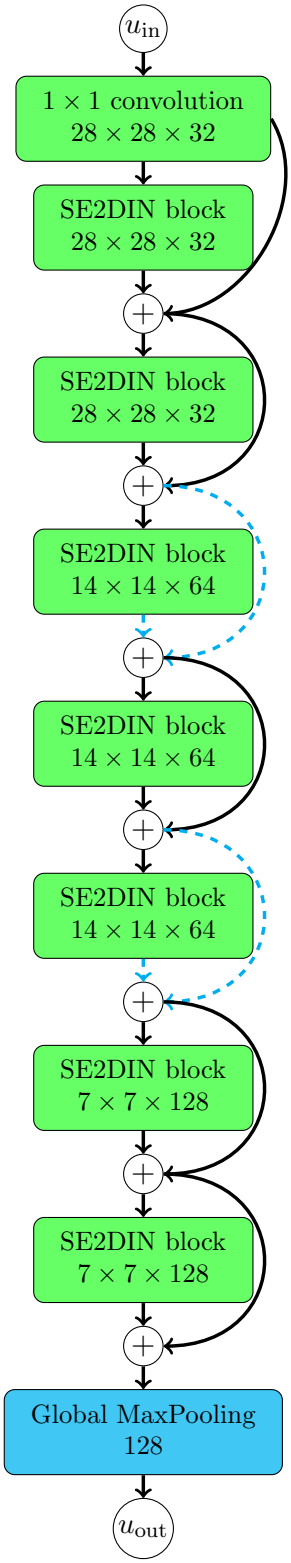
The performance of these networks in a context where invariance matters was evaluated on the MNIST and FashionMNIST datasets by training on the normal training set and testing on rotated test sets. Moreover the models were tested on randomly rotated digits from the MNIST-Rot dataset, and achieved competitive results, even when compared to methods which use data augmentation.

4.8 Résumé en Français

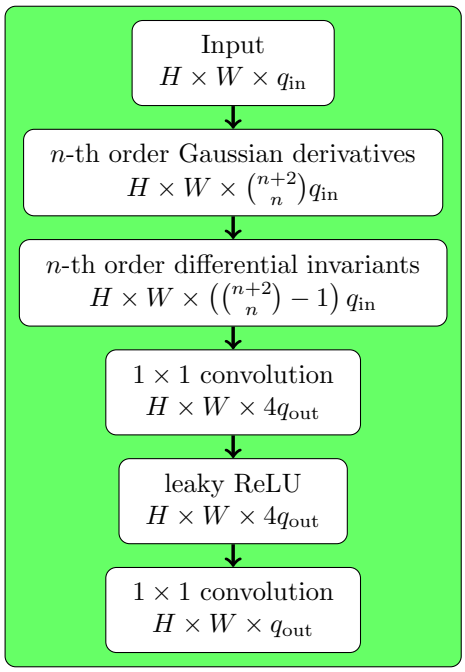
Ce Chapitre aborde la méthode du repère mobile, qui peut être utilisée pour obtenir des invariants différentiels. Nous avons appliqué la méthode pour dériver des invariants de l'action de SE(2), le groupe de Lie des rotations et des translations dans le plan, sur des signaux en deux dimensions. Les invariants différentiels calculés, appelés invariants fondamentaux, d'un certain degré n peuvent générer tous les autres invariants de degré n à partir de combinaisons fonctionnelles, et inversement toutes les combinaisons fonctionnelles de ces invariants sont invariants. Ces propriétés permettent la définition d'une couche de réseau de neurones, le bloc SE2DIN, qui calcule les dérivées Gaussiennes à partir des cartes de caractéristiques et utilise ces dérivées pour obtenir des invariants fondamentaux, qui sont à leur tour combinés par un Perceptron Multi-Couche (MLP) appliqué pixel par pixel. Cette formulation nous permet d'approximer un invariant différentiel arbitraire de degré n.

Le réseau de neurones, appelé SE2DINNet, obtenu à partir des invariants différentiels a été implémenté pour la classification d'images discrètes en utilisant les dérivées Gaussiennes, car elles sont en théorie et en pratique plus adaptées à cette tâche que les filtres de différences finies. Ce réseau peut être implémenté de manière efficace, car la complexité du calcul des invariants est similaire à la complexité d'une convolution *depthwise*.

Nous avons évalué la performance de cette architecture de réseaux de neurones dans un contexte où l'invariance est importante, plus spécifiquement dans les bases de données MNIST et FashionMNIST. Nous avons utilisé un protocole ou l'apprentissage est fait sur l'ensemble d'entraînement normal, mais le modèle est testé sur des images tournées. De plus, les modèles étaient testés sur la base de données de chiffres écrits à la main et tournés MNSIT-Rot (Larochelle et al., 2007), et ils ont obtenu des résultats compétitifs, même quand comparés à des méthodes qui utilisent l'augmentation de données.



(a) SE2DINNet



(b) SE2DIN block

subsampling + 1×1 convolution

Figure 4.16: SE2DINNet used for MNIST-Rot.

Chapter 5

Single-Moving-Frame Neural Networks

5.1 Introduction

The method based on differential invariants, presented in the previous chapter, defines an alternative to the group convolution to compute equivariant networks. In this chapter we build on this method and propose an alternative way to build equivariant neural networks based on the method of moving frames. Indeed, the ratios of polynomial functions involved in the computation of differential invariants can make it numerically unstable. In order to improve that, we define an equivariant network that only estimates one moving frame, instead of one per equivariant block, as before. This new approach is embedded in 3D CNNs, where one moving frame is computed from the Gaussian n-jets of the input image, and used to make each network block equivariant. This produces novel equivariant architectures for SE(3), that we call *SE(3)-Moving Frames* (SE3MovFNet), and which we apply to volumetric data. Empirically, we show that a SE3MovFNet improves the performance of competitive CNNs on a collection of datasets for 3D medical image classification.

The chapter is organized as follows. We discuss related work in the literature in Section 5.2. In Section 5.3 we introduce an alternate formulation of equivariant neural networks based on moving frames, in contrast to Chapter 4 we compute the moving frame only from the input function. In Section 5.4 we derive a moving frame in SE(3) and in Section 5.5 we introduce the SE3MovFNet based on the moving frames method. In Section 5.6 we validate de SE3MovFNet in a task of medical volume classification and overperform most of the benchmarks. We end the chapter with concluding remarks in Section 5.7.

5.2 Related Work

In the literature of group-equivariant networks there exist many approaches to plane rotation-equivariant networks, for example, (Cohen and Welling, 2016a; Worrall et al., 2017; Weiler et al., 2018b). Also on 2D rotation-equivariant networks, some approaches are based on differential operators (Shen et al., 2020; Jenner and Weiler, 2022; Sangalli et al., 2022a). In particular, the approach we present in this chapter is an extension of the moving frames-based SE(2)-equivariant neural network introduced in the previous chapter, as well as in Sangalli et al. (2022a). In the domain of 3D CNNs, two of the possible data representations are point clouds

(i.e. the domain is a finite set of points in $\mathbb{R}^3$) and volumetric data (i.e. the domain is a grid $\Omega \in \mathbb{R}^3$). Many approaches that seek equivariance to space rotations are for CNNs that process point clouds (Thomas et al., 2018; Chen et al., 2021; Melnyk et al., 2021; Thomas, 2020).

Our work focuses on defining SE(3)-equivariant networks for data based on voxels, i.e., volumetric data. Some other approaches that aim to achieve this result are: Worrall and Brostow (2018), which achieves equivariance to a discrete subgroup of SO(3); Weiler et al. (2018a), which uses a steerable filter basis based on spherical harmonics to learn general SE(3)-equivariant filters, and Shen et al. (2022), which proceeds similarly using filters based on partial differential operators. Our approach relies on differential operators like Shen et al. (2022), but instead of using a steerable filter basis we apply a moving frame to invariantize the network. This consists in evaluating each neighborhood rotated by a matrix $P \in \mathrm{SO}(3)$, which depends on the neighborhood, and is computed in the first layer.

5.3 Fixed Moving Frame

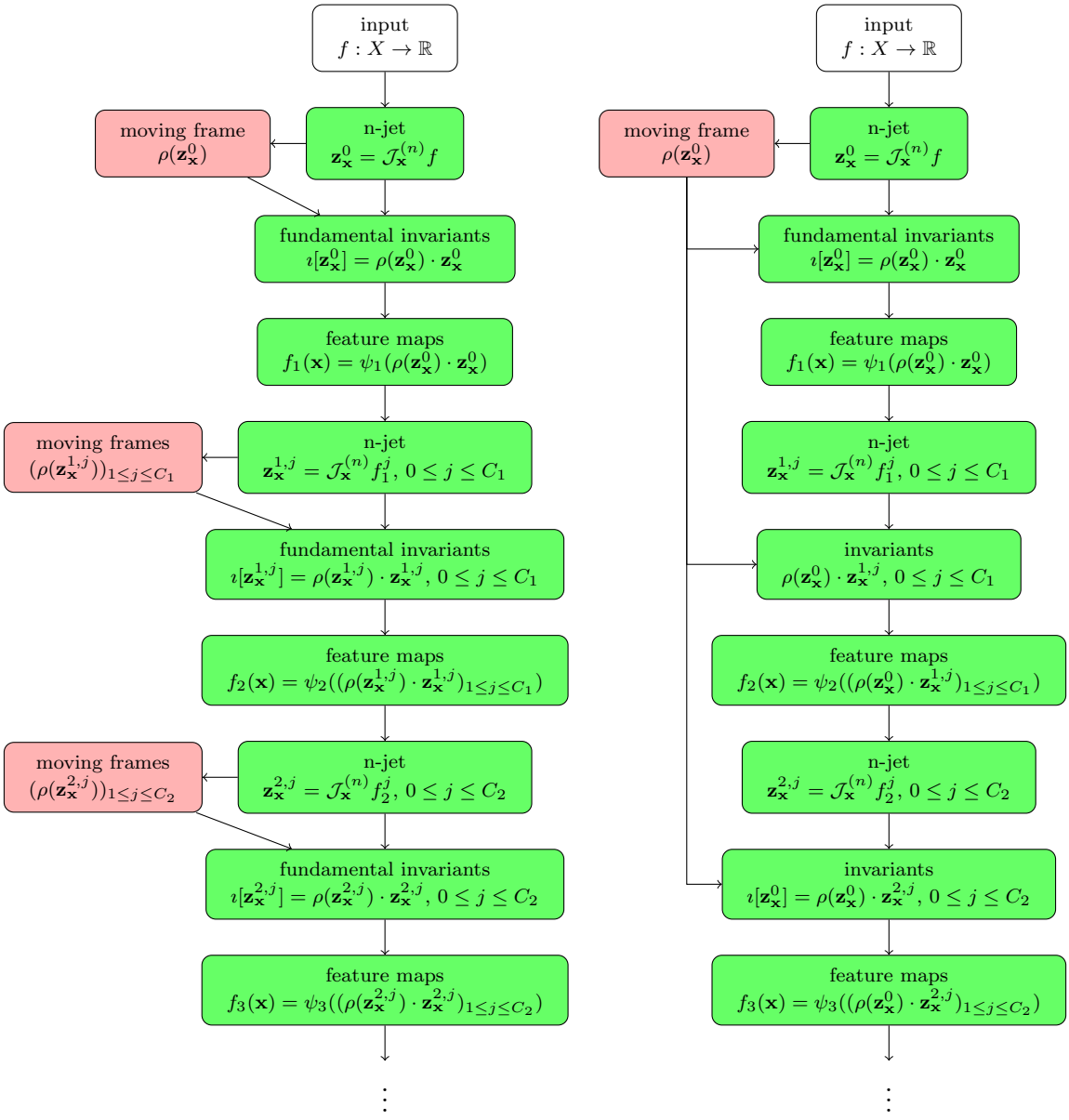
In Chapter 4 differential invariants were used to build equivariant layers of a neural network. The computation of the differential invariants done in (4.29) involves computing the prolonged action (4.28) and substituting the values of the moving frame on θ. Therefore the computation of a fundamental invariant at one specific point $\mathbf{x} \in \mathbb{R}^2$ $I_{ij}(\mathbf{x}) = I_{ij}(\mathbf{x}, f^{(n)}(\mathbf{x}))$ can be divided into three steps: the computation of the derivatives of f at $\mathbf{x}$, denoted $\mathcal{J}^{(n)}_{\mathbf{x}} f$, the computation of the moving frame $\rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f)$ from those derivatives, and the application of the moving frame to the derivatives $I(\mathbf{x}) = \rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f) \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f)$. A SE2DINNet, as defined in Chapter 4, is a serialization of these operators, as shown in Figure 5.1(a).

In in the context of neural networks, the method above has a significant disadvantage: the expression $\rho((\mathbf{x}, f^n(\mathbf{x})) \cdot (\mathbf{x}, f^n(\mathbf{x}))$ forms ratio of polynomials in the values of $f^{(n)}$ of potentially high order. High order polynomials in neural networks are not a good thing because during forward propagation the variance of the outputs differs from the variance of the inputs. A similar effect occurs during backpropagation. That in turn can cause the gradients to explode or vanish (Glorot and Bengio, 2010). In our experiments in Chapter 4, we added an activity regularizer to the layer that computes the invariants, which avoided in practice the numerical issues with the gradients, but this strategy was no longer efficient in our attempts of generalization to SE(3)-equivariant 3D networks.

Therefore, in the present chapter we propose an alternative relying on one fixed moving frame, but in order to introduce it let us first recapitulate the approach of Chapter 4, specifically Section 4.4.2, in the case of a 3D network equivariant to SE(3). In here we assume that we know a SE(3) moving frame for volumetric data. The moving frame will be explicitly derived in Section 5.4 using the second order derivatives. A two-layer neural network can be obtained using invariants of order two as follows:

1. Let the image $f \in C^\infty(X, Y) = C^\infty(\mathbb{R}^3, \mathbb{R})$ be the input to the network. First we compute for all $\mathbf{x}$,

$$\mathcal{J}^{(2)}_{\mathbf{x}} f = \left(f(\mathbf{x}), \frac{\partial f}{\partial x}(\mathbf{x}), \frac{\partial f}{\partial y}(\mathbf{x}), \frac{\partial^2 f}{\partial x^2}(\mathbf{x}), \frac{\partial^2 f}{\partial x \partial y}(\mathbf{x}), \frac{\partial^2 f}{\partial y^2}(\mathbf{x}) \right),$$



(a) SE2DINNet (b) Invariants from a fixed moving frame

Figure 5.1: Illustration of the difference of the approach of Chapter 4 and the approach we propose in this chapter where we compute a moving frame at the first layer and use it for subsequent layers. Similar diagrams can be shown in the case the network uses residual connections.

followed by the computation of the fundamental invariants of order 2, which we will denote $\imath[\mathbf{z}^0_{\mathbf{x}}] = \rho(\mathbf{x}, \mathcal{J}^{(2)}_{\mathbf{x}} f) \cdot (\mathbf{x}, \mathcal{J}^{(2)}_{\mathbf{x}} f) = \rho(\mathbf{z}^0_{\mathbf{x}}) \cdot \mathbf{z}^0_{\mathbf{x}}$ where $\mathbf{z}^0_{\mathbf{x}} = (\mathbf{x}, \mathcal{J}^{(2)}_{\mathbf{x}} f)$. Let $\psi_1 : J^2(\mathcal{M}) \to Y_1$, where $Y_1 = \mathbb{R}^q$, be a smooth map. In a deep learning context we assume ψ_1 to be a MLP with smooth activation functions. The first layer $\phi_1 : C^\infty(\mathbb{R}^3, Y) \to C^\infty(\mathbb{R}^3, Y_1)$, $Y_1 = \mathbb{R}^C$, is given by

$$\phi_1[f](\mathbf{x}) = \psi_1(\imath[\mathbf{z}^0_{\mathbf{x}}]) = \psi_1(\rho(\mathbf{z}^0_{\mathbf{x}}) \cdot \mathbf{z}^0_{\mathbf{x}}) = \imath[\psi_1](\mathbf{z}^0_{\mathbf{x}}) \tag{5.1}$$

and it is SE(3)-equivariant, because $\imath[\psi_1]$ is an invariant and we apply it to functions like in (4.18).

2. We build the second layer analogously. The output of the first layer is a signal

$$f_1 = (f_1^1, f_1^2, \dots, f^C) = \phi_1(f) : X \to Y_1.$$

We compute the derivatives

$$\mathcal{J}^{(2)}_{\mathbf{x}} f_1 = (\mathcal{J}^{(2)}_{\mathbf{x}}(f_1^1), \dots, \mathcal{J}^{(2)}_{\mathbf{x}}(f_1^C))$$

and the fundamental invariants $(\imath[\mathbf{z}^{1,1}_{\mathbf{x}}], \cdots, \imath[\mathbf{z}^{1,C}_{\mathbf{x}}])$ where $\mathbf{z}^{1,j}_{\mathbf{x}} = (\mathbf{x}, \mathcal{J}^{(2)}_{\mathbf{x}}(f_1^j))$ and $\imath[\mathbf{z}^{1,j}_{\mathbf{x}}] = \rho(\mathbf{z}^{1,j}_{\mathbf{x}}) \cdot \mathbf{z}^{1,j}_{\mathbf{x}}$. Let $\psi_2 : (J^n(\mathcal{M}))^C \to Y_2$ be a function given by a MLP. The second layer can be computed from the output of the first by

$$\begin{aligned} \phi_2'[f_1](\mathbf{x}) &= \psi_2\Big(\rho(\mathbf{z}^{1,1}_{\mathbf{x}}) \cdot \mathbf{z}^{1,1}_{\mathbf{x}}, \cdots, \rho(z^{1,C}_{\mathbf{x}}) \cdot z^{1,C}_{\mathbf{x}}\Big) \\ &= \psi_2(\imath[\mathbf{z}^{1,1}_{\mathbf{x}}], \cdots, \imath[z^{1,C}_{\mathbf{x}}]). \end{aligned} \tag{5.2}$$

Again this function is equivariant because it is a function of invariants. The second layer $\phi_2 = \phi_2' \circ \phi_1$ is equivariant because it is a composition of equivariant operators. This process can be repeated to obtain L equivariant layers $\phi_l = \phi_l' \circ \phi_{l-1}$, $0 \leq l \leq L$. It is illustrated in Figure 5.1(a).

The expressions above require the computation of certain non-linear combinations of the values of $\mathcal{J}^{(n)}_{\mathbf{x}} f$ that were established to be numerically unstable in the last section. Here we will propose an alternative that uses only a single moving frame computed from the input image and renders the network much more stable.

The alternative we propose is the following. Instead of computing the differential invariants at each layer, involving the computation of the moving frame based on the previous layer's feature maps, we compute the moving frame based only on the network input signal and compute all subsequent layers based on this moving frame.

Keeping the same target as before, which is to build a two-layer equivariant network, we can compute f_1 exactly as in step one. Now, from f_1 we compute $\mathcal{J}^{(2)}_{\mathbf{x}} f_1$ for all $\mathbf{x}$. Given some $\psi_2 : (J^2(\mathcal{M}))^C \to \mathbb{R}^{C'}$ (which again should be regarded as a MLP) we can compute the output of the second layer. In contrast to (5.2), however, we transform $\mathbf{z}^{1,j}_{\mathbf{x}}$ applying $\rho(\mathbf{z}^0_{\mathbf{x}}) = \rho(\mathbf{x}, \mathcal{J}^{(2)}_{\mathbf{x}} f)$, and not $\rho(\mathbf{z}^{1,j}_{\mathbf{x}})$, which yields

$$\phi_2'(f_1)(\mathbf{x}) = \psi_2(\rho(\mathbf{z}^0_{\mathbf{x}}) \cdot \mathbf{z}^{1,1}_{\mathbf{x}}, \dots, \rho(\mathbf{z}^0_{\mathbf{x}}) \cdot \mathbf{z}^{1,C}_{\mathbf{x}}). \tag{5.3}$$

The diagram of Figure 5.1(b) illustrates this method. The next result shows that repeated application of (5.3) defines a SE(n)-equivariant network.

Proposition 2. *Let $X = \mathbb{R}^d$, $Y = \mathbb{R}^{C_0} = \mathbb{R}$, $\rho : X \times Y_d^{(n)} \to SE(d)$ a moving frame, and $L \in \mathbb{N}^*$. For $1 \leq l \leq L$,*

- *let $Y_l = \mathbb{R}^{C_l}$, $C_l \geq 1$*
- *assume that $SE(d)$ acts on $X \times Y_l$ as $(\mathbf{x}, u) \mapsto (R\mathbf{x} + \mathbf{v}, u)$*

- *Let $\psi_l \in C^\infty(J^n(\mathcal{M})^{C_{l-1}}, Y_l)$ be L smooth maps.*

Let $\phi_1, \dots, \phi_L$ be the operators defined, for all $f \in C^\infty(X, Y)$, $\mathbf{x} \in X$, by

$$\phi_1[f](\mathbf{x}) = \psi_1\left(\rho\big(\mathbf{z}^0_{\mathbf{x}}\big) \cdot \mathbf{z}^0_{\mathbf{x}}\right) = \psi_1\left(\imath[\mathbf{z}^0_{\mathbf{x}}]\right) \tag{5.4}$$

and, for $1 < l \leq L$ either

$$\phi_l[f](\mathbf{x}) = \psi_l\left(\rho\big(\mathbf{z}^0_{\mathbf{x}}\big) \cdot \mathbf{z}^{l-1}_{\mathbf{x}}\right) \tag{5.5}$$

or (assuming $C_l = C_{l-1}$)

$$\phi_l[f](\mathbf{x}) = \phi_{l-1}[f](\mathbf{x}) + \psi_l\left(\rho\big(\mathbf{z}^0_{\mathbf{x}}\big) \cdot \mathbf{z}^{l-1}_{\mathbf{x}}\right) \tag{5.6}$$

where $\mathbf{z}^0_{\mathbf{x}} = (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f)$, $\mathbf{z}^l_{\mathbf{x}} = (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} \phi_l(f)_j)_{0 \leq j \leq C_l}$, $1 \leq l \leq L$, and the action of $\mathcal{G}$ on $J^n(\mathcal{M})^{C_l}$, is the action on $J^n(\mathcal{M})$ applied coordinate-wise.
Then each ϕ_l is $SE(d)$-equivariant, for $1 \leq l \leq L$.

Proof. For $l = 1$ we have

$$\phi_1(g \cdot f)(\mathbf{x}) = \psi_1\left(\rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f)) \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f))\right).$$

Noting $\mathbf{y} = g^{-1} \cdot \mathbf{x}$ and recalling (4.15), $g \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} f) = (g \cdot \mathbf{x}, \mathcal{J}^{(n)}_{(g \cdot \mathbf{x})}(g \cdot f))$, we can simplify

$$(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f)) = (g \cdot \mathbf{y}, \mathcal{J}^{(n)}_{(g \cdot \mathbf{y})}(g \cdot f)) = g \cdot (\mathbf{y}, \mathcal{J}^{(n)}_{\mathbf{y}} f) = g \cdot \mathbf{z}^0_{\mathbf{y}}. \tag{5.7}$$

Hence,

$$\begin{aligned}
\phi_1(g \cdot f)(\mathbf{x}) &= \psi_1\left(\rho(g \cdot \mathbf{z}^0_{\mathbf{y}}) \cdot (g \cdot \mathbf{z}^0_{\mathbf{y}})\right) \\
&= \psi_1\left(\imath[g \cdot \mathbf{z}^0_{\mathbf{y}}]\right) \\
&= \psi_1\left(\imath[\mathbf{z}^0_{\mathbf{y}}]\right) \quad \text{by invariance of } \imath \\
&= \phi_1(f)(\mathbf{y}) \quad \text{by (5.4)} \\
&= \phi_1(f)(g^{-1} \cdot \mathbf{x}) \\
&= (g \cdot \phi_1(f))(\mathbf{x}) \quad \text{by definition of the action on functions (4.11).}
\end{aligned} \tag{5.8}$$

Therefore, $\phi_1(g \cdot f) = g \cdot \phi_1(f)$.

Now, for $l > 1$, we have for the case (5.5),

$$\phi_l(g \cdot f)(\mathbf{x}) = \psi_l\left(\rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f)) \cdot (\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} \phi_{l-1}(g \cdot f)_j)_{1 \leq j \leq C_l}\right). \tag{5.9}$$

As shown earlier, $(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g \cdot f)) = g \cdot \mathbf{z}^0_{\mathbf{y}}$ with $\mathbf{y} = g^{-1} \cdot \mathbf{x}$. Similarly, assuming that ϕ_{l-1} is equivariant,

$$(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}} \phi_{l-1}(g \cdot f)_j) = g \cdot (\mathbf{y}, \mathcal{J}^{(n)}_{\mathbf{y}} \phi_{l-1}(f)_j)) = g \cdot \mathbf{z}^{l-1,j}_{\mathbf{y}}, \tag{5.10}$$

so that

$$\phi_l(g \cdot f)(\mathbf{x}) = \psi_l\left(\rho(g \cdot \mathbf{z}^0_{\mathbf{y}}) \cdot (g \cdot \mathbf{z}^{l-1,j}_{\mathbf{y}})_{1 \leq j \leq C_l}\right). \tag{5.11}$$

Since furthermore $\rho(g \cdot \mathbf{z}^0_{\mathbf{y}}) = \rho(\mathbf{z}^0_{\mathbf{y}}) \cdot g^{-1}$ by definition of a moving frame, we finally get

$$\begin{aligned}
\phi_l(g \cdot f)(\mathbf{x}) &= \psi_l\left(\rho(\mathbf{z}^0_{\mathbf{y}}) \cdot g^{-1} \cdot (g \cdot \mathbf{z}^{l-1,j}_{\mathbf{y}})_{1 \leq j \leq C_l}\right) \\
&= \psi_l\left(\rho(\mathbf{z}^0_{\mathbf{y}}) \cdot (\mathbf{z}^{l-1,j}_{\mathbf{y}})_{1 \leq j \leq C_l}\right) \\
&= \phi_l(f)(\mathbf{y}) = \phi_l(f)(g^{-1} \cdot \mathbf{x}) \\
&= (g \cdot \phi_l(f))(\mathbf{x}).
\end{aligned} \tag{5.12}$$

As for the case (5.6),

$$\begin{aligned}\phi_l(g\cdot f)(\mathbf{x}) &= \phi_{l-1}(g\cdot f)(\mathbf{x}) + \psi_l\left(\rho(\mathbf{x}, \mathcal{J}^{(n)}_{\mathbf{x}}(g\cdot f))\cdot \mathcal{J}^{(n)}_{\mathbf{x}}\phi_{l-1}(g\cdot f)\right)\\ &= \phi_{l-1}(f)(\mathbf{y}) + \psi_l\left(\rho(\mathbf{z}^0_{\mathbf{y}})\cdot \mathbf{z}^{l-1}_{\mathbf{y}}\right) \text{ like above, with } \mathbf{y} = g^{-1}\cdot\mathbf{x}\\ &= \phi_l(f)(\mathbf{y})\\ &= (g\cdot\phi_l(f))(\mathbf{x}).\end{aligned} \tag{5.13}$$

Therefore in all cases $\phi_l(g\cdot f) = g\cdot\phi_l(f)$ provided this is true for ϕ_{l-1}, and the proposition follows by induction. □

The proposition above shows that SE(d)-equivariant networks can be constructed this way using regular feedforward connection like in (5.5) and and using residual connections like in (5.6).

5.4 Moving Frames on SE(3)

The group SE(3) is the direct product of Lie groups SO(3) and $\mathbb{R}^3$ and since both are three-dimensional, then SE(3) is a 6-dimensional Lie group.

Here a *volume* refers to a signal on a 3-dimensional Euclidean domain, i.e. functions of the type $f:\mathbb{R}^3\to\mathbb{R}^C$. A reasoning similar to the one used to obtain SE(2)-equivariant operators on images in Chapter 4 can be used to produce SE(3)-equivariant operators on such volumes, but this time we use a cross-section based on restrictions of the Hessian matrix, instead of the derivatives of order one.

As such we begin by representing volumes as submanifolds of $\mathcal{M} = \mathbb{R}^3\times\mathbb{R}^C$. Firstly we consider the case $C=1$, but keeping in mind that for higher dimensions it is just a matter of channel-wise application. The action of SE(3) on $\mathcal{M}$ is then given by, $\forall(R,\mathbf{v})\in SE(3)$, $\forall(\mathbf{x},u)\in\mathcal{M}$

$$\pi_{R,\mathbf{v}}(\mathbf{x},u) = (R\cdot\mathbf{x}+\mathbf{v}, u). \tag{5.14}$$

If we proceed to extend $\mathcal{M}$ to the first-order jet space we will find that SE(3) does not act freely on $J^1(\mathcal{M})$. Indeed, the orbit of a point $(\mathbf{x},u,u_x,u_y,u_z)$ is the Cartesian product of $\mathbb{R}^3$, $\{u\}$ and a sphere with radius $\sqrt{u_x^2+u_y^2+u_z^2}$, hence it has dimension $5\neq\dim\mathrm{SE}(3)=6$. Therefore it is necessary to prolong the action to the second order jet-space in order to be able to obtain a moving frame. In this section we use a matrix notation for compactness: we denote $\nabla u = [u_x,u_y,u_z]^T$ and

$$Hu = \begin{bmatrix} u_{xx} & u_{xy} & u_{xz}\\ u_{xy} & u_{yy} & u_{yz}\\ u_{xz} & u_{yz} & u_{zz}\end{bmatrix}. \tag{5.15}$$

In that way, the coordinates of the second order jet-space are identified by $\mathbf{z}^{(2)} = (\mathbf{x},u^{(2)}) = (\mathbf{x},u,\nabla u, Hu)$ and the prolonged action can be expressed as

$$\pi_{R,\mathbf{v}}(\mathbf{z}^{(2)}) = \pi_{R,\mathbf{v}}(\mathbf{x},u,\nabla u,Hu) = (R\mathbf{x}+\mathbf{v}, u, R^T\nabla u, RHuR^T). \tag{5.16}$$

We could use a cross-section like the previous case that depends on the gradient, but it would also have to depend on one of the entries of Hu. However, it is simpler to use the coordinate

cross-section

$$\mathcal{K}' = \{(\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M}) | \mathbf{x} = 0, u_{xy} = u_{xz} = u_{yz} = 0\}, \tag{5.17}$$

i.e. $\mathbf{x} = 0$ and Hu is diagonal. Since Hu is symmetric, we can find a $P_{(\mathbf{x},u)} \in SO(3)$ [1] such that $P_{(\mathbf{x},u)} Hu P^T_{(\mathbf{x},u)}$ is diagonal. However there is more than one way to do so. Indeed we can permute any two columns of $P_{(\mathbf{x},u)}$ and we can multiply any two columns of $P_{(\mathbf{x},u)}$ by -1 and still map $(\mathbf{x}, u^{(n)})$ to $\mathcal{K}'$. In this way $\mathcal{K}'$ is not regular. To tackle the first problem we add the inequality $u_{xx} > u_{yy} > u_{zz}$ to the cross-section. Now if the transformed coordinates $P_{(\mathbf{x},u)} Hu P^T_{(\mathbf{x},u)}$ the inequality, and P' is obtained from permutating the columns of $P_{(\mathbf{x},u)}$, then $P' Hu P^{T'}$ does not satisfy the inequality. Furthermore, to solve the second problem we demand the the first to columns of $P_{(\mathbf{x},u)}$ are positively aligned with ∇u, i.e. let v be one of those columns, we constrain v such that $v^T \nabla u \geq 0$, which is equivalent to adding the inequalities $u_x \geq 0$ and $u_y \geq 0$ to the cross-section. Now we can define a regular cross-section

$$\mathcal{K} = \{(\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M}) | \mathbf{x} = 0, u_{xy} = u_{xz} = u_{yz} = 0, u_{xx} > u_{yy} > u_{zz}, u_x \geq 0, u_y \geq 0\}. \tag{5.18}$$

In particular the cross-section above is regular in an open set $U \subseteq J^n(\mathcal{M})$ where Hu has no repeated eigenvalues and $\nabla u \neq 0$. Now $\mathcal{K}$ defines a moving frame $\rho : (\mathbf{x}, u) \in \mathcal{M} \mapsto (P_{(\mathbf{x},u)}, -P_{(\mathbf{x},u)}\mathbf{x}) \in \mathcal{G}$. Then the non-zero fundamental invariants of the jet-space of order two are u, $v_i^T \cdot \nabla u$ and λ_i, $i = 1, 2, 3$, where the v_is are the eigenvectors of Hu (columns of $P_{(\mathbf{x},u)}$) and the λ_is are the eigenvalues of Hu (diagonal coefficients of $P_{(\mathbf{x},u)} Hu P^T_{(\mathbf{x},u)}$).

The cross-section $\mathcal{K}$ is not necessarily regular, but we can check that ρ is indeed a moving frame

5.5 SE3MovFNet

5.5.1 Gaussian Derivatives on Volumes

Similarly to what is done with the SE2DINNets in Chapter 4, derivatives are computed using the Gaussian derivative operators. In the 3-dimensional case the Gaussian derivatives of a function f are computed by

$$\frac{\partial^{i+j+k}}{\partial x^i \partial y^j \partial z^k}(f * G_\sigma) = f * \frac{\partial^{i+j+k}}{\partial x^i \partial y^j \partial z^k} G_\sigma = f * G^{i,j,k}_\sigma. \tag{5.19}$$

Figure 5.2 shows examples of Gaussian derivatives up to order 2.

We refer to the Gaussian n-jet of a volume as the Gaussian derivatives of order $\leq n$ $f^{(n)}_\sigma := (f * G^{i,j,k}_\sigma)_{0 \leq i+j+k \leq n}$. We can also identify the Gaussian n-jet by tensor coordinates. In particular for $n = 2$ we write $f^{(n)}_\sigma = (f_\sigma, \nabla_\sigma f, H_\sigma f)$ where $f_\sigma = f * G_\sigma$, ∇_σ is the Gaussian gradient i.e. derivatives of order one and $H_\sigma f$ is the Gaussian Hessian, i.e. derivatives of order two. Given an orthogonal matrix for each point $P : \Omega \to SO(3)$ (e.g. the matrices defining a moving frame) we denote the local action P by $(P \cdot f^{(n)}_\sigma)(\mathbf{x})$, e.g. for $n = 2$, $(P \cdot f^{(n)}_\sigma)(\mathbf{x}) = (f_\sigma(\mathbf{x}), P(\mathbf{x})^T \nabla_\sigma f(\mathbf{x}), P(\mathbf{x}) H_\sigma f(\mathbf{x}) P(\mathbf{x})^T)$ (here we omit the translational part of the action). If $f : \Omega \to \mathbb{R}^C$ is a multi-channel volume we can apply these operations channel-wise.

[1] if $\det P_{(\mathbf{x},u)} = -1$ we can multiply one of its rows by -1 so that the new matrix has determinant 1.

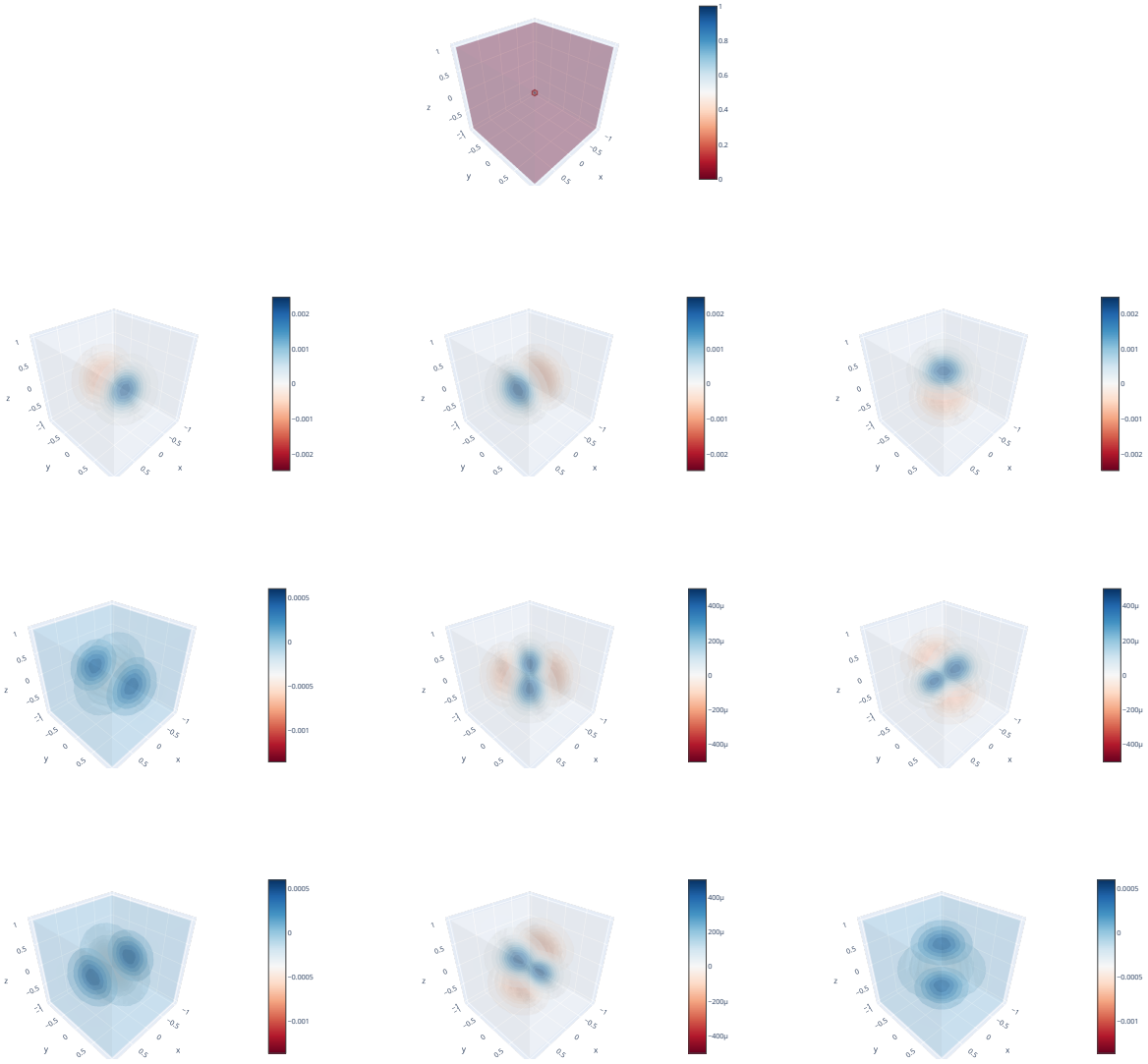
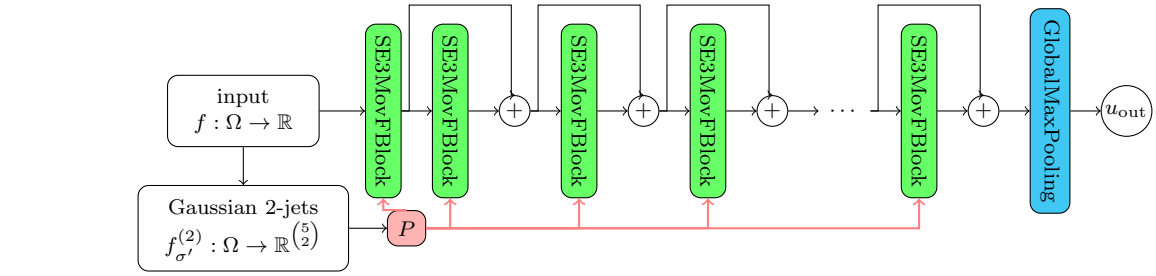


Figure 5.2: Examples of 3D Gaussian derivative filters of order ≤ 2.

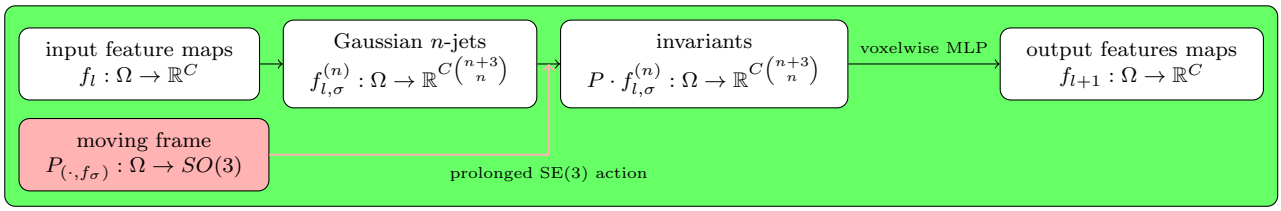
5.5.2 Architecture

Based on the exposition of Section 5.3, the general idea of our SE(3)-equivariant architecture, given an input signal $f : \Omega \to \mathbb{R}$ where $\Omega \subseteq \mathbb{R}^3$ is a 3-dimensional grid, is to first compute the matrices $P_{(\cdot, f_\sigma)} \in SO(3)$ of the moving frame for every $\mathbf{x} \in \Omega$. The matrices are computed by diagonalizing $H_{\sigma'} f(\mathbf{x})$ for every $\mathbf{x}$, i.e., find $P_{(\mathbf{x}, f_\sigma)}$ such that $P_{(\mathbf{x}, f_\sigma)} H_{\sigma'} f(\mathbf{x}) P^T_{(\mathbf{x}, f_\sigma)}$ is diagonal. $P_{(\mathbf{x}, f_\sigma)}$ must also have the values in the diagonal in decreasing order and with columns that are positively aligned with the gradient. If it is not possible to find such a matrix we set $P(\mathbf{x}) = 0$.

After computing P we compute blocks as shown in Figure 5.3b, which we call SE3MovF blocks, using the moving frame and the current features maps as input. The scale of each layer does not need to be necessarily the same, here we consider that a scale σ' is used to compute



(a) SE3MovFNet



(b) SE3MovFBlock

Figure 5.3: Example architecture of a SE3MovFNet, along with an example building block of the architecture.

the moving frames and a scale σ to compute the derivatives at each block.

A simple form of a global architecture, which we call SE3MovFNet, is in Figure 5.3a. The feature maps of each block are summed like in residual networks, which mimics a numerical PDE scheme (Ruthotto and Haber, 2020). The network in Figure 5.3 is specialized for a fixed number of channels, but by applying a $1 \times 1 \times 1$ convolution between blocks we can increase the number of feature maps of the next layer. Pooling may also be performed by subsampling after a block. The global max-pooling at the end renders the equivariant architecture invariant (Bronstein et al., 2021), which is interesting for a classification problem.

5.5.3 Complexity Analysis

We can perform a complexity analysis of the network define above.

Let us assume that the input is given as a signal $f : \Omega \to \mathbb{R}$ where $\Omega = \{0, \dots, W-1\} \times \{0, \dots, H-1\} \times \{0, \dots, D-1\}$ is a grid of size $W \times H \times D$, containing $N = W \cdot H \cdot D$ voxels. Moreover, let us assume that we compute the moving frame using Gaussian derivatives of scale σ' and the derivatives at other layers using σ, and that the discrete Gaussian derivative filters all have dimension $w' \times w' \times w'$, for the moving frame and $w \times w \times w$ for the other layers.

The computation of the moving frame is done as follows :

- compute all Gaussian derivatives of order 1 and 2 of f. Gaussian derivatives are separable, thus each one can be obtained by three convolutions with a filter of size w, which have cost a cost of $O(Nw')$ floating point operations (flops);
- compute the eigenvectors of the Hessian. Since the matrices have constant size 3×3 we consider this operation is done in constant time for each pixel and this step is done in $O(N)$ flops.

Figure 5.4: The eight tetris pieces in three dimensions.

So the computation of the moving frame is done in $O(Nw')$ flops.

From there on if we compute a layer with C input feature maps and C' output feature maps:

- this layer computes $\binom{n+3}{n}$ Gaussian derivatives for each input feature map, where n is the order of differentiation used, resulting in $O\left(\binom{n+3}{n}CwN\right)$ flops;
- to compute the prolonged group action, it can be verified that the equivariant group action can be expressed as polynomial in the partial derivatives, and thus it takes $O\left(\binom{n+3}{n}CN\right)$ flops;
- the previous step is followed by a L-layer multi-layer perceptron at each voxel. Assuming that the output dimension at each layer of the MLP is at most q we have that this step takes $O(CC' + LC'^2)$ flops.

The complexity of a layer of SE3MovF is the sum of the complexity of each step, i.e. it can be done in $O\left(\binom{n+3}{n}CNw + CC' + LC'^2\right)$ flops. In our experiments here we used $n = 2$ and $L = 2$ for all models, so the impact of those terms is very limited.

5.6 Experiments

5.6.1 Tetris

As a first test of the ability of the proposed layers to create equivariant/invariant networks, we apply them to the tetris pieces classification (Thomas et al., 2018; Weiler et al., 2018a). The training dataset contains the eight three-dimensional tetris pieces at a fixed orientation, like in Figure 5.4. The pieces are represented in a $40 \times 40 \times 40$ voxel grid by four Gaussian functions with mean at center of the cubes and standard deviation of half of a voxel. We train a SE3MovFNet with four blocks like Figure 5.3b applied with residual connections. Each block has 40 feature maps and uses leaky ReLU activations, also, between the two layers of the MLP there is a 30% dropout. We also train a CNN where the blocks are replaced with a 3×3 convolution with a leaky ReLU activation and the same quantity of feature maps. The last layer is a global average pooling. We constructed a rotated dataset by, for each piece, rotating it by a random rotation in SO(3) and adding the rotated piece to the test set, 30 times for each piece.

Both models achieve 100% accuracy in the non-rotated dataset, but on the rotated dataset the SE3MovFNet achieves $97\% \pm 2\%$ accuracy and the CNN achieves $19\% \pm 1\%$, as a result of 30 repetitions of the experiment. Overall this seems to confirm the rotational equivariance of the SE3MovFNet.

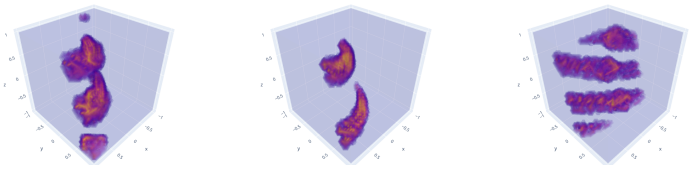
5.6.2 MedMNIST3D

The Moving Frame SE(3) networks are validated on the MedMNIST datasets (Yang et al., 2021a,b). MedMNIST refers to a collection of datasets for benchmarking of medical image analysis methods. They consist of 28×28 images and $28 \times 28 \times 28$ volumes. The datasets of interest here are those based on volumetric data. Here is a brief description of the 3D datasets, some examples can be seem on Figure 5.5, a more detailed one can be obtained from (Yang et al., 2021a,b).

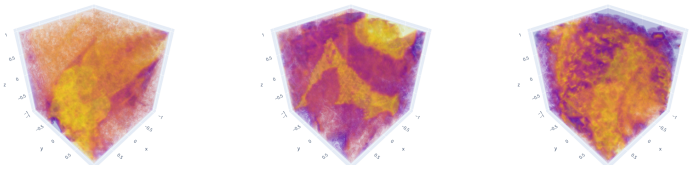
- **AdrenalMNIST3D.** Scans of adrenal glands labeled into normal adrenal gland and adrenal mass. The numbers of training/validation/test volumes is 1118/98/298.
- **NoduleMNIST3D.** Downsized thoratic CT scans. Contains two classes based on level of lung nodule malignancy. Has a 7:1:2 training/validation/test split.
- **OrganMNIST3D.** Multiclass classification of CT scans into 11 different organs.
- **VesselMNIST3D.** Entire brain vessels divided into healthy and aneurysm classes. 1694 healthy and 215 aneurysm. Has a 7:1:2 training/validation/test split.
- **FractureMNIST3D.** Rib fractures classified into 3 different classes.
- **SynapseMNIST3D.** Binary classification between excitatory and inhibitory synapses.

For each dataset we train a network with five SE3MovFr blocks with $16, 16, 32, 32, 64$ filters, using a stride of two in the second block. Voxelwise MLPs are computed as two subsequent $1 \times 1 \times 1$ convolutions followed by batch normalization (both) and leaky ReLU (only the first) and with the same number of neurons. We also train a CNN baseline with the same number of filters where each block consists of two $3 \times 3 \times 3$ convolutions followed by batch normalization and leaky ReLU. Input volumes are resized to $29 \times 29 \times 29$ so that subsampling by a factor of two is equivariant by rotations of $90°$ around the coordinate-axes. For both the SE3MovFNet and the CNN we also trained models with rotation data augmentation. The augmentation is performed by applying a random rotation in SO(3)

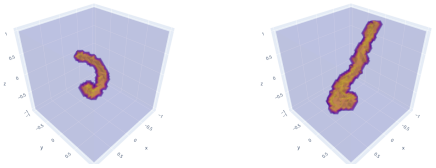
Overall results can be seen in Table 5.1. There we can see that the SE3MovFNet surpassed most of the benchmarks. Results of testing the models on rotated test sets are seen in Figures 5.6 and 5.7. Those results are obtained by taking each of the test sets of MedMNIST3D and rotating volumes around each of the coordinate axes by angles $i\frac{\pi}{12}$ for $i = 0, \ldots, 23$. In those results we observe the SE3MovFNet has perfect invariance for $90°$ rotations, evidenced by the periodicity of results, and a generally better equivariance than the CNN baseline with or without augmentation. It suffers, however, a significant loss of performance at orientations not multiple of $90°$. Moreover, while in some cases the data augmentation has a beneficial effect for the SE3MovFNet, it sometimes has the opposite effect of worsening rotation invariance. This could be a result of volumes being of very low resolution, as the discrete rotations can induce artifacts in rotated volumes.



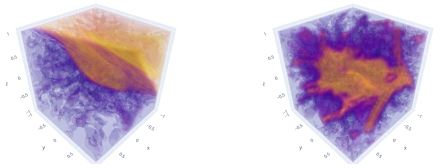
(a) FractureMNIST3D



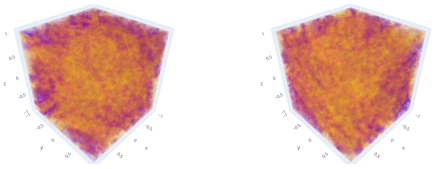
(b) OrganMNIST3D



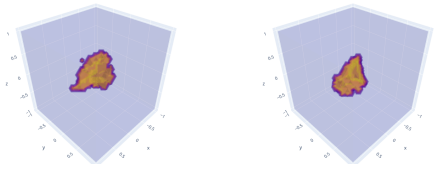
(c) VesselMNIST3D



(d) NoduleMNIST3D



(e) VesselMNIST3D

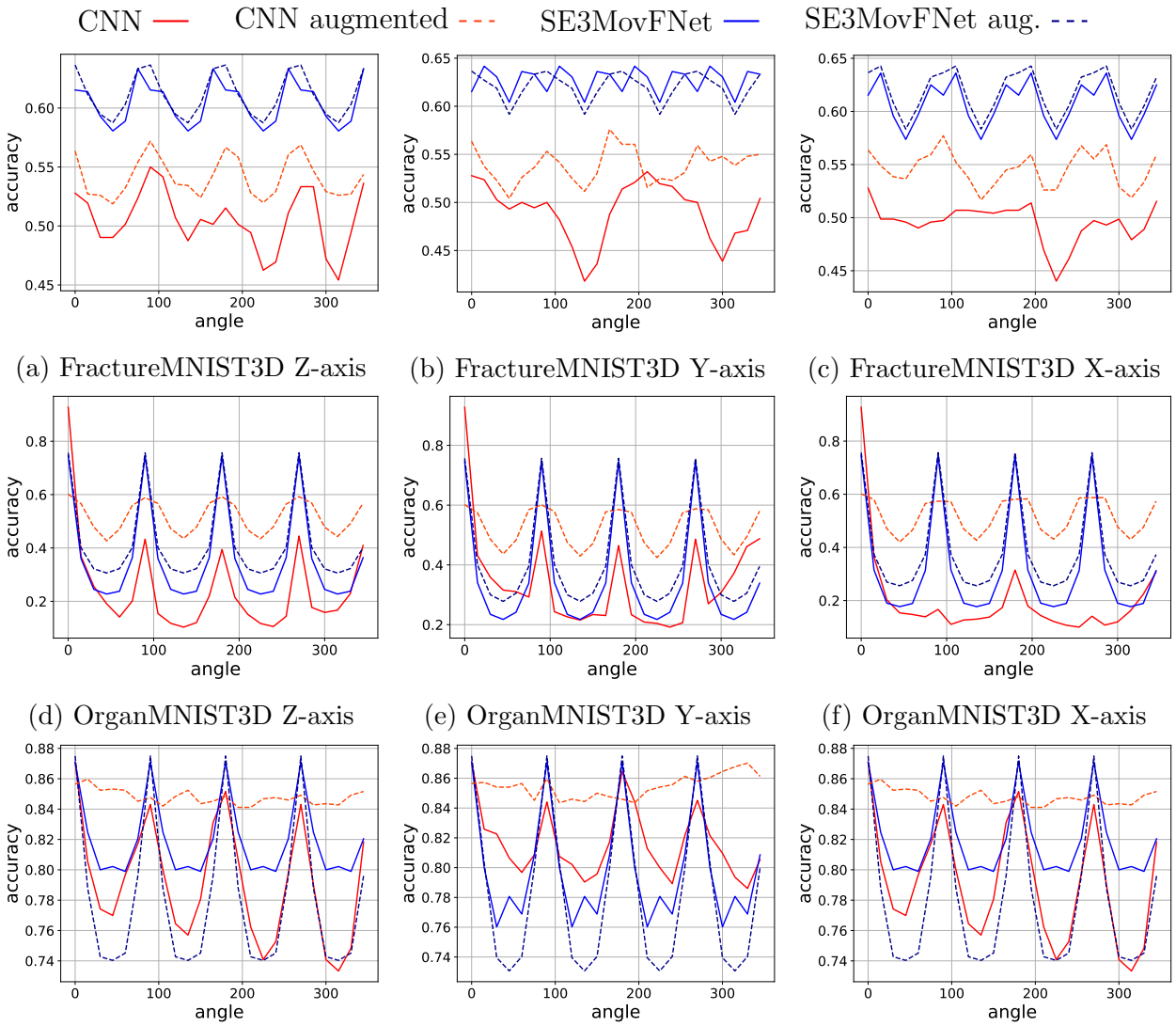


(f) NoduleMNIST3D

Figure 5.5: Some samples of different classes of each dataset.

	OrganMNIST3D	NoduleMNIST3D	FractureMNIST3D	AdrenalMNIST3D	VesselMNIST3D	SynapseMNIST3D
ResNet18 + 3D (Yang et al., 2021b)	0.907	0.844	0.508	0.721	0.877	0.745
ResNet18 + ACS (Yang et al., 2021b)	0.900	0.847	0.497	0.754	0.928	0.722
ResNet50 + 3D (Yang et al., 2021b)	0.883	0.847	0.484	0.745	0.918	0.795
ResNet50 + ACS (Yang et al., 2021b)	0.889	0.841	0.517	0.758	0.858	0.709
auto-sklearn (Yang et al., 2021a)	0.814	**0.914**	0.453	0.802	0.915	0.730
3DMedPT (Yu et al., 2021)	-	-	-	0.791	-	-
CNN baseline (ours)	**0.927**	0.871	0.528	0.824	0.949	0.775
SE3MovFrNet (ours)	0.745	0.871	0.615	0.815	0.953	**0.896**
CNN baseline, augmented (ours)	0.602	0.856	0.564	0.820	0.933	0.803
SE3MovFrNet, augmented (ours)	0.756	0.875	**0.636**	**0.830**	**0.958**	0.894

Table 5.1: Accuracies on the MedMNIST dataset compared to the benchmarks.

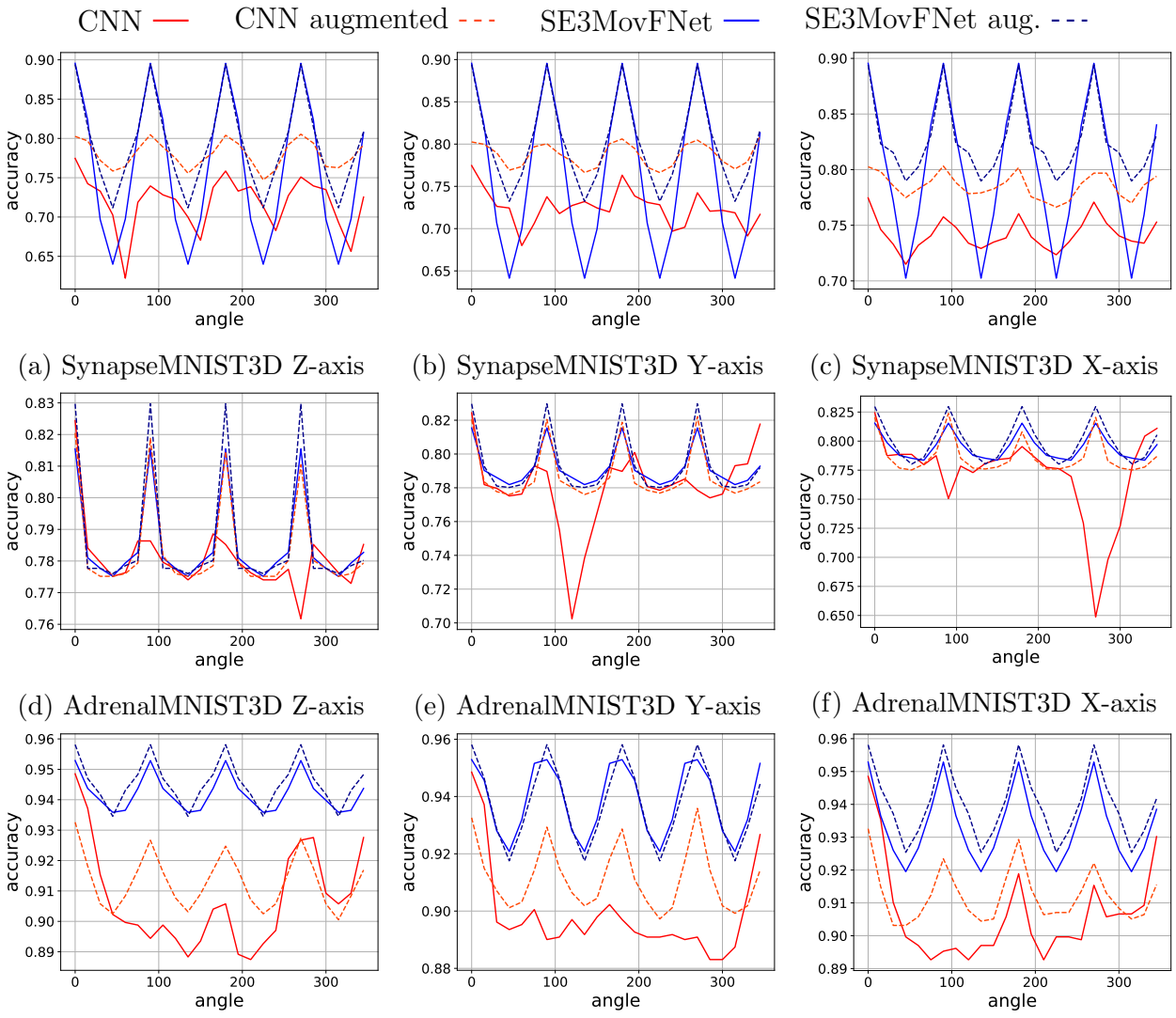


(a) FractureMNIST3D Z-axis (b) FractureMNIST3D Y-axis (c) FractureMNIST3D X-axis

(d) OrganMNIST3D Z-axis (e) OrganMNIST3D Y-axis (f) OrganMNIST3D X-axis

(g) NoduleMNIST3D Z-axis (h) NoduleMNIST3D Y-axis (i) NoduleMNIST3D X-axis

Figure 5.6: Predictions of models on half of the dataset of MedMNIST on the test set rotated by different angles around each of the coordinate axis.



(a) SynapseMNIST3D Z-axis (b) SynapseMNIST3D Y-axis (c) SynapseMNIST3D X-axis

(d) AdrenalMNIST3D Z-axis (e) AdrenalMNIST3D Y-axis (f) AdrenalMNIST3D X-axis

(g) VesselIST3D Z-axis (h) VesselIST3D Y-axis (i) VesselMNIST3D X-axis

Figure 5.7: Predictions of models on half of the dataset of MedMNIST on the test set rotated by different angles around each of the coordinate axis.

5.7 Conclusions

In this chapter we proposed an alternate moving-frame approach to group-equivariant networks to the approach of Chapter 4. In the latter, an equivariant network is obtained by computing differential invariants from the feature maps of the previous layer, which involves implicitly computing a moving frame for each pixel of those feature maps. In the approach of the present chapter, we compute only the moving frames for the input image and apply it to the feature maps of all layers. We showed that it is possible to define an equivariant neural network from this new approach , which is more stable numerically because the computed invariants are linear with respect to the feature maps of the previous layer.

We computed a moving frame for the action of SE(3) on volumes and applied the method above to construct a SE(3)-equivariant neural network. We implemented the architecture, referred to as SE3MovFNet, using Gaussian derivatives. The performance of the network is overall positive, as it attained the best results in four out of the six evaluated datasets of MedMNIST, and maintains a reasonable accuracy when images are rotated. The performance when the test set is rotated by an angle that is not a multiple of 90° around the coordinate axes, is much lower than otherwise. It might be interesting to identify the source of these problems, whether they come from artifacts from the discretization in low resolution or from the network itself.

5.8 Résumé en Français

Dans ce chapitre nous avons proposé une approche basée sur la méthode du repère mobile, pour obtenir un réseau équivariant par transformations de groupe. Cette approche représente une alternative à celle du Chapitre 4. Dans l'approche du chapitre précèdent, nous avons obtenu un réseau équivariant par le calcul des invariants différentiels à partir des cartes de caractéristiques de la couche précèdent, ce qui demande implicitement le calcul du repère mobile pour chaque pixel de ces cartes de caractéristiques. Dans ce chapitre, nous calculons le repère mobile seulement pour l'image d'entrée et nous l'appliquons aux cartes de caractéristiques de chaque couche. Nous avons montré qu'il est possible de définir un réseau équivariant à partir de cette nouvelle approche, qui est plus stable numériquement que l'approche du chapitre précédent, car les invariants sont linéaires par rapport aux cartes de caractéristiques de la couche précédente.

Nous avons calculé un repère mobile pour l'action de SE(3) (rotations et translations en 3 dimensions) sur les volumes et nous l'avons appliqué pour construire un réseau équivariant par l'action de SE(3). Nous avons implémenté la nouvelle architecture, qui a été appelée SE3MovFNet, avec les dérivées Gaussiennes.

Pour tester la performance et l'invariance de ces réseaux, nous avons utilisé MedMNIST3D (Yang et al., 2021b), une collection de bases de données d'imagerie médicale en faible résolution. La performance du réseau est plutôt positive, et il a atteint les meilleurs résultats sur quatre de six bases de données de MedMNIST3D, et la performance est encore maintenue dans un niveau raisonnable même quand les images de test sont tournées. Toutefois, la performance quand les images sont tournées par un angle qui n'est pas multiple de 90° autour des axes de coordonnées peut-être beaucoup plus faible que le cas contraire. Il serait intéressant de déterminer si ces

problèmes viennent de la discrétisation en faible résolution ou du réseau lui-même.

Chapter 6

Perspectives for Moving Frames Based Networks

6.1 Introduction

In this thesis a new class of group equivariant neural networks, based on moving frames, has been proposed. The approach was applied to obtain rotations and translations equivariant neural networks in two and three dimensions. The method of moving frames, however, is general enough to work on other domains and Lie groups and can be applied to obtain invariants in a variety of contexts.

In the examples presented in earlier chapters, the action of the Lie group was on the the domain of the signal, i.e. a spatial transformation, but we can equally define moving frames for actions on the output the values, e.g. group transformations on the grey levels or color space. In Section 6.2 we explore the idea of a neural network invariant to an affine transformation on the grey levels of the input image, or a contrast change. The network is tested in a simple toy example of digit classification where the contrast in the images is artificially changed and is shown to perform well under drastic contrast changes, while the CNN sees a visible drop when exposed to these changes.

In Section 6.3 neural networks equivariant to similarity transforms (i.e. rotations, translations and scalings) is proposed. The networks are similar to the SE2DIN in Chapter 4, but with the addition that the the layers receive a multi-scale input and compute differential invariants using re-scaled Gaussian derivatives, similar to Lindeberg (2022).

Finally, in Section 6.4 we propose a SE(3)-equivariant neural network for point clouds based on the method of moving frames. In that approach, instead of computing differential invariants, which is more challenging in point cloud data, we compute invariants on the spaces of k-tuples of points in $\mathbb{R}^3$. The approach we propose is equivalent to that of Thomas (2020), but the way we compute our moving frames is based on the covariance matrix of the points.

6.2 Contrast Change Invariant Networks

Let us consider a contrast change in an image. Again an image is a submanifold in $\mathcal{M} = \mathbb{R}^2 \times \mathbb{R}$. We are specifically considering grey-scale images in here, but the same reasoning can be applied

to multi-channel images, channel by channel. This time, the group of transformations we wish to have invariance to, is an affine transformation on the counter domain of our image (as well as translations on the input points). The group is given by $S \ltimes \mathbb{R}^2$ where $S = \{(\alpha, \beta) | \alpha > 0, \beta \in \mathbb{R}\}$ and S has composition law $(\alpha, \beta) \cdot (\alpha', \beta') = (\alpha\alpha', \alpha\beta' + \beta)$. It acts on $\mathcal{M}$ as

$$(\alpha, \beta, \mathbf{v}) \cdot (\mathbf{x}, u) \coloneqq (\mathbf{x} + \mathbf{v}, \alpha u + \beta). \tag{6.1}$$

when applied to a function $f : \mathbb{R}^2 \to \mathbb{R}$

$$[(\alpha, \beta, \mathbf{v}) \cdot f](\mathbf{x}) = \alpha f(\mathbf{x} - \mathbf{v}) + \beta. \tag{6.2}$$

This action can be interpreted as a horizontal translation (translation of $\mathbf{x}$) and a vertical affine transformations (affine transformation of $f(\mathbf{x})$).

In order for us to define a moving frame we derive the prolonged action in the jet-bundle. A simple computation reveals that for $n = i + j > 0$ and a smooth function $f : \mathbb{R}^2 \to \mathbb{R}$

$$\frac{\partial^n (\alpha, \beta, \mathbf{v}) \cdot f}{\partial x_1^i \partial x_2^j} = \widetilde{\frac{\partial^n f}{\partial x_1^i \partial x_2^j}} = \alpha \frac{\partial^n f}{\partial x_1^i \partial x_2^j}. \tag{6.3}$$

As such, the prolonged action of $(\alpha, \beta, \mathbf{v})$ on the jet space is given by $(\mathbf{x}, (u_{i,j})_{0 \leq |i+j| \leq n}) \mapsto (\tilde{\mathbf{x}}, (\tilde{u}_{i,j})_{0 \leq i+j \leq n})$ where $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{v}$ and $\tilde{u} = \tilde{u}_{0,0} = \alpha u + \beta$ are given by (6.1), and for $1 \leq i + j \leq n$

$$\tilde{u}_{i,j} = \alpha u_{i,j}.$$

From there we can define an appropriate cross-section in the jet bundle $J^n(\mathcal{M})$. For simplicity we compute a coordinate cross-section. In the translational part we choose the equation $\mathbf{x} = 0$, resulting in $\mathbf{v} = -\mathbf{x}$. Since β only influences u we add the equation $u = 0$ to the cross-section, resulting in $\beta = -\alpha u$. Now for determining α we can choose an equation of the form $|u_{i,j}| = 1$ resulting in $\alpha = (|u_{i,j}|)^{-1}$. This results in the cross-section $K_{i,j} = \{\mathbf{x} = 0, u = 0, |u_{i,j}| = 1\}$. The choice of i and j still remains, however.

We would like the invariant differential operators to have a consistent behaviour in all of the image, but choosing a single value for all of the domain would be problematic when $\frac{\partial^{i+j} u}{\partial x_1^i \partial x_2^j} = 0$. Our method to choose a cross-section is the following. Suppose that for some i, j such that $1 \leq i + j \leq n$ we have that $|u_{i,j}| > |u_{k,l}|$ for all $k, l \in \mathbb{N}$ such that $1 \leq k + l \leq n$ and $(k, l) \neq (i, j)$, then that is true for an open set containing $(\mathbf{x}, u^{(n)})$ and $K_{i,j}$ defines a moving frame

$$\rho(\mathbf{x}, u^{(n)}) = (|u_{i,j}^{-1}|, -|u_{i,j}^{-1}| u, -\mathbf{x}) = \left(\frac{1}{\max\limits_{1 \leq k+l \leq n} |u_{k,l}|}, -\frac{u}{\max\limits_{1 \leq k+l \leq n} |u_{k,l}|}, -\mathbf{x} \right) \in \mathcal{G}. \tag{6.4}$$

Note moving frame is not smooth everywhere. In particular, if $u_{i,j} = 0$ for all $1 \leq i + j \leq n$ or the maximum is attained at least twice, then the expression above is not smooth, but in practice it does not cause any problems as it is well defined and can be used to invariantize signals.

We have constant invariants $\mathcal{I}_{00} = \imath[u] = 0$, $H = \imath[\mathbf{x}] = 0$, $\mathcal{I}_{i^*,j^*} = 1$ where i^*, j^* are the maximum values in (6.4) and for $i + j \geq 1$ we have, for almost all $(\mathbf{x}, u^{(n)}) \in J^n(\mathcal{M})$,

$$\mathcal{I}_{i,j} = \frac{u_{i,j}}{\max\limits_{1 \leq k+l \leq n} |u_{k,l}|}. \tag{6.5}$$

For functions we implement the expression

$$\bar{F}_{i,j}(\mathbf{x}) = \frac{1}{\max\limits_{1\leq k+l\leq n} |\partial_{k,l} f| + \epsilon} \cdot \frac{\partial^n f(\mathbf{x})}{\partial^i x_1 \partial^j x_2} \tag{6.6}$$

where ϵ is a small positive constant to prevent division by 0.

A layer computing (6.6) is approximately invariant to change in contrast, depending on the value of ϵ. Therefore if we use the output of the invariants as the input to the next layer, the output of the next layer will also be (approximately) invariant. A contrast-invariant network can then be defined as a layer that computes (6.6) for example with Gaussian derivatives, followed by classical CNN layers.

Global Contrast Change Invariance Experiment. In order to test the efficacy of (6.5) in a neural network we prepare a benchmark task. Using the MNIST dataset we train a CNN and a contrast-invariant network obtained by first processing the data using (6.5) up to order two (where the derivatives are computed with Gaussian derivatives) and then using the same CNN architecture. Additionally we test a CNN where the input has been normalized following

$$\text{Normalize}(f)(\mathbf{x}) = \frac{f(\mathbf{x}) - \inf_{\mathbf{x}\in\Omega} f(\mathbf{x})}{\sup_{\mathbf{x}\in\Omega} f(\mathbf{x}) - \inf_{\mathbf{x}\in\Omega} f(\mathbf{x})}, \tag{6.7}$$

where Ω is the domain of the discrete signal and f is assumed to be bounded. The expression above should result in perfect invariant to contrast changes, but we will see later that it performs badly when compared to local contrast changes.

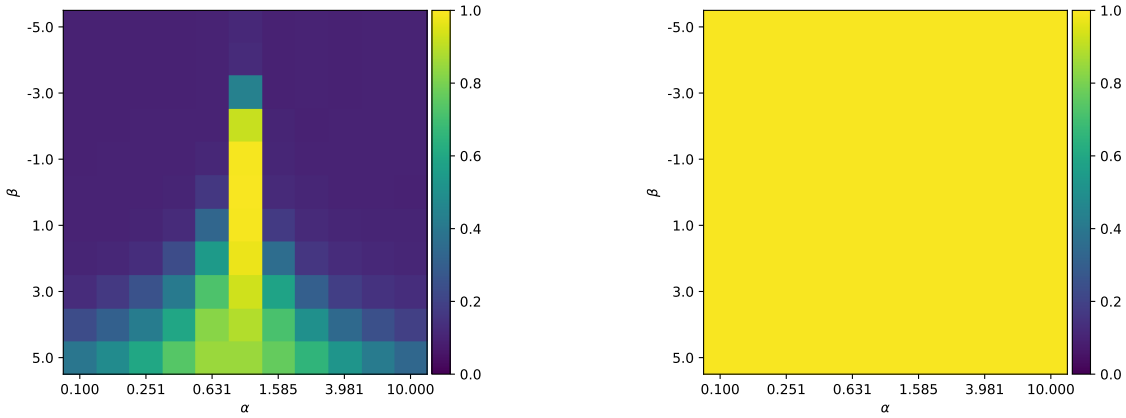
After training the models are tested in the MNIST test set, but the images have been transformed according to (6.1) with $\alpha \in [.1, 10]$ and $\beta \in [-5, 5]$. Results are displayed in Figure 6.1. As we can see, outside a small range the CNN's performance suffer significantly. The model based on the moving frame, on the other hand, deals mostly well with most of the contrast changes. The CNN with normalized input performs the same for every transformation as expects, and thus surpasses the moving frames based method in this case.

Local Contrast Change Invariance Experiment. Now we test the efficacy of the models when local contrast changes are applied. Indeed consider that the transformation in the function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by $f \mapsto \tilde{f}$ where

$$\tilde{f}(\mathbf{x}) = \alpha(\mathbf{x}) f(\mathbf{x}) + \beta(\mathbf{x}). \tag{6.8}$$

and $\alpha \in C^\infty(\mathbb{R}^2, \mathbb{R}_{\geq 0})$, $\beta \in C^\infty(\mathbb{R}^2, \mathbb{R})$. A transformation like that can more closely model lightning changes that can happen in natural images. A global normalization is not sufficient to attain invariance in this case. The moving frames approach, however, is local and should be approximately invariant to this transformation.

In order to test that, we use a test set obtained by taking each image from the MNIST test set and applying (6.8) where $\alpha(\mathbf{x})$ is given by $\alpha(\mathbf{x}) = \exp\left(\lambda \bar{G}_{\mu,\sigma}(\mathbf{x})\right)$, $\beta = 0$, where $\bar{G}_{\mu,\sigma}$ is a Gaussian function with mean μ and standard deviation σ, normalized to have values in $(0, 1]$, and $\gamma \in \mathbb{R}$. For each image we chose a random μ in a 12×12 square centered at the center of the image, σ uniformly chosen in the interval $[5, 8]$ and γ uniformly chosen in the interval $[-\log(15), \log(15)]$. Some example images are shown in Figure 6.2.



(a) CNN

(b) CNN with normalized input

(c) Moving-frame-based model

Figure 6.1: Accuracies obtained by a convolutional and a contrast invariant model with several contrast changes on the inputs. Results are averages of 20 repetitions of the experiment.

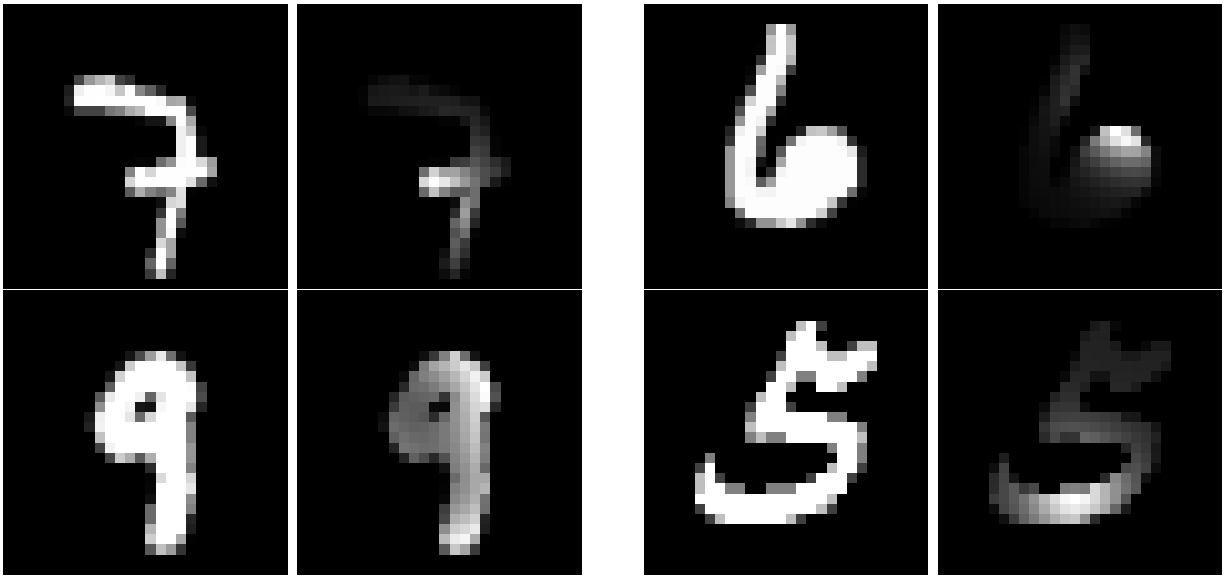


Figure 6.2: Original images and the transformed version we used to test local contrast change. Gray value scale is not the same for all images. White pixels are the largest grey value of the image and black the lowest.

After testing we obtained accuracies 0.981 ± 0.001 for the network based on moving frames, 0.78 ± 0.01 for the CNN and 0.143 ± 0.004 for the CNN with normalized input, confirming our expectations.

It could be interesting to try a similar methodology to other group actions on gray values or color spaces, for instance, a power law transformation $f \mapsto \alpha f^{\gamma}$ in the lightness component of a color image or general transformations in the color space of images. Moreover, a similar approach can be applied where instead of Gaussian derivatives we use convolutional or morphological filters.

6.3 Similarity Transformation Equivariant Network

The (special) *similarity transforms group* S(2) acting on the plane has elements of the form $(rR_\theta, \mathbf{v})$, where $r > 0$ is a real number, $R_\theta \in SO(2)$ is the rotation of angle $\theta \in [0, 2\pi)$ and $\mathbf{v} \in \mathbb{R}^2$. It acts in $\mathbf{x} \in \mathbb{R}^2$ by

$$\mathbf{x} \mapsto rR_\theta \mathbf{x} + \mathbf{v}. \tag{6.9}$$

This group can be seen as a product of a scaling group, $\mathbb{R}_{>0}$, and SE(2). In the vein of Lindeberg (2022), we can add scale equivariance to a network based on Gaussian derivatives. More specifically we first note a property of Gaussian derivative filters. Let $\pi_t^S(u)(x) = u(tx)$, $t > 0$ denote the action of the scaling group $\mathbb{R}_{>0}$

$$\begin{aligned}(\pi_t^S(u) * G_\sigma^{i,j})(\mathbf{x}) &= \tfrac{\partial^{i+j}}{\partial x_1^i \partial x_2^j}(\pi_t^S(u) * G_\sigma)(x) \\ &= \tfrac{\partial^{i+j}}{\partial x_1^i \partial x_2^j}(u * G_{t\sigma})(tx) \\ &= \pi_t^S(u * G_{t\sigma}^{i,j})(\mathbf{x}).\end{aligned} \tag{6.10}$$

We can see that the Gaussian derivatives have an interesting relationship with this scaling group, but it is not scale equivariance (at least not in a strict commutation of the same group action sense), as the scale of the Gaussian derivatives is changed when we change the order of operations.

Now suppose that as input to the Gaussian derivatives we have a function $u : \mathbb{R}_{>0} \times \mathbb{R}^2 \to \mathbb{R}$ and that $\pi_t^S(u)(s, x) = u(ts, t\mathbf{x})$. Let us define in that case the Gaussian derivatives as

$$(u *_S G_\sigma^{i,j})(s, \mathbf{x}) = (u(s, \cdot) * G_{s\sigma}^{i,j})(\mathbf{x}), \tag{6.11}$$

in this case we have

$$\begin{aligned}(\pi_t^S[u] *_S G_\sigma^{i,j})(s, \mathbf{x}) &= (\pi_t^S[u(s, \cdot)] * G_{s\sigma}^{i,j})(s, \mathbf{x}) \\ &= \pi_t^S[u(s, \cdot) * G_{ts\sigma}^{i,j}](\mathbf{x}) \\ &= (u(ts, \cdot) * G_{ts\sigma}^{i,j})(t\mathbf{x}) \\ &= (u *_S G_\sigma^{i,j})(ts, t\mathbf{x}) \\ &= \pi_t^S[u *_S G_\sigma^{i,j}](s, \mathbf{x}),\end{aligned} \tag{6.12}$$

therefore these Gaussian derivatives are scale-equivariant.

Of course, in order to apply these filters to our desired input domain, we need to first map a function $u : \mathbb{R}^2 \to \mathbb{R}$ to a function $\Lambda u : \mathbb{R}_{>0} \times \mathbb{R}^2 \to \mathbb{R}$, similarly to what was done with the lifting in Chapter 2. We wish as well that the operator Λ respects an equivariance relationship, namely $\pi_t^S[\Lambda u] = \Lambda \pi_t^S[u]$ (noting that the actions on the left and right hand sides are in different domains). Like in Chapter 2 we call Λ a *lifting*. A network consisting of a lifting and a $\mathbb{R}_{>0}$-equivariant operator ϕ satisfies $\phi \circ \Lambda \circ \pi_t^S = \pi_t^S \circ \phi \circ \Lambda$. The map $\Lambda u(s, \mathbf{x}) \coloneqq u(\mathbf{x})$ $\forall s \in \mathbb{R}_{>0}, \mathbf{x} \in \mathbb{R}^2$ works well for this purpose.

Finally, we can use these scaled Gaussian derivatives in order to compute differential invariants of SE(2) and show that they are both SE(2)-equivariant and $\mathbb{R}_{>0}$-equivariant, i.e. S(2)-equivariant. Firstly, let us define the action of SE(2) in images $u : \mathbb{R}_{>0} \times \mathbb{R}^2 \to \mathbb{R}$ as

$$\pi_{(R,\mathbf{v})}[u](s, \mathbf{x}) \coloneqq u(s, R_\theta \mathbf{x} + \mathbf{v}), \tag{6.13}$$

and a scaled n-th differential invariant as

$$I^S[u](s, \mathbf{x}) = I((G_\sigma^{i,j} *_S u)(s, \mathbf{x}), 0 \leq i + j \leq n) = I((G_{s\sigma}^{i,j} * u(s, \cdot))(x), 0 \leq i + j \leq n), \tag{6.14}$$

Given any SE(2) differential invariant I of the Jet-Space $J^n(\mathcal{M})$. We have

$$\begin{aligned}I^S(\pi_{\theta,\mathbf{v}}[u])(s, \mathbf{x}) &= I((G_{s\sigma}^{i,j} * \pi_{\theta,\mathbf{v}}[u](s, \cdot))(\mathbf{x}), 0 \leq i + j \leq n) \\ &= I((G_{s\sigma}^{i,j} * \pi_{\theta,\mathbf{v}}[u(s, \cdot)])(\mathbf{x}), 0 \leq i + j \leq n) \\ &= I(\pi_{\theta,\mathbf{v}}[(G_{s\sigma}^{i,j} * u(s, \cdot))](\mathbf{x}), 0 \leq i + j \leq n) \\ &= I((G_{s\sigma}^{i,j} * u(s, \cdot))(R_\theta \mathbf{x} + \mathbf{v}), 0 \leq i + j \leq n) \\ &= I^S(u)(s, R_\theta \mathbf{x} + \mathbf{v}) = \pi_{\theta,\mathbf{v}}[I^S(u)](s, \mathbf{x}),\end{aligned} \tag{6.15}$$

and

$$\begin{aligned}I^S(\pi_t^S[u])(s, \mathbf{x}) &= I((G_\sigma^{i,j} *_S \pi_t^S[u])(s, \mathbf{x}), 0 \leq i + j \leq n) \\ &= I(\pi_t^S[G_\sigma^{i,j} *_S u](s, \mathbf{x}), 0 \leq i + j \leq n) \\ &= I((G_\sigma^{i,j} *_S u)(ts, t\mathbf{x}), 0 \leq i + j \leq n) \\ &= I^S(u)(ts, t\mathbf{x}) = \pi_t^S[I^S(u)](s, \mathbf{x})\end{aligned} \tag{6.16}$$

In particular, this is true when I is a fundamental invariant or a functional combination of fundamental invariants.

Furthermore, we can extend these scaled differential invariants to be able to use the multi-scale information. We combine invariants of different scales by

$$I^S(u)(s,\mathbf{x}) = \sum_{i=1}^{C} \int_{-\infty}^{\infty} I_i^S(t,\mathbf{x}) h_i(ts) dt \tag{6.17}$$

for a compactly supported $h : \mathbb{R}^{>0} \to \mathbb{R}^C$. This can be computed as one-dimensional convolutions in parallel.

This method seems like an interesting extension of the SE2DIN blocks that has both rotation and scaling equivariance and uses multi-scale information, but for now it is merely a theoretical proposition that has not yet been tested in practice.

6.4 Moving Kernel Point Convolutions

Kernel Point Convolution

The Kernel Point Convolution (KPConv) (Thomas et al., 2019) is a convolutional layer for point cloud based data. Here we review it before we discuss its equivariant counterpart. In this chapter, we represent a point cloud as a finite set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} = \mathcal{P} \subseteq \mathbb{R}^3$. A signal $f : \mathcal{P} \to \mathbb{R}^C$ on the point cloud can be represented by a matrix $\mathcal{F} \in \mathbb{R}^{N \times C}$ where where its i-th row is given by $f_i = f(\mathbf{x}_i)$. The convolution of a signal by a filter $g : \mathbb{R}^3 \to \mathbb{R}^{C \times C'}$ is given by a signal $(f * g) : \mathcal{P} \to \mathbb{R}^{C'}$

$$(f * g)(\mathbf{x}) = \sum_{\mathbf{x}_i \in N_\mathbf{x}} g(\mathbf{x}_i - \mathbf{x}) f_i, \tag{6.18}$$

where $N_\mathbf{x} \subseteq \mathcal{P}$ is the neighborhood of a point $\mathbf{x} \in \mathcal{P}$.

The most important part of (6.18) is the filter g. KPConv defines g in terms of kernel points. Firstly, the support of g is set to be the closed ball $\bar{B}_r = \{\mathbf{y} \in \mathbb{R}^3 \mid \|\mathbf{y}\| \leq r\}$ for some $r > 0$. For a point $\mathbf{y} \in \bar{B}_r$, $g(\mathbf{y})$ is defined as

$$g(\mathbf{y}) = \sum_{k=1}^{K} h(\mathbf{y}, \mathbf{y}_k) W_k, \tag{6.19}$$

where $\mathbf{y}_1, \dots, \mathbf{y}_K$ are the so-called kernel points, $h : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ is a weighting function and $W_k \in \mathbb{R}^{C \times C'}$ is a matrix.

The weighting function h can be defined in different ways, in the KPConv paper we see some examples:

- Linear correlation:
$$h(\mathbf{y}, \mathbf{y}_k) = \max\left\{0, \frac{\|\mathbf{y} - \mathbf{y}_k\|}{\sigma}\right\}; \tag{6.20}$$
- Gaussian correlation
$$h(\mathbf{y}, \mathbf{y}_k) = \exp\left(-\frac{\|\mathbf{y} - \mathbf{y}_k\|^2}{\sigma^2}\right); \tag{6.21}$$

In Thomas et al. (2019) the method was further generalized to work with deformable kernels, as opposed to the rigid kernels described above. In this section, however, we only study the case of rigid kernels for the KPConv and for its rotation-equivariant version described below.

6.4.1 SE(3)-Equivariant KPConv

Now we propose to make the KPConv SE(3)-equivariant with the method of moving frames. The same reasoning was applied by Thomas (2020) albeit not using moving frames explicitly. Here we describe the method in terms of moving frames and propose a specific moving frame based on the covariance matrix of the points in a neighborhood.

In order to obtain equivariance we impose additional structure on the point cloud. We define an oriented point cloud as a point cloud $\mathcal{P}$ for which each point $\mathbf{x}_i \in \mathcal{P}$ is associated with a normal vector $\mathbf{n}_i$ belonging to the unit sphere S^2.

Let us write the neighborhood of a point $\mathbf{x}$ as a tuple $N_{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_M) \in \mathcal{P}^M$ where $\mathbf{x}_1 = \mathbf{x}$, and let us note $\mathbf{n}_1$ its corresponding normal vector. The space of possible neighborhoods of size M is given by $\mathcal{M} = (\mathbb{R}^3)^M$. We can define the action of SE(3) on it by

$$(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M) \mapsto (R\mathbf{x}_1 + \mathbf{v}, R\mathbf{x}_2 + \mathbf{v}, \dots, R\mathbf{x}_M + \mathbf{v}). \tag{6.22}$$

Define $\mu = \frac{1}{M}\sum_{i=1}^{M} \mathbf{x}_i$ and let X be the $3 \times M$ matrix having $(\mathbf{x}_i - \mu)/M$ as columns, i.e. $X = [\mathbf{x}_1 - \mu,\ \mathbf{x}_2 - \mu,\ \cdots,\ \mathbf{x}_M - \mu]$. Then

$$A = \frac{1}{M} X X^T \tag{6.23}$$

is the empirical covariance matrix of the neighborhood. If we apply the action of $(R, \mathbf{v})$ the transformed points become $\bar{\mathbf{x}}_i = R\mathbf{x}_i$ and the the transformed normal vectors $\bar{\mathbf{n}}_i = R\mathbf{n}_i$, we obtain $\bar{\mu}$, $\bar{X}$ and a new covariance matrix $\bar{A}$, given respectively by

$$\begin{gathered} \bar{\mu} = \tfrac{1}{M}\sum_{i=1}^{M} R\mathbf{x}_i = \tfrac{1}{M} R \sum_{i=1}^{M} \mathbf{x}_i = R\mu, \\ \bar{X} = [R(\mathbf{x}_1 - \mu)\ R(\mathbf{x}_2 - \mu)\ \cdots\ R(\mathbf{x}_M - \mu)] = RX, \quad \text{and} \\ \bar{A} = \tfrac{1}{M}\bar{X}\bar{X}^T = \tfrac{1}{M} R X X^T R^T = R A R^T. \end{gathered} \tag{6.24}$$

We define a cross-section in this space by the constraints $\mathbf{x}_1 = 0$ and $A = D$, where D is a diagonal matrix with values in the diagonal in descending order. This gives us six equations which must be solved, namely $\mathbf{x}_{1i} = 0$, $i = 1, 2, 3$ and $A_{12} = A_{13} = A_{23} = 0$ (note that A is symetrical and therefore this condition gives us $A_{21} = A_{31} = A_{32} = 0$) which should be enough since SE(3) is 6-dimensional. We can easily deduce that in order to solve the equations, we must have $\mathbf{v} = -R\mathbf{x}_1$. To determine R, first we can make an eigen decomposition $A = QDQ^T$, where D is sorted in descending order. If we set $R = Q^T$ we obtain $\bar{A} = QAQ^T = QQ^T DQQ^T = D$, therefore solving the moving frame equations. However, it turns out there are two solutions in this case. The simplest solution to this problem is to align the eigenvectors, i.e. the columns of Q, with the normal $\mathbf{n}_1$.

We propose the Moving KPConv (MKPConv) as

$$(f *_{\text{moving}} g)(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}_\mathbf{x}} g(R_\mathbf{x}(\mathbf{x}_i - \mathbf{x}))f_i. \tag{6.25}$$

where $R_\mathbf{x}$ is the matrix given by the moving frame at the neighborhood $N_\mathbf{x}$. Considering the function $F(N_\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}_\mathbf{x}}^{k} g(\mathbf{x}_i)f_i$, we have that

$$(f *_{\text{moving}} g)(\mathbf{x}) = \imath[F](N_\mathbf{x}). \tag{6.26}$$

Moreover, F is invariant w.r.t. permutations of $N_\mathbf{x}$ that fix $\mathbf{x}_1$, and therefore so is $\imath[F]$.

Similarly to the method in Chapter 5, we propose to compute the moving frame for one point cloud, the input point cloud.

The normal may not always be present in the data, and estimating the normal vector may sometimes result in a problem of orientation, as there are always two possibilities for normal vectors, facing opposite directions and the normal vectors may be inconsistent with each other. A solution is to always use both normal vectors and sum the result. When we do that it is impossible to distinguish between the result of a neighbordhood and its reflection, so this actually defines a E(3) equivariant operator when we applyto a signal on the ponit cloud.

6.4.2 Point Cloud Subsampling

In practice it is useful to subsample input point clouds to reduce the computations. Here below we mention two popular subsampling strategies, since we use the first one (Farthest Point Sampling) in our experiment in the next section, as an alternative to the second one (Grid Subsampling).

Farthest Point Sampling

Farthest point subsampling is a subsampling strategy that aims to obtain $M < N$ points in a point cloud that maximize the total point distance. In practice, it is computed by a greedy algorithm as follows:

1. Start by selecting a random point $\mathbf{p} \in \mathcal{P}$ and create $S = \{\mathbf{p}\}$;
2. Find a new point $\mathbf{q}^*$:
$$\mathbf{q}^* = \arg\max_{\mathbf{q}\in\mathcal{P}} \min_{\mathbf{p}\in S} \|\mathbf{q} - \mathbf{p}\|$$
3. Add $\mathbf{q}^*$ to S: $S = S \cup \{\mathbf{q}^*\}$;
4. If $|S| = M$ stop, otherwise got to step 2.

This subsampling approach is also referred to as *greedy clustering* as the points in S can form the centers for clusters in $\mathcal{P}$. An efficient implementation is detailed in Section 3.1 of Har-Peled and Mendel (2006).

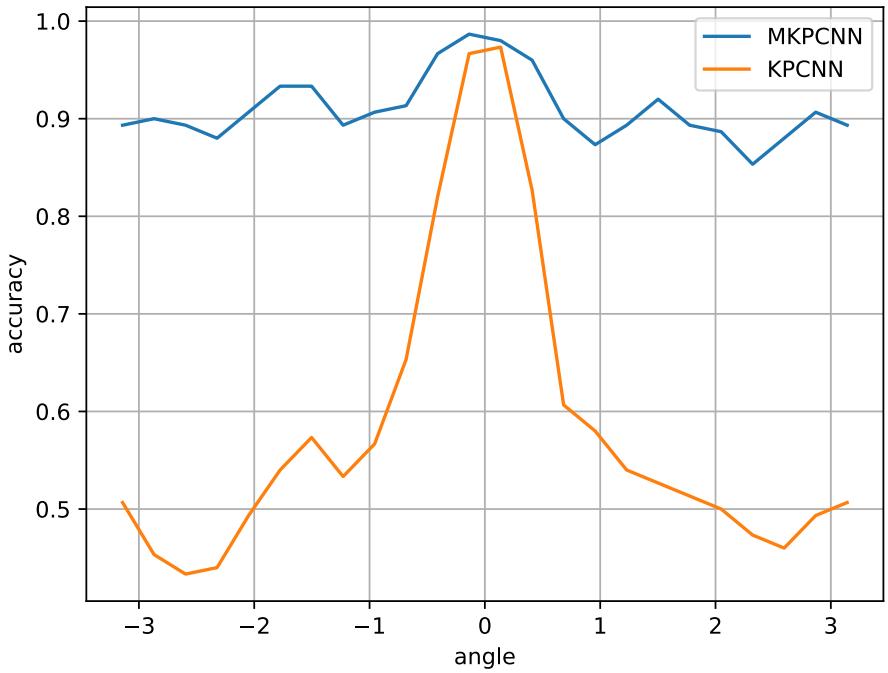


Figure 6.3: Accuracies of the MKPCNN and the KPCNN.

Grid Subsampling

In this approach, adopted by Thomas et al. (2019), a cell size s is chosen and the space is divided into a cubic grid of voxels of side s. For each voxel V_{ijk}, if there are points of $\mathcal{P}$ inside of V_{ijk} compute the barycenter of those points and add it to the subsampled cloud S.

6.4.3 Experiment

ModelNet

Firstly as a proof-of-concept we try out training an E(3)-equivariant KPCNN, and a regular KPCNN using only the classes 0 (airplane) and 1 (bed) of ModelNet (Wu et al., 2015). The networks are constructed with an architecture similar to that of the KPCNN in Thomas et al. (2019), but the subsampling is given by the farthest point sampling. Afterwards we test both in the task of predicting the classes at different orientations. The rotated images are obtained by first selecting a random unit vector in $\mathbb{R}^3$ for each point cloud and afterwards we rotate the points by different angles along those axes. We can see the results of the experiment in Figure 6.3. In this small experiment the E(3)-equivariant model improved the results of the non-equivariant one, for both the rotated and non-rotated test sets. The dataset, however, is very small and it is difficult to infer how the performance of the models scales with the size of the dataset, so additional experiments are necessary.

6.5 Conclusions

The aim of this chapter was to concretly show perspectives for equivariant neural networks based on the method of moving frames. More specifically, three possible applications of the method were shown.

The first one is a network invariant to affine contrast changes. A positive point of that network is that it is relatively straightforward to implement, as only the first layer needs to be based on differential invariants, and the following layers can be any CNN layers. It was shown in a simple experiment to overperform the CNN baseline when the constrast of the test set images is changed globally. It was also shown that it works well even when the contrast change is local.

The second of these approaches is a network equivariant to translations, rotations and rescalings. This approach is based on the Scale-Equivariant Gaussian derivative network of (Lindeberg, 2022) by computing the multi-scale Gaussian derivatives from the multi-scale feature maps of the previous layer.

In the last one we wrote in terms of moving frames the approach of (Thomas, 2020) to obtain SE(3)-equivariant neural network layer for point clouds. The particular moving frame we proposed is based on the covariance matrix of a neighborhood of points. Because SE(3)-equivariance required the orientation of normals vectors we also proposed a E(3)-equivariant network which does not depend on the orientation of normal vectors. We tested the proposed model in a simple experiment and obtained results that show a much higher invariance to rotations than the non-rotation-equivariant KPConv Thomas et al. (2019).

This chapter shows the generality of the moving frames approach for deriving invariants and equivariant network layers, and that it can be applied to a variety of input domains and symmetry groups.

6.6 Résumé en Français

L'objective de ce chapitre est de montrer concrètement des perspectives pour les réseaux de neurones basés sur la méthode du repère mobile. Plus particulièrement, trois applications possibles de la méthode ont été montrées.

La première est un réseau invariant par changement de contraste affine. Ce réseau est obtenu à partir de l'application des invariants différentiels du changement de contraste à l'entrée du réseau. Un premier point positif de ce réseau est la simplicité de son implémentation, car seulement la première couche doit être basée sur des invariants différentiels, et les couches suivantes peuvent être n'importe quelle couche de CNN. Nous avons montré dans une expérience simple que ce réseau surpasse un CNN de base quand le contraste change globalement. De plus les invariants marchent mieux qu'une normalisation des niveaux de gris quand le changement de contraste est local.

La deuxième est un réseau équivariant par translation, rotation et changement d'échelle. Cette approche est basée sur le réseau équivariant par changement d'échelle proposé par Lindeberg (2022) en calculant les dérivées Gaussiennes multi-échelle à partir des cartes des caractéristiques multi-échelle de la couche précédente.

Dans la dernière section nous avons réécrit, en termes de la méthode du repère mobile,

l'approche proposée par Thomas (2020) pour obtenir une couche équivariante par SE(3) appliquée aux nuages de points. Le repère mobile que nous avons proposé est basé sur la matrice de covariance d'un voisinage des points. Notre méthode requiert la connaissance des orientations des vecteurs normaux. Pour traiter cette question nous avons aussi proposé un réseau équivariant par E(3) (groupe des rotations, translations et réflexions en 3D) qui ne dépend pas de l'orientation des vecteurs normaux. Nous avons testé les modèles proposés dans une expérience simple et nous avons obtenu des résultats qui montrent un taux d'invariance par rotation beaucoup plus élevé par rapport aux réseaux KPConv qui ne sont pas équivariants par rotation (Thomas et al., 2019).

Ce chapitre montre la généralité de la méthode du repère mobile pour dériver des couches invariantes ou équivariantes. Il montre aussi qu'on peut appliquer cette méthode à une variété de domaines d'entrée et de groupes de symétrie.

Conclusions

The work done in this thesis focused on obtaining neural networks that are equivariant to transformations. There were two main topics that were explored with this end, the scale-equivariant networks based on scale-spaces and multi-scale operators in Chapters 2 and 3 and the group equivariant networks obtained from the method of moving frames from Chapters 4, 5 and 6. Here we discuss the main contributions and their perspectives for future applications.

- **Scale-Equivariant Networks Based on Morphological Scale-Spaces.** The extension of the semigroup scale-equivariant networks Worrall and Welling (2019) presented in Chapter 2 allow for the use of morphological scale-spaces as the lifting operators in the network. It was shown that the semigroup scale-equivariant networks are indeed able to generalize better than CNNs when the scale of the training set is not representative to the scale of the test set, and even though data augmented CNNs have sometimes better generalization results than the scale-equivariant ones, the scale-equivariant network are more robust to changes in the scale of the training set. Moreover, in experiments where the only available information is the shape of the objects and the scale of the training set is not representative of the scale of the test set, it is expected that morphological scale-spaces surpasses the Gaussian ones due to the morphological operator' capacity of preserving the shape in the presence of downsampling and this was confirmed empirically in a synthetic shape segmentation experiment.

  To further explore the advantages of scale-spaces other than the Gaussian one as the lifting layer, it would be interesting to experiment with trainable liftings that can parameterize many types of scale-spaces, such as the Lasry-Lions lifting we presented. Moreover, the semigroup scale-cross-correlation layers have a limiting factor of only dealing with integer scale discretizations, but other methods, e.g. Lindeberg (2022); Sosnovik et al. (2019) can deal with arbitrary scale discretizations, so it should be interesting to see if scale-space lifting layers can be combined with these methods to obtain improved results.

- **Scale-Equivariant U-Net.** The SEU-Net proposed in Chapter 3 is a segmentation network obtained through the application of the U-Net architecture, including its downsampling and upsampling layers, inside a scale-equivariant pipeline. We compare the SEU-Net, U-Net and a scale-equivariant network that does not perform upsampling in the equivariant pipeline, SResNet, in three segmentation experiments where we vary the scale of the test set but keep the scale of the training set: a strand segmentation task, the natural image segmentation task Oxford-IIIT Pet and the cell segmentation experiment DIC-C2DH-HeLa. In those experiments, it was shown that when the U-Net is applied to objects that do not reflect the scale distribution of the training set, it suffers a significant

decrease in performance, while SEU-Net was shown a much lesser decrease in performance in the same situation. When the U-Net is trained with a scale jittering containing the scales of the test sets, its overall performance falls as a result and still generalizes worse than the SEU-Net. Moreover the SEU-Net benefits from scale-jittering by small scale factors. When compared to the SResNet, the SEU-Net is shown to have better results at both scale 1 and unseen scales, justifying not only the use of scale-equivariant layers, but the arrangement of those layers in a U-Net like architecture.

As is the case of scale-equivariant networks with morphological liftings, it would be interesting to generalize the SEU-Net to layers at non-integer scale discretizations, as the discretizations by powers of 2 may be too rough for most applications.

- **Differential Invariant Blocks.** In Chapter 4, the method of moving frames was applied together with Gaussian derivative layers in order to propose neural network blocks based on differential invariant operators. This methodology was applied to obtain neural network blocks for image data that are equivariant to rotations and translations in the plane. The main block of the network, the SE2DIN block, is obtained by computing Gaussian derivatives from the input of the block, followed by the computation of fundamental differential invariants using the Gaussian derivatives and the functional combination of those invariant by a multi-layer perceptron to form new feature maps. The network constructed from these blocks, called SE2DINNet, is reasonably lightweight and the computation of its output takes a number of floating point operations similar to that of a CNN with the same number of parameters and layers. The method was evaluated in the task of rotated digit classification of MNIST-Rot and obtained competitive results when compared to similar models. The main difficulty of training the SE2DINNet was its numerical instability, an activity regularization was used to mitigate the problem but it did not address the cause of the issue.

  It would be interesting to test the SE2DINNet in more complicated tasks, but the numerical issues limits its applications. The method of moving frames is general enough to be applied to many different domains and many different symmetry groups, as was discussed in Chapter 6. Following the leads of that chapter it would be interesting to apply the methodology used for the SE2DINNets to define networks invariant to contrast changes, or more generally networks invariant to group transformations in the color space. Furthermore, scale-equivariant and rotation equivariant networks can be combined by exploiting the scale-equivariance properties of the Gaussian derivatives. Other than the approaches discussed in Chapter 6, some other areas where networks with differential invariants could be applied are in physics informed networks equivariant to Lorentz transformations Bogatskiy et al. (2020) and networks in the Poincaré disk, equivariant to the isometries of the disk for fisheye images Lagrave and Barbaresco (2022).

- **Single-Moving-Frame Networks.** The approach of Chapter 5 was proposed to further mitigate the numerical problems of the differential invariants networks. In the approach of the SE2DIN blocks, the differential invariants are implicitly the result of computing a moving frame from the Gaussian derivatives and applying the moving frame to the derivatives through the prolonged group action, which creates non-linear functions that may cause numerical issues. The proposal of Chapter 5 was to use a single moving frame,

computed from the Gaussian derivatives of the input image, to compute invariants. It was shown that this defines a rotation equivariant neural network and furthermore, the moving frame can be fed as input to the network and the application of the moving frame to the feature maps can be considered a linear operator with respect to the feature maps. A moving frame for the action of translations and rotations in volumes was applied with the methodology described above to obtain a SE(3)-equivariant network for volumes called SE3MovFNet. The networks were tested on datasets of low-resolution medical volumes classification. The SE3MovFNet achieved the state-of-the-art in four of the six tested datasets. It was shown to be perfectly equivariant to rotations by multiples of 90° degrees around the coordinate axes, but in some tasks its results are degraded for other rotations, which may be caused by the rotation of low-resolution of the volumes.

It would be interesting to investigate how the methodology of a single moving frame compares with the SE2DINNets in 2D computer vision tasks. Moreover, the approach of a single moving frame can be used to obtain SE(3)-equivariant for point clouds, as was shown in Chapter 6. This approach can also be applied to the Lorentz transformation-equivariant and the Poincaré disk isometriy-equivariant networks metioned above. Because the moving frames are computed before the rest of the network, a moving frame based on objects other than the jet-space can be investigated for these networks. In particular, the structure tensor Weickert (1998) may provide an alternative to the Hessian in order to find the rotation matrices, which would be interesting as it would depend only on first order derivatives and it also provides a description of the anisotropy of the images or volumes.

Overall these topics provided interesting results that motivate the study of equivariant networks, specially scale-equivariant networks and networks constructed from moving frames.

Bibliography

Abedalla, A., Abdullah, M., Al-Ayyoub, M., and Benkhelifa, E. (2020). The 2st-unet for pneumothorax segmentation in chest x-rays using resnet34 as a backbone for u-net. *arXiv preprint arXiv:2009.02805*.

Alvarez, L., Guichard, F., Lions, P. L., and Morel, J. M. (1993). Axioms and fundamental equations of image processing. *Archive for rational mechanics and analysis*, 123:199–257.

Angulo, J. (2014). Lipschitz regularization of images supported on surfaces using riemannian morphological operators. *HAL hal-01108130v2*.

Blusseau, S., Wielhorski, Y., Haddad, Z., and Velasco-Forero, S. (2022). Instance segmentation of 3d woven fabric from tomography images by deep learning and morphological pseudo-labeling. *Composites Part B: Engineering*, page 110333.

Bogatskiy, A., Anderson, B., Offermann, J. T., Roussi, M., Miller, D. W., and Kondor, R. (2020). Lorentz group equivariant neural network for particle physics. *International Conference on Machine Learning*.

Brockett, R. W. and Maragos, P. (1994). Evolution equations for continuous-scale morphological filtering. *IEEE transactions on Signal Processing*, 42(12):3377–3386.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Calabi, E., Olver, P. J., Shakiban, C., Tannenbaum, A., and Haker, S. (1998). Differential and numerically invariant signature curves applied to object recognition. *International Journal of Computer Vision*, 26(2):107–135.

Cartan, É. (1935). La méthode du repere mobile, la théorie des groupes continus, et les espaces généralisés. *Bull. Amer. Math. Soc*, 41:774.

Chen, H., Liu, S., Chen, W., Li, H., and Hill, R. (2021). Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14514–14523.

Chidester, B., Ton, T.-V., Tran, M.-T., Ma, J., and Do, M. N. (2019). Enhanced rotation-equivariant u-net for nuclear segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Cohen, T. and Welling, M. (2016a). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. Proceedings of Machine Learning Research.

Cohen, T. S., Geiger, M., and Weiler, M. (2019). A general theory of equivariant CNNs on homogeneous spaces. *Advances in neural information processing systems*, 32:9145–9156.

Cohen, T. S. and Welling, M. (2016b). Steerable CNNs. *arXiv preprint arXiv:1612.08498*.

Diop, E. H. and Angulo, J. (2015). Multiscale image analysis based on robust and adaptive morphological scale-spaces. *Image Analysis & Stereology*, 34(1):39–50.

Faugeras, O. (1993). Cartan's moving frame method and its application to the geometry and evolution of curves in the euclidean, affine and projective planes. In *Joint European-US Workshop on Applications of Invariance in Computer Vision*, pages 9–46. Springer.

Fels, M. and Olver, P. J. (1998). Moving coframes: I. a practical algorithm. *Acta Applicandae Mathematica*, 51(2):161–213.

Fels, M. and Olver, P. J. (1999). Moving coframes: II. regularization and theoretical foundations. *Acta Applicandae Mathematica*, 55(2):127–208.

Ghosh, R. and Gupta, A. K. (2019). Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Guttenberg, N., Virgo, N., Witkowski, O., Aoki, H., and Kanai, R. (2016). Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*.

Har-Peled, S. and Mendel, M. (2006). Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heijmans, H. (2002). Morphological scale-spaces, scale-invariance and lie groups. In *International Symposium on Mathematical Morphology*, pages 253–264. Citeseer.

Heijmans, H. J. and van den Boomgaard, R. (2002). Algebraic framework for linear and morphological scale-spaces. *Journal of Visual Communication and Image Representation*, 13(1-2):269–301.

Hubert, E. and Kogan, I. A. (2007). Smooth and algebraic invariants of a group action: local and global constructions. *Foundations of Computational Mathematics*, 7(4):455–493.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jackway, P. T. (1994). Properties of multiscale morphological smoothing by poweroids. *Pattern Recognition Letters*, 15(2):135–140.

Jacobsen, J.-H., Van Gemert, J., Lou, Z., and Smeulders, A. W. (2016). Structured receptive fields in CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619.

Jansson, Y. and Lindeberg, T. (2020). Exploring the ability of CNNs to generalise to previously unseen scales over wide scale ranges. *arXiv preprint arXiv:2004.01536*.

Jenner, E. and Weiler, M. (2022). Steerable partial differential operators for equivariant neural networks. In *International Conference of Learning Representations*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koenderink, J. J. and van Doorn, A. J. (1987). Representation of local geometry in the visual system. *Biological cybernetics*, 55(6):367–375.

Kondor, R. and Trivedi, S. (2018). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. Proceedings of Machine Learning Research.

Lagrave, P.-Y. and Barbaresco, F. (2022). Hyperbolic equivariant convolutional neural networks for fish-eye image processing. *arXiv preprint*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database.

Lee, J. M. (2013). Smooth manifolds. In *Introduction to smooth manifolds*, pages 1–31. Springer.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lindeberg, T. (1998). Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156.

Lindeberg, T. (1999). Principles for automatic scale selection. In *Handbook on Computer Vision and Applications : volume II*, pages 239–274. QC 20111024.

Lindeberg, T. (2022). Scale-covariant and scale-invariant gaussian derivative networks. *Journal of Mathematical Imaging and Vision*, 64(3):223–242.

Liu, R., Lin, Z., Zhang, W., and Su, Z. (2010). Learning pdes for image restoration via optimal control. In *European Conference on Computer Vision*, pages 115–128. Springer.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5048–5057.

Melnyk, P., Felsberg, M., and Wadenbäck, M. (2021). Embed me if you can: A geometric perceptron. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1276–1284.

Olver, P. J. (1993). *Applications of Lie groups to differential equations*, volume 107. Springer Science & Business Media.

Olver, P. J. (1995). *Equivalence, invariants and symmetry*. Cambridge University Press.

Olver, P. J. (2000). Moving frames and singularities of prolonged group actions. *Selecta Mathematica*, 6(1):41–77.

Olver, P. J. (2007). Generating differential invariants. *Journal of Mathematical Analysis and Applications*, 333(1):450–471.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pauwels, E. J., Van Gool, L. J., Fiddelaers, P., and Moons, T. (1995). An extended class of scale-invariant and recursive scale space filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):691–701.

Penaud-Polge, V., Velasco-Forero, S., Angulo, J., et al. (2022). Fully trainable gaussian derivative convolutional layer. In *29th IEEE International Conference on Image Processing (IEEE ICIP)*.

Roerdink, J. B. (2000). Group morphology. *Pattern Recognition*, 33(6):877–895.

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

Ruthotto, L. and Haber, E. (2020). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364.

Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2021). Scale equivariant neural networks with morphological scale-spaces. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 483–495. Springer.

Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022a). Differential invariants for SE(2)-equivariant networks. In *29th IEEE International Conference on Image Processing (IEEE ICIP)*, Bordeaux, France. (Oral Presentation).

Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2022b). Scale Equivariant U-Net. In *33rd British Machine Vision Conference*, London, United Kingdom.

Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2023). Moving frame net: SE(3)-equivariant network for volumes. In Sanborn, S., Shewmake, C., Azeglio, S., Di Bernardo, A., and Miolane, N., editors, *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 197 of *Proceedings of Machine Learning Research*, pages 81–97. PMLR.

Schmidt, M. and Weickert, J. (2016). Morphological counterparts of linear shift-invariant scale-spaces. *Journal of Mathematical Imaging and Vision*, 56(2):352–366.

Shen, Z., He, L., Lin, Z., and Ma, J. (2020). Pdo-econvs: Partial differential operator based equivariant convolutions. In *International Conference on Machine Learning*, pages 8697–8706. Proceedings of Machine Learning Research.

Shen, Z., Hong, T., She, Q., Ma, J., and Lin, Z. (2022). PDO-s3DCNNs: Partial differential operator based steerable 3D CNNs. In *International Conference on Machine Learning*, pages 19827–19846. Proceedings of Machine Learning Research.

Sosnovik, I., Moskalev, A., and Smeulders, A. (2021). Disco: accurate discrete scale convolutions. In *Proceedings of the 32nd British Machine Vision Conference*.

Sosnovik, I., Szmaja, M., and Smeulders, A. (2019). Scale-equivariant steerable networks. In *International Conference on Learning Representations*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Thomas, H. (2020). Rotation-invariant point convolution with multiple equivariant alignments. In *2020 International Conference on 3D Vision (3DV)*, pages 504–513. IEEE.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420.

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

Tuznik, S. L., Olver, P. J., and Tannenbaum, A. (2018). Affine differential invariants for invariant feature point detection. *arXiv preprint arXiv:1803.01669*.

Ulman, V., Maška, M., Magnusson, K. E., Ronneberger, O., Haubold, C., Harder, N., Matula, P., Matula, P., Svoboda, D., Radojevic, M., et al. (2017). An objective comparison of cell-tracking algorithms. *Nature methods*, 14(12):1141–1152.

Van Den Boomgaard, R. and Smeulders, A. (1994). The morphological structure of images: The differential equations of morphological scale-space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1101–1113.

Weickert, J. (1998). *Anisotropic Diffusion in Image Processing*, volume 1. Teubner Stuttgart.

Weiler, M. and Cesa, G. (2019). General E(2)-equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018a). 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, volume 31, pages 10381–10392.

Weiler, M., Hamprecht, F. A., and Storath, M. (2018b). Learning steerable filters for rotation equivariant CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858.

Witkin, A. (1984). Scale-space filtering: A new approach to multi-scale description. In *ICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 9, pages 150–153. IEEE.

Worrall, D. and Brostow, G. (2018). Cubenet: Equivariance to 3d rotation and translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 567–584.

Worrall, D. and Welling, M. (2019). Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems*, pages 7364–7376.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/.

Yang, J., Shi, R., and Ni, B. (2021a). Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., and Ni, B. (2021b). Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*.

Yu, J., Zhang, C., Wang, H., Zhang, D., Song, Y., Xiang, T., Liu, D., and Cai, W. (2021). 3d medical point transformer: Introducing convolution to attention networks for medical point cloud analysis. *arXiv preprint arXiv:2112.04863*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30.

Zhu, W., Qiu, Q., Calderbank, R., Sapiro, G., and Cheng, X. (2019). Scale-equivariant neural networks with decomposed convolutional filters. *arXiv preprint arXiv:1909.11193*.

RÉSUMÉ

Dans le contexte des réseaux de neurones, l'équivariance et l'invariance par des transformations peuvent induire une meilleure généeralisation à de nouvelles données si ces dernières contiennent les symétries correspondantes. En particulier, dans le champs de la vision par ordinateur, la plupart des tâches doivent tenir compte de symmetries géométriques. Ainsi, par exemple, la translation d'un objet dans une tâche de segmentation ne doit pas changer la classe de l'objet.
L'objetif principal de cette thèse est d'explorer et développer des réseaux de neurones qui sont équivariants par rapport à certaines transformations. Les deux principales méthodes qui ont été utilisées sont les réseaux équivariants par changement d'échelle basés sur la correlation-croisée sur le groupe des homothéties, et les réseaux équivariants par l'action d'un groupe de Lie basés sur la méthode des repères mobiles. La première méthode est basée sur la généralisation des réseaux équivariants par un semi-groupe d'échelles qui ont été proposés récemment, où les auteurs utilisent l'espace-échelle Gaussien pour transformer les images en des signaux sur un domaine des échelles et translations. La généralisation proposée dans cette thèse permet d'utiliser un espace-échelle beaucoup plus général. En particulier, les espaces-échelle morphologiques présentent un avantage quand la seule information disponible sur l'objet d'intérêt est sa géométrie. L'équivariance des opérateurs de sous-échantillonnage et suréchantillonnage est étudiée et ceux-ci sont appliqués avec les correlations-croisées d'échelle pour obtenir le SEU-Net, une version de U-Net équivariante par changement d'échelle qui améliore sa généralisation à des échelles non vues lors de l'entraînement.
La méthode du repère mobile est une approche classique pour obtenir des invariants différentiels par l'action d'un groupe de Lie sur une variété. Dans ce travail de thèse, l'approche a été appliquée à la construction d'un réseau de neurones équivariant par rotation et par translation. Le réseau proposé a été testé sur une tâche de classification de chiffres manuscrits tournés, et en dépit des certains problèmes numériques, le réseau a obtenu de bons résultats par rapport à des réseaux équivariants par rotation de taille similaire. Puis, pour éviter les problèmes numériques, un réseau qui utilise un seul repère mobile pour calculer des invariants a été proposé et appliqué pour créer des réseaux équivariants par rotations et translations pour les volumes en 3D. Le réseau a été testé sur un ensemble de bases de données pour la classification de volumes médicaux en faible résolution, et il a obtenu une performance à l'état-d'art dans la plupart des bases de données testées.

MOTS CLÉS

Équivariance, invariance, réseaux de neurones, vision par ordinateur, espaces-échelle.

ABSTRACT

In the context of neural networks, equivariance or invariance to transformations can induce a better generalization to new data as soon as the data is symmetric to the relevant transformations. In particular, in the realm of computer vision most tasks have some kind of geometrical symmetry, for example, a translation of an object in segmentation tasks usually does not change the class of the object.
The main objective of this thesis is exploring and developing neural networks that are equivariant to transformations. The two main frameworks which were used were the scale-equivariant networks based on scale-cross-correlation and the group-equivariant networks based on moving frames. The former is based on the generalization of the scale-semigroup-equivariant networks recently proposed which used the Gaussian scale-space as a way to map images from their original domain to a domain of scales and translations. The generalization proposed in this thesis allows for a much more general class of scale-spaces to be used as liftings and it is shown that morphological liftings are beneficial when only the only information available is the shape of objects. Equivariance of the upsampling and downsampling operators is studied and applied with the scale-cross-correlation to create an architecture similar to the U-Net, the SEU-Net, which was shown to improve generalization to unseen scales of the U-Net.
The method of moving frames is a classical approach to finding differential invariants in manifolds and in the present work it was applied to define neural network blocks that are equivariant to the action of a Lie group. In particular, it was applied to the definition of a neural network equivariant to rotations and translations of images. The proposed network was tested in the classification of rotated digits, and despite its numerical issues, it achieved results competitive with other rotation-equivariant models with similar size. Moreover, in order to deal with the numerical problems a solution which computes invariants from a single moving frame was proposed and applied to create a network equivariant to rotations and translations of 3D volumes. The 3D rotation-equivariant network was applied to tasks of low-resolution medical volume classification and achieved state-of-the-art results for most of the tested datasets.

KEYWORDS

Equivariance, invariance, neural networks, computer vision, scale-spaces.