Representation Learning *with* Structured Invariance

Artem Moskalev

Representation Learning with Structured Invariance

Artem Moskalev

This book was typeset by the author using $\LaTeX\,2_\varepsilon$.

Representation Learning with Structured Invariance

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op woensdag 20 maart 2024, te 10:00 uur

door

Artem Moskalev
geboren te Moskou, Rusland

Promotiecommissie

Promotor:	prof. dr. ir. A. W. M. Smeulders	Universiteit van Amsterdam
Copromotor:	dr. ir. Erik J. Bekkers	Universiteit van Amsterdam
Overige leden:	prof. dr. M. Shah	University of Central Florida
	prof. dr. M. Welling	Universiteit van Amsterdam
	dr. V. Fischer	Bosch
	prof. dr. C.G.M. Snoek	Universiteit van Amsterdam
	dr. P.S.M. Mettes	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

UNIVERSITY
OF AMSTERDAM

UvA - BOSCH
DELTA LAB

The work described in this thesis has been carried out within the the UvA-Bosch Delta lab of the University of Amsterdam.

CONTENTS

1 INTRODUCTION

1.1 FROM KNOWLEDGE TO REPRESENTATION

Real-world problems present themselves in a variety of facets and dimensions, requiring us to adapt quickly to tackle them. To do so, we build knowledge on specific problems which we learn to solve by repetition and knowledge on patterns in problems that we use to develop a solution strategy in general.

The game of chess serves as a good illustration. Following the first pawn move of $e2 - e4$, there emerge between 10^{111} and 10^{123} possible positions that can develop during the game. Each turn a player must explore this enormously large search space to pick an optimal strategy. Such a task, challenging even for modern-day chess engines, seems nearly impossible for humans to grasp. Yet, the 2021 World Chess championship witnessed an average accuracy of chess moves exceeding 97% [1]. This is possible because professional chess players rely on prior knowledge in the form of chess theory. Chess theory provides the guidelines for the game on what tactical goals players should prioritize while identifying others that can be sacrificed. For example, control of the center of the board and development of light chess figures such as bishop and knight as opposed to capturing material advantage at all costs. Therefore, chess theory, although not simple itself, significantly simplifies the search space for a winning strategy.

Language learning is another example. People starting to learn a new language initially face an overwhelming multitude of word forms and cases, each of which seems like an isolated entity governed by its own rule. As the learners progress with learning grammar, they begin to identify the underlying grammatical structures that govern the language. They realize how the multitude of word forms follows from a few specific rules, enabling them to generalize and apply these rules to new vocabulary effortlessly. Like this, grammar knowledge simplifies the understanding of language, providing a framework to navigate and comprehend the intricacies of language structures more with ease.

And a final example is in the realm of music. The intricacies of melodies, chords, and rhythms may seem overwhelming for a beginner learning to play a musical instrument. Nevertheless, as novices learn musical note theory, they begin to recognize the patterns and relationships between musical elements. They realize that different melodies and chord progressions are simply variations of established musical scales and harmonic principles. For example, certain notes within a scale create harmonies that sound pleasing when played together. This significantly simplifies the process of understanding and creating musical compositions.

1 ratio of moves considered strong by a chess engine

Although the tasks of winning in chess, learning a language, or composing a sonata seem drastically different in the skill set they require, we approach all of them in a similar conceptual manner. Relying on prior knowledge, we seek to represent the task in a space where the path toward the solution can be efficiently navigated. Representing a chess game through strategic layouts of a pawn structure and center control allows one to readily see the strength and weaknesses of a position to make a strong move. Representing the language through grammatical rules transforms the chaos of words into a structured system where one can understand the meaning of a sentence even if encountering an unfamiliar lexicon. Representing music through musical theory converts a jumble of notes into a structured composition where a musician can easily navigate to pick the next chord to play. In all those examples, finding a suitable representation is the essential step of solving the task efficiently. *We call the process of finding efficient representations - representation learning.*

Many everyday tasks, jobs, and routines involve some form of representation learning when we approach them. To drive a car, we learn a mental representation of traffic rules to navigate safely, and we use a map representation of a city layout to navigate efficiently. At our jobs, we learn to use infographics in the form of diagrams, flowcharts, or spreadsheets to represent complex processes or data to facilitate communicating them. In social interactions, we learn to interpret body language and facial expressions as representations of people's emotions and intentions to better understand other people. In each of those examples, we turn to representation learning to structure the information and simplify solving the task at hand.

Naturally, in the era of Artificial Intelligence (AI), automated representation learning is becoming the task of great interest and importance. Not only, the AI-driven representation learning is a crucial component for AI-driven problem solving, but it also can serve as a tool allowing to interpret and to explain predictions of an AI-agent [11]. Studying AI-driven representation learning is thus crucial for building trustworthy intelligent systems.

This thesis focuses on the process of representation learning in neural networks.

1.2 REPRESENTATION LEARNING AS A WAY OF EXPLAINING THE DATA

In the thesis, we view representation learning as a model describing the data subject to a specific downstream task. That is, describing a chess position to see how to checkmate an opponent, describing music harmonics to facilitate composing a sonata or describing an image to decide on the categories of objects present.

A useful framework to approach representation learning within this context is the Minimum Description Length principle [61, 143]. Minimum Description Length suggests seeking a model that provides the shortest, hence the simplest explanation of the data. With Minimum Description Length, representation emerges naturally as a condensed summary of the data subject to a downstream task. The best of such a summary would include relevant causal factors that affect the outcome but omits irrelevant information to enable the shortest description [87, 117].

Explaining the game of chess via such high-level structures as figure formations and control over the center allows for a simpler and a shorter explanation of a winning strategy compared to discussing each individual move. This way of structuring the game provides a better representation of chess according to the Minimum Description Length principle. Similarly, using a grammar to explain a language simplifies the description of numerous word forms using a few basic rules. Along the same lines, representing music as a collection of notes and scales readily allows describing millions of songs and melodies as a combination of several fundamental harmonics.

As an example, from the realm of AI, consider the task of image classification with Convolutional Neural Networks (CNNs) [102]. When we train a CNN to classify images, say predicting the type of animal on a photo, a network effectively learns to condense low-level raw pixel data into high-level features such as edges, textures, and shapes [86, 97, 194]. Those features are further condensed into high-level ones like the silhouette or characteristic facial features. The process continues until the data is distilled into a representation from which the type of the animal can be inferred directly. At this stage, the data is distilled into a summary that contains essential factors influencing the classification outcome but omits other irrelevant details, like the pose of an animal in an image, to enable the shortest description.

The Minimum Description Length principle can be seen as a practical realization of Occam's razor, a philosophical principle that advocates for simplicity when drawing hypotheses [150]. From a formal standpoint, Minimum Description Length is also closely related to Kolmogorov complexity [95, 169], which measures the length of the shortest program that can generate the observed data. In essence, the Minimum Description Length principle seeks to find a concise representation of the data while capturing its essential patterns and regularities, akin to compressing the data into a compact form; and Kolmogorov complexity provides a theoretical bound for a minimum amount of information needed to describe the data in the most compact form.

The Minimum Description Length principle underscores the nature of representation learning as a way of finding an efficient description of the data for the purpose of solving the downstream task of interest, be it the task of finding a winning move in chess, choosing a right note to complete musical composition, or classifying objects on images. In this thesis, we take inspiration from the Minimum Description Length perspective to analyze representation learning in neural networks.

1.3 INVARIANCE

The key underlying assumption of the Minimum Description Length principle is that the data comes with information redundant to solve the task of interest. Therefore the word *minimum* in Minimum Description Length, encourages the discard of this redundant information in representation space. The search for the *minimum* can be also understood from a probabilistic point of view. Any assumption introduced into a model has a probability of being wrong, hence it is most reasonable to limit assumptions to the core set, which consists of information strictly necessary for the task of interest. According to the Minimum Description Length principle, a good model is expected to reason on such a core set of assumptions, and only this core set, hence being *invariant* to all other factors of variations.

We are arriving to the concept of *invariance*. In the context of neural networks, a network $f : X \to Y$ is said to be invariant if there exists an invariance set $\mathcal{T}$ such that $f(x) = f(T(x))$ holds for $\forall T \in \mathcal{T}$, where $T : X \to X$. Put simply, invariance implies that a neural network produces the same output when the input undergoes a transformation from the invariance set $\mathcal{T}$. This property allows for generalizing learned patterns by constructing the representation space, where objects that share the same structure but may differ in other aspects, are mapped into the same point. In the task of image classification, for example, two objects of the same category may vary in size, position, and orientation, but a robust neural network would still map those objects into the same representation, filtering out variations irrelevant to the classification task.

In a broader context, invariance shares a common basis with the inductive approach to science [5] that involves deriving general principles from a collection of individual observations. Scientists use this method to build theories or models that describe natural phenomena. The goal is to find principles that hold true in various situations, identifying patterns and rules that are both universal and constant, unaffected by specific conditions. Invariance plays a key role in this process by focusing on elements that consistently impact the phenomenon while excluding factors that do not have a direct influence. This enables the formulated theory to be universally applicable, undeterred by varying circumstances.

For instance, the law of planetary motions applies universally to all celestial bodies orbiting around their stars, irrespective of the specific planet or star involved. Similarly, principles governing chemical reactions, such as rules of chemical kinetics, apply universally for all chemical compounds regardless of specific reaction conditions. Likewise, the law of supply and demand in economics holds true regardless of the specific goods or services being traded.

Along the same lines, when we train a neural network to solve a task, we want it to generalize across a whole range of input variations as long as truly causal factors remain intact. In this thesis, we study invariance as a key property in representation learning with neural networks that allows such generalizations.

Structure in invariance

In the thesis, we narrow our focus to a particular type of invariance called *structured invariance*. This kind of invariance emerges when the transformations that a neural network is invariant to follow an explicit mathematical structure. We choose to focus on the case where such a structure takes the form of a group [83]. By taking the group-theoretical framework, which has a well-established mathematical foundation, we aim for a tool for analyzing and understanding the invariance properties of representation learning in neural networks.

Groups are algebraic structures studied by group theory. In simple terms, a group is a set of elements coupled with an operation that combines any of two of those elements into a third one. Additionally, the group axioms should be met for a set with an operation to constitute a group. Those axioms are discussed in detail in Chapter 2.3.

Groups have practical relevance in describing many natural phenomena. In physics, for instance, the conservation laws are a manifestation of an underlying symmetry described by a group structure [165, 196]. In chemistry, the group theoretical approach is used to describe molecular geometry that governs molecular properties [81, 196]. Even in everyday routines, such as arranging furniture in a room, we unconsciously use the concept of symmetry groups to plan balanced and aesthetically pleasing arrangements. Due to this ubiquity of symmetries, neural networks inevitably face them when applied to real-world problems. It makes the mathematical apparatus of group theory relevant to analyze the invariance properties that emerge in neural networks' representations.

In this thesis, we aim to study representation learning in neural networks through the lens of structure and invariance. We further divide this topic into five research question.

Do neural networks learn structured invariance from the data?
Chapter 2: Investigating properties and limitations of learned invariance

Firstly, we study the invariance neural networks learn from the data. While, multiple prior works have demonstrated the outstanding ability of neural networks to solve tasks that requires structured invariance [13, 124, 132], the properties and limitations of such learned invariance remain largely unexplored. To what extent there is a structure in learned invariance? Does the invariance we observe in network's predictions imply genuinely invariant decision-making in a model? If we train a model to be invariant on a particular task, does it imply that a network will universally this learned invariance across other tasks?

To investigate the properties and limitations of learned invariance, in Chapter 2 we compare the invariance learned from the data to the oracle invariance achieved with group-invariance built into a model architecturally through the weight-tying [34]. While the weight-tying approach provides guaranteed invariance regardless of the operating conditions, we demonstrate that the invariance learned by unconstrained models is strongly conditioned on the input data. We further conduct perturbation analysis for networks with learned invariance revealing that even if a network learns to correctly classify samples on a group orbit, the underlying decision-making process does not attain genuine invariance. Instead, models without weight-tying constraints choose to learn a separate set of features for each of the transformations from a group orbit, even when invariance is enforced by strong data augmentation. As a result, the quality of learned invariance deteriorates quickly when applied out-of-distribution.

Can we infer invariance structure from learned representations?

Chapter 3: Extracting learned invariance by analyzing learned Lie group generators

Most of the existing methods to analyze invariance in neural networks focus on analyzing the invariance with respect to a group of transformations specified a priori [57, 59, 104]. This requires prior knowledge of the transformation group which is not always available in practice. In this research question, we focus on this limitation and ask if we can analyze the invariance learned in neural networks without such prior knowledge. Can we quantify the amount of structured invariance a neural network learns without exactly specifying the transformation group? Can we retrieve the transformation group to which a model has learned to be most invariant to?

To this end, we focus on Lie group invariance, which unifies a wide range of real-world continuous transformations like rotation, scaling, or translation. From here, we analyze the invariance criterion for Lie groups. We demonstrate that the learned invariance can be retrieved explicitly by solving a simple matrix equation with respect to a generator of the symmetry group. We further demonstrate that this approach allows evaluating learned invariance without any prior knowledge of the transformations in the data. Additionally, it provides a more nuanced analysis of the learned invariance as it allows to split the invariance error into invariance bias and variance. Invariance bias and variance provide a more nuanced understanding of the structure of learned invariance.

Can we improve contrastive representation learning with more structured supervision?

Chapter 4: Contrastive representation learning with structured assignments

In this research question, we focus on the contrastive approach to representation learning. Contrastive learning aims to train useful representation by contrasting pairs of samples from different views [82, 110, 133], aiming to train invariance to non-semantic variations in the data. The contrastive approach to representation learning has proven effective in a wide range of tasks ranging from visual [7,28,193] and audio domains [133] to natural language [50] and molecules [174]. Here, we investigate contrastive learning through the lens of invariance. The emerging question is what are the limitations of the invariance learned with contrastive approach? Can we improve contrastive representation learning by bringing more structure into the underlying invariance learning method?

We start by identifying the limitation of a standard contrastive learning methodology. We show that the pairwise contrast between data samples fails to capture both intra-set and inter-set structure, where the sets are formed as transformed views of the data. This inherently limits the information content of the supervisory signal available to train invariance. To address this, we propose a novel contrastive learning method based on combinatorial quadratic assignment theory [23, 51]. Different from pairwise contrastive learning, our method leverages both the intra-set and inter-set structure of transformed data views, which provides a more informative supervisory signal. We conduct experiments and show that bringing this additional structure to the supervision improves the quality of learned representations.

Can we improve object localization with structured representations?

Chapter 5: Scale equivariant representations for robust visual object tracking

Next, we turn to the practical advantage of structure and invariance in representation learning. For that, we consider the task of visual object tracking, which requires localizing specified objects throughout the video sequence. Object tracking is challenging because a target may undergo changes in scale, pose, or illumination, apart from less structured variants due to the weather, albedo change, and camera noise [33]. Overcoming these challenges requires a tracker to learn representations with a high degree of invariance to all those transformations, while still preserving the information that allows to discriminate the target against other objects in a sequence. With this, the task of object tracking provides a challenging environment to evaluate algorithms for robust representation learning.

In this research question, we ask if we can improve object tracking with structured representations. In particular, we focus on the scaling transformation, which as we demonstrate is one of the most challenging for the network to learn. We address this challenge by building scale equivariance into the model's representations a priori. This provides guaranteed processing of the scale regardless of the training. Building scale equivariance directly into the model is also beneficial as it saves the model's capacity to learn it from the data. We experimentally demonstrate the power of scale-equivariant networks to learn more robust representations, which results in more reliable tracking even when factors of variations other than scale are present.

Can we improve object localization with structured invariance prior?

Chapter 6: Relational prior for tracking groups of objects

In Chapter 6, we further delve into applying structured invariance to improve representation learning in object tracking. In the previous chapter, we focused on building the rigid structure in the form of equivariance into model's representations. Although it provides the advantage of guaranteed processing of the pre-defined transformation group, it can also over-constraint a model in cases, when the data does not exactly exhibit the exact symmetry with respect to this group. To address this, we explore the possibility of encouraging structured representations while simultaneously providing a model with the flexibility to diverge from this framework when the data does not align with the pre-established structure.

An instance of the invariance that holds locally, but not globally is spatial and temporal relations between objects in tracking. Two individuals may move together before diverging due to an obstacle. In this case, the walk-together relation acts as the invariant feature in the first part of the sequence but would over-constraint the model in the second part. To capture this dynamic invariance, we propose augmenting a neural network with a soft architectural prior, which encodes the structure of the desired invariance while allowing flexibility to diverge from it when necessary. To this end, we develop the relation encoding that can be plugged into a neural network to encode relations between objects. Relation encoding operates as a message passing on a dynamic spatio-temporal graph of objects to compute relational information, which is then aggregated into objects' representations using the self-attention mechanism [167]. The self-attention provides flexibility to consider different objects together or to treat them individually when beneficial. Experimental results on tracking benchmarks demonstrate that our method enhances tracking quality, especially in challenging cases when objects are occluded or move in a dense crowd.

CO-AUTHORSHIP AND ROLES

The thesis is composed of the following original contributions.

CHAPTER 2 This chapter is based on "On genuine invariance learning without weight-tying" [123]. Published in *2023 International Conference on Machine Learning workshop on Topology, Algebra, and Geometry in Machine Learning (ICML-TAGML)* by Artem Moskalev, Anna Sepliarskaia, Erik J. Bekkers and Arnold Smeulders.

Contributions of authors:

- Artem Moskalev: all aspects
- Anna Sepliarskaia: theoretical insights
- Erik J. Bekkers: theoretical insights and writing
- Arnold Smeulders: supervision and insights

CHAPTER 3 This chapter is based on "LieGG: Studying Learned Lie Group Generators" [124]. Published in *2022 Conference on Neural Information Processing Systems (NeurIPS)* by Artem Moskalev, Anna Sepliarskaia, Ivan Sosnovik, Arnold Smeulders **(Spotlight & Lightning Talk)**.

Contributions of authors:

- Artem Moskalev: all aspects
- Anna Sepliarskaia: theoretical insights
- Ivan Sosnovik: technical implementation
- Arnold Smeulders: supervision and insights

CHAPTER 4 This chapter is based on “Contrasting quadratic assignments for set-based representation learning” [127]. Published in *2022 European Conference on Computer Vision (ECCV)* by Artem Moskalev, Ivan Sosnovik, Volker Fischer, Arnold Smeulders.

Contributions of authors:

- Artem Moskalev: all aspects
- Ivan Sosnovik: technical implementation
- Volker Fischer: theoretical insights
- Arnold Smeulders: supervision and insights

CHAPTER 5 This chapter is based on “Scale Equivariance Improves Siamese Tracking” [155]. Published in *2021 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* by Ivan Sosnovik*, Artem Moskalev*, Arnold Smeulders.

Contributions of authors:

- Artem Moskalev: theoretical and technical implementation
- Ivan Sosnovik: all aspects
- Arnold Smeulders: supervision and insights

CHAPTER 6 This chapter is based on “Relational Prior for Multi-Object Tracking” [125]. Published in *2021 International Conference on Computer Vision workshop on Visual Inductive Priors (ICCV-VIPriors)* 2021 by Artem Moskalev, Ivan Sosnovik, Arnold Smeulders **(Oral)**.

Contributions of authors:

- Artem Moskalev: all aspects
- Ivan Sosnovik: technical implementation
- Arnold Smeulders: supervision and insights

ADDITIONAL CONTRIBUTIONS

Additional contributions made during the PhD program that are not foundational to this thesis include:

- “Two is a crowd: tracking relations in videos” [126]. Preprint 2021 by Artem Moskalev, Ivan Sosnovik and Arnold W. M. Smeulders.

- “Learning to Summarize Videos by Contrasting Clips” [152]. Preprint 2023 by Ivan Sosnovik, Artem Moskalev, Cees Kaandorp and Arnold W. M. Smeulders.
- “DISCO: Accurate Discrete Scale Convolutions” [153]. Published in *2021 The British Machine Vision Conference (BMVC)* by Ivan Sosnovik, Artem Moskalev and Arnold W. M. Smeulders (**Best Paper Award**).
- “How To Transform Kernels for Scale-Convolutions” [154]. Published in *2021 International Conference on Computer Vision (ICCV) 2nd Visual Inductive Priors for Data-Efficient Deep Learning Workshop* by Ivan Sosnovik, Artem Moskalev and Arnold W. M. Smeulders.

2

INVESTIGATING PROPERTIES AND LIMITATIONS OF LEARNED INVARIANCE

2.1 INTRODUCTION

The ability to abstract from irrelevant details and focus on core aspects is a foundational property of intelligent systems. Invariance, a crucial step of this abstraction process, enables neural networks to recognize patterns regardless of their transformations. Achieving effective invariance is vital for the robust performance of deep learning models.

There exist two approaches for invariance in neural networks: invariant weight-tying and learning invariance from data. Networks with built-in invariant weight-tying [10, 34, 156, 176, 179, 180] offer *genuine* invariance, but require knowledge of geometrical priors and incurs high computational and memory costs [153, 154]. Alternatively, neural networks can learn invariance directly from data. Recent works [13, 124, 132] demonstrate that neural networks successfully learn invariant priors without any architectural modifications. However, the nature of learned invariance remains largely unexplored, particularly regarding whether it resembles the genuine invariance of weight-tying methods at any level. Consequently, this raises concerns about how much we can rely on the learned invariance when operating conditions evolve. In this work, we investigate the properties of learned invariance to better understand its potential and limitations.

To investigate properties of the learned invariance, we adopt the group theoretical perspective and analyze invariance learning without weight-tying constraints. Firstly, we analyze the saliency maps of no weight-tying networks with learned invariance. We demonstrate that even when such networks learn to correctly classify samples on a group orbit, the underlying decision-making process does not attain genuine invariance, see Figure 1. Instead of learning genuinely invariant weight-tying, unconstrained networks choose to learn a separate set of features for each of the transformations from a group orbit, even when invariance is enforced by strong data augmentation. This results in learned invariance being strongly conditioned on the input data. Consequently, the effectiveness of learned invariance degrades rapidly when operating conditions evolve, e.g. under input distribution shift. This renders neural networks with learned invariance less reliable.

Secondly, we tackle the problem of aligning learned invariance with the genuine invariance of weight-tying networks. To do so, we propose several measures to quantify the invariance error; we next use those measures to regularize the task loss to promote genuine invariance learning. We conduct experiments with rotation and translation groups, and we show that the proposed regularization significantly aligns learned invariance

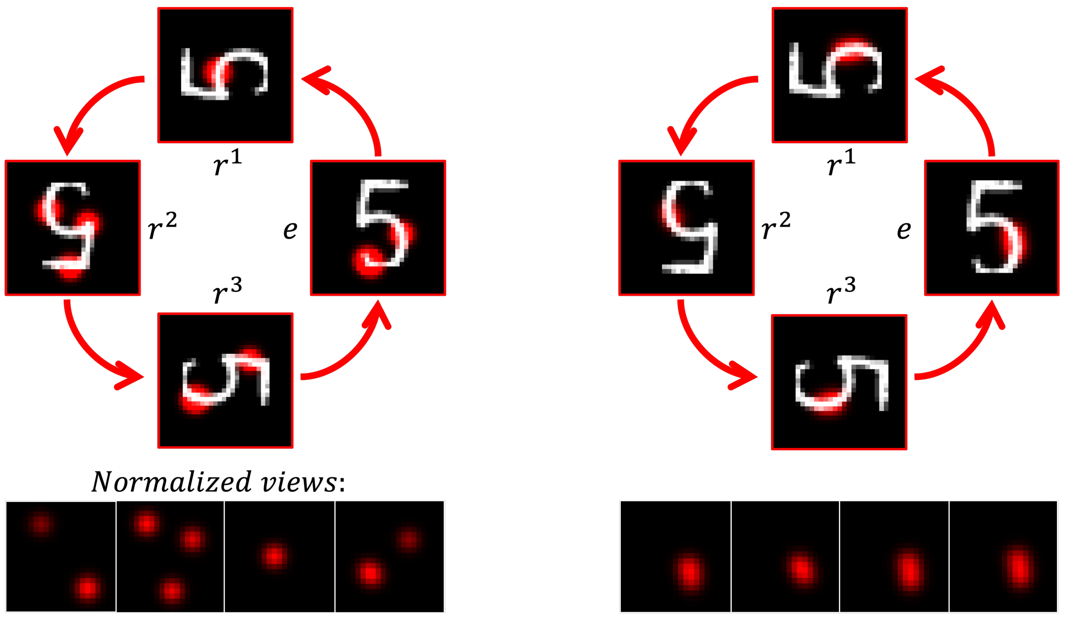


Figure 1: ***Left:*** *The network learns a separate set of features for each of the orientations as indicated by divergent saliency maps.* ***Right:*** *saliency maps of the networks with group-invariant weight-tying. In both cases, predicted class distributions are identical for all of the orientations.* ***Bottom row:*** *saliency maps normalized to a common orientation.*

with the genuine invariance achieved through the weight-tying. However, the alignment also induces performance decay on a downstream task. This presents a new challenging problem of achieving genuine invariance by learning through data augmentation and specialized losses, while also maintaining the downstream task performance.

Thirdly, we investigate the performance decay under the learned invariance. To this end, we analyze the training dynamics of the invariance error minimization from the perspective of the gradient flow. We show that minimizing the invariance error without weight-tying implicitly promotes attaining the invariance to a certain group of transformations by reducing the sensitivity to *any* input perturbation. This has an effect similar to training a network with a large weight decay, which motivates the performance drop. We conduct experiments and demonstrate that this phenomenon holds for various transformations and various forms of invariance error minimization.

To sum up, we make the following contributions:

- We demonstrate that data-driven invariance learning fails to learn genuine invariance as in weight-tying networks.
- We show that it is possible to attain genuine invariance through invariance regularization, but at the cost of the downstream task performance.
- We attribute the performance decay under learned invariance to the training dynamics of the invariance error minimization, which constrains the sensitivity of a network to input perturbations in general.

2.2 RELATED WORK

WEIGHT-TYING INVARIANCE Weight-tying is the approach for invariance in neural networks that is based on the concept of group equivariant networks [34]. Group equivariant networks explicitly embed equivariance, or invariance as a special case, for specific transformation groups into a network architecture. The principle traces back to convolutional networks [102] which incorporate translation symmetry. The scope of equivariant networks has since expanded to include other transformations such as rotations [34, 84, 176, 180], permutations [192], and scaling [10, 153–156, 179]. Another line of work focuses on advancing group equivariant networks by enabling them to learn symmetries directly from the data [4, 43, 147, 200]. This allows the model to adjust to specific symmetries present in the training dataset, eliminating the need for prior knowledge of geometrical priors. Yet, these methods still require modifying the architecture to train invariance.

In this work, we treat the weight-tying methods as oracle invariance learners and investigate whether networks without specific architectural modifications can learn the degree and quality of invariance comparable to the weight-tying approaches.

DATA-DRIVEN INVARIANCE LEARNING Another approach for achieving invariance is to learn it directly from the data. Recent and earlier works [13, 57, 99, 104, 124] demonstrate that neural networks can learn invariance without relying on specialized architectural modifications. Additionally, training with data augmentation has long been seen as a method to increase invariance of a model for input transformations [38, 137, 149]. Invariance learning that does not require specialized architectural modification is advantageous as it does not incur additional memory or computational costs. However, the nature of the learned invariance and its comparability to the genuine invariance obtained through the weight-tying remains an open question. The properties and reliability of such learned invariance are not well understood, which motivates the study in this paper.

2.3 LEARNING INVARIANCES FROM DATA

We take a group-symmetry perspective on data-driven invariance learning when the downstream task is classification. That is to say, we define a set of transformations to be a symmetry group a network needs to learn to be invariant to when classifying input signals. We start by briefly introducing group symmetry and invariance.

2.3.1 *Group symmetry*

GROUP A group $\langle \mathcal{G}, \circ \rangle$ is a set $\mathcal{G}$ with a group binary operation $\circ$ called the group product. For convenience, it is common to simplify the notation $a \circ b$ to ab. The group product combines two elements from $\mathcal{G}$ to a new element so that the following group axioms are satisfied. *Closure:* for all $a, b \in \mathcal{G}$, the element $ab \in \mathcal{G}$. *Associativity:* for all $a, b, c \in \mathcal{G}$, $(ab)c = a(bc)$. *Identity:* there is an element $e \in \mathcal{G}$ such that $ea = ae = a$ for every element $a \in \mathcal{G}$. *Inverse:* for each $a \in \mathcal{G}$ there exist $a^{-1} \in \mathcal{G}$ such that $a^{-1}a = aa^{-1} = e$.

GROUP ACTIONS & SYMMETRY Group actions are a way of describing symmetries of objects using groups. A group action of a group $\mathcal{G}$ on a set $\mathcal{X}$ maps each element $g \in \mathcal{G}$ and each element $x \in \mathcal{X}$ to an element of $\mathcal{X}$ in a way that is compatible with the group structure. In other words, $ex = x$ and $(g_1 g_2)x = g_1(g_2 x)$ for any $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$.

GROUP-INVARIANCE Group-invariance is a property of a function $f : \mathcal{X} \to \mathcal{Y}$ under a group action from a group $\mathcal{G}$. A function f is said to be group-invariant if $f(gx) = f(x)$ for $g \in \mathcal{G}$. This means that the value of f at x is unchanged by the action of any group element.

GROUP ORBIT The group orbit of an element $x \in \mathcal{X}$ under a group action from a group $\mathcal{G}$ is the set of all points in $\mathcal{X}$ that can be reached by applying the group action on x. More formally, the orbit of x is defined as the set $O_x = \{gx | g \in \mathcal{G}\}$. This concept encapsulates the idea that the group action can move the element x around within the set, and the orbit describes all the possible positions x can be moved to by the group action.

2.3.2 *Measuring learned invariance*

Next, we explain how to measure group-invariance learned by a neural networks from the data. We assume we are given a neural network $f : \mathcal{X} \to \mathcal{Y}$ that maps inputs to logits, a group $\mathcal{G}$ and the dataset $\mathcal{D}$. We define three types of measures: *(i) predictive distribution invariance* to measure the average change of a network's output distribution when a symmetry transformation is applied, *(ii) logit invariance* to measure the change of raw network's logits and *(iii) saliency invariance similarity* to evaluate the consistency of network's decisions under group transformations.

PREDICTIVE DISTRIBUTION INVARIANCE Since the downstream task of interest is classification, it is natural to measure the invariance by evaluating the shift of the predictive distribution when transformations from a group orbit are applied. Practically, we can utilize Kullback–Leibler divergence between output softmax-distributions of $f(x)$ and $f(gx)$. With this, we can write the predictive distribution invariance error DI_f:

$$DI_f(\mathcal{D}, \mathcal{G}) = \sum_{x \sim \mathcal{D}} \sum_{g \sim \mathcal{G}} D_{KL}(u_x \parallel q_{gx}) \tag{2.1}$$

where u_x and q_{gx} denote the *softmax* applied to the logits $f(x)$ and $f(gx)$ respectively.

Since DI_f operates directly on the level of predictive distributions, it is the most useful to evaluate the invariance tackled to the downstream classification task.

LOGIT INVARIANCE Next, we define the logit invariance error to measure the shift of raw logits under group actions. Practically, we utilize average squared L_2 distance between the logits $f(x)$ and $f(gx)$:

$$LI_f(\mathcal{D},\mathcal{G}) = \sum_{x\sim\mathcal{D}}\sum_{g\sim\mathcal{G}}\frac{1}{2}\|f(x) - f(gx)\|_2^2 \tag{2.2}$$

Note that the logit invariance error is a more strict invariance measure compared to $DI_f(\mathcal{D},\mathcal{G})$. This is due to a scalar addition invariance of the predictive softmax-distribution. That means $LI_f(\mathcal{D},\mathcal{G}) = 0$ implies $DI_f(\mathcal{D},\mathcal{G}) = 0$, but not vice versa. With this, the logit invariance error is the most useful to characterize the absolute invariance of a function to group transformations regardless of a particular downstream task.

SALIENCY INVARIANCE SIMILARITY Lastly, we propose saliency invariance similarity SI_f to measure the consistency of the decision-making process of a neural network under input transformations. Let $m_f : \mathcal{X}\times\mathcal{Y} \to \mathcal{S}$ be a saliency map function for the network f [128,157]. We then compute the similarity between $m_f(x)$ and $g^{-1}m_f(gx)$, where g^{-1} is needed to ensure a common orientation of the saliency maps. Practically, we adopt the cosine similarity and compute the average saliency similarity as:

$$SI_f(\mathcal{D},\mathcal{G}) = \sum_{x\sim\mathcal{D}}\sum_{g\sim\mathcal{G}}\frac{m_f(x)\cdot g^{-1}m_f(gx)}{\|m_f(x)\|_2\|g^{-1}m_f(gx)\|_2} \tag{2.3}$$

The saliency invariance similarity $SI_f(\mathcal{D},\mathcal{G})$ reflects how much the direction of the most important features, that a network bases its decisions on, change under transformations from a group orbit. Saliency invariance similarity differs from the previous two metrics as it considers the structure of the input data and not just the output of the network. This makes SI_f particularly useful to understand how group transformations alter a network's internal decision-making process.

2.3.3 *Invariance regularization*

CONSTRAINED INVARIANCE LEARNING We next consider the task of facilitating learning invariances from the data. The natural way to do so is to optimize the task performance subject to a low invariance error. Practically, with the dataset $\mathcal{D}$ and a group of interest $\mathcal{G}$, invariance learning boils down to the constrained optimization approach $\min_\theta \mathcal{L}_f(\mathcal{D})$ *s.t.* $I_f(\mathcal{D},\mathcal{G}) = 0$; then, to train a neural network, we can simply optimize the relaxation:

$$\min_\theta \mathcal{L}_f(\mathcal{D}) + \nu I_f(\mathcal{D},\mathcal{G}) \tag{2.4}$$

where θ denotes the parameters of f, $\mathcal{L}_f$ is a downstream task loss functions, $I_f(\mathcal{D},\mathcal{G})$ is an invariance regularizer with respect to the group $\mathcal{G}$ and ν regulates how much invariance we want to achieve at the training. Practically, we observed that using the logit invariance error as a regularizer provides better overall invariance and accuracy than other forms of the invariance error, see Section 2.4.3.

Adding an invariance-regularizer to the original loss yields a simple approach to facilitate data-driven invariance learning. We experimentally demonstrate that invariance regularization significantly improves the quality of learned invariance, closing the gap with the genuine invariance of weight-tying methods. However, we also observe that the improvement in the quality of invariance comes at the cost of downstream task performance, as we demonstrate in Section 2.4.3.

2.3.4 *Invariance-induced spectral decay*

In order to analyze the causes of performance decay under the invariance error minimization, we analyze the training dynamics of the learned invariance through the lens of its gradient flow. We show that a neural network opts for achieving the invariance to a particular transform group by reducing the sensitivity to any input variations. We use a maximum singular value $\sigma_{\max}$ of network's weights as a sensitivity measure [91, 190]; and we analyze the gradient flow for the logit invariance error with a class of linear neural networks. We firstly show that the logit invariance error minimization implicitly constrains the maximum singular value of network's weights, thereby reducing its input sensitivity. Then, we experimentally demonstrate that this result also holds for more complex neural networks and the various forms of invariance errors.

Consider a linear neural network $h(x) = Wx$. Without loss of generality, we analyze the sensitivity to the action of a single group element g, instead of the full orbit of the group $\mathcal{G}$. Let G be a linear representation of the group acting on x and consider invariance error minimization over t steps.

Proposition 1 (*Invariance-induced spectral decay*)**.** *Logit invariance error minimization implies $\sigma_{max}(W(t)) \leq \sigma_{max}(W(0))$ when $t \to \infty$.*

Proof. The optimization of the parameter matrix W takes the form of $W^{t+1} = W^t - \alpha \nabla LI^t$, where $\nabla LI^t = W^t(x - Gx)(x - Gx)^T$ is a gradient of the logit invariance error (Equation 2.2) at the time step t.

Let $\epsilon = x - Gx$ and $\Sigma = \epsilon\epsilon^T$. With the infinitesimally small learning rate α, we can write the gradient flow of W as:

$$\frac{d}{dt}W = -W\Sigma \tag{2.5}$$

For a fixed Σ we can solve the gradient flow above analytically as:

$$W(t) = W(0)\exp\left(-\Sigma t\right) \tag{2.6}$$

Next, we consider a maximum singular value $\sigma_{\max}(W) = ||W||_2$, when the model is trained, i.e. $W(t)$ with $t \to \infty$. Applying Cauchy–Schwarz we can write:

$$||W(t)||_2 \leq ||W(0)||_2 ||\exp\left(-\Sigma t\right)||_2 \tag{2.7}$$

With a spectral decomposition $\Sigma = U\Lambda U^T$, we can write $||\exp\left(-\Sigma t\right)||_2 = ||\exp\left(-\Lambda t\right)||_2$. Since Σ is a rank-one matrix, it contains all zero eigenvalues except of the one, which equates to $\lambda_{\max}(\Sigma) = \epsilon^T\epsilon$. Thus, eigenvalues of the matrix $\exp\left(-\Lambda t\right)$ are all ones

$\mathcal{G}$	Model	Acc. (%)	$LI \Downarrow$	$DI \Downarrow$	$SI \Uparrow$
$\mathbb{R}_4^2$	`WT`	94.6 ± 0.1	0.00 ± 0.0	0.0 ± 0.0	1.00 ± 0.00
	`DA`	94.0 ± 0.5	98.6 ± 5.4	0.3 ± 0.1	0.17 ± 0.04
	`IR`	87.9 ± 0.8	0.02 ± 0.0	0.0 ± 0.0	0.95 ± 0.03
$\mathbb{T}_3^2$	`WT`	96.6 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	1.00 ± 0.0
	`DA`	96.2 ± 0.2	50.8 ± 6.8	0.1 ± 0.0	0.57 ± 0.08
	`IR`	93.1 ± 0.1	0.01 ± 0.0	0.0 ± 0.0	0.95 ± 0.07

Table 1: Classification accuracy and invariance for the data augmentation [`DA`], weight-tying [`WT`], and the model trained with the logit invariance error as the regularizer [`IR`] on the Transformed-MNIST dataset. LI - logit invariance; DI - predictive distribution invariance; SI - saliency invariance similarity.

except of the eigenvalue, which equates to $\lambda_\epsilon(t) = \exp(-t \cdot \epsilon^T \epsilon)$. Note that $\lambda_\epsilon(t) \leq 1$, hence $\|\exp(-\Lambda t)\|_2 = 1$. Plugging into Equation 2.7 gives $\|W(t)\|_2 \leq \|W(0)\|_2$ with $t \to \infty$.

□

This reveals the non-increasing spectral norm constraint that invariance error minimization induces. Also, initialization routines for W, e.g. [56, 66], yield small $\|W(0)\|_2$ at the beginning of the training, further restricting the sensitivity of a network when optimizing for the low invariance.

2.4 EXPERIMENTS

In this section, we experimentally investigate the properties of learned group-invariance. As groups of interest we choose the $\mathbb{R}_4^2$ group of 4-fold rotations and the $\mathbb{T}_3^2$ group of 3-fold cyclic translations along the x-axis. We examine how well the learned invariance is aligned with the downstream task performance and the genuine invariance of weight-tying methods. Then, we analyze the reliability of the learned invariance under the data distribution drift. Lastly, we investigate the invariance-induced spectral decay phenomenon for various forms of the invariance error.

Implementation details

DATASETS We construct the Transforming-MNIST and Transforming-FMNIST datasets. Both dataset consist of MNIST and F-MNIST [185] (28×28 black an white images of clothing categories) with $\mathbb{R}_4^2$ or $\mathbb{T}_3^2$ group transformations applied. We additionally leave out the digit 9 from the Transforming-MNIST dataset to avoid the confusion with 6, when studying the rotation invariance. We also leave out the last class of the Transforming-FMNIST to make number of classes equal to the Transforming-MNIST. We extend the resolution from 28×28 to 36×36 by zero-padding data samples. We use $10k/50k/2k$ splits for *train / test / validation*. The datasets are normalized to zero mean and unit standard deviation.

MODELS We employ 5-layer perceptron with `ReLU` non-linearities and the hidden dimension of 128, resulting in total of 230*k* parameters. For the group-invariant model, we utilize group weight-tying with a pooling over a group to achieve the invariance. We only utilize group-invariant weight-tying for the first layer of a network.

TRAINING DETAILS We train all models for 300 epochs using Adam optimizer with the batch size of 512 and the learning rate of 0.0008. For all models, we use data augmentation with transformations randomly sampled from a group of interest. For invariance regularization, we employ logit invariance error as a regularizer $\mathcal{I}_f$. We tune the weighting of the regularizer, such that the resulting saliency invariance similarity $SI_f \geq 0.95$. Final models are selected based on the best validation accuracy.

SALIENCY MAP FUNCTION $\boldsymbol{m_f}$ The choice of the saliency map function to compute saliency invariance score is a hyper-parameter. We employ the following procedure to generate saliency maps of a network. First, we accumulate absolute values of integrated gradients [157] with respect to the target class prediction. Second, we threshold the values that are less than 0.9 of a maximum value of the absolute integrated gradient. Finally, we apply a Gaussian filter with a kernel size of 3 and a standard deviation of 1 to smooth the resulting saliency map.

2.4.1 *Learning invariance from the data*

In this experiment, we compare models that learn invariance with data augmentation [`DA`] and invariance regularization [`IR`], and models that have the invariant weight-tying built-in [`WT`]. We evaluate the models by the classification accuracy, the logit and distribution invariance errors, and also by the saliency invariance similarity. The results are reported in Table 1. All results are averaged over 4 common random seeds.

`WT` The networks with group-invariant weight tying deliver the highest classification accuracy in both $\mathbb{R}^2_4$ and $\mathbb{T}^2_3$ scenarios. We thus treat the weight-tying models as a performance upper-bound and an oracle for the genuine invariance when further analyzing models with invariance learned.

`DA` We observe that the models trained with data augmentation fail to learn genuine group invariance as indicated by high logit invariance error LI_f and lower saliency similarity score SI_f. Interestingly, these models still provide moderately low predictive distribution invariance error DI_f and high classification accuracy under group transformations. This implies that *neural networks can learn to solve an invariant task without learning a genuinely invariant decision making-process.*

To visualize this phenomenon, we depict the T-SNE of the latent space of the model with learned $\mathbb{R}^2_r$ invariance (Figure 2), and we trace different orientations of a sample in the latent space. We observe that different orientations of one sample can land far away from each other in the representation space, but still within the boundaries of its class. When such configuration fully satisfies the downstream task objective, there is apparently no reason for a network to learn genuine invariance.

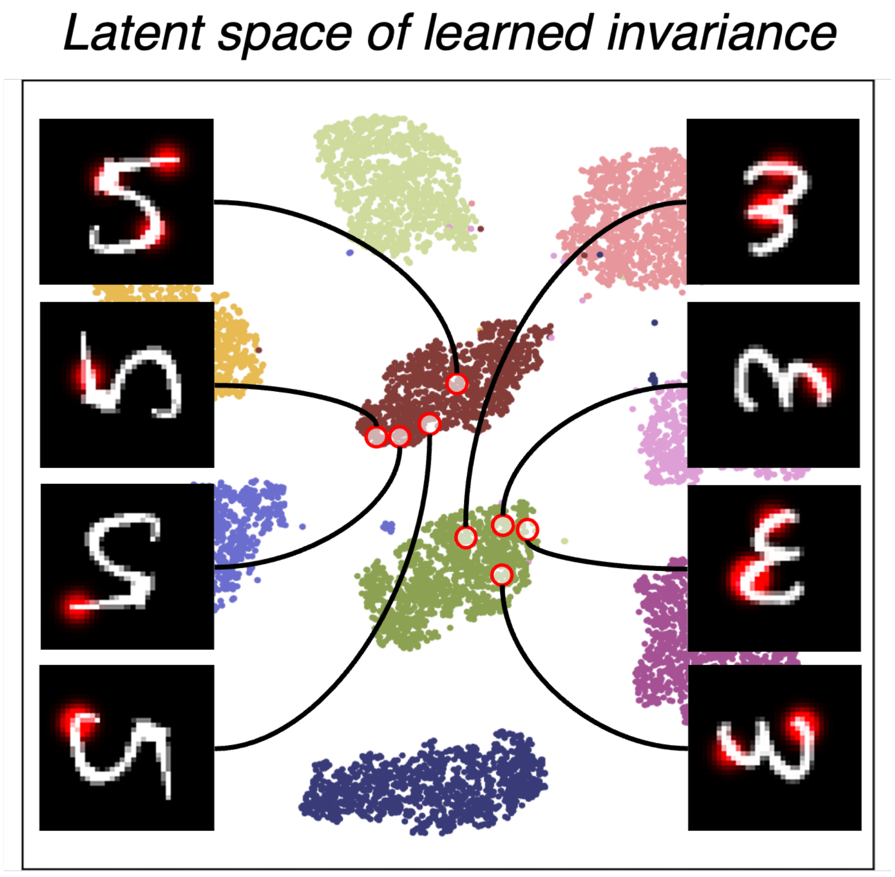


Figure 2: T-SNE of the representations in the pen-ultimate layer of the model with $\mathbb{R}_4^2$ learned invariance. Different orientations of a single sample are mapped to distant points in the latent space. The saliency of the network is highlighted by the red regions in the images.

IR The models trained with invariance regularization achieve low logit and distribution invariance errors on par with the weight-tying models. Also, high saliency invariance similarity indicates that invariance regularization guides a model towards learning genuinely invariance decision-making process. In Figure 3, we visualize examples of saliency maps of the network with $\mathbb{R}_4^2$ invariance learned by data augmentation and invariance regularization. In contrast to the saliency maps of the model trained solely with data augmentation, saliency maps of the model with invariance regularization are well-aligned over the group orbit.

2.4.2 *Reliability of learned invariance*

We next investigate the reliability of the models with the learned invariance when operating conditions evolve. We simulate changing operating conditions as the data distribution drift from Transforming-MNIST to Transforming-FMNIST datasets. Practically, we linearly interpolate between those two dataset as $\mathcal{D}_{1\to2}(\beta) = (1-\beta)\mathcal{D}_1 + \beta\mathcal{D}_2$ to obtain the dataset with the drift degree of β. We compare the models by measuring the invariance error and the accuracy drop ratio on the drifted dataset. The accuracy drop ratio on the drifted dataset $\mathcal{D}_{1\to2}(\beta)$ is computed as $\mathrm{Acc}(\mathcal{D}_{1\to2}(\beta))/\mathrm{Acc}(\mathcal{D}_1)$, where

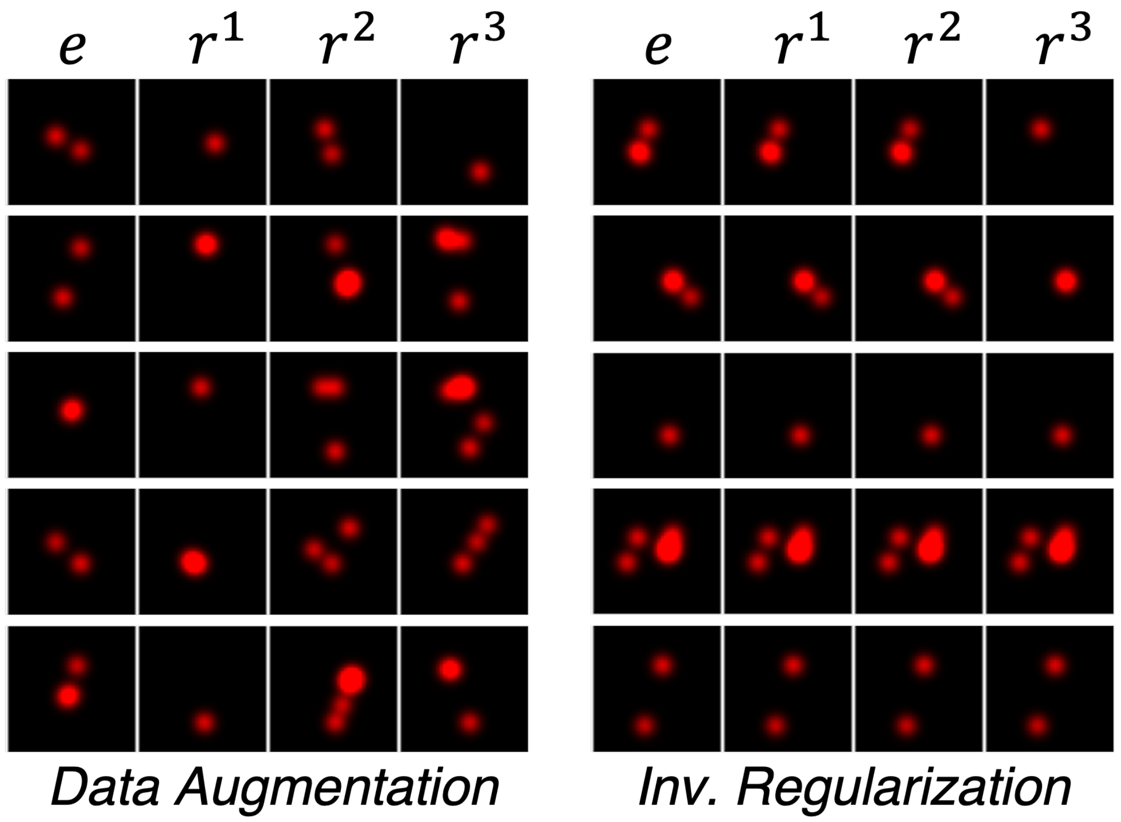


Figure 3: Saliency maps for the models with learned $\mathbb{R}_4^2$ invariance. Rows correspond to data samples and columns correspond to transformations from the group orbit applied to the sample. All saliency maps are realigned to a common orientation.

$\mathrm{Acc}(\mathcal{D})$ is the model's accuracy on the dataset $\mathcal{D}$. The accuracy ratio indicates how much of the original accuracy is preserved when a model is tested on the drifted dataset. The results are presented in Figure 4.

We observe that the invariance learned by data augmentation deteriorates rapidly, even under a slight degree of data drift. This turns into a major flaw if a user anticipates a certain level of invariance from the model, but then invariance instantly fails upon encountering unseen data. This also obscures the interpretability of predictions, thereby complicating the explainability of model decisions, even if accuracy is sustained. This yields invariance learned by data augmentation unreliable. *Conversely, models with weight-tying and invariance regularization maintain low invariance error even under substantial distribution drift.*

Also, we observe that networks with invariant weight-tying sustain higher classification accuracy under the distribution shifts. This also holds for the models trained with invariance regularization for $\mathbb{T}_3^2$, but interestingly, not for the $\mathbb{R}_4^2$ invariance. We hypothesize this can be attributed to the *accuracy-on-the-line* effect [121], where models with higher in-domain accuracy tend to also deliver higher accuracy on out-of-distribution data.

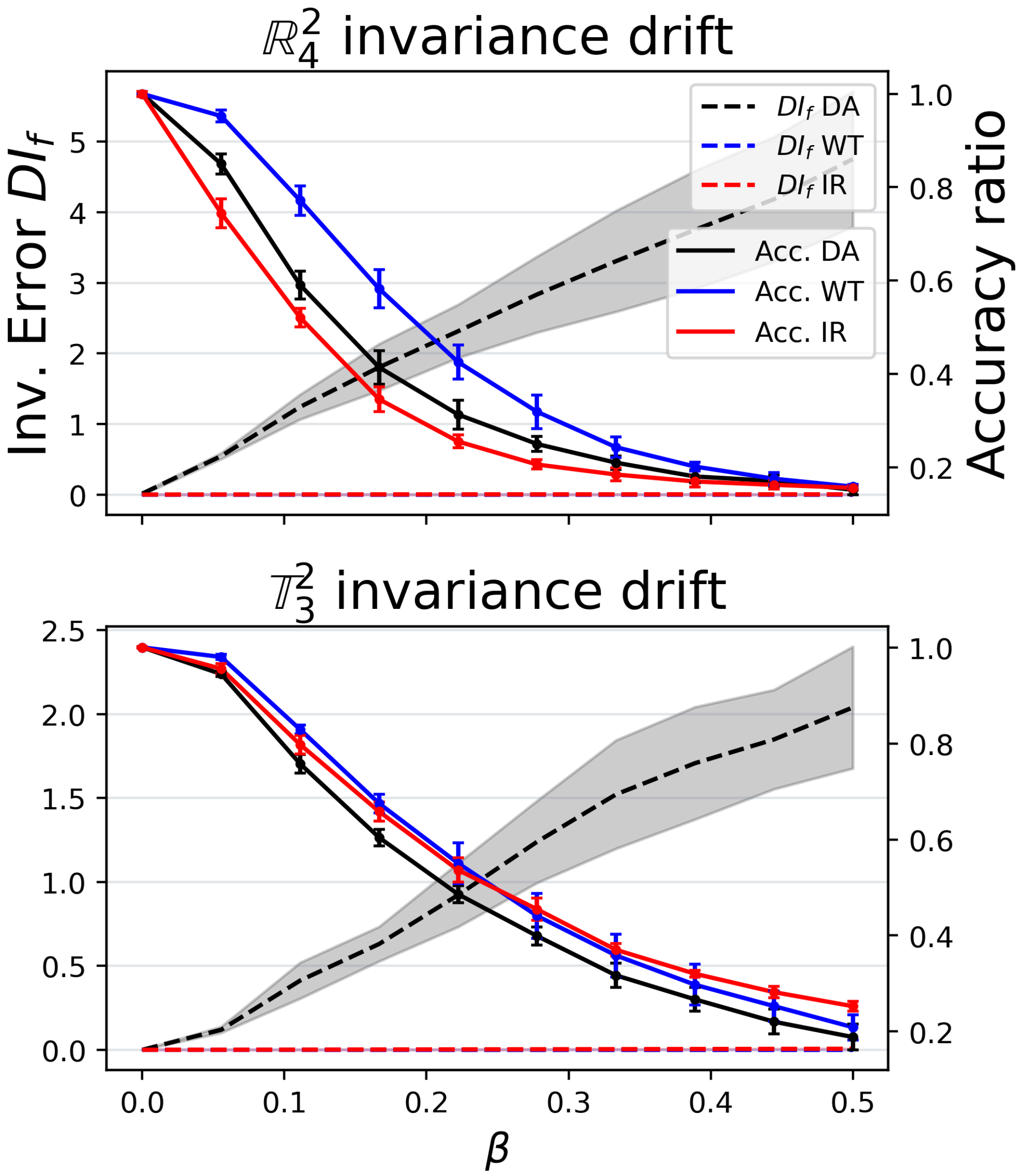


Figure 4: Predictive distribution invariance error DI_f (left y-axis) and classification accuracy drop ratio (right y-axis) for the data augmentation, weight-tying and invariance regularization models over increasing degree of the data distribution drift (x-axis).

2.4.3 *Invariance-induced spectral decay*

Lastly, we take a closer look at the invariance-induced spectral decay and we verify if it holds for different forms of invariance regularization. We investigate invariance regularization with *(i)* distribution invariance error with KL divergence, *(ii)* logit invariance error with squared L_2 distance, and *(iii)* logit invariance error with the squared L_2 distance normalized by the magnitude of the logits, i.e. $\|f(x) - f(gx)\|_2^2 / \|f(x)\|_2^2$. We tune the weighting of the regularizer for all of the models such that saliency invariance similarity $SI_f \geq 0.95$. We then evaluate the sensitivity of a network to input perturbations as a maximum singular value of its Jacobian $\sigma_{\max}(J)$; and we compare it to the sensitivity of

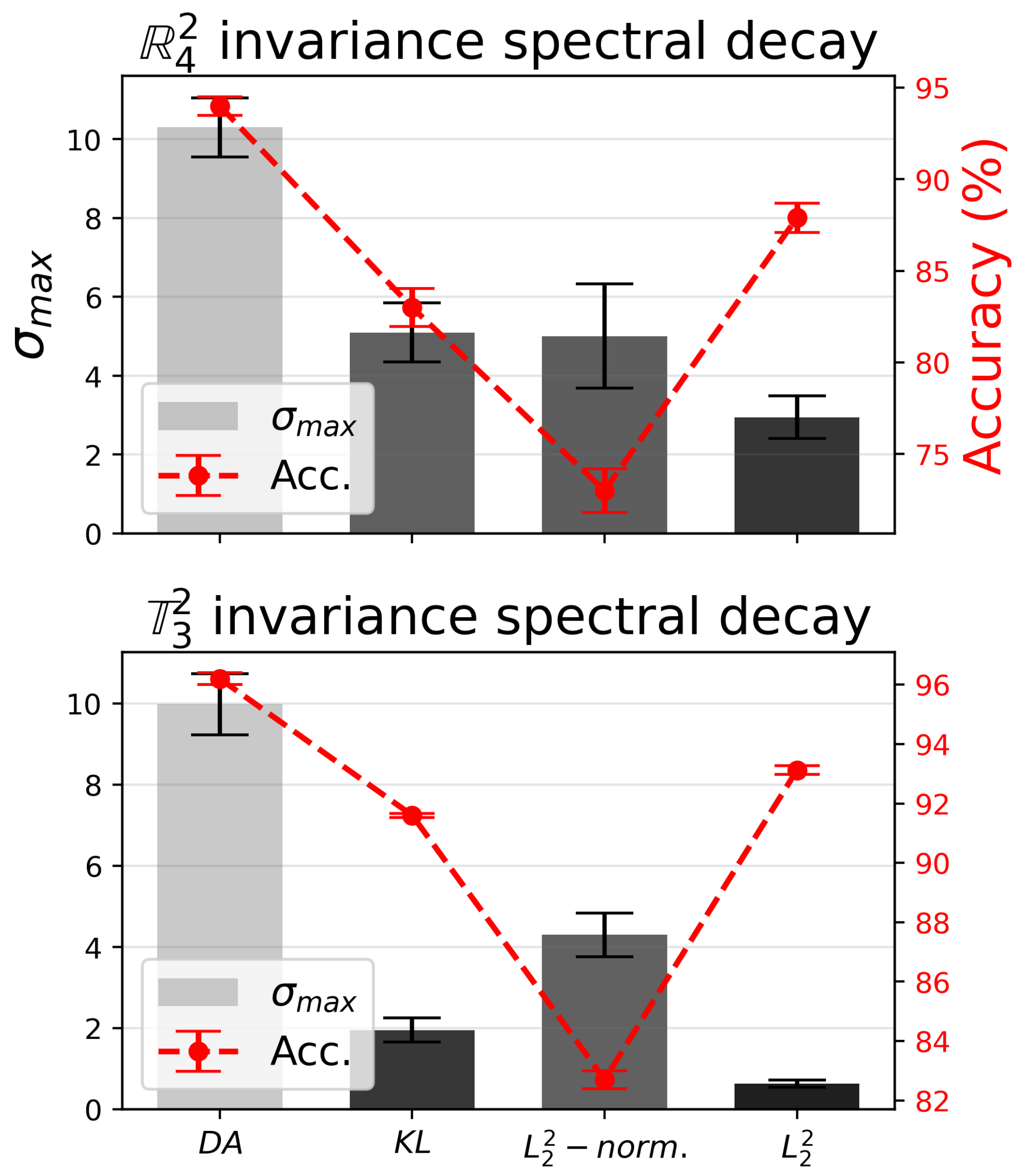


Figure 5: Sensitivity of a network to input perturbations measured by the maximum singular value of its Jacobian (left y-axis) for various forms of invariance regularization (x-axis). Models trained with invariance regularization come with overall reduced sensitivity to input perturbations.

the networks trained solely with data augmentation. The results are presented in Figure 5.

We observe that models trained with invariance regularization come with overall reduced sensitivity to input perturbations as indicated by considerably smaller $\sigma_{\max}(J)$. This phenomenon holds for both $\mathbb{R}^2_4$ and $\mathbb{T}^2_3$ groups and various forms of invariance regularization. Note that all forms of the invariance regularization we examine also induce an accuracy drop for the model.

2.5 DISCUSSION

SUMMARY Our study sheds light on the properties and limitations of data-driven invariance learning within neural networks. First, we proposed several measures to evaluate learned invariance: predictive distribution invariance and logit invariance errors, and saliency invariance similarity. With this, we study networks with learned group invariance and demonstrate that high performance and low invariance error do not guarantee a genuine invariant decision-making process. This leads to a notable risk, when learned invariance immediately fails beyond the training data distribution, making neural networks with learned invariance less reliable. Then, we showed that it is possible to promote genuine invariance learning by regularizing invariance during the training. Yet, such an approach leads to a spectral-decay phenomenon, when a network opts for reducing input sensitivity to all perturbations to achieve invariance to a specific group of transformations. These findings bring us a step closer to deciphering the intricate dynamics of learning inductive biases from the data.

BROADER IMPACT Our work, while primarily considering invariance to group symmetries, has potential implications for a much broader class of invariances. The increasing reliance on data-driven models, particularly in the era of large-scale machine learning, highlights the critical need to comprehend the properties of inductive biases that these models learn. Thus, understanding learned invariance, as one of the key inductive biases, becomes paramount for ensuring the fairness and interpretability of network's decisions.

LIMITATIONS AND FUTURE WORK While our study provides several key insights, some limitations remain. Firstly, the generalizability of our findings to other types of neural networks, other data modalities and other training regimes, e.g. self-supervised learning [28, 63, 127], remains an interesting future direction to explore. Secondly, the way we design invariance regularization assumes a known group of transformations, which may not always be accessible in practice. Future work could look into methods for learning genuine invariance to unknown transformations without architectural modification. Lastly, our results also highlight a trade-off between learning the genuine invariance and the downstream task performance, opening a direction for future research into strategies for mitigating this trade-off.

3

STUDYING LEARNED LIE GROUP GENERATORS

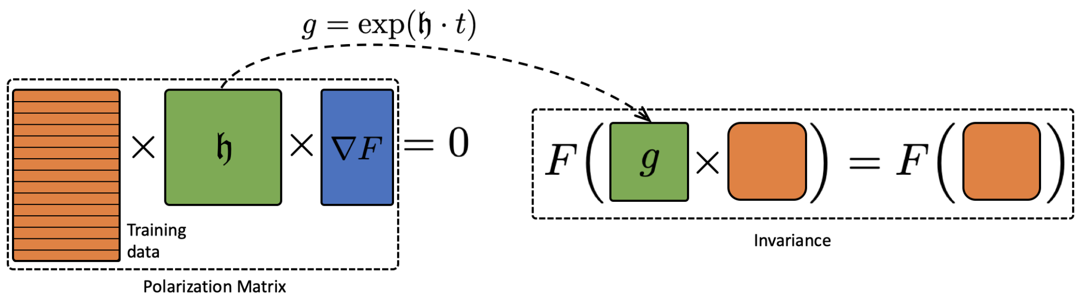
3.1 INTRODUCTION

Convolutional Neural Networks (CNNs) are efficient, for one because they can convert the translation symmetry in the data into a built-in translation-equivariance property of the network without exhausting the data to learn the equivariance. Group-equivariant networks generalize this property to rotation [34, 84, 176, 180], scale [10, 153, 156], and other symmetries defined by matrix groups [53]. Equipping a neural network with prior known symmetries has proved to be data-efficient.

Recent works [52, 90, 200] have demonstrated that hard coding the symmetry into a neural network does not always lead to a better generalization. Often, soft or learned equivariance is more advantageous mostly in terms of data efficiency and accuracy [171]. Moreover, architectures with no equivariance built-in, such as transformers [47] and multi-layer perceptron mixers [162], achieve a remarkable performance on a wide range of problems. Effectively, they learn their own geometrical priors without explicit symmetry constraints [37]. This raises the following questions to study in this paper. To what degree do neural networks learn symmetries? How accurately do learned symmetries reflect the true symmetries in the data? Can we support the capability of neural networks for learning symmetries? We present a method to study symmetries learned by neural networks to address the questions.

In the paper, we depart from Lie group theory and develop a method that can retrieve symmetries learned from the data by any neural network (Figure 6). Our method only makes the assumption that a model is differentiable, commonly met in neural networks. While previous works on analyzing symmetries in neural networks rely on empirical analysis of network representations [132] or on examination of a given set of transformations [57, 104, 153], our method outputs a generator of the corresponding Lie-group and allows to quantitatively evaluate how sensitive the network is in the direction of the learned symmetry We make the following contributions:

- From the perspective of Lie-groups, we propose the theory to study symmetrical properties of neural networks.
- We derive an efficient implementation of the method that allows improving the interpretability of the model by revealing symmetries it learned.
- With our method we conclude that models with more parameters and gradual fine-tuning learn more precise symmetry groups with a higher degree of invariance to them.



$g = \exp(\mathfrak{h} \cdot t)$
$\times$ $\mathfrak{h}$ $\times$ ∇F $= 0$
Training data
Polarization Matrix
$F(g \times \square) = F(\square)$
Invariance

Figure 6: We solve the matrix nullspace equation to derive an infinitesimal generator from a neural network and the training data. Due to Lie algebra - Lie group correspondence, we can calculate the class of transformations the model is invariant to by exponentiating the generator.

3.2 RELATED WORK

EQUIVARIANT AND INVARIANT NETWORKS The goal of equivariant and invariant networks is to build the symmetries into a neural network architecture as an inductive bias. Starting from the convolutional networks [102] that contain a translation symmetry, the concept of the equivariance was generalized to rotations [34, 84, 176, 180], permutations [192], scaling [10, 153–156, 179] and arbitrary matrix groups in [53]. Various methods have been proposed for learning symmetries directly from the training data [4, 200] among which a popular approach is to learn transformations by estimating the infinitesimal generators of symmetry groups [36,39,43,140,151]. These papers focus on building the symmetries into the models by modifying the architecture. In our work, we focus on the reverse question, i.e. given a network with a fixed architecture, what symmetries does it learn from the data?

SYMMETRIES IN NEURAL NETWORKS Another line of work is focused on interpreting symmetries in neural networks. In [132] Olah *et al.* take inspiration from biological circuit motifs [3] and study equivariant patterns learned by an unconstrained neural network on the image classification task. The authors demonstrate that the network learns rotation, hue, and scale symmetries when trained on ImageNet. In [57] Goodfellow *et al.* propose a number of empirical tests to study invariances to known transformations in a network, and demonstrate that auto-encoding architectures learn increasingly invariant features in the deeper layers. In [104] Lenc & Vedaldi quantify invariances and equivariance in the layers of convolutional networks to a pre-defined set of transformations. In the concurrent work of Gruver *et al.* [59], the authors propose using the Lie derivative to measure the local equivariance error to known symmetry groups. In contrast to these works, our method does not require knowing the set of transformations beforehand; and it also provides theoretical guarantees of the invariance from the perspective of the Lie groups theory.

NETWORK CONFIGURATION: WIDTH, DEPTH, NUMBER OF PARAMETERS Neural networks of various widths and depths have been studied through the lens of universal approximation theorem [92, 112], functional expressiveness [138] and by empirical analysis of learned representations [130]. These works focus on either characterizing learning capabilities of neural networks or on interpreting the differences between representations that models with various architectures learn. They, however, do not analyze how the symmetrical properties of a model depend on its width and depth.

Other works focus on analyzing a connection between a number of parameters and the generalization capability of networks [2, 166, 189, 195]. It is commonly argued that overparametrized architectures achieve better generalization bounds, i.e. provide smaller discrepancy between train and test performance and are thus preferred in many applications [195]. However, it is unclear if the models with more parameters and hence with a finer generalization capability also learn symmetries better. In our paper, we investigate this question for a family of feed-forward networks.

ROBUST LEARNING Various methods have been proposed to train more robust neural networks. While a naive training may lead to overfitting on the training subset, a well-organized pre-training helps to mitigate this issue [48]. Features extracted by a model pretrained on a bigger dataset often demonstrate more robust results [70]. While there are many effective techniques for training more robust feature extractors [12, 65, 204], the analysis of such methods from the invariance point of view has not been done before. We demonstrate that our method allows for better understanding and explaining what training regimes lead to more invariant, and thus more robust representations.

3.3 BACKGROUND

The focus of this paper is a *symmetry group*. A symmetry is a transformation of an object that leaves the object unchanged. The set of all such transformations of an object with composition as a binary operation forms a group. In this paper, the object of interest is a dataset and its symmetries. We study what kind of transformations map one data sample to another and how neural networks learn information about the symmetries.

The theory of Lie groups is a sweet spot of mathematics that helps to formalize symmetries and provides practical methods for studying them. Formally, a Lie group is a group that has the structure of a differential manifold. The tangent space in the identity element of a Lie group forms a vector space called Lie algebra. A Lie algebra determines a Lie group up to an isomorphism for simply connected Lie groups and, being a vector space, is a more convenient object to study than a Lie group. Thus, in order to recover a simply connected symmetry group, it is sufficient to understand its Lie algebra.

In this paper, we assume that the Lie group G is a matrix Lie group, i.e. is a closed subgroup of the general linear group of the degree n, denoted as $GL(n)$. It is defined as the set of $n \times n$ invertible matrices. The correspondence between the Lie group G and its Lie algebra $\mathfrak{g}$ in this case is given by the exponential map:

$$\mathfrak{g} = \{\mathfrak{h} \in M(n) : \; e^{t \cdot \mathfrak{h}} \in G \; \forall t \in \mathbb{R}\}, \tag{3.1}$$

where $M(n)$ is the set of all $n \times n$ matrices. Each such element $\mathfrak{h}$ from equation 3.1 is called an *infinitesimal generator* of the Lie group G.

3.4 METHOD

3.4.1 *Lie algebra invariance*

We introduce **LieGG**, a method to compute **Lie G**roup **G**enerators. LieGG extracts the infinitesimal generators of the Lie algebra for symmetries that a neural network learned from the data. To find the Lie algebra basis, we use the *discriminator* of the dataset, i.e. a function $F : \mathbb{R}^n \to \mathbb{R}$ such that $F(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is an element of the dataset $\mathcal{D}$. The discriminator function is naturally modeled by a neural network fitting the data subject to a downstream task. With this, we use the following criterion for a symmetry group:

Theorem 2 (*Lie algebra invariance*)**.** *Let G be a connected Lie group of linear transformations acting on an n-th dimensional manifold $\mathcal{X}$. Let $F : \mathcal{X} \to \mathbb{R}^l$, $l \leq n$, define a system of algebraic equations:*

$$F_\nu(\mathbf{x}) = 0, \ \nu = 1, \cdots, l$$

and assume that the system is of maximal rank, meaning that the Jacobian matrix $\left(\frac{\partial F_\nu}{\partial x_k}\right)$ is of rank l for every solution $\mathbf{x}$ of the system. Then G is a symmetry group of the system if and only if

$$\sum_{i=1,j=1}^{i=n,j=n} \frac{\partial F_\nu}{\partial x_i} \cdot \mathfrak{h}_{ij} \cdot x_j = 0, \ \textit{whenever } F_\nu(\mathbf{x}) = 0,$$

for $\nu = 1, \cdots, l$ and every infinitesimal generator $\mathfrak{h}$ of G, where $\mathfrak{h}_{ij}$ is an element of the matrix $\mathfrak{h}$ in the i-th row and j-th column.

Proof. Let G be a connected Lie group of transformations acting on the n-th dimensional manifold $\mathcal{X}$. Let $F : \mathcal{X} \to \mathbb{R}^l$, $l \leq n$. With this, we define a system of algebraic equations:

$$F_\nu(\mathbf{x}) = 0, \ \nu = 1, \cdots, l,$$

and assume that the system is of maximal rank, meaning that the Jacobian matrix $\left(\frac{\partial F_\nu}{\partial x_k}\right)$ is of rank l for every solution $\mathbf{x}$ of the system. Then G is a symmetry group of the system if and only if:

$$\mathfrak{h}[F_\nu(\mathbf{x})] = 0, \ \text{whenever } F_\nu(\mathbf{x}) = 0,$$

for every infinitesimal generator $\mathfrak{h}$ of G.

In our special case, the infinitesimal generator $\mathfrak{h}$ is a matrix, which generates the one parameter group of transformations equal to $\exp(\mathfrak{h} \cdot t)$. That means that in the theorem, $\mathfrak{h}$ is an element of the symmetry group if and only if $\frac{dF_\nu(\exp(\mathfrak{h} \cdot t) \cdot \mathbf{x})}{dt}|_{t=0} = 0$, whenever $F_\nu(\mathbf{x}) = 0$. But

$$\frac{dF_v(\exp(\mathfrak{h}\cdot t)\cdot \mathbf{x})}{dt}|_{t=0} = \sum_{i=1}^{i=n} \frac{\partial F_v(\mathbf{x})}{\partial x_i} \cdot \frac{(\exp(\mathfrak{h}\cdot t)\cdot \mathbf{x})_i}{dt}|_{t=0} =$$

$$\sum_{i=1}^{n} \frac{\partial F_v(\mathbf{x})}{\partial x_i} \cdot \sum_{j=1}^{n} \mathfrak{h}_{ij}\cdot x_j = \sum_{i=1,j=1}^{i=n,j=n} \frac{\partial F_v}{\partial x_i} \cdot \mathfrak{h}_{ij}\cdot x_j$$

□

EXAMPLE We illustrate the practical significance of the theorem with an example. Suppose the dataset D lies on a sphere in $\mathbb{R}^n$:

$$D = \{\mathbf{x} \in \mathbb{R}^n : x_1^2 + \cdots + x_n^2 = 1\}.$$

The discriminator for this example is the function $F(\mathbf{x}) = x_1^2 + \cdots + x_n^2 - 1$. According to the theorem, to find the Lie algebra of the symmetry group, we can find a basis of the solution of the following linear equations, where a_{ij} are the variables of interest:

$$\sum_{i,j} x_i \cdot a_{ij} \cdot x_j = 0,$$

when $x_1^2 + \cdots + x_n^2 = 1$.

We assert that the solutions are the family of matrices A such that $A + A^T = 0$. Indeed, for $\mathbf{x} : x_i = 1,\ x_j = 0,\ j \neq i$, the condition means that $a_{ii} = 0$. For $\mathbf{x} : x_i = \frac{1}{\sqrt{2}},\ x_j = \frac{1}{\sqrt{2}},\ x_k = 0,\ k \neq i, j$, it follows that $a_{ij} + a_{ji} = 0$. On the other hand, $A = -A^T$ implies $\sum_{i \neq j} (x_i \cdot a_{ij} \cdot x_j + x_j \cdot a_{ji} \cdot x_i) = \sum_{i \neq j} (x_i \cdot a_{ij} \cdot x_j - x_i \cdot a_{ij} \cdot x_j) = 0$. Note that the Lie algebra of the matrices $\{A : A + A^T = 0\}$ corresponds to the Lie group of matrices $\{B : B \cdot B^T = E\}$ which is the rotation group. Thus, the symmetry of a sphere is the rotation group.

3.4.2 *Nullspace invariance*

To compute a Lie algebra given a discriminator function $F : \mathbb{R}^n \to \mathbb{R}$ we use Theorem 2 and solve the system of linear equations, where each equation corresponds to one element in the dataset:

$$\sum_{i=1,j=1}^{i=n,j=n} \frac{\partial F}{\partial x_i} \cdot \mathfrak{h}_{ij} \cdot x_j = 0, \text{ for each point in the dataset.} \tag{3.2}$$

This is a system of linear equations with n^2 variables $\mathfrak{h}_{ij}$ and the number of equations equal to the number of points in the dataset. We can cast solving such system with respect to $\mathfrak{h}$ as a problem of finding a nullspace of the matrix $\mathcal{E}$. The matrix $\mathcal{E}$ has the number of rows equal to the number of points in the dataset and n^2 columns. Multiplying $\mathcal{E}$ with the vectorized representation of $\mathfrak{h}$ yields the system of equations in 3.2. We call the matrix $\mathcal{E}$ the *network polarization matrix*.

The problem of finding the nullspace basis of a matrix is a standard problem in the numerical analysis. It can be efficiently solved using the Singular Value Decomposition (SVD). Recall that we can write matrix $\mathcal{E}$ using the SVD in the following way $\mathcal{E} = U\Sigma V^T$, where U, V - are orthogonal matrices and Σ is a diagonal matrix with decreasing singular values. Thus, the columns in V corresponding to nearly-zero singular values encode the nullspace of the system of equations in 3.2 and hence form a Lie algebra basis.

Thereby, in practice, the calculation of LieGG consists of 3 steps: (i) training a neural network, (ii) calculation of the polarization matrix $\mathcal{E}$, (iii) computing singular vectors corresponding to almost zero singular values of the polarization matrix $\mathcal{E}$.

3.4.3 *Computing a Lie algebra of a group acting on* $\mathbb{R}^2$

Symmetries play an important role in computer vision problems, and we will describe LieGG for this case in more details. In various computer vision problems, it is assumed that the symmetry group G changes images from the dataset by acting on $\mathbb{R}^2$. For example, convolutional networks [102] and recently proposed equivariant networks for rotations [176, 180] assume that the group acts as a translation or rotation of an image. These neural networks perceive each image as a function f with non-zero values on some compact subset $\Omega \subset \mathbb{R}^2$, where the elements of the subset correspond to spatial coordinates. Thus, the space of images $\mathcal{I}$ is the space of all functions that map spatial coordinated to pixel values. In other words: $\mathcal{I} := \{f,\ f : \Omega \to \mathbb{R}\}$. A vision model $F : \mathcal{I} \to \mathbb{R}$ is a function on such a space of images.

We view an image is a function on a pixel grid $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ of the domain Ω, i.e. the image can be identified with the point in $\mathbb{R}^n$, which encodes the pixel values. In this case the model takes n real values as an input: $F(f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n))$.

It is important to note that Theorem 2 assumes that the group G acts continuously on the data manifold. However, this assumption is violated for the image data as an image is a non-continuous function of coordinates due to the discrete pixel grid. We overcome the discontinuity issue by preprocessing images with Gaussian smoothing.

To use LieGG for the image data, we explicitly write the condition that the matrix $\mathfrak{h}$ is an infinitesimal generator for the image data $\mathcal{I}$ on the grid $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. The condition reads as follows:

$$F(f(e^{\mathfrak{h}\cdot t} \circ \mathbf{x}_1), \cdots, f(e^{\mathfrak{h}\cdot t} \circ \mathbf{x}_n)) = 0 \leftrightarrow \sum_{p=1,k=1,j=1}^{p=n,k=2,j=2} \frac{\partial F}{\partial f(\mathbf{x}_p)} \cdot \frac{\partial f(\mathbf{x}_p)}{\partial x_p^k} \cdot \mathfrak{h}_{kj} \cdot x_p^j = 0 \tag{3.3}$$

where $t \in \mathbb{R}$ and x_p^j corresponds to the projection of $\mathbf{x}_p$ onto the j-th coordinate axis.

The resulting condition is the equation in 4 variables $\mathfrak{h}_{kj},\ k = 1, 2;\ j = 1, 2$. Hence, the polarization matrix has 4 columns. Thus, in equation 3.3 a space of potential image symmetries is reduced to a 4-dimensional spatial subgroup. This is to provide a prior for LieGG to retrieve meaningful symmetries from for images as the full-scale space of potential symmetries is overwhelming for the image data. This is a strong yet proven to useful prior for vision models [34, 80, 84, 176, 180].

In practice, the calculation of LieGG for vision models consists of the following steps: (i) preprocessing images by doing Gaussian smoothing, (ii) training a neural network, (iii) computing the polarization matrix using equation 3.3, (iv) computing singular vectors corresponding to almost zero singular values of the polarization matrix.

3.4.4 *Symmetry bias and symmetry variance*

The scope of LieGG is not limited to retrieving the generator of a learned symmetry group. With the polarization matrix $\mathcal{E}$, we can also evaluate the degree of the neural network's invariance to a learned symmetry. Furthermore, if a true symmetry in the data is known, we can apply our method to analyze how close is a learned symmetry to the true one. The former criterion we alias *symmetry variance*, and the later - *symmetry bias*.

Firstly, we show how to estimate the degree of the network invariance to the element g of a Lie group:

$$R(g) := \mathbb{E}(F(g \cdot \mathbf{x}) - F(\mathbf{x}))^2 \sim \frac{1}{|D|} \sum_{\mathbf{x}_i \in D} (F(g \cdot \mathbf{x}_i) - F(\mathbf{x}_i))^2 := \tilde{R}(g).$$

Let $\mathfrak{h}$ be an element from Lie algebra that corresponds to g, i.e.

$$\tilde{R}(g) = \frac{1}{|D|} \sum_{\mathbf{x}_i \in D} (F(\exp(\mathfrak{h}t) \cdot \mathbf{x}_i) - F(\mathbf{x}_i))^2 =$$
$$\frac{1}{|D|} \sum_{\mathbf{x}_i \in D} (t \cdot \sum_{k,j} \frac{\partial F(\mathbf{x}_i)}{\partial x_i^k} \cdot \mathfrak{h}_{kj} \cdot x_i^j + O(t^2))^2 =$$
$$\frac{1}{|D|} \sum_{\mathbf{x}_i \in D} t^2 \cdot (\sum_{k,j} \frac{\partial F(\mathbf{x}_i)}{\partial x_i^k} \cdot \mathfrak{h}_{kj} \cdot x_i^j)^2 + O(t^3)$$

The matrix with the i-th row equal to $\frac{\partial F(\mathbf{x}_i)}{\partial x_i^k} \cdot x_i^j$ equates to the polarization matrix $\mathcal{E}$. Then

$$\tilde{R}(g) = \frac{1}{|D|}(t^2 \cdot \|\mathcal{E}\bar{\mathfrak{h}}\|_2^2 + O(t^3)),$$

where $\bar{\mathfrak{h}}$ is the vectorized representation of the element of the Lie algebra.

To calculate the value of $\mathcal{E}\bar{\mathfrak{h}}$ we use the SVD of the matrix $\mathcal{E}$. Let $\mathcal{E} = U\Sigma V^T$, where U, V - are semi-unitary matrices and Σ is a diagonal matrix. With this, we can estimate $\tilde{R}(g)$ in the following way:

$$\tilde{R}(g) \sim \frac{1}{|D|} \|\mathcal{E}\bar{\mathfrak{h}}\|_2^2 = \frac{1}{|D|} \sum_l \sigma_l^2 (V^T \bar{\mathfrak{h}})_l^2 \tag{3.4}$$

where σ_l is the l-th singular value of the network polarization matrix $\mathcal{E}$.

In particular, the equation 3.4 reveals that singular values of the network polarization matrix reflect the degree of invariance to transformations encoded by the corresponding singular vectors. With this, we use the smallest singular value of the network polarization matrix to define the *symmetry variance* as $\mathcal{V}(\mathcal{E}) = \sigma_{min}(\mathcal{E})^2/|D|$. Symmetry variance estimates the degree of invariance in the direction of the symmetry, a network learned to be most invariant to. Moreover, when the true symmetries are known, the difference between the singular vectors and the generator of the true symmetry group equates to the *symmetry bias*, i.e. $\mathcal{B}_i(\mathcal{E}, \mathcal{G}) = \|\bar{v}_i(\mathcal{E}) - \mathcal{G}\|_F$, where $\bar{v}_i(\mathcal{E})$ returns the i-th singular vector reshaped to a matrix, and $\mathcal{G}$ is a ground truth symmetry group generator closest to the $\bar{v}_i(\mathcal{E})$, e.g. the closest skew-symmetric matrix for the rotation group.

Note that the proposed symmetry variance allows measuring learned invariance without the knowledge of a symmetry generator beforehand. This is in contrast to Gruver *et al.* [59], where the knowledge of a symmetry generator is required to measure learned equivariance.

3.5 EXPERIMENTS

3.5.1 *Retrieving learned symmetries*

SYNTHETIC REGRESSION Firstly, we conduct the experiment on the synthetic problem to test the ability of our method to accurately retrieve a symmetry learned by a network. To do so, we adopt the invariant regression task from [53] with clean $\mathrm{O}(5)$ symmetry built-in. The regression function $f : \mathbb{R}^{2\times 5} \to \mathbb{R}$ is given by:

$$f(\mathbf{x}_1, \mathbf{x}_2) = \sin(\|\mathbf{x}_1\|_2) - 0.5 \cdot \|\mathbf{x}_2\|_2^3 + \frac{\mathbf{x}_1^T \mathbf{x}_2}{\|\mathbf{x}_1\|_2 \, \|\mathbf{x}_2\|_2} \tag{3.5}$$

With this, we construct the training dataset to fit the regression function with the multi-layer perceptron. Here the MLP is the unconstrained feed-forward neural network with no prior knowledge of symmetries in the data. Once the model is converged, we apply our method to analyze how close are retrieved symmetries to clean $\mathrm{O}(5)$ group (symmetry bias), and how invariant is the network to the learned symmetries (symmetry variance). We also study how the ability of the network to learn symmetries depends on the number of available training samples. We report the results in Figure 7.

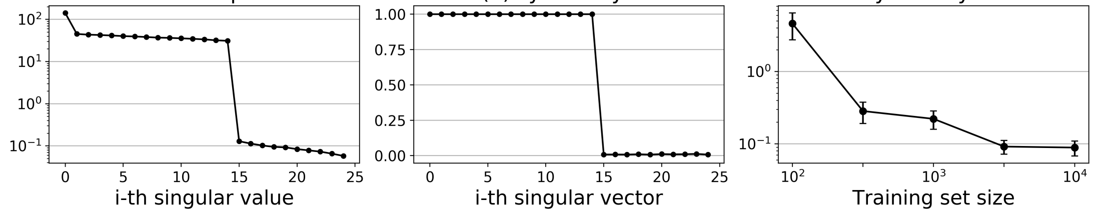
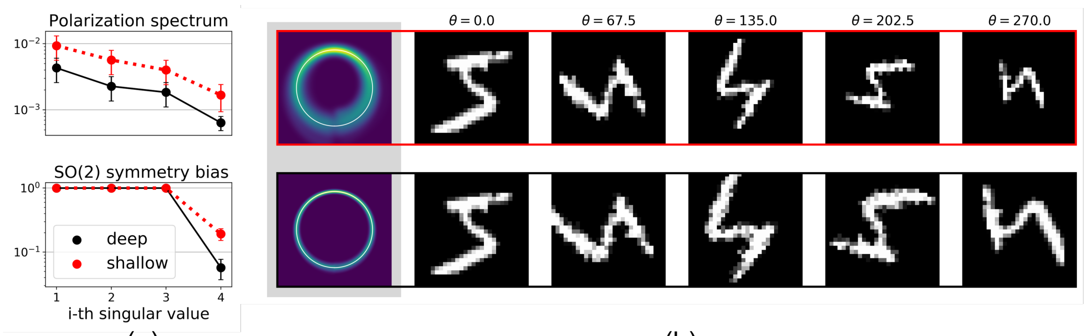


Figure 7: Singular values of the network polarization matrix and the symmetry biases per singular vector of the multi-layer perceptron fitted on $O(5)$ invariant regression task. ***(right)*** *Comparing the mean symmetry variance for various training set sizes.*



(a) (b)

*Figure 8: **(a)** Symmetrical properties of the shallow and deep networks: singular values of the network polarization matrix and $SO(2)$ symmetry bias. **(b)** Shaded: the kernel density of learned 2D rotations on a perfect circle. Non shaded: visualizing learned Lie algebras for various rotation angles (in columns). Top row: the shallow network. Bottom row: the deeper network.*

As can be seen from Figure 7a, the spectrum of the network polarization matrix is zeroing, indicating that the model becomes invariant to the transformations encoded by the corresponding singular vectors. Also, these singular vectors are the generators of the rotation group as indicated by zero symmetry bias with respect to $\mathrm{O}(5)$ group (Figure 7b). We also observed that the network accurately learns the symmetry group generators even with a small training set size. However, a larger training set size leads to a smaller symmetry variance, i.e. a higher degree of invariance. For an additional validation, we used LieGG to extract symmetries from the randomly initialized $\mathrm{O}(5)$ invariant EMLP [53] model. For this case, LieGG recorded zero symmetry variance and zero symmetry bias with respect to $\mathrm{O}(5)$ group without any model pre-training.

The experiment demonstrates that, on the synthetic regression task, our method can accurately retrieve symmetries learned or built-in. When symmetries are learned, the larger training dataset facilitates learning invariances for the model.

ROTATION MNIST In this experiment, we test the ability of our method to retrieve learned symmetries on the rotation MNIST dataset [101]. We train the feed-forward network to output the class of a digit invariant to 2D in-plane rotations. The symmetry group that the model should learn is thus $\mathrm{SO}(2)$. We note that different from the synthetic regression task, the symmetries in the rotation MNIST are much noisier since the rotations are coarsened by the projection onto a pixel grid. In addition, the network has a classification error, hence it serves as only an approximate data discriminator function. In this scenario, we test how well our method can retrieve noisy $\mathrm{SO}(2)$ symmetry.

We experiment with two architectures: the shallow 2-layer perceptron and the deeper 6-layer network with 4 times number of parameters. Once the network's performance plateaus on the validation split, we terminate the training, and apply our method to retrieve learned symmetries. We plot the spectrum of the network polarization matrix, $\mathrm{SO}(2)$ symmetry bias and visualize learned symmetries for the shallow and deep models in Figure 8.

From Figure 8a we observe that for both shallow and deep models, our method retrieves the rotation symmetry as indicated by the zeroing symmetry bias with respect to $\mathrm{SO}(2)$ group. Also, we noticed that the symmetry learned by the shallow model is noisier than the symmetry learned by the deeper network as indicated by the higher symmetry bias. We visualize this in Figure 8b by sampling learned Lie algebras and applying them to transform an input image. We see that the symmetry learned by the shallow model diverges from a clean rotation and also contains scaling and shearing. We also see, that on the rotation-MNIST, the spectrum of the network polarization matrix is zeroing not as fast as in the synthetic case. It indicates that the models, although do learn the rotation symmetry, do not become fully invariant to it.

This experiment demonstrates that our method can retrieve symmetries from the data even when an imprecise discriminator function is used.

3.5.2 *Symmetries in networks with different configurations*

Observing the differences in symmetrical properties for different network configurations for the rotation-MNIST, we further conduct the study on how the symmetry variance and the symmetry bias depend on width, depth, and a number of parameters in a neural network. In this experiment, we are interested to know if for some network configurations it is easier to learn symmetries from the data.

To obtain different configurations, we sample a number of parameters from the range of $[40000, 200000]$ with the step size of 20000, and vary the depth of a network from 1 to 5 hidden layers. We train all the networks until convergence on the validation split and apply LieGG to retrieve the symmetries and to record the symmetry variance and the symmetry bias. We also track the accuracy of the models to correlate it with the symmetry variance and the symmetry bias.

The results are reported in Figure 9. We summarize the analysis of how the configurations of the network influence its symmetrical properties:

NUMBER OF PARAMETERS: we observe that the networks with more parameters tend to learn to the higher degree of invariance than their shallow counterparts as indicated by consistently smaller symmetry variance. Also, we see that the networks with fewer parameters learn less precise symmetries as indicated by the higher symmetry bias.

DEPTH / WIDTH: our results indicate that sufficiently parameterized deep networks are more capable to learn invariances than the wide ones with the same number of parameters. In other words, deeper networks become more invariant in the direction of learned symmetries. At the same time, the same only partially holds for the quality of a learned symmetry group as indicated by the fluctuating symmetry bias, when the depth > 3. Overall, the network with a balanced width and depth achieves the smallest symmetry bias.

SYMMETRY VS. ACCURACY: we also test how the symmetry variance and the symmetry bias correlate with the performance of the models. As can be seen from Figure 9b, the symmetry variance and bias have moderate negative correlation coefficients $\rho_{\text{var}} = -0.49$ $(p\text{-}value < 1e^{-4})$ and $\rho_{\text{bias}} = -0.31$ $(p\text{-}value < 1e^{-4})$ with the classification accuracy respectively. The trend indicates that more accurate models tend to have lower

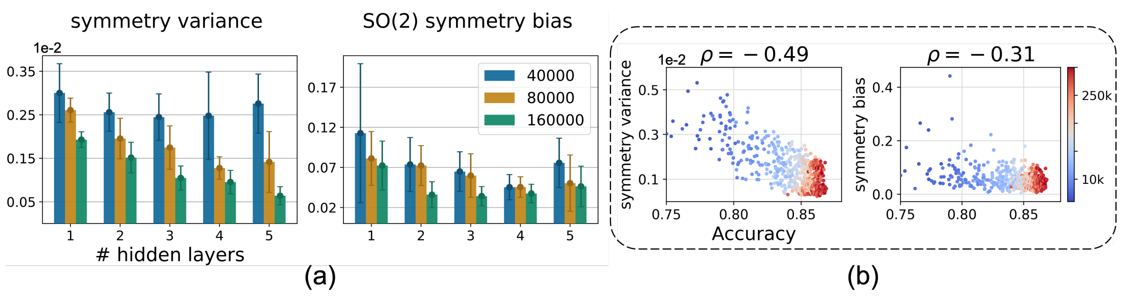


*Figure 9: **(a)** Symmetry variance and $SO(2)$ symmetry bias for the networks with different width, depth and number of parameters. **(b)** The correlation plot of the symmetry variance and the symmetry bias versus the test accuracy of the models. Each point corresponds to one network. The color indicates the number of parameters.*

symmetry variance and lower symmetry bias. However, the correlation is not strict. That is, it is possible to have neural networks, which learn symmetries differently, but perform equally well during the test.

3.5.3 *Tuning layer-wise symmetries in deep networks*

In the previous experiment, we analyzed how the symmetrical properties of the networks depend on the configuration of the model. Now, we ask if by modifying a training regime, we can facilitate learning symmetries for the network. In this experiment, we evaluate two known approaches to robust training: (i) pre-training with fine-tuning, and (ii) layer-wise training [12]. We also evaluate plain end-to-end training as a baseline.

To simulate the pre-training of a part of the network weights, we take randomly initialized 7 layers perceptron (mother network) and extract the sub-network that consists of the first 3 layers. We then attach the classification head to the sub-network and train it independently from the mother network on the rotation-MNIST dataset. After that, we re-inject the learned weights into the mother network and retrain the model. For the fine-tuning, we train all of the weights jointly. For the layer-wise training, we freeze the pre-trained part and only train the remaining layers. We then apply our method to study the symmetrical properties of the models trained with each of the regimes. We plot the symmetry variance and the symmetry bias after each layer of the network in Figure 10.

FINAL AND PEN-ULTIMATE LAYER SYMMETRIES: from Figure 10 we observe that for the different training regimes, the network learns approximately the same symmetries at the same degree of invariance in the final layer. Surprisingly, we also noticed that in the pen-ultimate layer, the symmetries for all of the training regimes come very close as measured by the symmetry variance and bias. Since the last layers are directly responsible for providing outputs for the downstream task, we hypothesize that *there exist a preferred optimal symmetry level for a neural network that depends on an architecture and the data, but not from a training regime of a model.*

INTERMEDIATE SYMMETRIES: on the other hand, we observed that *symmetries in the intermediate layers of the network do depend on the training regime*. Firstly,

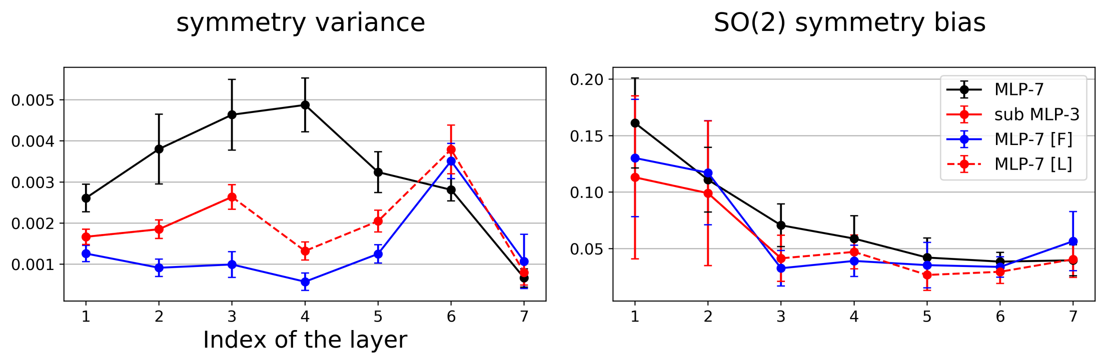


Figure 10: Symmetry variance and $SO(2)$ symmetry bias measured layer-wise for the network trained on the rotation-MNIST dataset. Different colors correspond to different training regimes of the model. Training scenarios are: ***MLP-7*** *for the 7-layer perceptron,* ***sub MLP-3*** *for the 3-layer sub-network,* ***MLP-7[F]*** *for the joint fine-tuning,* ***MLP-7[L]*** *for the layer-wise training.*

the pre-trained sub-network learns more accurate symmetry with the higher level of invariance compared to when it is trained as a part of the deeper model. We attribute it to the fact that symmetries are defined on a level of training labels, hence invariances learned by the model in the final layer should be kept at a minimum [178]. Secondly, after the fine-tuning or the layer-wise training, symmetries in the intermediate layers of the network are both more precise and utilized at the higher degree of invariance as indicated by the smaller symmetry bias and variance respectively. It indicates that *we can facilitate learning intermediate invariances in a neural network* by using a pre-training followed by the fine-tuning or the layer-wise learning approaches. This can be useful for a transfer learning when it is important to transfer as much inductive bias as possible into other models or to other tasks.

3.5.4 *Sample complexity*

LieGG computation requires finding a nullspace of the network polarization matrix. The dimensionality of the network polarization matrix depends on the number of samples in a dataset. Since the dimensionality of a full-scale dataset may be prohibitively large, we conduct the sample complexity study to investigate if the usage of all samples in a dataset is necessary to effectively retrieve the symmetries learned by a neural network.

We conduct the empirical sample complexity study for $O(5)$ synthetic regression and rotation-MNIST classification tasks. For the synthetic regression, we take the trained model from 3.5.1 and compute the symmetry variance and bias using only a fraction of the training samples. We then compare the results with the symmetry variance and bias computed from all available training data. The resulting empirical sample complexity is presented in Figure 11. For the rotation-MNIST, we follow the same procedure and study the sample complexity of LieGG applied to retrieve symmetries from the trained 6-layer perceptron from 3.5.1. The resulting empirical sample complexity for the rotation-MNIST is presented in Figure 12. Note that in both cases, the model is trained using all available training data. The sample complexity study is only to investigate how many samples are needed to effectively extract the symmetries from the trained model.

For the synthetic regression task, it suffices to use only 0.25% of the training samples to extract the symmetry as precisely as when utilizing all available training data. For the more challenging rotation-MNIST, we have to use at least 5% of the training samples to obtain a precise estimation of the symmetry bias. The results suggest that we can effectively extract the learned symmetry using only a fraction of the data to create the network polarization matrix.

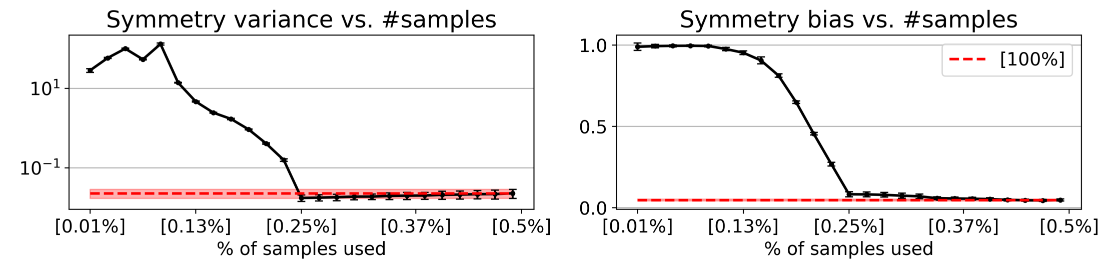


Figure 11: The sample complexity of LieGG for the $O(5)$ synthetic regression task. The red line indicates the symmetry variance and the symmetry bias computed using 100% of available samples.

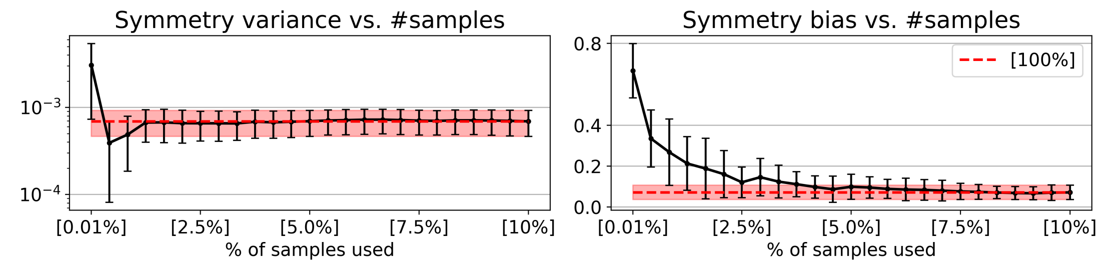


Figure 12: The sample complexity of LieGG for the rotation-MNIST classification task. The red line indicates the symmetry variance and the symmetry bias computed using 100% of available samples.

Implementation details

RETRIEVING SYMMETRIES FROM THE SYNTHETIC REGRESSION To fit the regression function we employ the multi-layer perceptron that consists of linear layers followed by Swish [139] activations with the following dimensions: $(10 \times 32) \rightarrow 3 * (32 \times 32) \rightarrow (32 \times 1)$. The network takes input coordinates and outputs the predicted function value. We use a mean squared error loss to supervise the model. We utilize Adam optimizer [93] with a learning rate of 10^{-3} and train the network for a total of $min(900000/N, 1000)$ epochs, where N is the size of the training dataset. [53] reports this training schedule is enough for convergence.

To compute the mean symmetry variance per a training set size, we average the last 10 singular values of the network polarization matrix. These singular values correspond to the $O(5)$ symmetry as indicated by the corresponding symmetry biases.

The algorithm to construct the network polarization matrix for the synthetic regression experiment is summarized next:

```
def polarization_matrix(model, data, dim = 5):
    # data: torch.FloatTensor(B, 2*dim)
    B = data.shape[0]
    data.requires_grad = True
    data.retain_grad()

    # compute network grads
    model.eval()
    y_pred = model(data)
    y_pred.backward(gradient=torch.ones_like(y_pred))

    # get grads and data per input dimension
    dF_1 = data.grad[...,:dim].view(B, dim, 1)
    data_1 = data[...,:dim].view(B, 1, dim)

    dF_2 = data.grad[...,dim:].view(B, dim, 1)
    data_2 = data[...,dim:].view(B, 1, dim)

    # collect into the network polarization matrix
    C = torch.bmm(dF_1, data_1) + torch.bmm(dF_2, data_2)

    return C
```

RETRIEVING SYMMETRIES FROM THE ROTATION-MNIST We employ two models: shallow (2-layer) and deep (6-layer) perceptrons with 40000 and 160000 parameters respectively. The hidden dimensions for the shallow and deep networks are 47 and 116 respectively. Both models are trained for 300 epochs with Adam optimizer with the learning rate set to 10^{-3}. We use a cross-entropy loss to supervise the model.

Symmetries are extracted from the models with the highest validation accuracy over 300 training epochs. Additionally, to account for a possible difference in the magnitudes of the network outputs, we normalize the network output logits to be unit L2 norm.

The algorithm to construct the network polarization matrix for the rotation MNIST experiment is summarized below:

```
def polarization_matrix_R2(model, data):
    # LieGG implementation with the groups acting on R^2
    # data: torch.FloatTensor(B, 28, 28)

    B, H, W = data.shape

    # compute image grads
    data_grad_x = data[:, 1:, :-1] - data[:, :-1, :-1]
    data_grad_y = data[:, :-1, 1:] - data[:, :-1, :-1]
    dI = torch.stack([data_grad_x, data_grad_y], -1)

    # compute network grads
    data = data.reshape(B, -1)
    data.requires_grad = True
    data.retain_grad()

    output = model(data)
    output.backward(torch.ones_like(output))

    dF = data.grad.reshape(B, H, W)
    dF = dF[:, :-1, :-1]

    # coordinate mask
    xy = torch.meshgrid(torch.arange(0, H-1), torch.arange(0, H-1))
    xy = torch.stack(xy, -1)
    xy = xy / (H // 2) - 1

    # collect into the network polarization matrix
    C = dF[..., None, None] * dI[..., None] * xy[None, :, :, None, :]
    C = C.view(B, -1, 2, 2).sum(1)

    return C
```

SYMMETRIES IN NETWORKS WITH DIFFERENT CONFIGURATIONS To evaluate networks with various configurations (width, depth, number of parameters), we consider the following family of architectures: $(784 \times hdim) \rightarrow p * (hdim \times hdim) \rightarrow (hdim \times 10)$, where p stands for the number of hidden layers and $hdim$ is the dimension of a hidden layer. Given the required number of parameters and the number of hidden layers, we can calculate the hidden dimension of the network. By varying the number of parameters and the number of hidden layers we create wide and deep networks.

3.6 CONCLUSION

SUMMARY We present LieGG - the approach to retrieve symmetries learned by neural networks. Conveniently, our method extracts symmetries directly in the form of corresponding Lie group generators. This allows to explicitly obtain symmetries learned by a neural network without specifying any set of transformations of interest beforehand. We next introduce the symmetry variance and the symmetry bias to evaluate symmetrical properties of neural networks by analyzing the network polarization matrix computed by LieGG. We make use of our approach by studying how the quality of the symmetry learned by the neural network depends on a network's configuration and a training regime. By improving the interpretability of symmetry learning, we identify configurations and training regimes that facilitate training invariances in neural networks.

LIMITATIONS AND FUTURE WORK The proposed method can facilitate the interpretability of neural networks and help to design efficient ways of incorporating symmetries into a model. However, some limitations remain. In particular, LieGG is currently limited to instances of $GL(n)$ connected group, which hinders the analysis of complex invariances, e.g. where a transformation can not be described as a group action. It is an intriguing direction for future research to extend a mechanism of retrieving symmetries to more complex transformations. Nonetheless, LieGG provides an important building block on the way to increasing the interpretability of symmetry learning in neural networks. Also, in our analysis, we focus on the specific family of models, i.e. feed-forward multi-layer perceptions trained on the datasets with well-controlled factors of variations. We assume that further large-scale analysis of LieGG in complex scenarios and with more advanced models can provide a more general understanding of symmetry learning in neural networks.

4

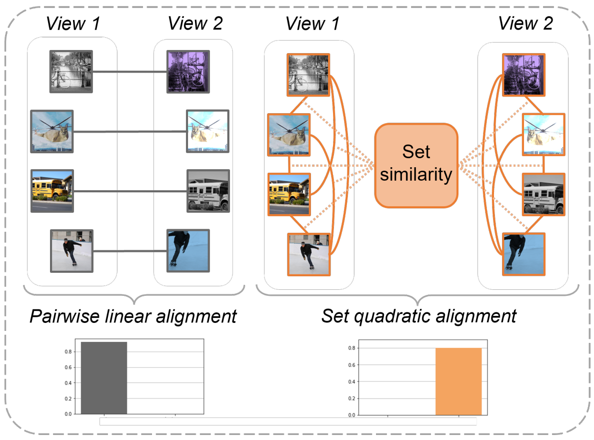
CONTRASTIVE REPRESENTATION LEARNING WITH STRUCTURED ASSIGNMENTS

4.1 INTRODUCTION

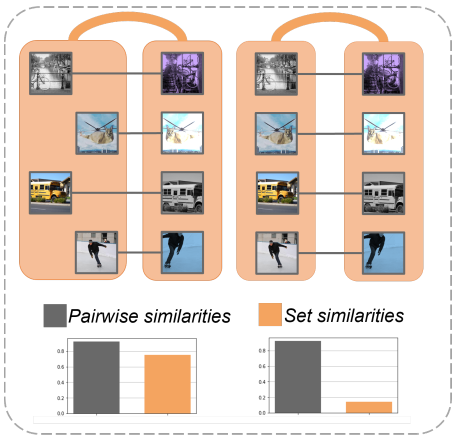
The common approach to contrastive learning is to maximize the agreement between individual views of the data [82, 110]. The views are ordered in pairs, such that they either positive, encoding different views of the same object, or negative, corresponding to views of different objects. The pairs are compared against one another by a contrastive loss-objective [28, 133, 172]. Contrastive learning was successfully applied among others in metric learning [72], self-supervised classification [28, 64], and pre-training for dense prediction tasks [173]. However, when two views of an object are drastically different, the view of object A will resemble the same view of object B much more than it resembles the other view of object A. By comparing individual pairs, the common patterns in the differences between two views will not be exploited. In this paper, we propose to go beyond contrasting individual pairs of objects and focus on contrasting sets of objects.

In contrastive learning, the best alignment of objects follows from maximizing the total similarity over positive pairs, while the negative pairs are needed to encourage diversity. The diversity in representations is there to avoid collapse [163]. We note that one number, the total similarity from contrasting individual pairs, cannot both account for the inter-set similarities of the objects from the same view and the intra-set similarities with objects from another view. Therefore, considering the total similarity over pairs essentially limits the information content of the supervisory signal available to the model. We aim to exploit the information from contrasting sets of objects rather than contrasting objects pairwise only, further illustrated in Figure 13a. From the perspective of set assignment theory [23], two sets may expose the same linear pairwise alignment costs, yet have different internal structures and hence have different set alignment costs, see Figure 13b. Therefore, contrastive learning from sets provides a richer supervisory signal.

To contrast objects as sets, we turn to combinatorial quadratic assignment theory [23] designed to evaluate set and graph similarities. Quadratic assignment has advanced from linear assignment [24], which relies on pairwise similarities between objects. We pose contrastive learning by structural risk minimization [164] in assignment problems. In this, pairwise contrastive learning methods emerge from the linear assignment case. We aim to extend the contrastive objective to the set level by generalizing the underlying assignment problem from linear to quadratic, as illustrated in Figure 13a. Since directly computing set similarities from quadratic alignment is computationally expensive, we derive an efficient approximation. It can be implemented as a regularizer for existing contrastive learning methods. We provide a theory for the proposed method from the per-



(a) Contrasting pairs and contrasting sets



(b) Comparing alignment costs

Figure 13: (a) Contrasting views by similarity over pairs (pairwise linear alignment) versus similarity over sets with set-alignment (set-wise quadratic alignment). (b) Comparing total pairwise similarities versus set similarities for different configurations of representation graphs. In both configurations, the total similarity over pairs remains the same, being unable to discriminate between the internal structures of different views as quadratic alignment can.

spective of assignment problems and dependence maximization. And, we experimentally demonstrate the advantages of contrasting objects by quadratic alignment.

We make the following contributions:

- Using combinatorial assignment theory, we propose to learn representations by contrasting objects as sets rather than as pairs of individual instances.
- We propose a computationally efficient implementation, where the set contrastive part is implemented as a regularizer for existing contrastive learning methods.
- We demonstrate that set-contrastive regularization improves recent contrastive learning methods for the tasks of metric learning and self-supervised classification.

As a byproduct of viewing representation learning through the lens of combinatorial assignment theory, we additionally propose SparseCLR contrastive loss, a modification of the popular InfoNCE [133] objective. Different from InfoNCE, SparseCLR enjoys sparse support over negative pairs, which permits better use of hard negative examples. Our experiments demonstrate that such an approach improves the quality of learned representations for self-supervised classification.

4.2 RELATED WORK

Contrastive learning

In contrastive learning, a model is trained to align representations of the data from different views (modalities), which dates back to the work of Becker *et al.* [9]. Contrastive learning is made up of a joint embedding architecture like Siamese networks [22] and a contrastive objective function. The network maps different views of the data into the embedding space, where the alignment between embeddings is measured by contrasting positive and negative pairs. The pairs either from a complete set of observations [28, 29, 134, 136] or from partially-complete views [78, 109, 187, 188]. And as a contrastive objective, contrastive loss functions are used [28, 64, 133, 172]. Van den Oord *et al.* [133] derived the InfoNCE contrastive loss as a mutual information lower bound between different views of the signal. In computer vision, Chen *et al.* apply the InfoNCE to contrast the images with their distorted version. In PIRL [122], the authors propose to maintain a memory bank of image representations to improve the generalization of the model by sampling negative samples better. Along the same lines, MoCO [31, 64] proposes a running queue of negative samples and a momentum-encoder to increase the contrasting effect of negative pairs. All of these contrastive methods are based on measuring the alignment between individual pairs of objects. This approach does not account for patterns in the views of objects beyond contrasting them pairwise. We aim to extend contrastive learning to include set similarities.

Information maximization methods

In contrastive learning information maximization methods aim to improve contrastive representations by maximizing the information content of the embeddings [7]. In W-MSE [49], batch representations are passed through a whitening, Karhunen-Loève transformation before computing the contrastive loss. Such transformations help to avoid collapsing representations when the contrastive objective can be minimized without learning the discriminative representations. In [193], the authors follow the redundancy-reduction principle to design a loss function to bring the cross-correlation matrix of representations from different views close to identity. In the recent work of Zbontar *et al.*, the authors extend the above formulation of Barlow Twins with an explicit variance term, which helps to stabilize the training. In this paper, we also seek to maximize the information content of the embeddings but we aim to do so by computing the rich representation of set similarities between data views. In contrast to existing methods, our approach does not require any additional transformations like whitening and can be easily incorporated into other contrastive learning methods.

Distillation methods

Another branch of self-supervised methods is not based on contrastive learning but rather based on knowledge distillation [73]. Instead of contrasting positive and negative pairs of samples, these methods try to predict one positive view from another. We discuss them here for completeness. BYOL [58] uses a student network that learns to predict the output of a teacher network. SimSiam [32] simplifies the BYOL by demonstrating that the stop-gradient operation suffices to learn a generalized representation from distillation. In SWaV [27], the authors combine online clustering and distillation by predicting swapped cluster assignments. These self-supervised methods have demonstrated promising results in the task of learning representations. We prefer to follow a different path of contrastive learning, where we do not exclude the possibility that set-based methods may also be applicable to distillation.

Assignment theory

Generally, optimal assignment problems can be categorized into linear and higher-order assignments. Linear assignment problems can be viewed as a transportation problem over a bipartite graph, where the transportation distances are defined on individual pairs of nodes. Linear assignment problems have many real-world applications such as scheduling or vehicle routing, we refer to [24]. Higher-order or in particular Quadratic assignment problems [23, 51] extend the domain of the transportation distances from individual pairs to sets of objects. Where the linear assignment problem seeks to optimally match individual instances, its quadratic counterpart matches the inter-connected graphs of instances, bringing additional structure to the task. In this work, we aim to exploit linear and quadratic assignment theory to rethink contrastive learning.

Structured predictions

In a structured prediction problem, the goal is to learn a structured object such as a sequence, a tree, or a permutation matrix [6]. Training a structure is non-trivial as one has to compute the loss on the manifold defined by the structured output space. In the seminal work of Tsochantaridis *et al.* [164], the authors derive a structured loss for support vector machines and apply it to a sequence labeling task. Later, structured prediction was applied to learn parameters of constraint optimization problems [20, 62]. In this work, we utilize structured prediction principles to implement set similarities into contrastive losses.

4.3 BACKGROUND

We start by formally introducing contrastive learning and linear assignment problems. Then, we argue about the connection between these two problems and demonstrate how one leads to another. This connection is essential for the derivation of our contrastive method.

4.3.1 *Contrastive representation learning*

Consider a dataset $\mathcal{D} = \{d_i\}_{i=1}^{N}$ and an encoder function $f_\theta : \mathcal{D} \longrightarrow \mathbb{R}^{N \times E}$, which outputs an embedding vector of dimension E for each of the objects in $\mathcal{D}$. The task is to learn a such f_θ that embeddings of objects belonging to the same category are pulled together, while the embeddings of objects from different categories are pushed apart.

As category labels are not available during training, the common strategy is to maximize the agreement between different views of the instances, i.e. to maximize the mutual information between different data modalities. The mutual information between two latent variables X, Y, can be expressed as:

$$MI(X,Y) = H(X) - H(X|Y) \tag{4.1}$$

where X and Y correspond to representations of two different views of a dataset. Minimizing the conditional entropy $H(X|Y)$ aims at reducing uncertainty in the representations from one view given the representations of another, while $H(X)$ enforces the diversity in representations and prevents trivial solutions. In practice, sample-based estimators of the mutual information such as InfoNCE [133] or NT-Xent [28] are used to maximize the objective in (4.1).

4.3.2 *Linear assignment problem*

Given two input sets, $\mathcal{A} = \{a_i\}_{i=1}^N$ and $\mathcal{B} = \{b_j\}_{j=1}^N$, we define the inter-set similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ between each element in set $\mathcal{A}$ and each element in $\mathcal{B}$. This similarity matrix encodes *pairwise* distances between the elements of the sets, *i.e.* $[\mathbf{S}]_{i,j} = \phi(a_i, b_j)$, where ϕ is a distance metric. The goal of the linear assignment problem is to find a one-to-one assignment $\hat{y}(\mathbf{S})$, such that the sum of distances between assigned elements is optimal. Formally:

$$\hat{y}(\mathbf{S}) = \underset{\mathbf{Y} \in \Pi}{\operatorname{argmin}}\, tr(\mathbf{S}\mathbf{Y}^T) \tag{4.2}$$

where Π corresponds to a set of all $N \times N$ permutation matrices.

The linear assignment problem in (4.2) is also known as the bipartite graph matching problem. It can be efficiently solved by linear programming algorithms [17].

4.3.3 *Learning to produce correct assignments*

Consider two sets, $\mathcal{D}_{\mathcal{Z}_1}$ and $\mathcal{D}_{\mathcal{Z}_2}$, which encode two different views of the dataset $\mathcal{D}$ under two different views $\mathcal{Z}_1$ and $\mathcal{Z}_2$. By design, the two sets consist of the same instances, but the modalities and the order of the objects may differ. With the encoder, which maximizes the mutual information, objects in both sets can be uniquely associated with one another by comparing their representations as the uncertainty term in (4.1) should be minimized. In other words, $\hat{y}(\mathbf{S}) = \mathbf{Y}_{gt}$, where $\mathbf{Y}_{gt} \in \Pi$ is a ground truth assignment between elements of $\mathcal{D}_{\mathcal{Z}_1}$ and $\mathcal{D}_{\mathcal{Z}_2}$.

Thus, a natural objective to supervise an encoder function is to train it to produce the correct assignment between the different views of the data. As the assignment emerges as a result of the optimization problem, we employ a structured prediction framework [164] to define a structural loss from the linear assignment problem as follows:

$$L(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \min_{\mathbf{Y} \in \Pi} tr(\mathbf{S}\mathbf{Y}^T) \tag{4.3}$$

where $\mathbf{S} = \phi\big(f_\theta(\mathcal{D}_{\mathcal{Z}_1}), f_\theta(\mathcal{D}_{\mathcal{Z}_2})\big)$. Note that $L \geq 0$ and $L = 0$ only when the similarities produced by an encoder lead to the correct optimal assignment. Intuitively, the structured linear[1] assignment loss in (4.3) encodes the discrepancy between the true cost of the assignment and the cost induced by an encoder.

By minimizing the objective in (4.3), we train the encoder f_θ to correctly assign objects from one view to another. In practice, it is desirable that such assignment is robust under small perturbations in the input data. The straightforward way to do so is to enforce a separation margin m in the input similarities as $\mathbf{S}_m = \mathbf{S} + m\mathbf{Y}_{gt}$. Then, the structured linear assignment loss with separation margin reduces to a known margin Triplet loss [72].

Proposition 1. *The structured linear assignment loss $L(\mathbf{S}_m, \mathbf{Y}_{gt})$ with separation margin $m \geq 0$ is equivalent to the margin triplet loss.*

Proof. Let $\mathbf{Y}^* = \hat{y}(\mathbf{S}_m)$ be the solution of the optimal linear assignment problem given the similarity matrix $\mathbf{S}_m$. With this, we can rewrite $L(\mathbf{S}_m, \mathbf{Y}_{gt})$ as follows:

$$L(\mathbf{S}_m, \mathbf{Y}_{gt}) = \sum_i \sum_j ([\mathbf{Y}_{gt}]_{ij} - [\mathbf{Y}^*]_{ij})[\mathbf{S}_m]_{ij} = \sum_i \sum_j q_{ij}[\mathbf{S}_m]_{ij} \tag{4.4}$$

Note that the term q_{ij} can only take the values in $\{-1, 0, 1\}$ as both matrices $\mathbf{Y}_{gt}$ and $\mathbf{Y}^*$ are binary. Consider the case when $q_{ij} = 0$, which indicates that the objects i and j are matched correctly and that $[\mathbf{S}]_{ij} + m \leq [\mathbf{S}]_{ik}$ for any $k \neq j$. The latter implies that the distance between the positive pair (i, j) is at least m smaller than the distance with all other pairs in the batch.

Next, note that the optimal value of the loss function $L(\mathbf{S}_m, \mathbf{Y}_{gt}) = 0$ is achieved when all q_{ij} equate to zero, which implies that at the optimum, distances in all positives pairs are at least m smaller than the distances in all negative pairs. This optimality condition reassembles the formulation of margin Triplet loss [72]. □

MINING STRATEGIES. By default, the loss in (4.3) enforces a one-to-one negative pair mining strategy due to the structural domain constraint $\mathbf{Y} \in \Pi$. By relaxing this domain constraint to row-stochastic binary matrices, we arrive at the known batch-hard mining [72]. This is essential to have a computationally tractable implementation of structured assignment losses.

SMOOTHNESS. An immediate issue when directly optimizing the structured linear assignment loss is the non-smoothness of the minimum function in (4.3). It is known that optimizing smoothed functions can be more efficient than directly optimizing their non-smooth counterparts [8]. The common way to smooth a minimum is by *log-sum-exp* approximation. Thus, we can obtain a smoothed version of structured linear assignment loss:

$$L_\tau(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) + \tau \log \sum_{\mathbf{Y} \in \Pi} \exp(-\frac{1}{\tau} tr(\mathbf{S}\mathbf{Y}^T)) \tag{4.5}$$

where τ is a temperature parameter controlling the degree of smoothness. Practically, the formulation in (4.5) requires summing $N!$ terms, which makes it computationally intractable under the default structural constraints $\mathbf{Y} \in \Pi$. Fortunately, similar to the

1 We emphasize that the term *linear* here is only used with regard to an underlying assignment problem.

non-smooth case, we can utilize batch-hard mining, which leads to $O(N^2)$ computational complexity. The smoothed structured linear assignment loss with batch-hard mining reduces to known normalized-temperature cross entropy [28] also known as the InfoNCE [133] loss.

Proposition 2. *The smoothed structured linear assignment loss $L_\tau(\boldsymbol{S}, \boldsymbol{Y}_{gt})$ with batch-hard mining is equivalent to the normalized-temperature cross entropy loss.*

Proof. Recall that the batch-hard mining emerges from relaxing constraint $\mathbf{Y} \in \Pi$ into $\mathbf{Y} \in \mathcal{R}$, where $\mathcal{R}$ is a set of binary row-stochastic matrices. We can rewrite $L_\tau(\mathbf{S}_m, \mathbf{Y}_{gt})$ with batch-hard mining as:

$$\begin{aligned} \hat{L}_\tau(\mathbf{S}, \mathbf{Y}_{gt}) &= \sum_{(p,k)\in\mathbf{Y}_{gt}} [\mathbf{S}]_{pk} + \tau \sum_i log\Big(\sum_j exp(-\frac{1}{\tau}[\mathbf{S}]_{ij})\Big) \\ &= \sum_{(p,k)\in\mathbf{Y}_{gt}} \phi(p,k) + \tau \sum_i log\Big(\sum_j exp(-\frac{1}{\tau}\phi(i,j))\Big) \end{aligned} \tag{4.6}$$

where $\phi(p,k)$ is a distance measure between the samples p and k, and τ is the parameter controlling a degree of smoothness. Note that in (4.6) we assume that the distance $\phi(p,k)$ decreases when the similarity between the samples increases.

The first and the second terms in (4.6) relates to the alignment and distribution terms [30] and reassemble the formulation of the normalized temperature cross entropy loss (InfoNCE) [28, 133] up to a normalization constant. □

CONNECTION TO MUTUAL INFORMATION It is known that the InfoNCE objective is a lower bound on the mutual information between the representations of data modalities [133]. Thus, Propositions 1 and 2 reveal a connection between mutual information maximization and minimization of structured losses for assignmnet problems. In fact, the assignment cost $tr(\mathbf{S}\mathbf{Y}_{gt}^T)$ in (4.2) is related to the conditional entropy $H(X|Y)$, while min $_{\mathbf{Y}\in\Pi} tr(\mathbf{S}\mathbf{Y}^T)$ aims to maximize the diversity in representations. This connection allows for consideration of contrastive representation learning methods based on InfoNCE as a special case the structured linear assignment loss.

4.4 EXTENDING CONTRASTIVE LOSSES

We next demonstrate how to exploit the connection between contrastive learning and assignment problems to extend contrastive losses the set level. As a byproduct of this connection, we also derive the SparseCLR contrastive objective.

4.4.1 *Contrastive learning with quadratic assignments on sets*

Quadratic assignment problem

As in the linear case, we are given two input sets $\mathcal{A}, \mathcal{B}$ and the inter-set similarity matrix $\mathbf{S}$. For the Quadratic Assignment Problem (QAP), we define intra-set similarity matrices $\mathbf{S}_{\mathcal{A}}$ and $\mathbf{S}_{\mathcal{B}}$ measuring similarities within the sets $\mathcal{A}$ and $\mathcal{B}$ respectively, *i.e.* $[\mathbf{S}_{\mathcal{A}}]_{ij} = \phi(a_i, a_j)$. A goal of the quadratic assignment problem is to find a *one-to-one* assignment $\hat{y}_Q(\mathbf{S}, \mathbf{S}_{\mathcal{A}}, \mathbf{S}_{\mathcal{B}}) \in \Pi$ that maximizes the set similarity between $\mathcal{A}, \mathcal{B}$, where the set similarity is defined as follows:

$$Q(\mathbf{S}, \mathbf{S}_{\mathcal{A}}, \mathbf{S}_{\mathcal{B}}) = \min_{\mathbf{Y} \in \Pi} \left\{ tr(\mathbf{S}\mathbf{Y}^T) + tr(\mathbf{S}_{\mathcal{A}}\mathbf{Y}\mathbf{S}_{\mathcal{B}}^T\mathbf{Y}^T) \right\} \tag{4.7}$$

Compared to the linear assignment problem, the quadratic term $Q(\mathbf{S}, \mathbf{S}_{\mathcal{A}}, \mathbf{S}_{\mathcal{B}})$ in (4.7) additionally measures the discrepancy in internal structures between sets $\mathbf{S}_{\mathcal{A}}$ and $\mathbf{S}_{\mathcal{B}}$.

Learning with quadratic assignments

Following the similar steps as for the linear assignment problem in Section 4.3.3, we next define the structured quadratic assignment loss by extending the linear assignment problem in (4.3) with the quadratic term:

$$L_{QAP} = tr(\mathbf{S}\mathbf{Y}_{gt}^T) - Q(\mathbf{S}, \mathbf{S}_{\mathcal{A}}, \mathbf{S}_{\mathcal{B}}) \tag{4.8}$$

Minimization of the structured quadratic assignment loss in (4.8) encourages the encoder to learn representations resulting in the same solutions of the linear and quadratic assignment problems, which is only possible when the inter-set and intra-set similarities are sufficiently close [23]. Note that we do not use a ground truth quadratic assignment in L_{QAP}, but a ground truth linear assignment objective is used for the supervision. This is due to a quadratic nature of Q, where minimizing the discrepancy between ground truth and predicted assignments is a subtle optimization objective, e.g. for an equidistant set of points the costs of all quadratic assignments are identical.

To compute L_{QAP}, we first need to evaluate the quadratic term $Q(\mathbf{S}, \mathbf{S}_{\mathcal{A}}, \mathbf{S}_{\mathcal{B}})$, which requires solving the quadratic assignment. This problem is known to be notoriously hard to solve exactly even when an input dimensionality is moderate [23]. To alleviate this, we use a computationally tractable upper-bound:

$$\begin{aligned} L_{QAP} &\le tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \min_{\mathbf{Y} \in \Pi} tr(\mathbf{S}\mathbf{Y}^T) - \min_{\mathbf{Y} \in \Pi} tr(\mathbf{S}_{\mathcal{A}}\mathbf{Y}\mathbf{S}_{\mathcal{B}}^T\mathbf{Y}^T) \\ &\le L(\mathbf{S}, \mathbf{Y}_{gt}) - \langle \lambda_{\mathcal{A}}, \lambda_{\mathcal{B}} \rangle_- \end{aligned} \tag{4.9}$$

where $\langle \lambda_{\mathcal{A}}, \lambda_{\mathcal{B}} \rangle_-$ corresponds to a minimum dot product between eigenvalues of matrices S_A and S_B.

The first inequality is derived from Jensen's inequality for (4.8) and the second inequality holds due to the fact that $\langle \lambda_F, \lambda_D \rangle_- \le tr(\mathbf{F}\mathbf{X}\mathbf{D}^T\mathbf{X}^T) \le \langle \lambda_F, \lambda_D \rangle_+$ for symmetric matrices $\mathbf{F}, \mathbf{D}$ and $\mathbf{X} \in \Pi$ as shown in *Theorem 5* by Burkard [23]. The above derivations are for the case when the similarity metric is a valid distance function, i.e. minimizing a distances leads to maximizing a similarity. The derivation for the reverse case can be done analogously by replacing *min* with *max* in the optimal assignment problem formulation.

Optimizing the upper-bound in (4.9) is computationally tractable compared to optimizing the exact version of L_{QAP}. Another advantage is that the upper-bound in (4.9) breaks down towards minimizing the sum of the structured linear assignment loss $L(\mathbf{S}, \mathbf{Y}_{gt})$ and the regularizing term, which accounts for the set similarity. This allows to easily modify existing contrastive learning approaches like those in [28, 72, 119, 133] that are based on pairwise similarities and thus stem from the linear assignment problem by simply plugging in the regularizing term. We provide a simple code example demonstrating InfoNCE with the quadratic assignment regularization below:

```
# f: encoder network
# alpha: weighting for the pairwise contrastive part
# beta: weighting for the set contrastive part
# N: batch size
# E: dimensionality of the embeddings

for x in loader: # load a batch with N samples
    # two randomly augmented views of x
    y_a, y_b = augment(x)

    # compute embeddings
    z_a = f(y_a) # NxE
    z_b = f(y_b) # NxE

    # compute inter-set and intra-set similarities
    S_AB = similarity(z_a, z_b) # NxN
    S_A  = similarity(z_a, z_a) # NxN
    S_B  = similarity(z_b, z_b) # NxN

    # compute pairwise contrastive InfoNCE loss
    pairwise_term = infonce(S_AB)

    # compute eigenvalues
    eigs_a = eigenvalues(S_A) #N
    eigs_b = eigenvalues(S_B) #N

    # compute QARe from minimum dot product of eigenvalues
    eigs_a_sorted = sort(eigs_a, descending = True) #N
    eigs_b_sorted = sort(eigs_b, descending = False) #N
    qare = -1*(eigs_a_sorted.T@eigs_b_sorted)

    # combine pairwise contrastive loss with QARe
    loss = alpha*pairwise_term + beta*qare/(N^2)

    # optimization step
    loss.backward()
    optimizer.step()
```

Computational complexity

Computing the upper-bound in (4.9) has a computational complexity of $O(N^3)$ as one needs to compute eigenvalues of the intra-set similarity matrices. This is opposed to (4.8) that requires directly computing quadratic assignments, for which it is known there exist no polynomial-time algorithms [23]. The computational complexity can be pushed further down to $O(k^2N)$ by evaluating the only top-k eigenvalues instead of computing all eigenvalues.

4.4.2 *SparseCLR*

In Section 4.3.3 we noted that the smoothness of a loss function is a desirable property for optimization and that the *log-sum-exp* smoothing of the structured linear assignment loss yields the normalized temperature cross-entropy objective as a special case. Such an approach, however, is known to have limitations. Specifically, the *log-sum-exp* smoothing yields dense support over samples, being unable to completely rule out irrelevant examples [131]. That can be statistically and computationally overwhelming and can distract the model from fully utilizing information from hard negative examples.

We propose to use *sparsemax* instead of *log-sum-exp* approximation to alleviate the non-smoothness of the structured linear assignment objective, yet keep the sparse support of the loss function. Here *sparsemax* is defined as the minimum distance Euclidean projection to a k-dimensional simplex as in [116, 131].

Let $\tilde{x} :\in \mathbb{R}^{N!}$ be a vector that consists of the realizations of all possible assignment costs for the similarity matrix $\mathbf{S}$. With this we can define SparseCLR as follows:

$$L_{sparse}(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \frac{1}{2} \sum_{j \in \Omega(\tilde{x})} \left(\tilde{x}_j^2 - \mathcal{T}^2(-\tilde{x}) \right) \tag{4.10}$$

where $\Omega(X) = \{j \in X : \ sparsemax(X)_j > 0\}$ is the support of *sparsemax* function, and $\mathcal{T}$ denotes the thresholding operator [116]. In practice, to avoid summing over a factorial number of terms, as well as for other methods, we use batch-hard mining strategy resulting in $O(N^2)$ computational complexity for SparseCLR.

4.4.3 *Formulations of contrastive losses with batch-hard mining*

So far, the proposed contrastive losses have been formulated in a one-to-one mining form, which emerges when considering assignment problems. Next, we present the formulations of the proposed contrastive losses with batch-hard mining strategy. To this end, we start with the derivation for the case of the structured linear assignment loss, while the derivations for the smoothed structured linear assignment and SparseCLR losses can be done analogously.

We start from the original structured linear assignment loss with one-to-one mining:

$$L(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \min_{\mathbf{Y} \in \Pi} tr(\mathbf{S}\mathbf{Y}^T) \tag{4.11}$$

To derive the loss under the batch-hard mining strategy, the structural constraint $\mathbf{Y} \in \Pi$ is relaxed to $\mathbf{Y} \in \mathcal{R}$, where $\mathcal{R}$ is a set of row-stochastic binary matrices, i.e. $[\mathbf{Y}]_{ij} \in \{0, 1\}$ for $\forall(i, j)$ and $\sum_j [\mathbf{Y}]_{ij} = 1$ for $\forall i$. With this, the structured linear assignment loss with the batch-hard mining is defined as follows:

$$\begin{aligned}
\hat{L}(\mathbf{S}, \mathbf{Y}_{gt}) &= tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \min_{\mathbf{Y} \in \mathcal{R}} tr(\mathbf{S}\mathbf{Y}^T) \\
&= tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \min_{u_1 \dots u_N} \sum_i (\sum_j [\mathbf{S}]_{ij} [u_i]_j) \\
&= tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \sum_i \min_{u_i} (\sum_j [\mathbf{S}]_{ij} [u_i]_j) \\
&= tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \sum_i \min_j [\mathbf{S}_{ij}]
\end{aligned} \tag{4.12}$$

where the first inequality follows from the fact that the rows $u_1 \dots u_N$ of $\mathbf{Y} \in \mathcal{R}$ are independent of each other, and the last inequality is due to u_i is a binary vector containing a one-hot encoding of the minimum index.

► Smoothed structured linear assignment loss $L_\tau(\mathbf{S}, \mathbf{Y}_{gt})$ with batch-hard-mining:

$$\hat{L}_\tau(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) + \tau \sum_i \log \sum_j \exp(-\frac{1}{\tau}[\mathbf{S}]_{ij}) \tag{4.13}$$

► SparseCLR contrastive loss $L_{sparse}(\mathbf{S}, \mathbf{Y}_{gt})$ with batch-hard mining:

$$\hat{L}_{sparse}(\mathbf{S}, \mathbf{Y}_{gt}) = tr(\mathbf{S}\mathbf{Y}_{gt}^T) - \frac{1}{2} \sum_i \sum_{j \in \Omega(h_i)} \left([h_i]_j^2 - \mathcal{T}^2(-h_i) \right) \tag{4.14}$$

where $h_i \in \mathbb{R}^N$ correspond to a i-th row of the similarity matrix $\mathbf{S}$, and $\Omega(X) = \{j \in X : \; sparsemax(X)_j > 0\}$ is the support of *sparsemax* function [116]. The thresholding operator $\mathcal{T}$ is defined as:

$$\mathcal{T}(z) = \frac{(\sum_{j \in \Omega(z)} z_j) - 1}{|\Omega(z)|} \tag{4.15}$$

And the *sparsemax* function is defined as:

$$sparsemax(z) = \underset{p \in \Delta^{N-1}}{\operatorname{argmin}} \, ||p - z||^2 \tag{4.16}$$

with Δ^{N-1} being $N - 1$ dimensional simplex.

4.5 EXPERIMENTS

In this section, we evaluate the quality of representations trained with and without our Quadratic Assignment Regularization (QARe). We also evaluate the performance of SparseCLR and compare it with other contrastive losses. As we consider the representation learning from the perspective of the assignment theory, we firstly conduct the

Method	Triplet	SparseCLR	InfoNCE	NTLogistic
pairwise	54.85±0.77	53.03±0.43	57.87±1.19	40.30±0.42
+QARe	**58.48±1.67**	**54.84±1.40**	**61.96±1.38**	**43.48±1.41**

Table 2: Matching accuracy for instances from different views of CUHK-03 dataset. The pairwise corresponds to the contrastive losses acting on the level of pairs. +QARe denotes methods with quadratic assignment regularization. Best results are in bold.

instance matching experiment, where the goal is to learn to predict the correct assignment between different views of the dataset. Then, we test the proposed method on the task of self-supervised classification and compare it with other contrastive learning approaches. Next, we present ablation studies to visualize the role of the weighting term, when combining QARe with the baseline contrastive learning method.

4.5.1 *Matching instances from different views*

In this experiment, the goal is to train representations of objects from different views, such that the learned representations provide a correct matching between identities in the dataset. This problem is closely related to a metric learning and can be solved by contrasting views of the data [72].

Data

We adopt the CUHK-03 dataset [107] designed for the ReID [72] task. CUHK-03 consist of 1467 different identities recorded each from front and back views. To train and test the models, we randomly divide the dataset into *70/15/15* train/test/validation splits.

Evaluation

We evaluate the quality of representations learned with contrastive losses by computing the matching accuracy between front and back views. In practice, we first obtain the embeddings of the views of the instances from the encoder and then compute their inter-view similarity matrix using the Euclidean distance. Given this, the matching accuracy is defined as the mean Hamming distance between the optimal assignment from the inter-view similarity matrix and the ground truth assignment. We report the average matching accuracy and the standard deviation over 3 runs with the same fixed random seeds. As a baseline, we chose Triplet with batch-hard mining [72], InfoNCE [133] and NTLogistic [119] contrastive losses, which we extend with the proposed QARe.

Implementation details

BACKBONE As the encoder, we use ResNet-18 [67] with the top classification layer replaced by the linear projection head that outputs feature vectors of dimension 64. To obtain representations, we normalize the feature vectors to be L2 unit-norm.

TRAINING We train the models for 50 epochs using Adam optimizer with the cosine annealing without restarts [111]. Initial learning rate is 0.01 and a batch size is set to 128. During the training, we apply random color jittering and horizontal flip augmentations. To compute the test matching accuracy, we select the best model over the epochs based on the validation split. During the training, we apply a simple augmentation strategy: horizontal flip with a 50% chance and color jittering. For the latter, the brightness, contrast, saturation, and hue are sampled from a uniform distribution $U[0, 0.1]$.

HYPER-PARAMETERS OF THE LOSS FUNCTIONS We use 4 baseline contrastive losses: margin Triplet [72], InfoNCE [133], NT-Logistic [119] and the proposed SparseCLR losses, which we extend with the Quadratic Assignment Regularization (QARe). The margin parameter for Triplet loss is set to 0.5, the temperature parameters for InfoNCE and NT-Logistic are both set to 0.05. We combine QARe and the backbone contrastive learning loss by taking their convex combination, where the QARe term is weighted by the constant $\beta \geq 0$. The exact values of β are $0.4/0.5/0.2/0.3$ for margin Triplet / InfoNCE / NT-Logistic / SparseCLR.

Results

The results are reported in Table 2. As can be seen, adding QARe regularization to a baseline contrastive loss leads to a better matching accuracy, which indicates an improved quality of representations. Notably, QARe delivers 3.6% improvement when combined with Triplet loss and 4.1% increase in accuracy with the InfoNCE objective. This demonstrates that the quadratic assignment regularization on sets helps the model to learn generalized representation space.

4.5.2 *Self-supervised classification*

Next, we evaluate the quadratic assignments on sets in the task of self-supervised classification. The goal in this experiment is to learn a representation space from an unlabeled data that provides a linear separation between classes in a dataset. This is a common problem to test contrastive learning methods.

Evaluation

We follow the standard linear probing protocol [94]. The evaluation is performed by retrieving embeddings from a contrastively trained backbone and fitting a linear classifier on top of the embeddings. Since linear classifiers have a low discriminative power on their own, they heavily rely on input representations. Thus, a higher classification accuracy indicates a better quality of learned representations. As the baseline for pairwise contrastive learning methods, we select popular SimCLR and proposed SparseCLR. We extend the methods to the set level by adding the quadratic assignment regularization. We compare the set-level contrastive methods with other self-supervised [26, 54] and contrastive approaches [27, 28].

	Conv-4			ResNet-32		
Method	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**
Supervised (upper bound)	80.46±0.39	49.29±0.85	36.47±0.36	90.87±0.41	65.32±0.22	50.09±0.32
Random Weights (lower bound)	32.92±1.88	10.79±0.59	6.19±0.13	27.47±0.83	7.65±0.44	3.24±0.43
DeepCluster [26]	42.88±0.21	21.03±1.56	12.60±1.23	43.31±0.62	20.44±0.80	11.64±0.21
RotationNet [54]	56.73±1.71	27.45±0.80	18.40±0.95	62.00±0.79	29.02±0.18	14.73±0.48
Deep InfoMax [74]	44.60±0.27	22.74±0.21	14.19±0.13	47.13±0.45	24.07±0.05	17.51±0.15
SimCLR* [28]	59.64±0.93	30.75±0.23	21.15±0.16	76.45±0.10	40.18±0.13	23.44±0.27
SimCLR+QARe (Ours)	**61.54±1.17**	**31.18±0.41**	**21.33±0.18**	**76.85±0.39**	**41.20±0.34**	**25.05±0.18**

*Table 3: Self-supervised training and linear probing for a self-supervised classification. Average classification accuracy (percentage) and standard deviation over 3 runs with common fixed random seeds are reported. * we report the results from our reimplementation of SimCLR.*

Implementation details

BACKBONES We use two models: Conv-4 and ResNet-32. The Conv-4 model involves 4 blocks. The first 3 blocks consist of 8, 16, and 32 feature maps respectively. Each performs a convolution with a kernel size of 3, a stride of 1, and a padding of 1 pixel, followed by a batch-normalization layer, a ReLU, and an average pooling layer with a kernel size of 2 and a stride of 2 pixels. The fourth block performs the same operations but instead of an average pooling, an adaptive average pooling is used. For ResNet-32, we use off-the-shelf Pytorch implementation as described in [67]. We initialize the model using standard Xavier initialization [56] and set batch-normalization weights and biases to 1 and 0 respectively.

TRAINING We train the models for 200 epochs using Adam optimizer with a learning rate of 0.001 and a batch size is set to 128. For contrastive learning methods, we adopt a set of augmentations from [28] to form different views of the data. For linear probing, we freeze the encoder weights and train a logistic regression on top of the learned representation for 100 epochs with Adam optimizer, using a learning rate of 0.001 and a batch size of 128. We apply the following augmentations to images: horizontal flip with a 50% chance, random crop-resize, grayscale conversion with a 20% chance, and color jitter with an 80% chance. Random crop-resize consists of cropping the given image from 0.08 to 1.0 of the original size, then changing its aspect ratio from 3/4 to 4/3 of the original, and finally resizing to input shape using a bilinear interpolation. For color jitter, the brightness, contrast and saturation are sampled from $U[0, 0.8]$, and the hue is sampled from $U[0, 0.2]$.

We perform training and testing on standard CIFAR-10, CIFAR-100, and tiny-ImageNet datasets. For each dataset, we follow the same training procedure and fine-tune the optimal weighting of the quadratic assignment regularizer.

HYPER-PARAMETERS OF THE LOSS FUNCTIONS We combine quadratic assignment regularization and the backbone contrastive learning loss by summing with weighting the QARe term by a constant $\beta \geq 0$. The values of β for each architecture and dataset are listed in Table 4. The temperature parameter for both SimCLR [29] and SimCLR+QARe is set to 0.05.

	Conv-4			ResNet-32		
Method	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**
(β) SimCLR+QARe	0.5	0.375	0.875	1.125	1.125	1.125
(β) SparseCLR+QARe	0.125	1.25	0.5	1	0.875	1.25

Table 4: QARe weighting for the architectures and the datasets for self-supervised classification experiment.

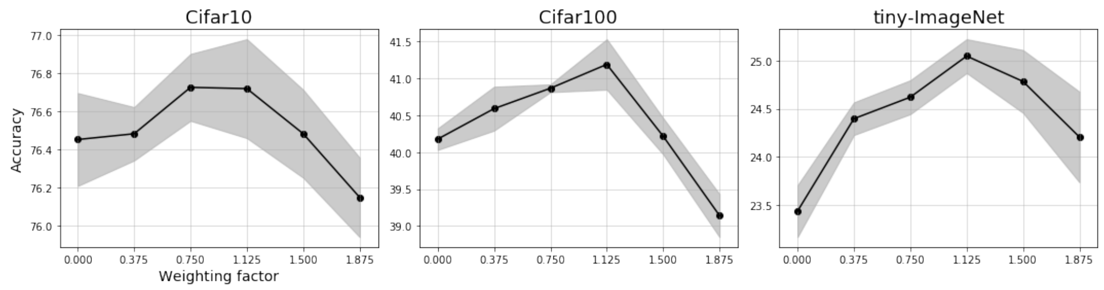


Figure 14: Influence of the weighting of the QARe term in the self-supervised classification for ResNet-32 trained with SimCLR. The accuracy is recorded over 3 runs with common fixed random seeds.

QARe results

The results for the quadratic assignment regularization are reported in Table 3 for SimCLR and in Table 5 for proposed SparseCLR. For SimCLR, adding QARe delivers 1.9% accuracy improvement on CIFAR-10 for the shallow encoder network and up to 1.6 % improvement for ResNet-32 on tiny-ImageNet. We observed a consistent improvement in other dataset-architecture setups, except tiny-ImageNet with shallow Conv-4 encoder, where the performance gain from adding QARe is modest. For this case, we investigated the training behavior of the models and observed that both SimCLR and SimCLR+QARe for Conv-4 architecture very quickly saturate to a saddle point, where the quality of representations stops improving. Since this does not happen with ResNet-32 architecture, we attribute this phenomenon to the limited discriminative power of the shallow Conv-4 backbone, which can not be extended by regularizing the loss.

For SparseCLR, we observed the same overall pattern. As can be seen from Table 5, extending SparseCLR to set level with QARe delivers 1.6% accuracy improvement for ResNet-32 on tiny-ImageNet and also steadily improve the performance on other datasets. This indicates that QARe helps to provide a richer supervisory signal for the model to train representations.

SparseCLR results

Next, we compare SimCLR against the proposed SparseCLR. As can be seen in Table 5, SparseCLR consistently improves over the baseline on CIFAR-100 and tiny-ImageNet datasets, where it delivers 2.4% improvement. Surprisingly, we noticed a significant drop in performance for ResNet-32 on CIFAR-10. For this case, we investigated the training behavior of the model and observed that in the case of CIFAR-10, the batch often includes many false negative examples, which results in uninformative negative

	Conv-4			ResNet-32		
Method	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**	**CIFAR-10**	**CIFAR-100**	**tiny-ImageNet**
SimCLR* [28]	59.64±0.93	30.75±0.23	21.15±0.16	**76.45±0.10**	40.18±0.13	23.44±0.27
SparseCLR (Ours)	59.86±0.56	32.04±0.26	**22.41±0.39**	70.37±0.12	41.05±0.20	25.87±0.59
SparseCLR+QARe (Ours)	**61.06±0.39**	**33.05±0.32**	**22.45±0.30**	71.39±0.30	**42.07±0.35**	**27.03±0.40**

Table 5: Self-supervised training on unlabeled data and linear evaluation on labeled data. Comparing SimCLR [28] with the proposed SparseCLR and SparseCLR with QARe. Average accuracy and standard deviation over three runs over 3 runs with common fixed random seeds.

pairs. Since SparseCLR assigns the probability mass only to a few hardest negative pairs, a lot of false-negative examples in the batch impede obtaining a clear supervisory signal. We assume the problem can be alleviated by using false-negative cancellation techniques [79].

Ablation study

Here we illustrate how the weighting of the QARe term influences the quality of representations learned with SimCLR under a linear probing evaluation. We search for the optimal weighting in the range from 0 to 1.875 with a step of 0.125. The results are depicted in Figure 14. In practice, we observed that QARe is not too sensitive to a weighting and the values in the range 0.75-1.25 provide consistent improvement.

Time and memory complexity

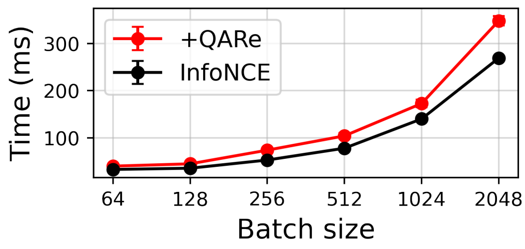


Figure 15: A training iteration time complexity of InfoNCE with and without the quadratic assignment regularization.

In addition to computational complexity, we provide an empirical analysis of how the proposed regularization influences the time and memory complexity of a baseline method (InfoNCE). We observe 13% increase in training time for the batch size of 256, and up to 29% for the batch size of 2048 (Figure 15). The increase in memory consumption is negligible (+0.78% for the batch size of 2048).

4.6 DISCUSSION

In this work, we present set contrastive learning method based on quadratic assignments. Different from other contrastive learning approaches, our method works on the level of set similarities as opposed to only pairwise similarities, which allows improving the information content of the supervisory signal. For derivation, we view contrastive learning through the lens of combinatorial assignment theory. We show how pairwise contrastive methods emerge from learning to produce correct optimal assignments and then extend them to a set level by generalizing an underlying assignment problem, implemented as a regularization for existing methods. As a byproduct of viewing representation learning through the lens of assignment theory, we additionally propose SparceCLR contrastive loss.

Our experiments in instance matching and self-supervised classification suggest that adding quadratic assignment regularization improves the quality of representations learned by backbone methods. We suppose, that our approach would be most useful in the problems where the joint analysis of objects and their groups appears naturally and labeling is not readily available.

5

SCALE EQUIVARIANT REPRESENTATIONS FOR ROBUST VISUAL OBJECT TRACKING

5.1 INTRODUCTION

Siamese trackers turn tracking into similarity estimation between a template and the candidate regions in the frame. The Siamese networks are successful because the similarity function is powerful: it can learn the variances of appearance very effectively, to such a degree that even the association of the frontside of an unknown object to its backside is usually successful. And, once the similarity is effective, the location of the candidate region is reduced to simply selecting the most similar candidate.

Mathematically, one of the key ingredients of the success of the similarity function is translation *equivariance*, i.e. a translation in the input image is to result in the proportional translation in feature space. Non-translation-equivariant architectures will induce a positional bias during training, so the location of the target will be hard to recover from the feature space [105, 199]. In real-life scenarios, the target will undergo more transformations than just translation, and, unless the network has an internal mechanism to handle them, the similarity may degrade. We start from the position that equivariance to common transformations should be the guiding principle in designing conceptually simple yet robust trackers. To that end, we focus on scale equivariance for trackers in this paper.

Measuring scale precisely is crucial when the camera zooms its lens or when the target moves into depth. However, scale is also important in distinguishing among objects in general. In following a marching band or in analyzing a soccer game, or when many objects in the video have a similar appearance (a crowd, team sports), the similarity power of Siamese trackers has a hard time locating the right target. In such circumstances, spatial-scale equivariance will provide a richer and hence more discriminative descriptor, which is essential to differentiate among several similar candidates in an image. And, even, as we will demonstrate, when the sequence does not show variation over scale, proper scale measurement is important to keep the target bounding box stable in size.

The common way to implement scale into a tracker is to train the network on a large dataset where scale variations occur naturally. However, as was noted in [100], such training procedures may lead to learning groups of re-scaled duplicates of almost the same filters. As a consequence, inter-scale similarity estimation becomes unreliable, see Figure 16 top. Scale-equivariant models have an internal notion of scale and built-in weight sharing among different filter scales. Thus, scale equivariance aims to produce the same distinction for all sizes, see Figure Figure 16 bottom.

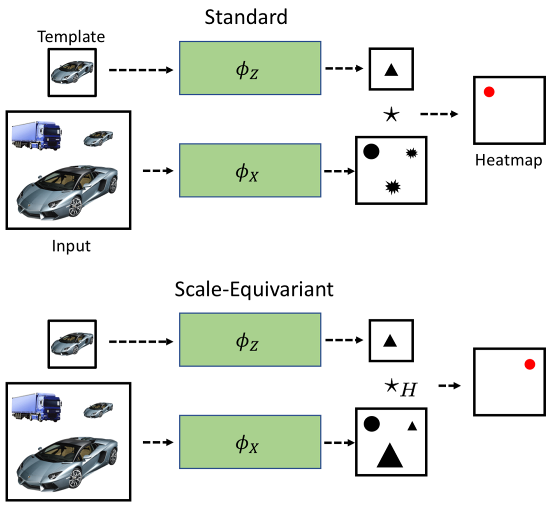


Figure 16: The standard version (top) and the scale-equivariant version (bottom) of a basic tracker. The scale-equivariant tracker has an internal notion of scale which allows for the distinction between similar objects which only differ in scale. The operator $\star$ denotes convolution, and $\star_H$ stands for scale-convolution.

In this paper, we aim to equip the Siamese network with spatial and scale equivariance built-in from the start to capture the natural variations of the target *a priori*. We aim to improve a broad class of tracking algorithms by enhancing their capacity of candidate distinction. We adopt recent advances [156] in convolutional neural networks (CNNs) which handle scale variations explicitly and efficiently.

While scale-equivariant convolutional models have lead to success in image classification [156, 180], we focus on their usefulness in *object localization*. Where scale estimation has been used in the localization for tracking, it typically relies on brute-force multi-scale detection with an obvious computational burden [16, 42], or on a separate network to estimate the scale [40, 106]. Both approaches will require attention to avoid bias and the propagation thereof through the network. Our new method treats scale and scale equivariance as a desirable fundamental property, which makes the algorithm conceptually easier. Hence, scale equivariance should be easy to merge into an existing network for tracking. Then, scale equivariance will enhance the performance of the tracker without further modification of the network or extensive data augmentation during the learning phase.

We make the following contributions:

- We propose the theory for scale-equivariant Siamese trackers and provide a simple recipe of how to make a wide range of existing trackers scale-equivariant.
- We propose building blocks necessary for efficient implementation of scale equivariance into modern Siamese trackers and implement a scale-equivariant extension of the recent SiamFC+ [199] tracker.
- We demonstrate the advantage of scale-equivariant Siamese trackers over their conventional counterparts on popular benchmarks for sequences with and without apparent scale changes.

5.2 RELATED WORK

SIAMESE TRACKING The challenge of learning to track arbitrary objects can be addressed by deep similarity learning [16]. The common approach is to employ Siamese networks to compute the embeddings of the original patches. The embeddings are then fused to obtain a location estimate. Such formulation is general, allowing for a favourable flexibility in the design of the tracker. In [16] Bertinetto *et al.* employ off-line trained CNNs as feature extractors. The authors compare dot-product similarities between the feature map of the template with the maps coming from the current frame and measure similarities on multiple scales. Held *et al.* [69] suggest a detection-based Siamese tracker, where the similarity function is modeled as a fully-connected network. Extensive data augmentation is applied to learn a similarity function, which generalizes for various object transformations. Li *et al.* [106] consider tracking as a one-shot detection problem to design Siamese region-proposal-networks [141] by fusing the features from a fully-convolutional backbone. The recent ATOM [40] and DIMP [19] trackers employ a multi-stage tracking framework, where an object is coarsely localized by the online *classification* branch, and subsequently refined in its position by the *estimation* branch. From a Siamese perspective, in both [19, 40] the object embeddings are first fused to produce an initial location and subsequently processed by the IoU-Net [85] to enhance the precision of the bounding box.

The aforementioned references have laid the foundation for most of the state-of-the-art trackers. These methods share an implicit or explicit attention to translation equivariance for feature extraction. The decisive role of translation equivariance is noted in [16, 105, 199]. Bertinetto *et al.* [16] utilize fully-convolutional networks where the output directly commutes with a shift in the input image as a function of the total stride. Li *et al.* [105] suggest a training strategy to eliminate the spatial bias introduced in non-fully-convolutional backbones. Along the same line, Zhang and Peng [199] demonstrated that deep state-of-the-art models developed for classification are not directly applicable for localization. And hence these models are not directly applicable to tracking as they induce positional bias, which breaks strict translation equivariance. We argue that transformations, other then translation, such as rotation may be equally important for certain classes of videos like sports and following objects in the sea or in the sky. And we argue that scale transformation is common in the majority of sequences due to the changing distances between objects and the camera. In this paper, we take on the latter class of transformations for tracking.

EQUIVARIANT CNNS Various works on transformation-equivariant convolutional networks have been published recently. They extend the built-in property of translation-equivariance of conventional CNNs to a broader set of transformations. Mostly considered was roto-translation, as demonstrated on image classification [34, 35, 75, 145, 146, 176], image segmentation [177] and edge detection [180].

One of the first works on scale-translation-equivariant convolutional networks was by Marcos *et al.* [115]. In order to process images on multiple scales, the authors resize and convolve the input of each layer multiple times, forming a stack of features which corresponds to variety of scales. The output of such a convolutional layer is a vector whose length encodes the maximum response in each position among different scales.

The direction of the vector is derived from the scale, which gave the maximum. The method has almost no restrictions in the choice of admissible scales. As this approach relies on rescaling the image, the obtained models are significantly slower compared to conventional CNNs. Thus, this approach is not suitable for being applied effectively in visual object tracking.

Worrall & Welling [179] propose Deep Scale-Spaces, an equivariant model which generalizes the concept of scale-space to deep networks. The approach uses filter dilation to analyze the images on different scales. It is almost as fast as a conventional CNN with the same width and depth. As the method is restricted to integer scale factors it is unsuited to applications in tracking where the scene dictates arbitrary scale factors.

Almost simultaneously, three papers [10, 156, 203] were proposed to implement scale-translation-equivariant networks with arbitrary scales. What they have in common is that they use a pre-calculated and fixed basis defined on multiple scales. All filters are then calculated as a linear combination of the basis and trainable weights. As a result, no rescaling is used. We prefer to use [156], as Sosnovik *et al.* propose an approach for building general scale-translation-equivariant networks with an algorithm for the fast implementation of the scale-convolution.

To date, the application of scale-equivariant networks was mostly demonstrated in image classification. Almost no attention was paid to tasks that involve object localization, such as visual object tracking. As we have noted above, it is a fundamentally different case. To the best of our knowledge, we demonstrate the first application of transformation-equivariant CNNs to visual object tracking.

5.3 SCALE-EQUIVARIANT TRACKING

In this work, we consider a wide range of modern trackers which can be described by the following formula:

$$h(z, x) = \phi_X(x) \star \phi_Z(z) \tag{5.1}$$

where z, x are the template and the input frame, and ϕ_X, ϕ_Z are the functions which process them, and $\star$ is the convolution operator which implements a connection between two signals. The resulting value $h(z, x)$ is a heatmap that can be converted into a prediction by relatively simple calculations. Functions ϕ_X, ϕ_Z here can be parametrized as feed-forward neural networks. For our analysis, it is both suitable if the weights of these networks are fixed or updated during training or inference. This pipeline describes the majority of Siamese trackers such as [16, 105, 106] and the trackers based on correlation filters [41, 42].

5.3.1 *Convolution is all you need*

Let us consider some mapping g. It is equivariant under a transformation L if and only if there exists L' such that $g \circ L = L' \circ g$. If L' is the identity mapping, then the function g is invariant under this transformation. A function of multiple variables is equivariant when it is equivariant with respect to each of the variables. In our analysis, we consider only transformations that form a transformation group, in other words, $L \in G$.

Theorem 3 (***Siamese equivariance***)**.** *A function given by Equation 5.1 is equivariant under a transformation L from group G if and only if ϕ_X and ϕ_Z are constructed from G-equivariant convolutional layers and $\star$ is the G-convolution.*

Proof. Let us fix $z = z_0$ and introduce a function $h_X = h(x, z_0) = \phi_X(x) \star \phi_Z(z_0)$. This function is a feed-forward neural network. All its layers but the last one are contained in ϕ_X and the last layer is a convolution with $\phi_Z(z_0)$. According to [96] a feed-forward neural network is equivariant under transformations from G if and only if it is constructed from G-equivariant convolutional layers. Thus, the function h_X is equivariant under transformations from G if and only if

- The function ϕ_X is constructed from G-equivariant convolutional layers
- The convolution $\star$ is the G-convolution

If we then fix $x = x_0$, we can show that a function $h_Z = h(x_0, z) = \phi_X(x_0) \star \phi_Z(z)$ is equivariant under transformations from G if and only if

- The function ϕ_Z is constructed from G-equivariant convolutional layers
- The convolution $\star$ is the G-convolution

The function h is equivariant under G if and only if both the function h_X and the function h_Z are equivariant. □

A simple interpretation of this theorem is that *a tracker is equivariant to transformations from G if and only if it is fully G-convolutional.* The necessity of fully-convolutional trackers is well-known in tracking community and is related to the ability of the tracker to capture the main variations in the video — the translation. In this paper, we seek to extend this ability to scale variations as well. Which, due to Theorem 3 boils down to using scale-convolution and building fully scale-translation convolutional trackers.

5.3.2 *Scale Modules*

Given a function $f : \mathbb{R} \rightarrow \mathbb{R}$, a scale transformation is defined as follows:

$$L_s[f](t) = f(s^{-1}t), \quad \forall s \geq 0 \tag{5.2}$$

where cases with $s > 1$ are referred to as upscale and with $s < 1$ as downscale. Standard convolutional layers and convolutional networks are translation equivariant but not scale-equivariant [156].

PARAMETRIC SCALE-CONVOLUTION In order to build scale-equivariant convolutional networks, we follow the method proposed by Sosnovik *et al.* [156]. We begin by choosing a complete basis of functions defined on multiple scales. Choosing the center of the function to be the point $(0, 0)$ in coordinates (u, v), we use functions of the following form:

$$\psi_{\sigma nm}(u, v) = A\frac{1}{\sigma^2}H_n\Big(\frac{u}{\sigma}\Big)H_m\Big(\frac{v}{\sigma}\Big)e^{-\frac{u^2+v^2}{2\sigma^2}} \tag{5.3}$$

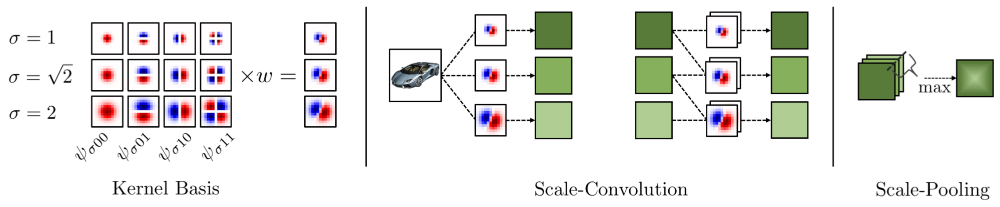


Figure 17: Left: convolutional kernels use a fixed kernel basis on multiple scales, each with a set of trainable weights. Middle: a representation of scale-convolution using Equation 5.6 for the first and all subsequent layers. Right: a scheme of scale-pooling, which transforms a 3D-signal into a 2D one without losing scale equivariance. As an example, we use a basis of 4 functions and 3 scales with a step of $\sqrt{2}$. Only one channel of each convolutional layer is demonstrated for simplicity.

Here H_n is a Hermite polynomial of the n-th order, and A is a constant used for normalization. In order to build a basis of N functions, we iterate over increasing pairs of n and m. As the basis is complete, the number of functions N is equal to the number of pixels in the original filter. We build such a basis for a chosen set of equidistant scales σ and fix it:

$$\Psi_\sigma = \left\{\psi_{\sigma 00},\ \psi_{\sigma 01},\ \psi_{\sigma 10},\ \psi_{\sigma 11}\dots\right\} \tag{5.4}$$

Kernels of convolutional layers are parametrized by trainable weights w in the following way:

$$\kappa_\sigma = \sum_i \Psi_{\sigma i} w_i \tag{5.5}$$

As a result, each kernel is defined on multiple scales and no image interpolation is used. Given a function of scale and translation $f(s,t)$ and a kernel $\kappa_\sigma(s,t)$, a scale convolution is defined as:

$$[f \star_H \kappa_\sigma](s,t) = \sum_{s'} [f(s',\cdot) \star \kappa_{s\cdot\sigma}(s^{-1}s',\cdot)](t) \tag{5.6}$$

The result of this operation is a stack of features each of which corresponds to a different scale. We end up with a 3-dimensional representation of the signal — 2-dimensional translation + scale. We follow [156] and denote scale-convolution as $\star_H$ in order to distinguish it with the standard one. Figure 17 demonstrates how a kernel basis is formed and how scale-convolutional layers work.

FAST 1×1 SCALE-CONVOLUTION An essential building block of many backbone deep networks such as ResNets [67] and Wide ResNets [191] is a 1×1 convolutional layer. We follow the interpretation of these layers proposed in [108] — it is a linear combination of channels. Thus, it has no spatial resolution. In order to build a scale-equivariant counterpart of 1×1 convolution, we do not utilize a kernel basis. As we pointed out before, the signal is stored as a 3 dimensional tensor for each channel. Therefore, for a kernel defined on N_S scales, the convolution of the signal with this kernel is just a 3-dimensional convolution with a kernel of size 1×1 in spatial dimension, and with N_S values in depth. This approach for 1×1 scale-convolution is faster than the special case of the algorithm proposed in [156].

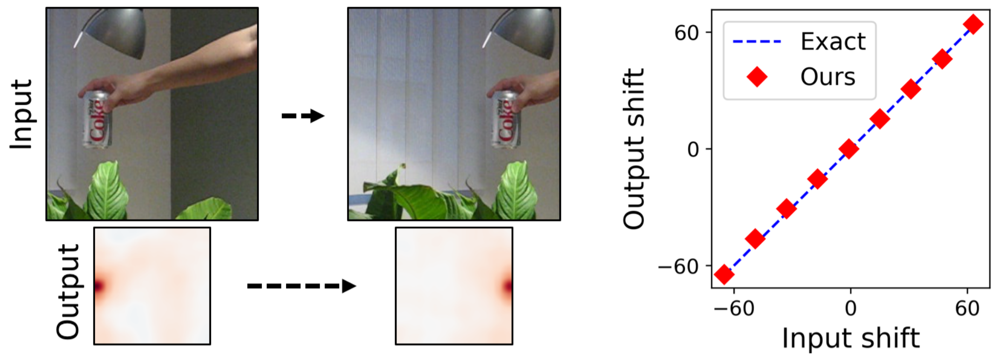


Figure 18: Left: two samples from the simulated sequence. The input image is a translated and cropped version of the source image. The output is the heatmap produced by the proposed model. The red color represents the place where the object is detected. Right: correspondence between the input and the output shifts.

PADDING Although zero padding is a standard approach in image classification for saving the spatial resolution of the image, it worsens the localization properties of convolutional trackers [105, 199]. Nevertheless, a simple replacement of standard convolutional layers with scale-equivariant ones in very deep models is not possible without padding. Scale-equivariant convolutional layers have kernels of a bigger spatial extent because they are defined on multiple scales. For these reasons, we use circular padding during training and zero padding during testing in our models. We experimentally validate (Figure 18) that the proposed padding scheme has minimal effect on translation equivariance.

SCALE-POOLING In order to capture correlations between different scales and to transform a 3-dimensional signal into a 2-dimensional one, we utilize global max pooling along the scale axis. This operation does not eliminate the scale-equivariant properties of the network. We found that it is useful to additionally incorporate this module in the places where conventional CNNs have spatial max pooling or strides. The mechanism of scale-pooling is illustrated in Figure 17.

NON-PARAMETRIC SCALE-CONVOLUTION The convolutional operation which results in the heatmap of a tracker is non-parametric. Both the input and the kernel come from neural networks. Thus, the approach described in Equation 5.6 is not suitable for this case. Given two functions f_1, f_2 of scale and translation the non-parametric scale convolution is defined as follows:

$$[f_1 \star_H f_2](s,t) = L_{s^{-1}}[L_s[f_1] \star f_2](t) \tag{5.7}$$

Here L_s is rescaling implemented as bicubic interpolation. Although it is a relatively slow operation, it is used only once in the tracker and does not heavily affect the inference time.

Proof. A function given by Equation 5.7 is equivariant under scale transformations of f_1, indeed

$$\begin{aligned}[L_{\hat{s}}[f_1] \star_H f_2](s,t) &= L_{s^{-1}}[L_{s\hat{s}}[f_1] \star f_2](t) \\ &= L_{\hat{s}} L_{(s\hat{s})^{-1}}[L_{s\hat{s}}[f_1] \star f_2](t) \\ &= L_{\hat{s}}[f_1 \star_H f_2](s\hat{s},t)\end{aligned} \tag{5.8}$$

For a pair of scale and translation $s, \hat{t}$ we have the following property of the joint transformation $L_s T_{\hat{t}} = T_{\hat{t}s} L_s$ from [156], where $T_{\hat{t}}$ is the translation operator defined as $T_{\hat{t}}[f](t) = f(t-\hat{t})$. Now we can show the following:

$$\begin{aligned}[T_{\hat{t}}[f_1] \star_H f_2](s,t) &= L_{s^{-1}}[L_s[T_{\hat{t}}[f_1]] \star f_2](t) \\ &= L_{s^{-1}}[T_{\hat{t}s} L_s[f_1] \star f_2](t) \\ &= L_{s^{-1}} T_{\hat{t}s}[L_s[f_1] \star f_2](t) \\ &= T_{\hat{t}} L_{s^{-1}}[L_s[f_1] \star f_2](t) \\ &= T_{\hat{t}}[f_1 \star_H f_2](t)\end{aligned} \tag{5.9}$$

Therefore, a function given by Equation 5.7 is also equivariant under translations of f_1. The equivariance of the function with respect to a joint transformation follows from the equivariance to each of the transformations separately [156].

We proved the equivariance with respect to f_1. The proof with respect to f_2 is analogous. □

5.3.3 *Extending a Tracker to Scale Equivariance*

We present a recipe to extend a tracker to scale equivariance.

1. The first step is to estimate to what degree objects change in size in this domain, and then to select a set of scales $\sigma_1, \sigma_2, \ldots \sigma_N$. This is a domain-specific hyperparameter. For example, a domain with significant scale variations requires a broader span of scales, while for more smooth sequences, the set may consist of just 3 scales around 1.

2. For a tracker which can be described by Equation 5.1, derive ϕ_X and ϕ_Z.

3. For the networks represented by ϕ_X and ϕ_Z, all convolutional layers need to be replaced with scale-convolutional layers. The basis for these layers is based on the chosen scales $\sigma_1, \sigma_2, \ldots \sigma_N$.

4. (Optional) Scale-pooling can be included to additionally capture inter-scale correlations between all scales.

5. The connection operation $\star$ needs to be replaced with a non-parametric scale-convolution.

6. (Optional) If the tracker only searches over spatial locations, scale-pooling needs to be included at the very end.

The obtained tracker produces a heatmap $h(z, x)$ defined on scale and translation. Therefore, each position is assigned a vector of features that has both the measure of similarity and the scale relation between the candidate and the template. If additional scale-pooling is included, then all scale information is just aggregated in the similarity score.

Note that the overall structure of the tracker, as well as the training and inference procedures are not changed. Thus, the recipe allows for a simple extension of a tracker with little cost of modification.

5.4 SCALE-EQUIVARIANT SIAMFC

While the proposed algorithm is applicable to a wide range of trackers, in this work, we focus on Siamese trackers. As a baseline we choose SiamFC [16]. This model serves as a starting point for modifications for the many modern high-performance Siamese trackers.

Architecture

Given the recipe, here we discuss the actual implementation of the scale-equivariant SiamFC tracker (SE-SiamFC).

In the first step of the recipe, we assess the range of scales in the domain (dataset). In sequences presented in most of the tracking benchmarks, like OTB or VOT, objects change their size relatively slowly from one frame to the other. The maximum scale change usually does not exceed a factor of 1.5 – 2. Therefore, we use 3 scales with a step of $\sqrt{2}$ as the basis for the scale-convolutions. The next step in the recipe is to represent the tracker as it is done in Equation 5.1. SiamFC localizes the object as the coordinate *argmax* of the heatmap $h(z, x) = \phi_Z(z) \star \phi_X(x)$, where $\phi_Z = \phi_X$ are convolutional Siamese backbones. Next, in step number 3, we modify the backbones by replacing standard convolutions by scale-equivariant convolutions. We follow step 4 and utilize scale-pooling in the backbones in order to capture additional scale correlations between features of various scales. According to step 5, the connecting correlation is replaced with non-parametric scale-convolution. SiamFC computes its similarity function as a 2-dimensional map, therefore, we follow step 6 and add extra scale-pooling in order to transform a 3-dimensional heatmap into a 2-dimensional one. Now, we can use exactly the same inference algorithm as in the original paper [16]. We use the standard approach of scale estimation, based on the greedy selection of the best similarity for 3 different scales.

Weight Initialization

An important ingredient of a successful model training is the initialization of its weights. A common approach is to use weights from an Imagenet [45] pre-trained model [105,106, 199]. In our case, however, this requires additional steps, as there are no available scale-equivariant models pre-trained on the Imagenet. We present a method for initializing a scale-equivariant model with weights from a pre-trained conventional CNN. The key idea

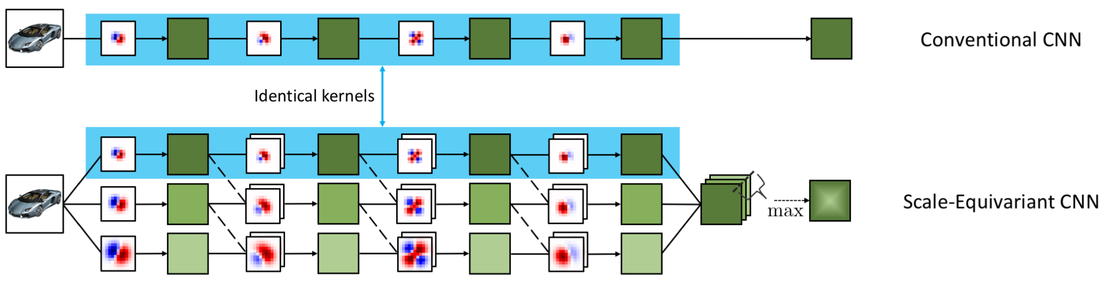


Figure 19: The visualization of the weight initialization scheme from a pretrained model. Dashed connections are initialized with 0.

is that a scale-equivariant network built according to Section 5.3.3 contains a sub-network that is identical to the one of the non-scale-equivariant counterpart. As the kernels of scale-equivariant models are parameterized with a fixed basis and trainable weights, our task is to initialize these weights.

We begin by initializing the inter-scale correlations by setting to 0 all weights responsible for these connections. At this moment, up to scale-pooling, the scale-equivariant model consists of several networks parallel to, yet disconnected from one another, where the only difference is the size of their filters. For the convolutional layers with a non-unitary spatial extent, we initialize the weights such that the kernels of the smallest scale match those of the source model. Given a source kernel $\kappa'(u,v)$ and a basis $\Psi_{\sigma i}(u,v)$ with $\sigma = 1$, weights w_i are chosen to satisfy the linear system derived from Equation 5.5:

$$\kappa_1(u,v) = \sum_i \Psi_{1i}(u,v) w_i = \kappa'(u,v), \quad \forall u,v \tag{5.10}$$

As the basis is complete by construction, its matrix form is invertible. The system has a unique solution with respect to w_i:

$$w_i = \sum_{u,v} \Psi^{-1}_{1i}(u,v)\kappa'(u,v) \tag{5.11}$$

All 1×1 scale-convolutional layers are identical to standard 1×1 convolutions after zeroing out inter-scale correlations. We copy these weights from the source model. We provide an illustration of the proposed initialization method in Figure 19.

5.5 EXPERIMENTS AND RESULTS

5.5.1 *Translation-Scaling MNIST*

To test the ability of a tracker to cope with translation and scaling, we conduct an experiment on a simulated dataset with controlled factors of variation. We construct the datasets of translating (T-MNIST) and translating-scaling (S-MNIST) digits.

In particular, to form a sequence, we randomly sample up to 8 MNIST digits with backgrounds from the GOT10k dataset [77]. Then, on each of the digits in the sequence independently, a smoothed Brownian motion model induces a random translation. Simul-

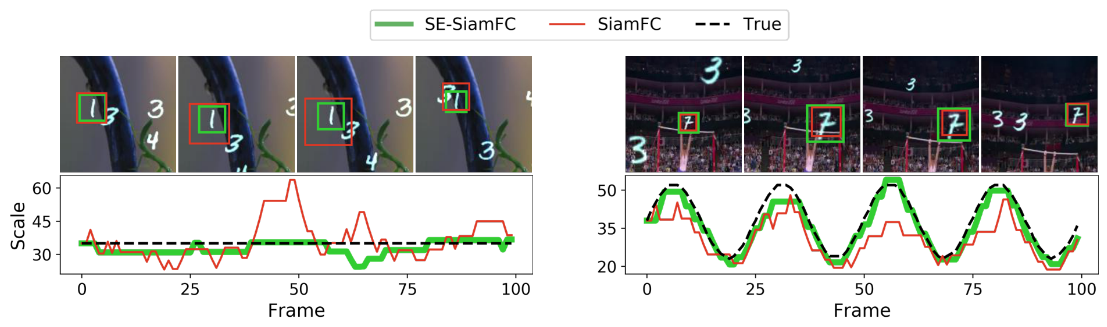


Figure 20: Top: examples of simulated T-MNIST and S-MNIST sequences. Bottom: scale estimation for equivariant and non-equivariant models. In the S-MNIST example, SE-SiamFC can estimate the scale more accurately. In the T-MNIST example, our model better preserves the scale of the target unchanged, while the non-scale-equivariant model is prone to oscillations in its scale estimate.

Tracker	T/T	T/S	S/T	S/S	# Params
SiamFC	0.64	0.62	0.64	0.63	999 K
SE-SiamFC	**0.76**	**0.69**	**0.77**	**0.70**	999 K

Table 6: AUC for models trained on T-MNIST and S-MNIST. T/S indicates that the model was trained on T-MNIST and tested on S-MNIST datasets. Bold numbers represent the best result for each of the training/testing scenarios.

taneously, for S-MNIST, a smooth scale change in the range $[0.67, 1.5]$ is induced by the sine rule:

$$s_i(t) = \frac{h-l}{2}\Big[\sin\big(\frac{t}{4}+\beta_i\big)+1\big)\Big]+l \tag{5.12}$$

where $s_i(t)$ is the scale factor of the i-th digit in the t-th frame, h, l are upper and lower bounds for scaling, and $\beta_i \in [0, 100]$ is a phase, sampled randomly for each of the digits. In total, we simulate 1000 sequences for training and 100 for validation. Each sequence has a length of 100 frames.

We compare two configurations of the tracker: (i) SiamFC with a shallow backbone and (ii) its scale-equivariant version SE-SiamFC. We conduct the experiments according to 2×2 scenarios: the models are trained on either S-MNIST or T-MNIST and are subsequently tested on either of them. Both models have 4 convolutional layers and cross-correlation or non-parametric scale-convolution to merge features from two siamese branches. We use 3×3 filter dimension and the stride of 2 in each of the convolutional layers, besides the last one, where the stride of 1 is used. Channel dimension goes as $3 \to 96 \to 128 \to 256 \to 256$. We train the models for 50 epochs using SGD with a mini-batch of 8 images and exponentially decay the learning rate from 10^{-2} to 10^{-5}. We set the momentum to 0.9 and the weight decay to 0.5^{-4}. A binary cross-entropy loss as in [16] is used.

The results are reported in Table 6. As can be seen, the equivariant version outperforms its non-equivariant counterpart in all scenarios. The experiment on S-MNIST, varying the scale of an artificial object, shows that the scale-equivariant model has a superior ability to precisely follow the change in scale compared to the conventional one. The experiment on T-MNIST shows that (proper) measurement of scale is important even in the case when the sequence does not show a change in scale, where the observed scale in SE-SiamFC fluctuates much less than it does in the baseline (see Figure 20).

Tracker	Year	OTB-2013		OTB-2015		VOT2016			VOT2017		
		AUC	Prec.	AUC	Prec.	EAO	A	R	EAO	A	R
SINT [161]	2016	0.64	0.85	-	-	-	-	-	-	-	-
SiamFC [16]	2016	0.61	0.81	0.58	0.77	0.24	0.53	0.46	0.19	0.50	0.59
DSiam [60]	2017	0.64	0.81	-	-	-	-	-	-	-	-
StructSiam [198]	2018	0.64	0.88	0.62	0.85	0.26	-	-	-	-	-
TriSiam [46]	2018	0.62	0.82	0.59	0.78	-	-	-	0.20	-	-
SiamRPN [106]	2018	-	-	0.64	0.85	0.34	0.56	0.26	0.24	0.49	0.46
SiamFC+ [199]	2019	0.67	0.88	0.64	0.85	0.30	0.54	0.38	0.23	0.50	0.49
SE-SiamFC	Ours	**0.68**	0.90	**0.66**	0.88	**0.36**	0.59	0.24	**0.27**	0.54	0.38

Table 7: Performance comparisons on OTB-2013, OTB-2015, VOT2016, and VOT2017 benchmarks. Bold numbers represent the best result for each of the benchmarks.

5.5.2 *Benchmarking*

We compare the scale-equivariant tracker against a non-equivariant baseline on popular tracking benchmarks. We test SE-SiamFC with a backbone from [199] against other popular Siamese trackers on OTB-2013, OTB-2015, VOT2016, and VOT2017. The benchmarks are chosen to allow direct comparison with the baseline [199].

IMPLEMENTATION DETAILS The parameters of our model are initialized with weights pre-trained on Imagenet by a method described in Section 5.4. We use the same training procedure as in the baseline. Table 8 provides the details on the architecture used.

The pairs for training are collected from the GOT10k [77] dataset. We adopt the same prepossessing and augmentation techniques as in [199]. The inference procedure remains unchanged compared to the baseline.

OTB We test on the OTB-2013 [182] and OTB-2015 [183] benchmarks. Each of the sequences in the OTB datasets carries labels from 11 categories of difficulty in tracking the sequence. Examples of these labels include: occlusion, scale variation, in-pane rotation, *etc.* We employ a standard one-pass evaluation (OPE) protocol to compare our method with other trackers by the area under the success curve (AUC) and precision.

The results are reported in Table 7. Our scale-equivariant tracker outperforms its non-equivariant counterpart by more than 3% on OTB-2015 in both AUC and precision, and by 1.4% on OTB-2013. When summarized at each label of difficulty (see Figure 21), the proposed scale-equivariant tracker is seen to improve all sequence types, not only those labeled with “scale variation”.

Stage	SiamFC+	SE-SiamFC
Conv1	$\left[7\times7, 64, s=2\right]$	$\left[7\times7, 64, s=2\right]$
Conv2	max pool $\left[2\times2, s=2\right]$	
	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64, i=2\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
Conv3	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 128, sp\\3\times3, 128\\1\times1, 512\end{bmatrix}\times3$
Connect.	Cross-correlation	Non-parametric scale-convolution
# Params	1.44 M	1.45 M

Table 8: Architectures used in OTB/VOT experiments. All convolutions in SE-SiamFC are scale-convolutions. `s` *refers to stride,* `sp` *denotes scale pooling,* `i` *— is the size of the kernel in a scale dimension.*

We attribute this to the fact that the “scale variation” tag in the OTB benchmark only indicates the sequences with a relatively big change in scale factors, while up to a certain degree, scaling is present in almost any video sequence. Moreover, scaling may be present implicitly, in the form of the same patterns being observed on multiple scales. An ability of our model to exploit this leads to better utilization of trainable parameters and a more discriminative Siamese similarity as a result.

VOT We next evaluate our tracker on VOT2016 and VOT2017 datasets [98]. The performance is evaluated in terms of average bounding box overlap ratio (A), and the robustness (R). These two metrics are combined into the Expected Average Overlap (EAO), which is used to rank the overall performance.

The results are reported in Table 7. On VOT2016 our scale-equivariant model shows an improvement from 0.30 to 0.36 in terms of EAO, which is a 20% gain compared to the non-equivariant baseline. On VOT2017, the increase in EAO is 17%.

We qualitatively investigated the sequences with the largest performance gain and observed that the most challenging factor for our baseline is the rapid scaling of the object. Even when the target is not completely lost, the imprecise bounding box heavily influences the overlap with the ground truth and the final EAO. Our scale-equivariant model better adapts to the fast scaling and delivers tighter bounding boxes.

5.5.3 *Ablation Study*

We conduct an ablation study on the OTB-2013 benchmark to investigate the impact of scale step, weight initialization, and fast 1×1 scale-convolution. We also test the baseline

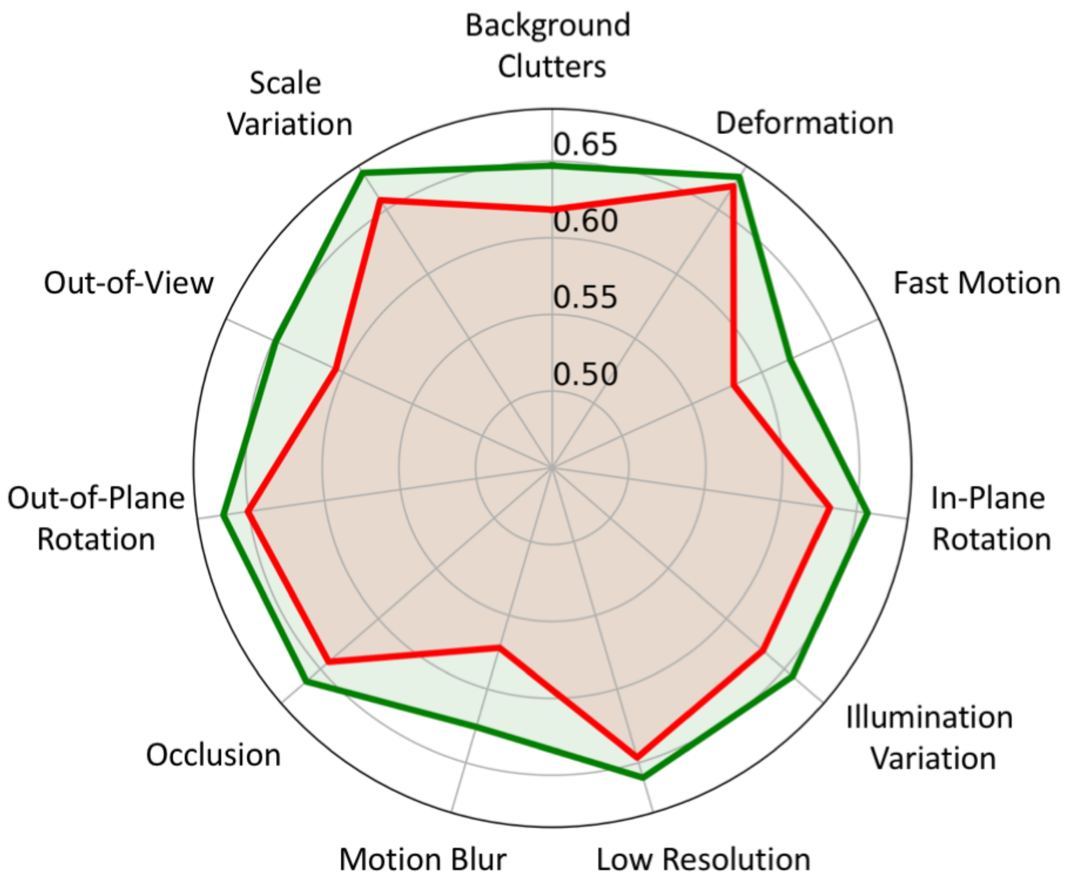


Figure 21: Comparison of AUC on OTB-2013 with different factors of variations. The red polygon corresponds to the baseline SiamFC+ and the green polygon — to SE-SiamFC.

Model	Pretrained weigths	AUC
SiamFC+ (aug. 5%)	✓	0.668
SiamFC+ (aug. 20%)	✓	0.668
SiamFC+ (aug. 50%)	✓	0.664
SE-SiamFC $\sigma = 1.2$	✓	0.677
SE-SiamFC $\sigma = 1.3$	✓	0.680
SE-SiamFC $\sigma = 1.4$	✓	**0.681**
SE-SiamFC $\sigma = 1.5$	✓	0.678
SE-SiamFC $\sigma = 1.4$	✗	0.553

Table 9: Ablation study on the OTB-2013 benchmark. The parameter σ stands for the step between scales in scale-equivariant models. Bold numbers represent the best result.

SiamFC+ model with various levels of scale data augmentation during the training. We follow the same training and testing procedure as in Section 5.5.2 for all experiments. In the weight initialization experiment, however, we do not use gradual weights unfreezing, but train the whole model end-to-end from the first epoch.

SCALE STEP We investigate the impact of scale step σ, which defines a set of scales our model operates on. We train and test SE-SiamFC with various scale steps. Results are shown in Table 9. It can be seen that the resulting method outperforms the baseline on a range of scale steps. We empirically found that $\sigma = 1.4$ achieves the best performance.

SCALE DATA AUGMENTATION Data augmentation is a common way to improve model generalization over different variations. Since our method is focused on scale, we compare SE-SiamFC against a baseline trained with different levels of scale data augmentation. Our results indicate (Table 9) that scale augmentation does not improve the performance of the conventional non-equivariant tracker.

WEIGHT INITIALIZATION We train and test SE-SiamFC model, where weights initialized randomly [56, 177]. As can be seen from Table 9, random initialization results in a *19%* performance drop compared to the proposed initialization technique.

FAST 1×1 SCALE-CONVOLUTION We compare the speed of 1×1 scale-convolution from [156] and the proposed fast implementation. Implementation from [156] requires 450 / 1650 μs, while our implementation requires 67 / 750 μs for forward / backward pass respectively, which is more than 6 times faster. In our experiments, the usage of fast 1×1 scale-convolution results in 30 – 40% speedup of a tracker.

5.6 DISCUSSION

In this work, we argue about the usefulness of additional scale equivariance in visual object tracking for the purpose of enhancing Siamese similarity estimation. We present a general theory that applies to a wide range of modern Siamese trackers, as well as all the components to turn an existing tracker into a scale-equivariant version. Moreover, we prove that the presented components are both necessary and sufficient to achieve built-in scale-translation equivariance. We sum up the theory by developing a simple recipe for extending existing trackers to scale equivariance. We apply it to develop SE-SiamFC — a scale-equivariant modification of the popular SiamFC tracker.

We experimentally demonstrate that our scale-equivariant tracker outperforms its conventional counterpart on OTB and VOT benchmarks and on the synthetically generated T-MNIST and S-MNIST datasets, where T-MNIST is designed to keep the object at a constant scale, and S-MNIST varies the scale in a known manner.

The experiments on T-MNIST and S-MNIST show the importance of proper scale measurement for all sequences, regardless of whether they have scale change or not. For the standard OTB and VOT benchmarks, our tracker proves the power of scale equivariance. It is seen to not only improves the tracking in the case of scaling, but also when other factors of variations are present (see Figure 21). It affects the performance in two ways: it prevents erroneous jumps to similar objects at a different size and it provides a better consistent estimate of the scale.

6

RELATIONAL PRIOR FOR TRACKING GROUPS OF OBJECTS

6.1 INTRODUCTION

For online multi-object tracking, when objects are part of a group, the frequent mutual occlusions make individual tracking harder. Rather than rejecting that information, identifying group membership is interesting by itself, where in principle the group is easier to identify having more uniquely identifying characteristics than an individual object would. In this paper we set out to exploit group relations for multi-object tracking.

When tracking pedestrians online in a crowd, following one specifically is generally harder than following all members of a family, just because their *combination* offers good distinction: a mother and a small child holding hands. Occlusion may hamper the complete view of one of the targets but then the characteristics of related members may be borrowed to render approximate tracking for the occluded one like parents with a child in shopping malls and other forms of crowd control.

Online multi-object tracking has recently made great progress with tracking-by-regression [14, 113, 170, 175, 186, 197, 201]. These methods track each object separately until, at a crossroad of tracks, a mechanism is called upon to determine which object continues on what track. The current methods demonstrate good speed and good accuracy. They do not, however, consider inter-object relations, which may cause tracking to become unreliable especially when the interaction between bodies becomes dense where occlusion becomes a major obstacle, as in Figure 22.

We draw inspiration from offline multi-object tracking, where the whole video is available for the analysis. In [88, 89, 158–160], the trajectories are derived by running a graph optimization on the object detections. While the structure of the graph encodes the

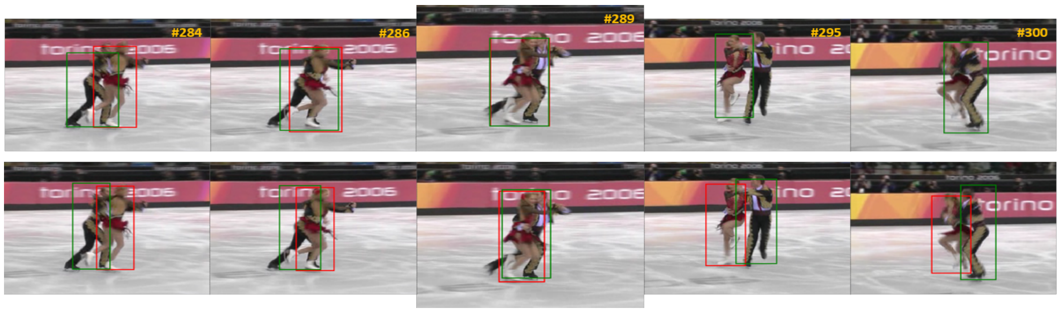


Figure 22: Top: multi-object tracking with Tracktor [14], where independent trajectories are assumed making tracking hard when the targets come close. Bottom: the same tracker [14] equipped with relation-reasoning is more robust to occlusions.

inter-object relations in these offline trackers, their capability of finding relations relies heavily on having all detections in the video at once, combining information *before and after* dense interactions. This offline information blocks the methods unsuited for online multi-object tracking as the detections of the future are not yet available.

In this work, we extend current tracking-by-regression methods with online group relations. Inspired by offline graph-based video analysis, we learn to encode inter-object relations from limited data *a priori*. In our relation encoding module, a message passing algorithm is running over a dynamic object-graph to produce relation embeddings, which encode the group structure for each object. The message passing is performed over space and time to capture the complex dynamics of group behaviour. The relation encoding module is implemented as a plug-in extension for tracking-by-regression methods.

We make the following contributions:

- We develop a method to encode inter-object relations online in dense scenes by running spatial-temporal message passing.
- We demonstrate the virtue of relations to improve the tracking of objects in dense scenes by adding relations on top of current tracking-by-regression methods, even tracking objects with low visibility.
- We demonstrate that the degree of inter-object relations can be estimated online during the tracking.

6.2 RELATED WORK

MULTI-OBJECT TRACKING BY GRAPH ASSOCIATION Many the multi-object trackers first apply an object detector on the whole sequence, then link the detections across frames on the basis of a best match criterion [88, 89, 158–160]. They follow the tracking by a graph association paradigm. The matching is usually posed as an offline graph association problem connecting the detections into trajectories. In [158, 159], the authors solve association as a multicut problem, where trajectories are derived from a dense graph of detections by extracting weighted subgraphs. Along the same lines, in [89], Keuper *et al.* propose a multicut formulation to decompose a dense detection graph into a set of trajectories. To better handle occlusions, Tang *et al.* [160] further extend multicuts with lifted edges. More recently, Braso and Leal-Taixé [21] proposed a neural graph formulation to directly predicts the links between targets given the association graph. Such an approach allows combining feature extraction and association into one end-to-end tracking framework, which dramatically improves association quality. At the core of their method is the time-aware message-passing algorithm, which is used to aggregate the positional information from both past and future frames.

Graph associations are powerful as they reason about groups of detections while taking inter-object relations into account. However, their offline nature limits the real-life application of such methods. Also, it is not straightforward to combine these offline graph association algorithms with online end-to-end tracking-by-regression models. In this work, we take inspiration from these graph association works and develop a new method for relation encoding, which learns to encode the dynamics of multiple objects for online tracking. Our relation encoding module is fully end-to-end compatible with modern trackers.

MULTI-OBJECT TRACKING BY REGRESSION ASSOCIATION Recently, a family of methods called tracking-by-regression has become the state-of-the-art approach in multi-object tracking. The key idea is to assess the association of detections to previously detected objects by utilizing the regression head of the object detector. In the pioneering work of Bergmann *et al.* [14], tracking is based on the second stage of the Faster R-CNN [141] object detector with the previous positions of detected objects as proposals. Later, more sophisticated object detectors were used [113, 175, 197, 201]. In [201] Zhou *et al.* modify CenterNet [202] for multi-object tracking. In [113], authors modify the lightweight RetinaNet [55] for faster inference. In [175, 197], the authors extend the detector with ReID embeddings, which allows for better identity preservation in case of occlusion.

In all these works, objects are tracked independently of one another. When scenes become crowded or filled with similar targets, independent tracking under frequent and heavy occlusion becomes hard or impossible. Whereas the above methods function well generally, they tend to break when individual cues are no longer available (Figure 22). For these hard but frequent circumstances, one has to employ relations in tracking. In this paper, we propose a simple yet effective extension for regression-based multi-object trackers to improve tracking robustness in dense interaction.

REASONING ON RELATIONS From the literature on relations in computer vision, we focus in particular on self-attention [103, 167] and graph neural networks (GNN) [168, 184]. Their ability to use structural information in the input has motivated Hu *et al.* [76] to develop a self-attention relation module to remove duplicates in the task of object detection. In [25] Cai *et al.* use graph convolution to propagate relational information to refine a predicted pose. Narasimhan *et al.* [129] rely on graph-structured representations to encode relations for visual question answering. In [118], Materzynska *et al.* model interactions in a subject-object graph representation for action recognition.

Another line of work specifically focuses on modeling social behavior to reason about the relations and interactions between objects. Pioneering work from Helbing and Molnar [68] presented a pedestrian social dynamics model with attractive and repulsive forces. The authors used simulated data to demonstrate that modeling social forces is useful for predicting the dynamics of interacting pedestrians. In [135] Pellegrini *et al.* proposed a learnable social behavior mechanism based on crowd simulation to predict the pedestrian trajectories, which they showed to be useful in the case of occlusions. In [1] Alahi *et al.* model the social context by pooling the hidden states of long short-term memory (LSTM) units. Such a strategy allows combining local and global context for the task of pedestrian motion forecasting.

These works demonstrate how interactions and relational cues improve the analysis in complex systems. Inspired by the success of relation reasoning for various tasks, we expand relation cues to multi-object online tracking as a plug-in extension for state-of-the-art tracking-by-regression algorithms and with the potential online applicability for all the above purposes.

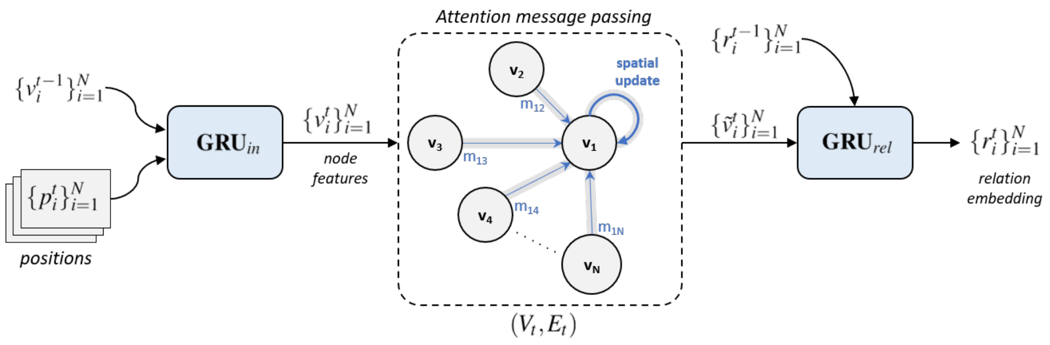


Figure 23: Computing relation embeddings $\{r_i^t\}_{i=1}^N$ for N object from time step $t-1$ to t. Input coordinates are passed through an input Gating Recurrent Unit (GRU) cell to produce node features. Message passing is performed on top of the relational graph to update internal node representations. Finally, node representations are passed through another GRU-cell, which emulates message passing along temporal edges.

METRICS OF MULTI-OBJECT TRACKING Classical multi-object tracking evaluation methods include CLEAR MOT metrics [15], the percentage of mostly tracked (MT) and mostly lost (ML) objects [181] and IDF1 score [144]. These metrics assess various aspects of tracking, while the recently proposed HOTA metric [114] aims to provide the measure of overall performance. None of these metrics, however, capture, the dense interaction of objects, which is one of the circumstances which makes multi-object tracking interesting. To do so, in this work we also consider the decomposition of the HOTA metric over various localization thresholds.

6.3 ENCODING RELATIONS

To encode inter-object relations, the relation encoding module takes a set of tracked instances as input and produces *relation embeddings* by running a message passing algorithm over the spatial-temporal graph. Figure 23 renders the architecture of the module.

6.3.1 *Graph-attention message passing*

Inter-object relations are modulated by running message passing over the relational graph. The procedure consists of 4 steps: compute input node features, compute messages between spatial nodes, aggregate messages and compute spatial-temporal updates of node representations. This procedure is recurrently performed for each time step until the end of the graph is reached.

NODE FEATURES To construct the input feature we use bounding box coordinates of the detection and the positional offset with respect to the previous time step. Let $p_i^t \in \mathbb{R}^4$ be the bounding box of the *i-th* object at time t, the input feature $v_i^t \in \mathbb{R}^F$ for the node is then computed as:

$$\tilde{p}_i^t = \sigma(\mathbf{W}_{in}[p_i^t || p_i^t - p_i^{t-1}] + \mathbf{b}_{in}) \tag{6.1}$$

$$v_i^t = \mathbf{GRU}_{in}(\tilde{p}_i^t, v_i^{t-1}) \tag{6.2}$$

where $\mathbf{W}_{in}, \mathbf{b}_{in}$ are learnable parameters, σ is a non-linearity and $||$ denotes concatenation operator. The initial hidden states of the $\mathbf{GRU}_{in}$ cell are set to zeros.

MESSAGE SENDING A message between two nodes of the graph is designed to encode their pairwise interaction. We define the message as a function of both the sending and the receiving nodes i and j, respectively. To make the message aware of the geometry of the graph, we also include the distance D_{ij}^t between the objects as an additional input for the message function. The message $m_{ij}^t : \mathbb{R}^F \times \mathbb{R}^F \times \mathbb{R} \to \mathbb{R}^F$ is calculated as:

$$m_{ij}^t = \sigma\Big(\mathbf{W}_{m_2}(\sigma(\mathbf{W}_{m_1}[v_i^t || v_j^t || D_{ij}^t] + \mathbf{b}_{m_1})) + \mathbf{b}_{m_2}\Big) \tag{6.3}$$

AGGREGATING MESSAGES When the messages have been computed, they are gathered in an aggregated message. An aggregation function should be permutation equivariant with respect to the neighbors' features. In this work, we follow the graph attention approach [168], which computes attention between features to weigh them according to their importance. The attention mechanism $\alpha_{ij}^t : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}_+$ computes the attention coefficients as:

$$\alpha_{ij}^t = \frac{\exp(\text{LeakyReLU}\big([\mathbf{W}_{a_1} v_i^t]^T [\mathbf{W}_{a_2} v_j^t]\big))}{\sum_{j \in \mathcal{N}_i} \exp(\text{LeakyReLU}\big([\mathbf{W}_{a_1} v_i^t]^T [\mathbf{W}_{a_2} v_j^t]\big))} \tag{6.4}$$

where $\mathcal{N}_i$ denotes the set of the nodes *spatially* adjacent to *i-th* node in the graph. Temporal edges are not considered at this stage. The attention coefficients are then used to compute a linear combination of the corresponding neighbors' representation into an aggregated feature.

SPATIAL-TEMPORAL UPDATE In the final step, we update node representations spatially and temporally. For the spatial update, we concatenate the self-feature of the node with the aggregated message from its neighbors and pass it through the perceptron. The temporal update is performed by passing the features through the GRU-cell. Formally:

$$\textit{(spatial)} \quad \tilde{v}_i^t = \sigma(\mathbf{W}_u[v_i^t || \sum_{j \in \mathcal{N}_i} \alpha_{ij}^t m_{ij}^t] + \mathbf{b}_u) \tag{6.5}$$

$$\textit{(temporal)} \quad r_i^t = \mathbf{GRU}_{rel}(\tilde{v}_i^t, r_i^{t-1}) \tag{6.6}$$

We call the resulting feature $r_i^t \in \mathbb{R}^F$ *relation embedding* of the *i-th* node at time t. Relation embeddings at $t = 0$ are all set to zero vectors.

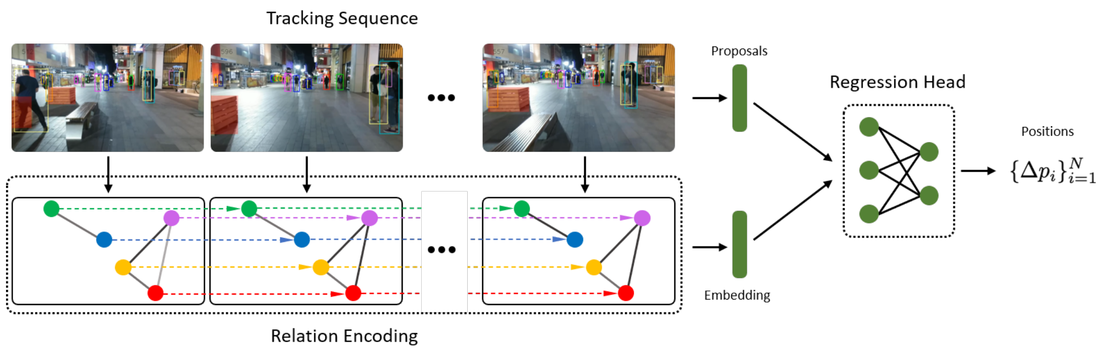


Figure 24: The relational graph for the REM is built on top of the tracked objects. The constructed graph is used to compute relation embeddings, which guide the regression head of a backbone tracker.

As can be seen in (6.5) and (6.6), in our implementation the temporal updates follow the spatial update. Early experiments demonstrated that such an approach is slightly superior to when the temporal update is performed first (data not shown).

6.3.2 *Relation-importance*

To answer the question *to what degree object i relates to object j at time t*, we define a relation-importance function $R_{ij}^t : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}_+$:

$$R_{ij}^t = \mathbb{1}[D_{ij}^t \leq d_{th}]\phi(r_i^t, r_{i\oslash j}^t) \tag{6.7}$$

where ϕ can be any bounded distance function and $r_{i\oslash j}^t$ denotes the relation encoding of *i-th* object computed by excluding the *j-th* node from the set of neighbors. In early experiments, we found $\phi(x, y) = 1 - s^2(x, y)$ to work well as a distance function, where $s^2(x, y)$ stands for the squared cosine similarity between vectors x, y.

Higher values for R_{ij}^t indicate a higher degree of relation between these objects. Note that we can also compute relation-importance by adopting the attention weights from (6.4). However, we observed that such an approach yields less intuitive relations as it does not take the information encoded in the messages into account. In addition, it restricts the relations to be *symmetric*, while the formulation from (6.7) permits *asymmetric* relations.

6.3.3 *Utilizing relations for tracking*

Next, we explore two purposes of using relations in tracking: *(i)* extending tracking-by-regression models to reason about the object's position based both on appearance and relation cues, *(ii)* recovering the position of the objects purely from their relation embeddings, which is useful in the case of occlusions.

RELATION-AWARE TRACKING-BY-REGRESSION To make a tracker aware of relations, we condition the predicted positions of the objects on their relation embeddings. To that end, we concatenate the appearance features extracted from proposal regions with the relation embeddings of the corresponding objects. The positional offset is then predicted by passing the combined feature via the regression head of the object detector. The model can be seen as the REM with the tracker attached to graph nodes. Such a framework applies to a wide range of trackers [14, 113, 175, 197, 201]. It does not require modification of the tracker other than adjusting the regression head.

We illustrate a pipeline of a tracking-by-regression extended with relation reasoning in Figure 24. The graph is built dynamically from the tracked instances. When the object enters or leaves the scene, the corresponding node in the relational graph is updated.

TRACKING-BY-RELATIONS ONLY When the object is heavily occluded, appearance features become unreliable, causing a tracking failure. As relation embeddings produced by the relation module do not depend on the objects' appearance, they can be used to track objects under severe occlusion. To that end, we train an output MLP to directly regress the coordinates of the occluded objects from their relation embeddings. Then, we run the relation-aware tracker on the whole sequence and apply tracking-by-relations only to the occluded objects, when the rest of the objects are tracked both based on appearance and relational cues. This way, the model can *explain* hard occluded cases through easy non-occluded ones.

6.4 EXPERIMENTS

We evaluate our relation-aware tracker on the MOTChallenge benchmarks MOT17 [120] and MOT20 [44].

Implementation details

We use Tracktor[1] as our baseline model as it provides good speed-accuracy balance. We extend Tracktor to relation-aware RelTracktor by plugging in the REM. To do so, we modify the regression head of the tracker to take the concatenated relation-appearance feature instead of just the appearance feature as the input. The rest of the tracker remains unchanged.

We use Xavier initialization [56] for the relation encoding module. We also initialize the modified regression head from the backbone tracker. We then jointly train the modified regression head and the relational module. To do so, we randomly sample $T = 10$ consecutive frames from MOT17/MOT20 datasets, compute relation embeddings and feed them into the regression head together with appearance features to refine bounding boxes at time step $T + 1$. We train for 50 epochs using the Adam optimizer [93] with a learning rate of 0.0001 while setting $d_{th} = 15$ to build relational graphs. We choose $F = 128$ for a dimension of the relation embedding vectors. Generalized intersection over union [142] is used as a loss function. We highlight that only the relational module and the regression head are trained, while the rest of the model is kept as is.

For the tracking-by relations, we extract the positional information from the relation embeddings outputted by REM. To that end, we pass the relation embeddings through an output MLP : $\mathbb{R}^F \to \mathbb{R}^4$, which predicts the coordinates of the object. We set the dimension of the relation embedding $r_i^t \in \mathbb{R}^{128}$. Then, the output MLP is a two-layer

1 `https://github.com/phil-bergmann/tracking_wo_bnw`

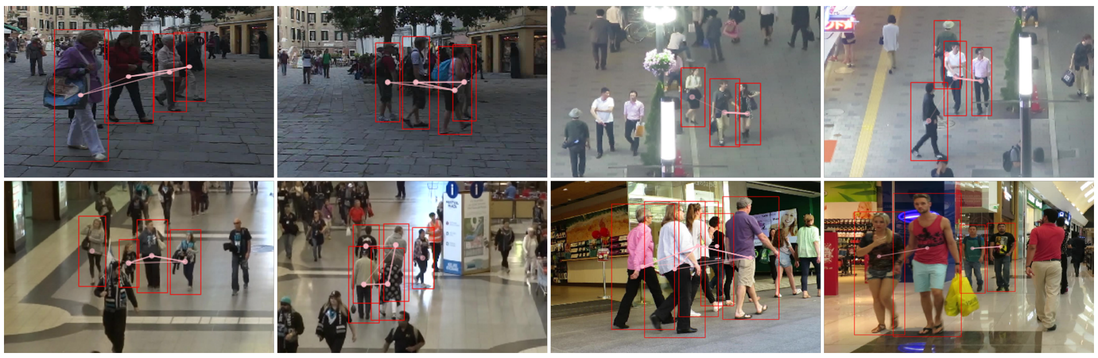


Figure 25: A visualization of relation-importance weights during the tracking on the MOT17/MOT20 benchmarks.

feed-forward network `128` → `64` → `4` with LeakyReLU non-linearities. The output MLP is jointly trained with the main model by minimizing the generalized intersection over union [142] with respect to the ground truth bounding boxes.

Datasets and evaluation metrics

The MOT17 benchmark consists of 7 train and 7 test sequences, which contain pedestrians with annotated full-body bounding boxes. It also specifies the degree of visibility for each annotated instance in the train split. The MOT20 benchmark contains 4 train and 4 test sequences of moving pedestrians in unconstrained environments with bounding boxes, covering the visible part of the objects.

Following [14], we evaluate the multi-object tracking quality in a public and private detection setting. We employ standard the MOT metrics and the HOTA metric [114] as an indicator of the overall performance. We additionally analyze HOTA over different localization thresholds. In a scene that contains a lot of dense interactions, a higher HOTA under low localization thresholds indicates more robust tracking of bodily interacting objects.

6.4.1 *Discovering relations*

We start by testing the ability of the relation encoding module to catch inter-object relations in the scene. As the MOT-benchmark does not provide explicit annotation for the relations, we conduct a qualitative study. In particular, from the relation-aware tracker we compute relation-importance weights (Equation 6.7) for the objects in each time-step. We then visualize the top relations in Figure 25 by the same color.

As can be seen, objects moving in a group tend to have stronger relations. We also observe that relations often are formed between coherently moving objects, even if they are not a part of what we would assess as a group (on our social experience). Such relations are inevitable as we do not employ social rules into the model, but still these assessed relations are useful as the trajectories of such ad-hoc groups can still be explained together.

	Method	HOTA ↑	IDF1 ↑	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDs ↓
MOT17	**RelTracktor** *(Ours)*	**45.8**	**56.5**	**57.2**	**79.0**	21.9	**34.3**	2170
	Tracktor [14]	44.8	55.1	56.3	78.8	21.1	35.3	1987
	deepMOT [186]	42.4	53.8	53.7	77.2	19.4	36.6	1947
	eHAF [148]	-	54.7	51.8	-	**23.4**	37.9	1834
	FWT [71]	-	47.6	51.3	-	21.4	35.2	2648
	jCC [88]	-	54.5	51.2	-	20.9	37.0	**1802**
MOT20	**RelTracktor** *(Ours)*	**43.4**	**53.0**	**54.1**	79.2	**36.7**	**22.6**	2196
	Tracktor [14]	42.1	52.7	52.6	**79.9**	29.4	26.7	**1648**
	SORT20 [18]	36.1	45.1	42.7	78.5	16.7	26.2	4470

Table 10: Performance comparison on MOT17 and MOT20 with public evaluation protocol. The relation-aware RelTracktor model outperforms the baseline model with no relations. RelTracktor improves the conventional tracker in various aspects as indicated by a range of metrics, while the HOTA improvement indicates the rise in the overall tracking quality.

6.4.2 *Relation-aware tracking-by-regression*

We compare the relation-aware RelTracktor versus the baseline method from [14] and other trackers. We run the tracker on the test subset of MOT benchmarks and submit results to the evaluation server. Results are presented in Table 10.

On the MOT17-benchmark, the relation-aware tracker shows an improvement in all metrics compared to the Tracktor baseline. In particular, a higher IDF1 score indicates that our model robustly preserves the identities of the objects throughout the sequence, while also providing more accurate localization as indicated by the MOTP score. Specifically, we observed that when the baseline model fails for close objects in a group, the plain tracker drifts to follow the non-occluded member of the group and loses the target. The relation aware extension of the tracker, on the other hand, is more robust in such scenarios.

On the MOT20-benchmark, the relation-aware tracker demonstrates 1.3% increase in the overall HOTA score. Although the baseline tracker provides slightly higher localization precision as indicated by MOTP score, its relation-aware extension is much more robust and is able to track targets longer as indicated by the higher percentage of mostly tracked objects (MT).

We also evaluate the relation-aware model in a private detection setting. Results in Table 11 indicate that relations improve the overall performance. The higher percentage of mostly tracked objects indicates improved robustness of the relation-aware tracker.

In both public and private scenarios, our relation-aware model has a higher number of ID switches compared to the baseline. We attribute it to the fact that our RelTracktor has a higher recall compared to the conventional model. On MOT17/MOT20 benchmark, the baseline tracker has a recall of 58.3 / 54.3 compared to 59.5 /60.0 of RelTracktor in the public detection setting. And 63.0 /58.1 versus 65.6 /65.8 for the conventional and relation-aware model in private settings respectively.

	Method	HOTA ↑	IDF1 ↑	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDs ↓
MOT17	**RelTracktor** *(Ours)*	**46.9**	**57.5**	**60.0**	**78.3**	**28.8**	**27.1**	3586
	Tracktor [14]	46.4	56.9	58.9	78.0	25.6	28.4	**2859**
MOT20	**RelTracktor** *(Ours)*	**43.0**	**51.3**	**58.1**	**79.3**	**44.0**	**15.9**	3619
	Tracktor [14]	41.2	49.9	54.3	**79.3**	34.1	21.6	**2841**

Table 11: Performance comparison on MOT17 and MOT20 with private evaluation protocol. Relation-aware model outperforms its conventional counterpart as indicated by the HOTA metric.

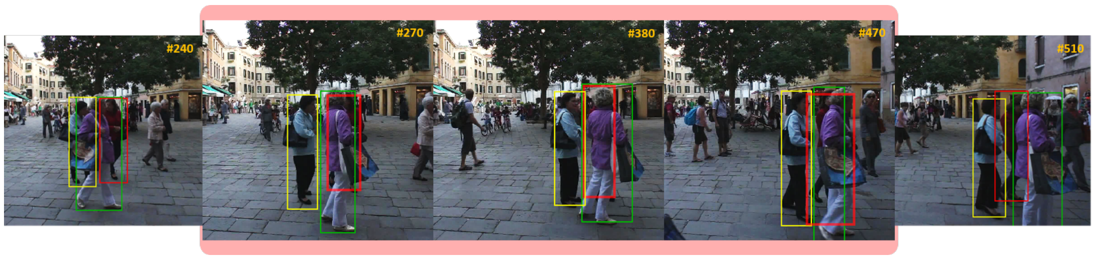


Figure 26: Tracking occluded object (red bounding box) by its relations with neighbors. The position of the object in the shaded region is recovered from its relation embedding.

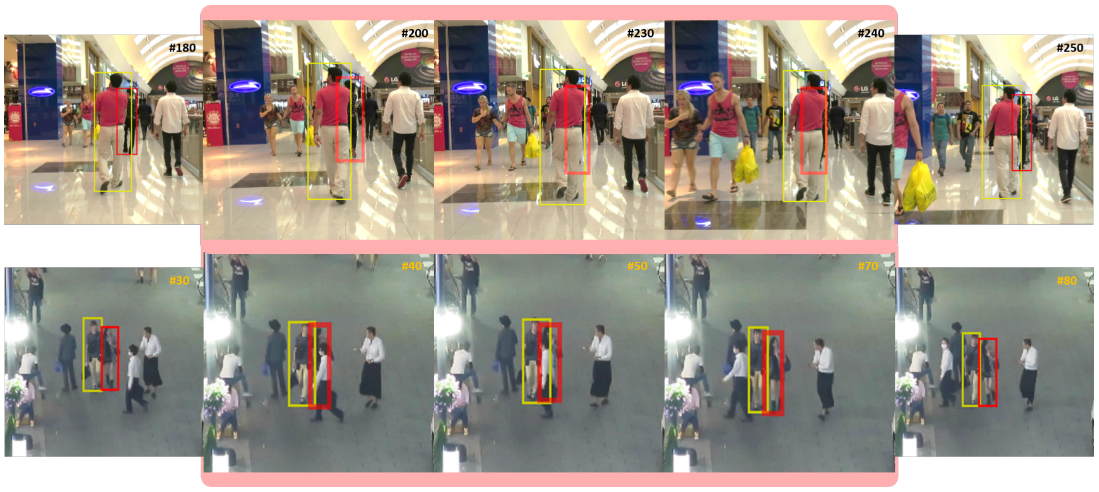


Figure 27: Tracking occluded object (red bounding box) by its relations with neighbors. The position of the object in the shaded region is recovered from its relation embedding.

6.4.3 *Tracking-by-relations*

We next demonstrate that the position of out of sight objects can be recovered based purely on relation cues. To do so, we select an instance, which undergoes occlusion while moving in a group of related objects. We then apply the approach described in Section 6.2. Practically, we do not kill the trajectories of non-visible objects, but continue to track them based only on the relational cues.

Figures 26 and 27 demonstrate qualitative results. As can be seen, the position of the occluded object can be approximately recovered from the group relations. It indicates

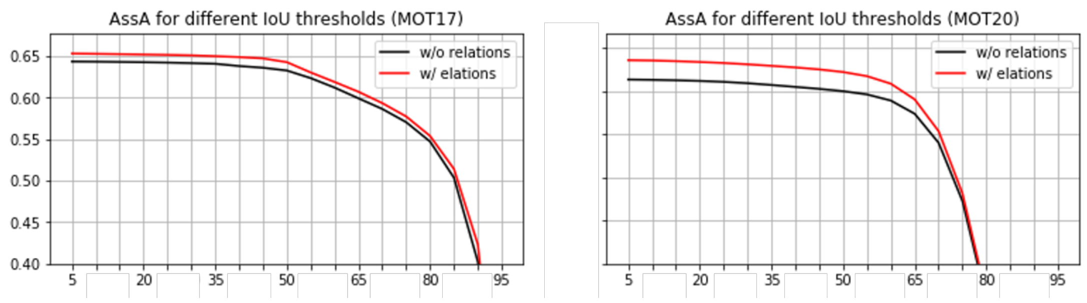


Figure 28: Association Accuracy (AssA) of the relation-aware and the baseline tracker over different localization thresholds.

that relation embeddings can be learned to encode the geometric prior about the relative positions of the objects.

6.4.4 *Analyzing higher-order tracking accuracy*

To investigate the ability of the relation-aware model to provide robust tracking in the dense scenes, we decompose and analyze Higher-Order Tracking Accuracy (HOTA) [114] metrics over various localization thresholds α. From [114]:

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha} \tag{6.8}$$

where $DetA_\alpha, AssA_\alpha$ stand for detection and association accuracy for a given α. Since the classification part responsible for the quality of detection remains unchanged, we only analyze the association accuracy. Association accuracy for tracking-by-regression methods measures the ability of the regression head of the tracker to preserve identities frame by frame. We plot $AssA_\alpha$ for the baseline and relation-aware trackers at Figure 28.

Low localization thresholds permit the association of loose bounding boxes. It can deteriorate the association quality when predicted bounding boxes are densely overlapped, which is a common case in crowded scenarios. Thus, the higher association accuracy of relation-aware tracker (Figure 28) under low localization thresholds indicates the better ability to preserve identities of densely interacting objects.

Ablation study

We conduct an ablation study on the MOT17-FRCNN and MOT20 train splits to investigate an impact of the distance used to build the relational graph d_{th}, with higher d_{th} resulting in a relational graph of a larger spatial extent. Since the relation-aware model is fine-tuned for the tracking task, we also compare against the fine-tuned baseline method from [14].

As can be seen in Table 12 and Table 13, the relation-aware tracker outperforms the baseline method and its fine-tuned version. We observed that using the relational graphs of a larger spatial extent generally results in higher performance. Intuitively, lower d_{th} restricts to encode relation only in small groups, missing the possible distant connections. At the same time, a very large d_{th} leads to a slight decrease in performance. We attribute it to the fact that in very big relational graphs, formed relations are less sharp because of the attention aggregation mechanism.

Method	IDF1	MOTA	MOTP
Base	65.0	61.7	89.4
Base-tuned	66.7	61.8	89.5
Rel $d_{th} = 10$	66.9	61.9	89.5
Rel $d_{th} = 20$	**67.1**	**62.0**	**89.5**
Rel $d_{th} = 30$	**67.1**	**62.0**	**89.5**
Rel $d_{th} = 40$	66.9	61.9	89.4

Table 12: Comparing the baseline and relation-aware models over different distance threshold used to build relational graph on MOT17 dataset. Rel $d_{th} = 20$ stands for the relation-aware tracker with distance threshold of 20 to build relational graph.

Method	IDF1	MOTA	MOTP
Base	60.7	66.4	89.1
Base-tuned	62.4	66.5	89.2
Rel $d_{th} = 5$	62.6	66.5	89.2
Rel $d_{th} = 10$	**62.7**	**66.7**	**89.2**
Rel $d_{th} = 20$	**62.7**	**66.7**	**89.2**
Rel $d_{th} = 30$	62.5	66.5	89.2

Table 13: The baseline and relation-aware models' results with different distance threshold used to build relational graph on MOT20 dataset.

6.5 DISCUSSION

In this work, we demonstrate that learning inter-object relations is important for robust multi-object tracking. We develop a plug-in relation encoding module, which encodes relations by running a message passing over a spatial-temporal graph of tracked instances.

We experimentally demonstrate that extending a backbone multi-object tracker with REM improves tracking quality. We also investigate the ability of the proposed method to track heavily occluded objects based on the relational cues, when appearance information is unreliable. Our experiments suggest that relational information is important and should not be left out from the analysis.

We suppose that REM would be the most useful in problems, where video analysis of crowded scenes with regular occlusions is required. For example, multi-object tracking in crowded streets, underground, airports, and railway stations. Moreover, one can use the proposed relation encoding module not only for tracking but also for action recognition or scene anomaly detection. We leave these questions for further research.

7

SUMMARY AND CONCLUSION

SUMMARY

In the thesis, we study the invariance as a key ingredient for efficient representation learning with neural networks. Specifically, we focus on structured invariance, where the invariance follows an explicit mathematical structure such as the structure of a symmetry group. We considered the following research questions:

Do neural networks learn structured invariance from the data?
Chapter 2: Investigating properties and limitations of learned invariance

For the first research question, we compared the invariance of unconstrained neural networks trained from the data with the oracle invariance in neural networks under invariant weight-tying constraints. To this end, we proposed several metrics to measure learned invariance: logit invariance, predictive distribution invariance, and saliency invariance. Our experiments revealed that the invariance learned by unconstrained neural networks is strongly conditioned on the input data. This causes learned invariance to rapidly decay when evaluated out-of-distribution. Then, we tackled the task of aligning data-driven invariance learning with the genuine invariance of weight-tying models. To do so, we proposed to regularize the invariance error during the training. We demonstrated that such a simple strategy significantly improves the quality of learned invariance. However, improved invariance came at the cost of a downstream task performance due to the implicit regularization of the invariance error optimization. We further showed that such implicit regularization is due to the spectral decay phenomenon, where a network opts for reducing input sensitivity to all perturbations to achieve invariance to a specific group of transformations.

Can we infer invariance structure from learned representations?
Chapter 3: Extracting learned invariance by analyzing learned Lie group generators

For this research question, we developed a method to retrieve the group of invariance learned by a neural network from the data. In particular, we focused on the setting, where invariance comes in the form of a Lie group. We leveraged Lie group - Lie algebra correspondence to develop a method to retrieve learned invariance by solving a simple nullspace equation, where the solution corresponds to the generator of a learned symmetry group. With this, we proposed two new metrics to evaluate the invariant properties of a neural network - symmetry bias and variance, which allows for a more nuanced analysis of learned invariance.

Can we improve contrastive representation learning with more structured supervision?

Chapter 4: Contrastive representation learning with structured assignments

For this research question, we considered the task of contrastive representation learning through the lens of assignment theory. We demonstrated that standard pair-wise contrastive learning methods stem from linear assignment problems. From here, we proposed a way to generalize contrastive representation learning beyond pair-wise contrast by generalizing an underlying assignment problem from linear to quadratic. This allows additional structural information, which is missed by standard methods based on pair-wise contrast. Our experiments suggest that going beyond pair-wise contrast can significantly improve the quality of learned representations in metric learning and self-supervised classification setups.

Can we improve object localization with structured representations?

Chapter 5: Scale equivariant representations for robust visual object tracking

For this research question, we tackled the problem of visual object tracking, and we developed a method to structure objects' representations with scale equivariance. To do so, we designed a scale-equivariant siamese network that is adopted for the task of tracking via a scale-equivariant template matching mechanism. We experimentally demonstrated that in-built scale-equivariance significantly increases the precision of object size estimation in a wide range of circumstances. This leads to a more accurate position estimation and alleviates the accumulation of tracking errors throughout the sequence. As a result, the tracker with scale-equivariant representations allows for significantly more robust and precise object localization.

Can we improve object localization with structured invariance prior?

Chapter 6: Relational prior for tracking groups of objects

Lastly, for this research question, we developed the relation encoding mechanism that acts as a soft prior to structure tracker's representations towards invariance in the form of consistent spatio-temporal relations between objects in a scene. We demonstrated that our relation prior significantly improves localization accuracy, especially in crowded scenes with a high degree of visual occlusions. Furthermore, the proposed relation encoding allows tracking even fully occluded objects as long as they are in relation to other non-occluded objects in a scene.

SAMENVATTING

In de scriptie bestuderen we invariantie als een sleutelingrediënt voor het leren van efficiënte representaties met neurale netwerken. Specifiek richten wij ons op gestructureerde invariantie, waarbij de invariantie een expliciete wiskundige structuur volgt, zoals de structuur van een symmetriegroep. We hebben de volgende onderzoeksvragen overwogen:

Leren neurale netwerken gestructureerde invariantie uit data?

Hoofdstuk 2: Het onderzoeken van de eigenschappen en beperkingen van geleerde invariantie

Voor de eerste onderzoeksvraag vergeleken we de invariantie van neurale netwerken zonder beperkingen getraind op data met de orakel-invariantie in neurale netwerken onder invariante gewichtskoppelingsbeperkingen. Hiertoe stelden we verschillende metrieken op om de geleerde invariantie te meten: logit invariantie, voorspellende distributie invariantie, en opvallendheids invariantie. Onze experimenten onthulden dat de door onbeperkte neurale netwerken geleerde invariantie sterk afhankelijk is van de invoergegevens. Dit zorgt ervoor dat geleerde invariantie snel afneemt wanneer deze buiten de data distributie wordt geëvalueerd. Vervolgens pakten we de taak aan om datagestuurde invariantie-learning af te stemmen op de echte invariantie van gewichtskoppelingsmodellen. Hiervoor stelden we voor om de invariantiefout tijdens de training te regulariseren. We toonden aan dat deze eenvoudige strategie de kwaliteit van geleerde invariantie aanzienlijk verbetert. Echter, verbeterde invariantie ging ten koste van de prestaties op de taken die volgden, doordat de impliciete regularisatie van de invariantiefoutoptimalisatie. We toonden verder aan dat deze impliciete regularisatie te wijten is aan het fenomeen van spectrale afname, waarbij een netwerk ervoor kiest om de gevoeligheid voor alle verstoringen te verminderen om invariantie te bereiken voor een specifieke groep transformaties.

Kunnen we de invariantiestructuur afleiden uit geleerde representaties?

Hoofdstuk 3: De geleerde invariantie extraheren door het analyseren van geleerde Lie-groepgeneratoren

Voor dit onderzoek ontwikkelden we een methode om de groep van invariantie die door een neuraal netwerk uit de gegevens is geleerd, op te zoeken. Wij richtte ons specifiek op de situatie, waarbij de invariantie in de vorm van een Lie-groep aanneemt. We maakten gebruik van de overeenkomst tussen Lie-groepen en Lie-algebra’s om een methode te ontwikkelen om de geleerde invariantie terug te halen doormiddel van het oplossen van een eenvoudige nulruimtevergelijking, waarbij de oplossing overeenkomt met de generator van een geleerde symmetriegroep. Hiermee stelden we twee nieuwe metrieken voor om de invariante eigenschappen van een neuraal netwerk te evalueren - de symmetriebias en de variantie, wat een meer genuanceerde analyse van de geleerde invariantie mogelijk maakt.

Kunnen we contrastief representatieleren verbeteren met meer gestructureerd toezicht?

Hoofdstuk 4: Het Contrastief leren van representaties met gestructureerde toewijzingen

Voor deze onderzoeksvraag beschouwden we de taak van het contrastief representatieleren door de lens van toewijzingstheorie. We toonden aan dat standaard paarsgewijze contrastieve leermethoden voortkomen uit lineaire toewijzingsproblemen. Vanuit deze aanname stelden we een manier voor om contrastief representatieleren te generaliseren voorbij paarsgewijze contrast door een onderliggend toewijzingsprobleem te generaliseren van lineair naar kwadratisch. Dit maakt extra structurele informatie mogelijk, die gemist wordt door standaardmethoden gebaseerd op het paarsgewijze contrast. Onze experimenten suggereren dat het voorbij gaan van het paarsgewijze contrast de kwaliteit van de geleerde representaties in metriek leren en zelf-toezichthoudende classificatieopstellingen aanzienlijk kan verbeteren.

Kunnen we objectlokalisatie verbeteren met gestructureerde representaties?

Hoofdstuk 5: Schaal-equivariante representaties voor robuuste visuele objecttracking

Voor deze onderzoeksvraag pakten we het probleem van visuele objecttracking aan. We ontwikkelden een methode om objectrepresentaties te structureren met schaalequivariantie. Hiertoe ontwierpen we een schaal-equivariante siamees netwerk dat was aangepast voor de taak van tracking via een schaal-equivariante template matching mechanisme. We toonden experimenteel aan dat ingebouwde schaalequivariantie de precisie van objectgrootte-schatting aanzienlijk verhoogt in een breed scala van omstandigheden. Dit leidt tot een nauwkeurigere positiebepaling en vermindert de accumulatie van trackingfouten gedurende de reeks. Als resultaat maakt de tracker met schaal-equivariante representaties significant robuustere en nauwkeurigere objectlokalisatie mogelijk.

Kunnen we objectlokalisatie verbeteren met gestructureerde invariantie als a-prior verdeling?

Hoofdstuk 6: Relationele a-priori verdeling voor het tracken van groepen objecten

Ten slotte hebben we voor deze onderzoeksvraag het relatiecoderingsmechanisme ontwikkeld dat fungeert als een zachte a-prior verdeling om trackerrepresentaties te structureren naar invariantie in de vorm van consistente ruimte- en tijdrelaties tussen objecten in een scène. We toonden aan dat onze relationele a-prior verdeling de lokaliseringsnauwkeurigheid aanzienlijk verbetert, vooral in drukke scènes met een hoge mate van visuele occlusies. Bovendien maakt de voorgestelde relatiecodering het tracken van zelfs volledig geoccludeerde objecten mogelijk zolang ze in relatie staan tot andere niet-geoccludeerde objecten in een scène.

CONCLUSION

In this thesis, we explored representation learning with a focus on structure and invariance. First, we analyzed the structure of invariance as learned by neural networks from the data. We concluded that neural networks learn relevant invariances from the data when they are integral to the downstream task. Yet, neural networks often struggle to accurately capture the full range the relevant invariances when no geometric priors are available. Therefore, we conclude that using geometric prior at the learning stage is crucial to ensure reliable invariance. Next, we proposed several methods to improve representation learning in neural networks through more structured supervision and the integration of invariance priors. We observed that structuring the supervision to promote invariance as well as incorporating the invariance information a priori considerably improves the quality of learned representations which indicates the pivotal role of invariance in efficient representation learning.

While advancing our understanding of structured invariance in neural networks, many open research questions and challenges remain. One of the open challenges is to model diverse real-world invariances that do not follow a well-defined group structure. Alternative to group-theoretical invariance, category theory offers a more general notion of *naturality* to describe transformations that preserve the internal structure of the categories (domains) involved. This provides a potentially more flexible approach to understanding the invariance in a wider range of applications. Another open research direction is *deformable* invariance where the network can dynamically adapt to the symmetry structure in the data. This requires a neural network that can detect various symmetry patterns across data points and then align its processing mechanism to be consistent with those patterns.

In the broader context, our work emphasizes the significance of representation learning not just in terms of encoding relevant data features, but also in terms of learning to disregard irrelevant ones. These can be seen as two dual approaches to representation learning with respect to the minimum description length principle. That is, to obtain an optimal data representation, we can either focus on enhancing the relevant features to overshadow the irrelevant ones or on suppressing the irrelevant features to highlight the relevant ones. While both approaches strive for similar endpoints, they do so along very different routes, and it remains to study which of the ways is more advantageous and in which scenarios.

8

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor, Professor Arnold Smeulders. What I learned from you extends far beyond the realm of science; it encompasses life wisdom, the art of communication, the skill of asking the right questions, and the insight to plan and set priorities smartly. Also, I have gained a surprising amount of knowledge about sailing and boats – a somewhat unexpected bonus from a Ph.D. in machine learning. But seriously, I consider myself so fortunate that I had an opportunity to work with Arnold and learn from all his experience. Arnold, thank you for not only being an exceptional guide in my academic pursuit but also a mentor in the broader journey of life.

I am grateful to Professor Cees Snoek for his unwavering guidance and support. Our conversations really helped me to find the direction whenever I felt lost. More than a director, Cees has been the architect of a motivating and welcoming atmosphere at VISLab, making it a place where I felt truly inspired.

I want to extend my sincere thanks to those who shared their wisdom and offered guidance throughout my journey: Erik Bakkers, Pascal Mettes, Volker Fischer, Michael Pfeiffer, Jan-Willem van de Meent, Patrick Forré, Herke van Hoof, and Taco Cohen. The lessons I have learned from each of you have enriched my personal and professional growth, and for that, I am deeply grateful.

My heartfelt thanks go to my friend and a great scientist, Ivan Sosnovik, whose influence was pivotal in shaping my growth as a researcher during my PhD. Ivan, your exceptional skill of unraveling complex ideas into simpler, understandable parts and explaining them with such ease is truly remarkable. I believe your ability to bring clarity to complexity is what made our joint work so productive, and your sense of humor is what made it so fun.

I would like to thank my friend and exceptionally talented mathematician, Anna Sepliarskaia. Anna's talent in representation theory is matched only by her kickboxing and juggling skills. Anna, the depth of our discussions has been immensely valuable, offering me insights into mathematics and life.

I want to express my sincere gratitude to my Delta lab mates Elise van der Pol, Andy Keller (a.k.a. Neural Andy), Sindy Lowe, Emiel Hoogeboom, Victor Garcia Satorras, Metod Jazbec, Daan Roos, Mona Schirmer, Ronny Velastegui, Rajeev Verma, Xiaoyan Xing, Melis Öcal, and Alexander Timans. I am extremely fortunate to have had the opportunity to pursue my PhD alongside such bright minds as you guys.

My deepest gratitude to all the amazing people from AMLab and VISLab that I had the privilege to work alongside with: Babak Esmaeili, David Ruhe, Sharvaree Vadgama, Putri van der Linden, Grigory Bartosh, Teodora Pandeva, Max Zhdanov, Amber Brands, Aritra Bhowmik, Leonard Bereska, Alex Gabel, Evgenii Egorov, David Knigge, Miltiadis Kofinas, Mina Ghadimiatigh, Riaan Zoetmulder, Mohammadreza Salehi, Andreas Panteli, Shuo Chen, Yunlu Chen, Tao Hu, Mert Kilickaya, Zenglin Shi, William Thong, David Zhang, and Deepak Gupta. Your diverse perspectives and collaborative spirit made every day in the lab a learning adventure. Special shout out to Evgenii for all the great chess games we played – excited for another round soon!

I want to thank my friends and fellow PhD students Ivona Najdenkoska, Mehmet Altinkaya, Robert-jan Bruintjes, Fida Thoker, Mohammad Derakhshani, and Sadaf Gulshad. Our conversations were not just fun but also very insightful, especially when it came to culture, science, and philosophy. It was absolutely great doing our PhDs together and sharing laughs and chats about the roller-coaster ride of a PhD life. Thanks for making it such a fun ride!

I am grateful to Anna Rozhavskaya for all of her unwavering support and empathy throughout the course of my PhD journey. Anna's patience and understanding, especially during the most demanding phases of my research, have been a constant source of strength and motivation for me.

I want to express my deep gratitude to my dear family for providing me with the foundation that has shaped me into the person I am today. Knowing that you are all well and happy brings calmness to my mind and happiness to my heart, giving me a solid foundation to set and achieve my goals. I am thankful to my father for teaching me the importance of optimism and finding joy in life, no matter the situation. My heartfelt appreciation goes to my mother, whose constant warmth and support exemplify the essence of unconditional love. To my dear sister, I am immensely grateful for the deep trust we share. I am truly fortunate to have a sister like you.

BIBLIOGRAPHY

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers, 2018.

[3] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall/CRC, 2006.

[4] F. Anselmi, G. Evangelopoulos, L. Rosasco, and T. Poggio. Symmetry-adapted representation learning. *Pattern Recognition*, 86:201–208, 2019.

[5] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[6] G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

[7] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

[8] A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[9] S. Becker and G. E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.

[10] E. J. Bekkers. B-spline cnns on lie groups. In *International Conference on Learning Representations*, 2020.

[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[12] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and U. Montreal. Greedy layer-wise training of deep networks. volume 19, 01 2007.

[13] G. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. Learning invariances in neural networks from training data. *Advances in neural information processing systems*, 33:17605–17616, 2020.

[14] P. Bergmann, T. Meinhardt, and L. Leal-Taixé. Tracking without bells and whistles. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[15] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *J. Image Video Process.*, 2008, Jan. 2008.

[16] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[17] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1998.

[18] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[19] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte. Learning discriminative model prediction for tracking. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[20] M. Blondel, A. Martins, and V. Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 606–615. PMLR, 2019.

[21] G. Brasó and L. Leal-Taixé. Learning a neural solver for multiple object tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[22] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.

[23] R. E. Burkard. *Quadratic Assignment Problems*, pages 2741–2814. Springer New York, 2013.

[24] R. E. Burkard and E. Çela. Linear assignment problems and extensions. In *Handbook of Combinatorial Optimization*, 1999.

[25] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, J. Yuan, and N. M. Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2281, 2019.

[26] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[27] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924, 2020.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[29] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[30] T. Chen, C. Luo, and L. Li. Intriguing properties of contrastive losses. In *Advances in Neural Information Processing Systems*, volume 34, pages 11834–11845, 2021.

[31] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[32] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[33] D. M. Chu and A. W. Smeulders. Thirteen hard cases in visual tracking. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 103–110. IEEE, 2010.

[34] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

[35] T. S. Cohen and M. Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[36] M. Connor and C. Rozell. Representing closed transformation paths in encoded network latent space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3666–3675, 2020.

[37] J.-B. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

[38] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[39] B. Culpepper and B. Olshausen. Learning transport operators for image manifolds. *Advances in neural information processing systems*, 22, 2009.

[40] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Atom: Accurate tracking by overlap maximization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[41] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[42] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European conference on computer vision*, pages 472–488. Springer, 2016.

[43] N. Dehmamy, R. Walters, Y. Liu, D. Wang, and R. Yu. Automatic symmetry discovery with lie algebra convolutional network. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[44] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes, 2020.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[46] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 459–474, 2018.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[48] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[49] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024, 2021.

[50] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*, 2020.

[51] G. Finke, R. E. Burkard, and F. Rendl. Quadratic assignment problems. In *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 61–82. North-Holland, 1987.

[52] M. Finzi, G. Benton, and A. G. Wilson. Residual pathway priors for soft equivariance constraints. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[53] M. Finzi, M. Welling, and A. G. Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *Arxiv*, 2021.

[54] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[55] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[56] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[57] I. Goodfellow, H. Lee, Q. Le, A. Saxe, and A. Ng. Measuring invariances in deep networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[58] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[59] N. Gruver, M. Finzi, M. Goldblum, and A. G. Wilson. The lie derivative for measuring learned equivariance, 2022.

[60] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1763–1771, 2017.

[61] M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[62] T. Hazan, J. Keshet, and D. McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[63] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

[64] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[65] K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019.

[66] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[67] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[68] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51, 05 1998.

[69] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference Computer Vision (ECCV)*, 2016.

[70] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

[71] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn. Improvements to frank-wolfe optimization for multi-detector multi-object tracking. *CVPR*, 05 2017.

[72] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[73] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[74] D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. ICLR, 2019.

[75] E. Hoogeboom, J. W. Peters, T. S. Cohen, and M. Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018.

[76] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *CVPR*, pages 3588–3597, 2018.

[77] L. Huang, X. Zhao, and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2019.

[78] Z. Huang, P. Hu, J. T. Zhou, J. Lv, and X. Peng. Partially view-aligned clustering. *Advances in Neural Information Processing Systems*, 33:2892–2902, 2020.

[79] T. Huynh, S. Kornblith, M. R. Walter, M. Maire, and M. Khademi. Boosting contrastive self-supervised learning with false negative cancellation. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[80] J.-H. Jacobsen, B. de Brabandere, and A. W. M. Smeulders. Dynamic steerable blocks in deep residual networks. In *British Machine Vision Conference*, 2017.

[81] H. H. Jaffé and M. Orchin. *Symmetry in chemistry*. Courier Corporation, 2002.

[82] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1), 2020.

[83] G. D. James. *The representation theory of the symmetric groups*, volume 682. Springer, 2006.

[84] E. Jenner and M. Weiler. Steerable partial differential operators for equivariant neural networks. In *International Conference on Learning Representations*, 2022.

[85] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[86] M. Kaiser, C. C. Hilgetag, and R. Kötter. Hierarchy and dynamics of neural networks. *Frontiers in neuroinformatics*, 4:112, 2010.

[87] D. Kaltenpoth and J. Vreeken. We are not your real parents: Telling causal from confounded using mdl. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 199–207. SIAM, 2019.

[88] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion segmentation multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):140–153, 2020.

[89] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects, 2016.

[90] N. Khetan, T. Arora, S. U. Rehman, and D. K. Gupta. Implicit equivariance in convolutional networks, 2021.

[91] V. Khrulkov and I. Oseledets. Art of singular vectors and universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[92] P. Kidger and T. Lyons. Universal Approximation with Deep Narrow Networks. In *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, 09–12 Jul 2020.

[93] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[94] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[95] A. N. Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7, 1965.

[96] R. Kondor and S. Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

[97] N. Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1:417–446, 2015.

[98] M. Kristan, A. Leonardis, J. Matas, and M. F. et. The visual object tracking vot2017 challenge results. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1949–1972, 2017.

[99] H. Kvinge, T. Emerson, G. Jorgenson, S. Vasquez, T. Doster, and J. Lew. In what ways are deep neural networks invariant and how should we measure this? In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[100] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 289–297, 2016.

[101] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 473–480, New York, NY, USA, 2007. Association for Computing Machinery.

[102] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345. Springer Verlag, 1999. International Workshop on Shape, Contour and Grouping in Computer Vision ; Conference date: 26-05-1998 Through 29-05-1998.

[103] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019.

[104] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence, 2014.

[105] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. *arXiv preprint arXiv:1812.11703*, 2018.

[106] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[107] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 152–159, 2014.

[108] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[109] Y. Lin, Y. Gou, Z. Liu, B. Li, J. Lv, and X. Peng. Completer: Incomplete multi-view clustering via contrastive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[110] R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21:105–117, 03 1988.

[111] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[112] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

[113] Z. Lu, V. Rathod, R. Votel, and J. Huang. Retinatrack: Online single stage joint detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[114] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, pages 1–31, 2020.

[115] D. Marcos, B. Kellenberger, S. Lobry, and D. Tuia. Scale equivariance in cnns with vector fields. *arXiv preprint arXiv:1807.11783*, 2018.

[116] A. F. T. Martins and R. F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, 2016.

[117] A. Marx and J. Vreeken. Formally justifying mdl-based inference of cause and effect. *arXiv preprint arXiv:2105.01902*, 2021.

[118] J. Materzynska, T. Xiao, R. Herzig, H. Xu, X. Wang, and T. Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *CVPR*, 2020.

[119] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[120] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831, 2016.

[121] J. P. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, pages 7721–7735. PMLR, 2021.

[122] I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6707–6717, 2020.

[123] A. Moskalev, A. Sepliarskaia, E. J. Bekkers, and A. Smeulders. On genuine invariance learning without weight-tying. In *ICML workshop on Topology, Algebra, and Geometry in Machine Learning*, 2023.

[124] A. Moskalev, A. Sepliarskaia, I. Sosnovik, and A. Smeulders. Liegg: Studying learned lie group generators. In *Advances in Neural Information Processing Systems*, 2022.

[125] A. Moskalev, I. Sosnovik, and A. Smeulders. Relational prior for multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1081–1085, 2021.

[126] A. Moskalev, I. Sosnovik, and A. Smeulders. Two is a crowd: tracking relations in videos. *arXiv preprint arXiv:2108.05331*, 2021.

[127] A. Moskalev, I. Sosnovik, F. Volker, and A. Smeulders. Contrasting quadratic assignments for set-based representation learning. In *European Conference on Computer Vision*, 2022.

[128] T. N. Mundhenk, B. Y. Chen, and G. Friedland. Efficient saliency maps for explainable ai. *arXiv preprint arXiv:1911.11293*, 2019.

[129] M. Narasimhan, S. Lazebnik, and A. Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *NeurIPS*, 2018.

[130] T. Nguyen, M. Raghu, and S. Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021.

[131] V. Niculae, A. Martins, M. Blondel, and C. Cardie. Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning (ICML)*, pages 3799–3808, 2018.

[132] C. Olah, N. Cammarata, C. Voss, L. Schubert, and G. Goh. Naturally occurring equivariance in neural networks. *Distill*, 2020. https://distill.pub/2020/circuits/equivariance.

[133] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[134] M. Patacchiola and A. Storkey. Self-supervised relational reasoning for representation learning. In *Advances in Neural Information Processing Systems*, 2020.

[135] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009.

[136] X. Peng, Z. Huang, J. Lv, H. Zhu, and J. T. Zhou. Comic: Multi-view clustering without parameter selection. In *International Conference on Machine Learning (ICML)*, 2019.

[137] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[138] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 06–11 Aug 2017.

[139] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions, 2017.

[140] R. Rao and D. Ruderman. Learning lie groups for invariant visual perception. *Advances in neural information processing systems*, 11, 1998.

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

[142] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union. 2019.

[143] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[144] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *European Conference on Computer Vision*, pages 17–35, 2016.

[145] D. W. Romero, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn. Attentive group equivariant convolutional networks. *arXiv preprint arXiv:2002.03830*, 2020.

[146] D. W. Romero and M. Hoogendoorn. Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. *arXiv preprint arXiv:1911.07849*, 2019.

[147] S. Sanborn, C. A. Shewmake, B. Olshausen, and C. J. Hillar. Bispectral neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[148] H. Sheng, Y. Zhang, J. Chen, Z. Xiong, and J. Zhang. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(11):3269–3280, 2019.

[149] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[150] E. Sober. *Ockham's razors*. Cambridge University Press, 2015.

[151] J. Sohl-Dickstein, C. M. Wang, and B. A. Olshausen. An unsupervised algorithm for learning lie group transformations. *arXiv preprint arXiv:1001.1027*, 2010.

[152] I. Sosnovik, A. Moskalev, C. Kaandorp, and A. Smeulders. Learning to summarize videos by contrasting clips. *arXiv preprint arXiv:2301.05213*, 2023.

[153] I. Sosnovik, A. Moskalev, and A. Smeulders. Disco: accurate discrete scale convolutions. *arXiv preprint arXiv:2106.02733*, 2021.

[154] I. Sosnovik, A. Moskalev, and A. Smeulders. How to transform kernels for scale-convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1092–1097, 2021.

[155] I. Sosnovik, A. Moskalev, and A. W. Smeulders. Scale equivariance improves siamese tracking. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2765–2774, January 2021.

[156] I. Sosnovik, M. Szmaja, and A. Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations*, 2020.

[157] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[158] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5033–5041, 2015.

[159] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-person tracking by multicut and deep matching. In *Computer Vision – ECCV 2016 Workshops*, pages 100–111, 2016.

[160] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person re-identification. In *CVPR*, pages 3701–3710, 2017.

[161] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429, 2016.

[162] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[163] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[164] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(50):1453–1484, 2005.

[165] W.-K. Tung. *Group theory in physics*, volume 1. World Scientific, 1985.

[166] S. Vakili, M. Bromberg, J. Garcia, D.-s. Shiu, and A. Bernacchia. Uniform generalization bounds for overparameterized neural networks, 2021.

[167] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[168] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[169] P. M. Vitányi and M. Li. Minimum description length induction, bayesianism, and kolmogorov complexity. *IEEE Transactions on information theory*, 46(2):446–464, 2000.

[170] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. MOTS: Multi-object tracking and segmentation. In *CVPR*, 2019.

[171] R. Wang, R. Walters, and R. Yu. Approximately equivariant networks for imperfectly symmetric dynamics. *arXiv preprint arXiv:2201.11969*, 2022.

[172] T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning (ICML)*, 2020.

[173] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li. Dense contrastive learning for self-supervised visual pre-training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[174] Y. Wang, J. Wang, Z. Cao, and A. Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022.

[175] Z. Wang, L. Zheng, Y. Liu, and S. Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019.

[176] M. Weiler and G. Cesa. General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[177] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[178] S. Wojtowytsch and E. Weinan. On the emergence of tetrahedral symmetry in the final and penultimate layers of neural network classifiers. *arXiv preprint arXiv:2012.05420*, 2020.

[179] D. Worrall and M. Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[180] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7168–7177, 2017.

[181] B. Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 951–958, 2006.

[182] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[183] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[184] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[185] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[186] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé, and X. Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6787–6796, 2020.

[187] M. Yang, Y. Li, P. Hu, J. Bai, J. C. Lv, and X. Peng. Robust multi-view clustering with incomplete information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[188] M. Yang, Y. Li, Z. Huang, Z. Liu, P. Hu, and X. Peng. Partially view-aligned representation learning with noise-robust contrastive loss. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[189] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[190] Y. Yoshida and T. Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[191] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[192] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets, 2017.

[193] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, 2021.

[194] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[195] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[196] X. Zhang, L. Wang, J. Helwig, Y. Luo, C. Fu, Y. Xie, M. Liu, Y. Lin, Z. Xu, K. Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023.

[197] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[198] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 351–366, 2018.

[199] Z. Zhang and H. Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4591–4600, 2019.

[200] A. Zhou, T. Knowles, and C. Finn. Meta-learning symmetries by reparameterization, 2020.

[201] X. Zhou, V. Koltun, and P. Krähenbühl. Tracking objects as points. *ECCV*, 2020.

[202] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[203] W. Zhu, Q. Qiu, R. Calderbank, G. Sapiro, and X. Cheng. Scale-equivariant neural networks with decomposed convolutional filters. *arXiv preprint arXiv:1909.11193*, 2019.

[204] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845, 2020.