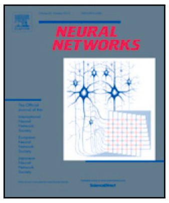
Neural Networks 174 (2024) 106229

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

Efficient learning of Scale-Adaptive Nearly Affine Invariant Networks

Zhengyang Shen[a,1], Yeqing Qiu[b,c,1], Jialun Liu[a], Lingshen He[d], Zhouchen Lin[d,e,f,*]

[a] Baidu Inc, Beijing, 100871, China
[b] Shenzhen Research Institute of Big Data, Shenzhen, 518172, China
[c] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, 518172, China
[d] National Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing, 100871, China
[e] Institute for Artificial Intelligence, Peking University, Peking, 100871, China
[f] Peng Cheng Laboratory, Shenzhen, 518000, China

ARTICLE INFO

Keywords:
Representation learning
Affine invariance
Self-supervised method
Data augmentation

ABSTRACT

Recent research has demonstrated the significance of incorporating invariance into neural networks. However, existing methods require direct sampling over the entire transformation set, notably computationally taxing for large groups like the affine group. In this study, we propose a more efficient approach by addressing the invariances of the subgroups within a larger group. For tackling affine invariance, we split it into the Euclidean group $E(n)$ and uni-axial scaling group $US(n)$, handling invariance individually. We employ an $E(n)$-invariant model for $E(n)$-invariance and average model outputs over data augmented from a $US(n)$ distribution for $US(n)$-invariance. Our method maintains a favorable computational complexity of $\mathcal{O}(N^2)$ in 2D and $\mathcal{O}(N^4)$ in 3D scenarios, in contrast to the $\mathcal{O}(N^6)$ (2D) and $\mathcal{O}(N^{12})$ (3D) complexities of averaged models. Crucially, the scale range for augmentation adapts during training to avoid excessive scale invariance. This is the first time nearly exact affine invariance is incorporated into neural networks without directly sampling the entire group. Extensive experiments unequivocally confirm its superiority, achieving new state-of-the-art results in affNIST and SIM2MNIST classifications while consuming less than 15% of inference time and fewer computational resources and model parameters compared to averaged models.

1. Introduction

Ideally, a machine learning model should adeptly extract task-specific features while effectively disregarding irrelevant information. For instance, applying modest geometric transformations to an image should not essentially effect the features for classification. In other words, an ideal model is sought to be invariant *w.r.t.* these transformations. It is worth noting that CNN models inherently exhibit translation invariance after global spatial pooling, which contributes to their superiority over fully-connected neural networks that lack this property. However, vanilla CNNs fall short of achieving invariance over more intricate transformations, such as affine transformations.

To enhance invariance, a common approach is the utilization of data augmentation (DA). For affine transformations encompassing rotations, reflections, scalings, shears, and more, multiple augmentations are typically sampled and then introduced into the model. Nevertheless, this method places a substantial learning burden on the network, and the invariance it offers lacks theoretical guarantees. Some alternative techniques (Benton, Finzi, Izmailov, & Wilson, 2020; Immer, van der Ouderaa, Rätsch, Fortuin, & van der Wilk, 2022; Laptev, Savinov, Buhmann, & Pollefeys, 2016; Miao et al., 2023) involve averaging model outputs across data augmented by the considered transformation group. For instance, Lila (Immer et al., 2022) and Augerino (Benton et al., 2020) are representative models that employ the reparameterization technique to represent the distribution of the transformation. However, it is important to note that the number of required samples scales with the size of the considered transformation group, rendering them computationally expensive for the large affine group.

In addition to averaged models, several works (Cohen & Welling, 2016; Finzi, Stanton, Izmailov, & Wilson, 2020; Sosnovik, Szmaja, & Smeulders, 2020; Weiler, Hamprecht, & Storath, 2018) have employed group convolutions to address invariance over relatively simple groups like rotations and scalings. These approaches are designed to introduce hard-coded symmetries into neural networks by means of group convolutions. Finzi et al. (2020) introduce a universal technique to build a convolutional layer with equivariance to transformations from any designated Lie group featuring a surjective exponential map.

[*] Corresponding author at: National Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing, 100871, China.
E-mail address: zlin@pku.edu.cn (Z. Lin).
[1] These authors contributed equally.

https://doi.org/10.1016/j.neunet.2024.106229
Received 25 September 2023; Received in revised form 4 March 2024; Accepted 7 March 2024
Available online 11 March 2024
0893-6080/© 2024 Published by Elsevier Ltd.

Table 1
Summary of notations in this paper.

Symbol	Description	Symbol	Description
$\mathcal{G}$	Transformation group	π_g	Group action of g
g	Transformation in $\mathcal{G}$	f	Input data
$US(n)$	Uni-axial scaling group	Ψ	Network model
$IS(n)$	Isotropic scaling group	μ_θ	Parameterized distribution of group
$SO(n)$	Special orthogonal group	θ	Parameter of the distribution
$T(n)$	Translation group	ϵ_i	Learnable parameters
$E(n)$	Euclidean group	S	Transformation set
$\mathrm{Aff}(n)$	Affine group	s	Transformation in S
$\mathcal{G}_i$	Subgroup of $\mathcal{G}$	U	Uniform distribution
g_i	Elements in $\mathcal{G}_i$	N	Number of samples
$\mathcal{L}_{CE}$	Classification loss	dim	Dimensions of the group
$\mathcal{L}_R$	Regularization loss	σ	Parameter of the uni-axial scaling
$\mathcal{L}_{var}$	Variance penalty loss	Σ_σ	Transformation matrix of uni-axial scaling

However, their method does not support invariance to affine transformations. MacDonald, Ramasinghe, and Lucey (2022) proposed a method based on group convolutions to achieve affine equivariance by sampling from the Haar measure of the group using the Metropolis Sampling method. However, as the depth of the network increases, the computational complexity of the evaluation function grows exponentially, significantly reducing the utility and efficiency of the model. Furthermore, these model sizes also scale with the group sizes, rendering them computationally expensive for the large affine group. Some alternative methods propose the use of steerable CNNs (Cohen & Welling, 2017; Shen, Hong, She, Ma, & Lin, 2022; Weiler & Cesa, 2019; Weiler, Geiger, Welling, Boomsma, & Cohen, 2018). These networks can decouple computational costs from group size by analytically computing invariant filters. However, it is important to note that these methods are primarily applicable to the Euclidean group $E(n)$ and its subgroups, which consist of rigid transformations. They cannot be readily applied to the more general affine group.

In general, we have observed that achieving complex invariance can be simplified by breaking it down into simpler components. In the context of our work, where we focus on achieving affine invariance, we recognize that an affine transformation can be decomposed into a combination of uni-axial scalings and rigid transformations. Consequently, we address affine invariance by separately handling the invariance of uni-axial scalings and rigid transformations. To ensure invariance to uni-axial scalings, we compute the model output averaged over uni-axial scaling augmentations. For achieving invariance to rigid transformations, we utilize Rigid transformation Invariant CNNs (RI-CNNs) (Shen et al., 2022; Weiler & Cesa, 2019) as the backbones, complemented by a self-supervision regularization, which nearly guarantees the invariance of the entire model over rigid transformations. Compared to methods such as Augerino (Benton et al., 2020) and Lila (Immer et al., 2022), which achieve invariance by averaging vanilla models over affine transformations, our approach is significantly more efficient on numbers of parameters, inference time and computation occupation. This efficiency arises from the fact that the former methods consider all affine augmentations primarily, whereas ours only needs sampling over the uni-axial scalings.

We also take into consideration the degree of invariance. Typically, rigid transformations do not change the label of an image. However, excessive scaling along a single axis can distort an image significantly, making it difficult to recognize. Therefore, it is vital to confine uni-axial scale invariance within a reasonable range to prevent unwarranted biases in neural networks that could lead to performance degradation. Taking a cue from Augerino (Benton et al., 2020), we employ an adaptive training approach to determine the acceptable range of uni-axial scale invariance. This adaptive process automatically excludes augmentations that result from excessive scaling. Consequently, we introduce the Scale-Adaptive Nearly Affine Invariant Network (SANAIN).

Our experiments serve as validation, confirming that our method effectively maintains affine invariance while outperforming counterparts like Augerino (Benton et al., 2020) and Lila (Immer et al., 2022), all with a substantially reduced computational load. Furthermore, our SANAINs significantly enhance the performance of RI-CNNs with a manageable increase in computational cost, thanks to their superior invariance incorporation. To evaluate the performance on the practical scenarios, we conduct experiments on the ImageNet-1k (Deng et al., 2009). Notably, our approach attains new state-of-the-art (SOTA) results in affNIST and SIM2MNIST classification tasks. The notations in this paper are in Table 1.

Our contributions are summarized as follows:

- To the best of our knowledge, this represents the first successful attempt at achieving intricate affine invariance at an affordable computational cost. We accomplish this by breaking down complex invariances into more manageable components. Thanks to this technique, the computational complexity of SANAIN is only $\mathcal{O}(N^2)$ in 2D and $\mathcal{O}(N^4)$ in 3D, significantly more efficient than the $\mathcal{O}(N^6)$ and $\mathcal{O}(N^{12})$ complexities of averaged models (Benton et al., 2020; Immer et al., 2022) in 2D and 3D, not to mention the staggering $\mathcal{O}(N^{6L})$ and $\mathcal{O}(N^{12L})$ complexity associated with MacDonald et al.'s method (MacDonald et al., 2022) in these two cases.
- We employ adaptive learning techniques to determine the range of uni-axial scale invariance based on data, allowing us to precisely define the extent of affine invariance.
- In our experimental evaluations, our models effectively preserve affine invariance, surpassing previous methods on four comprehensive benchmarks, spanning both 2D and 3D scenarios. In contrast to the approach of averaging model outputs with augmented data, as seen in Lila and Augerino, our models demand less than 15% of the processing time and significantly fewer computational resources. Notably, we attain new state-of-the-art results in affNIST and SIM2MNIST classifications.

2. Related work

2.1. RI-CNNs

In a general context, RI-CNNs are designed to introduce essential hard-coded rigid transformation invariance into CNNs without the need for explicit learning. Initially applied to 2D images, Cohen & Welling (Cohen & Welling, 2016) introduced the concept of group correlation to incorporate 4-fold rotation invariance into CNNs. Subsequent research endeavors (Hoogeboom, Peters, Cohen, & Welling, 2018; Marcos, Volpi, Komodakis, & Tuia, 2017; Shen, He, Lin, & Ma, 2020; Weiler, Hamprecht, & Storath, 2018; Zhou, Ye, Qiu, & Jiao, 2017) have aimed to leverage larger rotation groups for enhanced invariance. Notably, Worrall & Brostow (Worrall & Brostow, 2018) extended group correlations into the realm of 3D imaging, introducing CubeNet, an architecture invariant to transformations within the cube group. A similar approach has found applications in the field of medical image analysis (Winkels & Cohen, 2019).

The aforementioned approaches predominantly achieve invariance through the utilization of group correlations, where the model's size scales in accordance with the size of the transformation group. Consequently, these methods face practical limitations when applied to large or continuous groups. An alternative direction was explored by Worrall et al., who introduced H-Nets (Worrall, Garbin, Turmukhambetov, & Brostow, 2017). H-Nets are capable of producing invariant features concerning arbitrary 2D rotations. In an effort to disentangle computational costs from group sizes, Cohen and Welling introduced steerable CNNs (Cohen & Welling, 2017). These models represent feature spaces as feature fields, and invariant filters are derived by solving specific constraints. In the 2D domain, E2CNNs (Weiler & Cesa, 2019) and the work by Jenner (Jenner & Weiler, 2021) are among the most versatile steerable CNNs, as they can handle arbitrary feature fields of $E(2)$ and its subgroups. In the 3D realm, PDO-s3DCNNs (Shen et al.,

2022) and Cesa's research (Cesa, Lang, & Weiler, 2021) stand as the most comprehensive, as they can accommodate any subgroup of $E(3)$. However, it is important to note that these techniques are primarily geared towards addressing rigid transformations and are not equipped to handle more general affine transformations.

2.2. Affine invariant models

The most commonly employed technique for enforcing invariance is data augmentation, as demonstrated, for instance, in Krizhevsky, Sutskever, and Hinton (2017). The fundamental concept involves enriching the training dataset with transformed samples, including affine augmentations. Jaderberg et al. introduced spatial transformer networks (STN) (Jaderberg, Simonyan, Zisserman, et al., 2015), which utilize a differentiable module to actively transform feature maps. In Sabour, Frosst, and Hinton (2017) and Hinton, Sabour, and Frosst (2018), capsules are employed to represent location information and enforce invariance. Guo, Zhu, Liu, and Yin (2019) adopted an unsupervised approach to learn equivariant features, while Xu, Wang, Sullivan, and Zhang (2020) utilized a multi-scale maxout CNN (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013) to acquire affine-invariant representations. AffNet (Mishkin, Radenovic, & Matas, 2018) and ASLFeat Luo et al. (2020) specifically focused on learning local affine invariance for image matching tasks. In summary, these methods achieve invariance primarily through the learning process, which can impose a substantial computational burden on networks, particularly when dealing with a significantly large transformation group like the affine group.

To directly integrate invariance into models, Laptev et al. (2016) employed parallel Siamese architectures for the considered transformation group and applied an invariant pooling operator to their outputs to establish invariance. Building upon this concept, Benton et al. (2020) introduced Augerino, which can further adapt to the extent of invariance in data during training. Following Augerino (Benton et al., 2020), Immer et al. (2022) proposed Lila, a gradient-based method for automated data augmentation selection. However, their models' sizes scale proportionally with the group sizes, rendering them computationally demanding when applied to the affine group in practical scenarios. Esteves, Allenblanchette, Zhou and and Daniilidis (2018) introduced polar transformer networks (PTN), which leveraged STN to ensure translation invariance, subsequently utilizing group correlations to achieve invariance over rotations and dilations by transforming inputs into log-polar coordinates. Nevertheless, PTN essentially only achieves invariance over 2D similarity transformations and cannot accommodate other transformations within the affine group, such as uni-axial scalings. MacDonald et al. (2022) introduced a module designed to maintain equivariance for arbitrary Lie groups. However, as network layers increase, the computational complexity of the evaluation function grows exponentially, significantly reducing the model's practicality and efficiency.

3. Preliminaries

3.1. Affine invariance

The affine group is a relatively large group. It represents affine transformations and encompasses translations and all invertible linear transformations, such as rotations, scalings, shears, and more. It is typically denoted as $\mathrm{Aff}(n)$ when defined over $\mathbb{R}^n$. Formally, for a given vector $x \in \mathbb{R}^n$, an affine transformation g operates on x as follows:

$$gx = Ax + t, \tag{1}$$

where the linear transformation is represented by $A \in \mathrm{GL}(n, \mathbb{R})$, and the translation transformation by $t \in \mathbb{R}^n$. Here, $\mathrm{GL}(n, \mathbb{R})$ denotes the general linear transformation group, encompassing all invertible linear transformations expressible as invertible real matrices of size $n \times n$.

Table 2
This table illustrates the dimensions of different transformation groups, emphasizing the substantial size difference between the affine group and the others.

	$US(n)$	$IS(n)$	$SO(n)$	$T(n)$	$E(n)$	$\mathrm{Aff}(n)$
$n = 2$	1	1	1	2	3	6
$n = 3$	1	1	3	3	6	12

When A is constrained to belong to the orthogonal transformation group $O(n)$, meaning $A^T A = I_n$, the affine group simplifies to the rigid transformation group, also known as the Euclidean group $E(n)$. Specifically, the affine group includes several subgroups, such as the rigid transformation group $E(n)$, the special orthogonal transformation/rotation group $SO(n)$, the translation group $T(n)$, the isotropic scaling group $IS(n)$ and the uni-axial scaling group $US(n)$. The uni-axial scaling group performs scaling along a single axis. In Table 2, we provide the dimensions of these groups for ease of comparison. Notably, the affine group is considerably larger than its subgroups, especially as the dimensions increases, which results in significantly higher computational costs when dealing with it.

In general, the invariance of a model reflects its robustness to transformations. Essentially, a model is considered invariant to a transformation set $\mathcal{G}$ if its outputs remain unchanged when a transformation is applied to the inputs. Formally, for a model Ψ to be considered invariant, it should satisfy the condition that for all transformations g within the set $\mathcal{G}$, the following holds:

$$\Psi[\pi_g[f]] = \Psi[f], \tag{2}$$

where $f : \mathbb{R}^n \to \mathbb{R}^K$ represents the input data function, K is the number of channels, and π_g describes how the transformation g acts on the inputs, as given by the equation:

$$\pi_g[f](x) = f(g^{-1}x), \tag{3}$$

where $x \in \mathbb{R}^n$ is the coordinate. Ψ is deemed affine invariant if this condition, as described above, holds when the transformation set $\mathcal{G}$ comprises affine transformations, denoted as $\mathrm{Aff}(n)$.

3.2. Averaged models for invariance

In order to construct a model that exhibits invariance over a defined set of transformations, aggregating the outcomes of a model over inputs transformed by every operation within the transformation set is a readily available and user-friendly approach. To the best of our knowledge, Augerino (Benton et al., 2020) and Lila (Immer et al., 2022) represent notable examples of employing data augmentation to learn invariance.

The Augerino model (Benton et al., 2020) achieves invariance over a transformation set through augmentation. This approach involves sampling multiple augmentations from a distribution and subsequently applying these augmentations to an input, resulting in several augmented input samples. These augmented samples are then processed through the model, and the final prediction is derived by averaging the model outputs.

Formally, we consider working with a transformation set S. Given a neural network Ψ, we can create a new model $\bar{\Psi}$, which approximates invariance to transformations in S by averaging the outputs using a parameterized distribution μ_θ over the transformations $g \in S$. In other words,

$$\bar{\Psi}[f] = \mathbb{E}_{g \sim \mu_\theta} \Psi[\pi_g[f]]. \tag{4}$$

Specifically, the set of affine transformations constitutes an algebraic structure known as a Lie Group. For 2D affine transformations, we can parameterize the distribution $g \sim \mu_\theta$ as follows:

$$g_\epsilon = \exp\left(\sum_{i=1}^{6} \epsilon_i \theta_i G_i\right), \tag{5}$$

where exp represents the matrix exponential function, $\epsilon_i \sim U[-1,1]$, and $U[-1,1]$ signifies the uniform distribution within the range of -1 to 1. The terms $G_1, \dots, G_6$ correspond to translation in the x-direction, translation in the y-direction, rotation, scaling in the x-direction, scaling in the y-direction, and shearing, respectively. The θ_i values denote the learnable bounds of a uniform distribution across the various exponential generators G_i within the Lie Algebra.

In practice, the implementation of Eq. (4) involves approximating it by calculating the model output $\Psi[\pi_g[f]]$ averaged over a finite number of samples from $g \sim \mu_\theta$, using a Monte Carlo estimator. For the 2D affine group, we consider taking N samples for each $\epsilon_i \in U[-1,1]$. This would require generating N^6 transformed augmentations, resulting in significant storage and computational overhead, especially when N is relatively large. Consequently, Benton et al. (2020) found success primarily with the 2D rotation group in practical scenarios. This is because the 2D rotation group has just one dimension, and hence, only requires N samples. However, when applied to more complex groups, its performance significantly degrades. Moreover, when attempting to employ this method with larger transformation groups, such as the 3D affine group, which has a dimension of 12, the computational complexity becomes prohibitively high at $\mathcal{O}(N^{12})$.

Lila (Immer et al., 2022) represents an improved iteration of Augerino (Benton et al., 2020). It approaches augmentation as an invariance issue within the priors of neural network functions, employing Bayesian model selection techniques. They optimize the objective using a differentiable Kronecker-factored Laplace approximation, eliminating the need for manual intervention or validation. However, similar to Augerino (Benton et al., 2020), achieving invariance over a significantly larger transformation group remains highly inefficient due to the substantial computational demands involved.

4. Affine invariant neural networks

4.1. Overview

Acknowledging the current limitations of existing techniques in efficiently incorporating affine invariance into models without imposing extensive computational overhead, our objective is to provide a viable solution. We recognize that the intricate invariance of complex transformations can be effectively achieved by decomposing it into more manageable components with simpler group invariance. Therefore, our primary focus in this endeavor is to introduce a practical approach that involves breaking down the complex affine invariance, offering a method to tackle this challenge.

As detailed in Section 4.2, we contend that an affine transformation can be dissected into a composition of uni-axial scalings and rigid transformations. Consequently, we address these two types of invariances individually to attain affine invariance. An overview of architecture of SANAIN is presented in Fig. 1. Specifically, for uni-axial scale invariance, we sample multiple uni-axial scale augmentations of an input from a learnable distribution and subsequently input them into a parallel Siamese network, with their outputs being averaged. For the other form of invariance, we employ RI-CNNs as the backbones of the parallel Siamese network, and the invariance of the entire network over rigid transformations is nearly ensured through self-supervised regularization. During training, our network is supervised by three loss terms: a cross-entropy loss for classification, a regularization loss to encourage the broadness of the learnable augmentation distribution, and a variance loss to reduce output variance while enforcing rigid transformation invariance across the entire network in a self-supervised manner. Additionally, the extent of uni-axial scaling can be adaptively adjusted. Subsequent sections delve into the design principles and the major components of our framework in detail.

4.2. Decomposition of the affine invariance

We observe that complex invariance can be obtained by decomposing it into simpler components. We denote a complex group as $\mathcal{G}$. This group $\mathcal{G}$ can be decomposed into several subgroups $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n$. This implies that any element g in $\mathcal{G}$ can be represented as:

$$g = g_1 g_2 \cdots g_m, \tag{6}$$

where for any i, there exists a corresponding j such that $g_i \in \mathcal{G}_j$.

To attain $\mathcal{G}$ invariance, we can deconstruct it into the individual invariances of $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n$. Suppose there exists a model Ψ that maintains invariance over $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n$, which implies:

$$\forall g_i \in \mathcal{G}_i, \quad \Psi[\pi_{g_i}[f]] = \Psi[f] \tag{7}$$

Then, for all elements $g \in \mathcal{G}$, we have:

$$\begin{aligned}\Psi[\pi_g[f]] &= \Psi[\pi_{g_1 g_2 \cdots g_m}[f]] \\ &= \Psi[\pi_{g_1}\pi_{g_2} \cdots \pi_{g_m}[f]] \\ &= \Psi[\pi_{g_2} \cdots \pi_{g_m}[f]] \\ &\cdots \\ &= \Psi[f].\end{aligned} \tag{8}$$

Thus, achieving $\mathcal{G}$ invariance is realized through the invariance of $\mathcal{G}_1, \dots, \mathcal{G}_n$.

Intuitively, when considering the complex group $\mathcal{G}$ and its components $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n$, if the following inequality holds:

$$\dim(\mathcal{G}_1) + \dim(\mathcal{G}_2) + \cdots + \dim(\mathcal{G}_n) < \dim(\mathcal{G}), \tag{9}$$

it becomes more efficient to address the individual components rather than dealing with the larger group $\mathcal{G}$ as a whole.

Taking the affine group as an example, we firstly leverage some mathematical tools to simplify it. Formally, we have the following conclusion.

Proposition 1. *An affine transformation can be decomposed as a composition of rigid transformations and uni-axial scalings.*

The proof of Proposition 1 is presented in Appendix. According to Proposition 1, for any affine transformation $g \in \mathrm{Aff}(n)$, it can be decomposed as

$$g = e_1 s_1 e_2 s_2 \cdots e_m s_m, \tag{10}$$

where $e_i \in E(n)$ and $s_i \in US(n)$. Suppose that a model Ψ is invariant over both $E(n)$ and $US(n)$, i.e.,

$$\forall e \in E(n), s \in US(n), \quad \Psi[\pi_e[f]] = \Psi[\pi_s[f]] = \Psi[f], \tag{11}$$

it is easy to deduce that

$$\Psi[\pi_g[f]] = \Psi[\pi_{e_1 s_1 e_2 s_2 \cdots e_m s_m}[f]] = \Psi[f], \tag{12}$$

i.e., this model is naturally invariant over affine transformations. As a result, we can separately address uni-axial scale and rigid transformation invariances, which are considerably more manageable, ultimately achieving affine invariance.

Some prior works (Esteves, Allenblanchette, Zhou & and Daniilidis, 2018; Xu et al., 2020) have also devised invariant neural networks by handling certain basic invariances independently, such as translations, dilations, and rotations. However, they primarily handle isotropic scalings, where all axes are scaled uniformly, rather than uni-axial scalings. Specifically, they decompose the 2D similarity transformations group SIM(2) into translation group $T(2)$, rotation group $SO(2)$, and the isotropic scaling group $IS(2)$. This decomposition results in:

$$\dim(\mathrm{SIM}(2)) = \dim(T(2)) + \dim(SO(2)) + \dim(IS(2)) = 2 + 1 + 1 = 4. \tag{13}$$

Hence, this decomposition does not significantly improve the efficiency of calculating invariance for SIM(2). Additionally, their approaches are restricted to the 2D similarity transformations group SIM(2), rather than addressing the entire affine group, which distinguishes them from our approach.

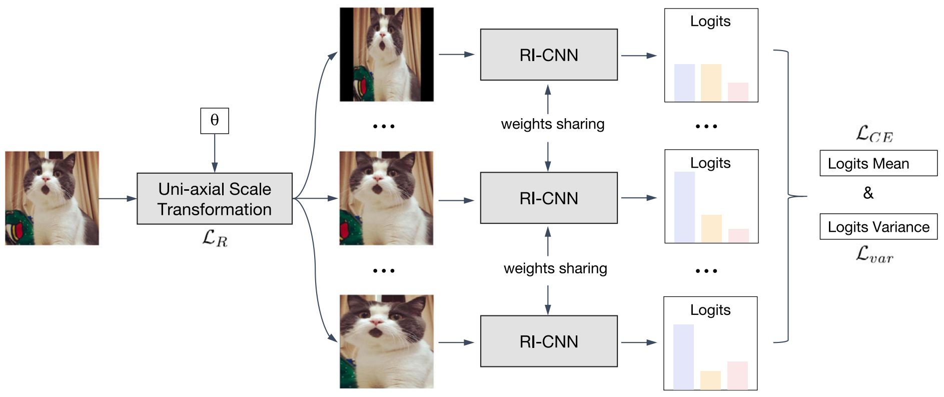


Fig. 1. The architecture of our SANAIN model. Firstly, multiple augmentations are sampled from a learnable distribution over the uni-axial scale transformation group and applied to an input, resulting in multiple augmented inputs. These augmented inputs then pass through RI-CNNs with shared weights. The final prediction is obtained by averaging over the multiple outputs. Our model is supervised by three loss terms: a regularization loss $\mathcal{L}_R$ to encourage the diversity of the augmentation distribution, a cross-entropy loss $\mathcal{L}_{CE}$ for classification, and a variance loss $\mathcal{L}_{var}$ to enforce rigid transformation invariance across the entire network by reducing output variance.

4.3. SANAIN

Now, our initial consideration pertains to the degree of affine invariance, which will guide our approach to handling rigid transformations and uni-axial scale invariance separately.

For uni-axial scale transformations, we employ the concept of an average model, denoted as follows:

$$\bar{\Psi}[f] = \mathbb{E}_s \Psi[\pi_s[f]], \tag{14}$$

where s denotes the uni-axial scale transformation. It is evident from Eq. (14) that $\bar{\Psi}[f]$ remains invariant to uni-axial scale transformations.

When addressing rigid transformations such as rotations and translations, they inherently do not alter the label of an image. However, when in the case of uni-axial scalings, if an image undergoes an extreme scaling along a single axis, it can become severely distorted and may no longer be recognizable to humans. Consequently, it is essential to constrain the extent of uni-axial scale invariance within a reasonable range. Excessive invariance, as mentioned earlier, should be avoided because it can introduce unexpected biases into the model, ultimately degrading its performance.

In accordance with the approach introduced by Augerino (Benton et al., 2020), our methodology addresses the aforementioned challenge through adaptive learning of uni-axial scaling extent. To be specific, we parameterize uni-axial scaling as follows:

$$\Sigma_\sigma = \begin{bmatrix} \sigma & 0 \\ 0 & 1 \end{bmatrix}, \tag{15}$$

where σ represents the scaling factor applied to the first dimension of the input. To constrain the scale factor σ within a reasonable range for maintaining uni-axial scale invariance, we limit σ to the interval $[\exp(-\theta), \exp(\theta)]$. To ensure the parameter θ remains positive, we introduce a learned parameter $\tilde{\theta}$, where θ is defined as $\theta = \log(1 + \exp(\tilde{\theta}))$. Subsequently, we reparameterize the scale as $\sigma = \exp(\epsilon\theta)$, where $\epsilon \sim U[-1, 1]$. Consequently, our adaptation of Augerino for uni-axial scale invariance can be represented as follows:

$$\bar{\Psi}[f] = \mathbb{E}_s \Psi[\pi_s[f]] = \mathbb{E}_{\epsilon \sim U[-1,1]} \Psi[\pi_{\Sigma_{\exp(\epsilon\theta)}}[f]], \tag{16}$$

where Ψ signifies the backbone models with shared weights.

In practice, it is essential to recognize that a less constrained model often achieves a lower training loss when employing the cross-entropy loss $\mathcal{L}_{CE}$ for classification. Interestingly, this outcome contradicts the prior expectation that a model should inherently maintain a certain degree of invariance. To address this concern and steer the training process towards solutions that inherently incorporate invariance, we draw inspiration from Benton et al. (2020) and introduce a regularization penalty into the network loss function. This penalty is designed to encourage broader distributions over augmentations, and it can be formally expressed as:

$$\mathcal{L}_R = -\|\theta\|_2. \tag{17}$$

To tackle the challenge of achieving rigid transformation invariance, we adopt RI-CNNs as the backbone networks Ψ in the formulation presented in Eq. (16), which have been extensively explored in prior research, with notable contributions from studies such as Cesa et al. (2021), Jenner and Weiler (2021), Shen et al. (2022), Weiler and Cesa (2019).

RI-CNNs are particularly noteworthy for their capability to attain $E(n)$-invariance, a property that greatly contributes to robust affine invariance. However, empirical findings from studies like (Shen et al., 2022; Weiler & Cesa, 2019) suggest that models invariant to discrete subgroups may exhibit superior performance, despite achieving only approximate invariance over $E(n)$. As a result, we adopt different RI-CNNs as needed to implement the desired level of invariance in our approach.

We now delve into the investigation of the model Ψ for its rigid transformation invariance. We assume that the inputs f follow a rotation-invariant distribution, which leads to the following analysis:

$$\begin{aligned}
&\mathbb{E}_f \mathbb{E}_e |\bar{\Psi}[f] - \bar{\Psi}[\pi_e[f]]| \\
=&\mathbb{E}_f \mathbb{E}_e |\bar{\Psi}[f] - \Psi[f] + \Psi[\pi_e[f]] - \bar{\Psi}[\pi_e[f]]| \\
\le&\mathbb{E}_f |\bar{\Psi}[f] - \Psi[f]| + \mathbb{E}_f \mathbb{E}_e |\Psi[\pi_e[f]] - \bar{\Psi}[\pi_e[f]]| \\
=&\mathbb{E}_f |\mathbb{E}_s \Psi[\pi_s[f]] - \Psi[f]| + \mathbb{E}_f \mathbb{E}_e |\Psi[\pi_e[f]] - \mathbb{E}_s \Psi[\pi_s \pi_e[f]]| \\
=&2\mathbb{E}_f |\Psi[f] - \mathbb{E}_s \Psi[\pi_s[f]]|.
\end{aligned}$$

Employing Chebyshev's inequality, we find that for any $\delta > 0$:

$$Pr(|\Psi[f] - \mathbb{E}_s \Psi[\pi_s[f]]| > \delta) < \frac{Var_s(\Psi[\pi_s[f]])}{\delta^2}, \tag{18}$$

where $Var(\cdot)$ represents the average of the pointwise variance of output features. Therefore, enforcing rigid transformation invariance can be achieved by driving the above variance towards zero during training. This objective can be realized by incorporating a penalty loss term $\mathcal{L}_{var}$. Ultimately, the loss function takes the form:

$$\begin{aligned}
Loss =&\mathcal{L}_{CE} + \lambda \mathcal{L}_R + \mu \mathcal{L}_{var} \\
=&l(\mathbb{E}_s \Psi[\pi_s[f]]) - \lambda \|\theta\|_2 + \mu Var_s \Psi[\pi_s[f]],
\end{aligned} \tag{19}$$

where l represents the cross-entropy loss utilized for classification, and λ and μ are associated weights, which are set to 0.1 and 0.001, respectively. During training, multiple samples are drawn from the learnable augmentation distribution to estimate expectations and variance in Eq. (19), in our experiments. During testing, we sample multiple transformations from the discovered distribution and make predictions by averaging over the predictions generated by the transformed inputs, approximating Eq. (16).

Table 3
Computational complexity of **testing** comparison for achieving affine invariance between output-averaged-based Lila, group correlations-based methods proposed by MacDonald et al. and SANAIN. In this table, N represents the number of samples for each degree of freedom.

	Aff(2)	Aff(3)
Augerino (Benton et al., 2020)	$\mathcal{O}(N^6)$	$\mathcal{O}(N^{12})$
Lila (Immer et al., 2022)	$\mathcal{O}(N^6)$	$\mathcal{O}(N^{12})$
MacDonald et al. (2022)	$\mathcal{O}(N^{6L})$	$\mathcal{O}(N^{12L})$
SANAIN (ours)	$\leq \mathcal{O}(N^2)$	$\leq \mathcal{O}(N^4)$

4.4. Computational complexity analysis

In essence, an RI-CNN, implemented using group correlations, employs one kernel for each transformation considered, except for translations, since a CNN naturally exhibits translation equivariance. In the case of $E(2)$ with one additional degree of freedom, if we sample N transformations for this degree of freedom, we require N kernels. Consequently, SANAIN, which is invariant over Aff(2), exhibits a computational complexity of only $\mathcal{O}(N^2)$, with N kernels for RI-CNNs and N for uni-axial scale copies. Notably, when it comes to the 3D affine group, the computational complexity of SANAIN is $O(N^4)$, with N^3 dedicated to RI-CNNs and N for copies. It is important to highlight that our computational cost can be further reduced by employing steerable CNNs (Shen et al., 2022; Weiler & Cesa, 2019) as the backbones, as they are more efficient than group correlations.

As discussed in Section 2.2, while there are numerous models robust to affine transformations, most existing methods do not exhibit true invariance to such transformations, such as PTN (Esteves, Allen-blanchette, Zhou & and Daniilidis, 2018) and STN (Jaderberg et al., 2015). Therefore, we only compare the performance of these models with our method in practical experiments and do not compare the computational complexity here.

There are two main categories of methods for achieving invariance to transformation groups in existing implementations. The first type relies on averaged models, where the approach involves averaging the model outputs for data augmentation to achieve invariance to affine transformations. Models relying on averaged outputs, such as Augerino (Benton et al., 2020) and Lila (Immer et al., 2022), exhibit much higher computational complexity. For 2D affine invariance, the testing-stage model complexity is $\mathcal{O}(N^6)$. In the case of 3D, the testing-stage model's sampling complexity increases to $\mathcal{O}(N^{12})$.

The second mainstream approach is the design of neural networks based on group correlation. To the best of our knowledge, currently, only the method proposed by MacDonald et al. (2022) achieves invariance to affine transformations through group correlation. They attempted to achieve affine invariance using group correlations by sampling from the Haar measure from the Lie algebra. Nevertheless, because the discretized set for estimating the integral of group correlation does not form a group, elements sampled for calculating the group correlation may fall outside the set, which brings substantial sampling. Therefore, as the number of layers increases, combinatorial explosion occurs in the sampling process, resulting in computational complexities of $\mathcal{O}(N^{6L})$ for 2D and $\mathcal{O}(N^{12L})$ for 3D, where L represents the number of layers in the neural network. Consequently, their method is primarily suitable for shallow networks and faces challenges when applied to more complex tasks. A summary of the computational complexity comparison is presented in Table 3.

5. Experiments

5.1. Testing affine invariance

We evaluate the affine invariance of our approach using the MNIST dataset, a commonly used dataset of handwritten digits with 28 × 28 pixels, drawn from 10 classes. The training and test sets contain 60,000 and 10,000 images, respectively. We set a fixed uni-scale transformation range for data augmentation and train our model using a combination of cross-entropy loss and variance loss. This training approach aims to generate distinctive invariant features, preventing model degradation. For the analysis of invariance errors, we randomly sample $n = 10{,}000$ affine transformations g_i and calculate the invariance error using the following formula:

$$\text{Error} = \frac{1}{n}\sum_{i=1}^{n} \frac{\|\bar{\Psi}\left[\pi_{g_i}[I_i]\right] - \bar{\Psi}[I_i]\|_2}{\|\bar{\Psi}[I_i]\|_2}, \tag{20}$$

where I_i represents the test samples, and $\bar{\Psi}$ is defined in Eq. (16), with a predefined range of uni-axial scaling between 0.8 and 1.25. This range ensures that transformed digits remain within the image boundaries and remain recognizable to humans. Our choice of backbone networks Ψ was $SO(2)$-steerable CNNs, which exhibit exact invariance to arbitrary 2D rotations.

Our model is trained for 10 epochs, and thereafter, we assess its affine invariance performance. As depicted in Fig. 2(a), our model nearly achieves affine invariance, with an invariance error of less than 0.1 in most cases. Interestingly, as the network depth increases, the level of invariance improves. This observation aligns with the intuition that a larger network capacity aids in better invariance learning during training. It is important to note that using only 2 augmentations provides satisfactory affine invariance, leading us to employ 2 copies in subsequent experiments to strike a balance between performance and computational cost. Next, we conduct a comparative analysis of our approach against its counterparts, specifically, averaged models designed for achieving affine invariance. We maintain uniformity in backbone sizes and training details to ensure a fair comparison. Setting the number of augmentations to 4, we find that these models fail to attain the same level of invariance as our method, which utilizes only 2 augmentations. This discrepancy can be attributed to the considerably larger affine group, making it challenging for Eq. (4) to provide accurate approximations with a limited number of samples. In an attempt to enhance their performance, we increase the number of augmentations to 8 and 12, albeit at a significantly higher computational cost. Nevertheless, even with these additional augmentations, their affine invariance still falls short of ours, validating the effectiveness and efficiency of our proposed method.

Notably, the quality of affine invariance degrades considerably when $\mathcal{L}_{var}$ is omitted (as shown in Fig. 2(b) and (c)). In this scenario, rotation invariance cannot be guaranteed, highlighting the essential role and effectiveness of incorporating $\mathcal{L}_{var}$ in our approach.

5.2. MNIST variants

The AffNIST dataset is created by applying various reasonable affine transformations to images from the MNIST dataset. The resulting images are resized to 40 × 40 pixels while preserving the original MNIST content. Specifically, the vertical and horizontal expansion scales are uniformly sampled from the range of 0.8 to 1.2. The dataset is divided into training (50k samples), validation (10k samples), and test (10k samples) sets.

For this dataset, we utilize $SO(2)$-steerable CNNs (Weiler & Cesa, 2019) with 7 layers as the backbone for our SANAIN model. To address uni-axial scale invariance, we employ 2 augmentations. Hyperparameters are selected based on achieving the lowest validation error during training. Our models are trained using the Adam optimizer (Kingma & Ba, 2015) for a total of 150 epochs, with a batch size of 64 and a weight decay of 0.0001. The initial learning rate is set to 0.001 and is decayed by a factor of 0.7 every 10 epochs.

Several approaches (Jin, Lazarow, & Tu, 2017; Lee, Xu, Fan, & Tu, 2018) have attempted to address the challenges posed by this dataset through augmenting the original training data to enhance its affine

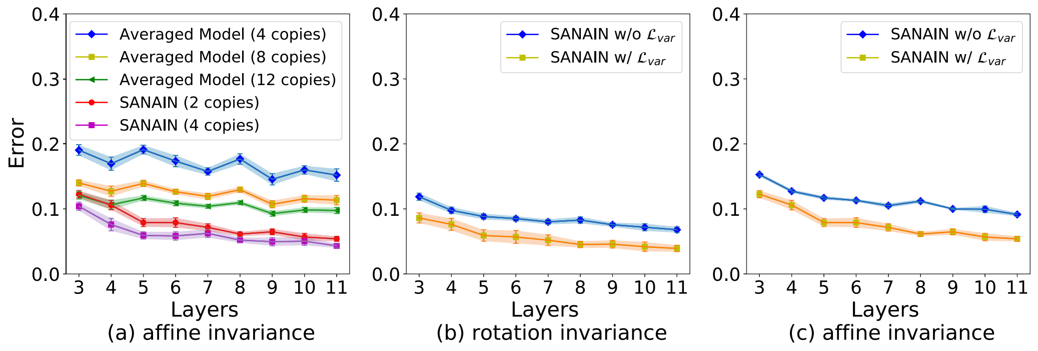


Fig. 2. (a) illustrates the affine invariance error comparison between the affine invariant averaged model and our SANAIN. Our method effectively preserves affine invariance with significantly fewer copies compared to the averaged model. Panels (b) and (c) display the rotation invariance error and affine invariance error of SANAIN, both with and without $\mathcal{L}_{var}$.

Table 4
Test error rates on affNIST.

Method	Test err. (%)
Macdonald et al.(w/ DA) (MacDonald et al., 2022)	2.15
CNN (w/ DA)	1.65
ICN (w/ DA) (Jin et al., 2017)	1.54
WINN (w/ DA) (Lee et al., 2018)	1.48
ITN (w/ DA) (Zhao, Tian, Fowlkes, Shen, & Yuille, 2020)	1.09
$SO(2)$-steerable CNN (Weiler & Cesa, 2019)	1.01
(Xu et al., 2020)	0.92
Augerino (Benton et al., 2020)	0.81
Lila (Immer et al., 2022)	0.72
SANAIN (ours)	**0.56**

Table 5
Test error rates on SIM2MNIST.

Method	Test err. (%)	Params
Polar CNN (Esteves, Allenblanchette, Zhou & and Daniilidis, 2018)	15.46	129k
CNN	11.73	129k
STN (Jaderberg et al., 2015)	12.35	150k
Augerino (Benton et al., 2020)	10.21	2.35M
H-Net (Worrall et al., 2017)	9.28	44k
Lila (Immer et al., 2022)	8.54	2.35M
PTN (Esteves, Allenblanchette, Zhou & and Daniilidis, 2018)	5.03	134k
SANAIN (ours)	**3.98**	132k

robustness. However, when compared to these DA-based methods, the $SO(2)$-steerable approach exhibits superior performance, underscoring the advantage of incorporating hard-coded invariance into networks rather than relying solely on data augmentation. MacDonald et al. (2022) employed improved group correlations to achieve affine invariance. Nevertheless, they can only employ very shallow network, which results in very poor performance. Also, their framework lacks the capacity to adaptively learn the appropriate degree of invariance. Xu et al. (2020) tackled scalings and rotations jointly in CNNs to learn affine invariance, achieving a test error rate of 0.92%. However, it is important to note that their method primarily addresses similarity transformations. Augerino (Benton et al., 2020) achieves a test error rate of 0.81%. Lila serves as a compelling alternative to Augerino and has demonstrated performance improvements, achieving a test error rate of 0.72%. In our experiments, we evaluate the performance of Augerino and Lila with a comprehensive set of 31 augmentations. To ensure a fair comparison, the backbone architecture used for Augerino and Lila aligns with the backbone employed in SANAIN in terms of channel count.

As illustrated in Table 4, our SANAIN achieves an impressive test error rate of 0.56% on the affNIST dataset, signifying a remarkable performance improvement. Notably, this outperformance is achieved with just two copies, a notably lower number when compared to 31 copies employed by vanilla Augerino (0.81%) and Lila (0.72%). It is worth highlighting that our model learns a uni-axial scale range of $[0.79, 1.26]$, which closely matches the actual scale range of $[0.8, 1.2]$ within which the original images are generated.

SIM2MNIST, a more challenging variant of the MNIST dataset introduced in Esteves, Allenblanchette, Zhou and and Daniilidis (2018), incorporates random transformations from the 2D similarity transformations group SIM(2). This group encompasses translations, dilations, and rotations, making the task notably complex. The images in this dataset are larger at 96 × 96 pixels, featuring arbitrary 2D rotations. Scale factors vary between 1 and 2.4, and the digits can appear anywhere within the images. The dataset is partitioned into training (10k samples), validation (5k samples), and test (50k samples) sets.

For this demanding task, we leverage a more expressive C_8-steerable CNNs architecture, comprising 10 layers. Remarkably, given that this dataset primarily involves similarity transformations, we adaptively focus on learning dilation invariance rather than uni-axial scaling invariance. This adaptation is achieved by modifying the uni-axial scaling operations in Eq. (16) to accommodate dilations. In our experiments on SIM2MNIST, we utilize 2 augmentations to address dilation invariance. We meticulously tune our hyperparameters to minimize validation error, ensuring consistency with the settings used in the affNIST experiments.

The results for SIM2MNIST are presented in Table 5, where we observe notable distinctions between various methods. STN (Jaderberg et al., 2015) exhibits subpar performance due to the inherent difficulty in achieving complete affine invariance through the learning of a differential module. H-Net (Worrall et al., 2017), while invariant over rotations, lacks invariance over dilations, rendering it unsuitable for this dataset. Approaches employing output averaging, such as Augerino (Benton et al., 2020) and Lila (Immer et al., 2022), attain only modest accuracy at 10.21% and 8.54%, respectively, when utilizing 31 copies for transformation invariance. PTN (Esteves, Allenblanchette, Zhou & and Daniilidis, 2018) projects images into log-polar coordinates, achieving similar transformation invariance but is impeded by an implicit oversampling near the origins. This leads to a degradation in model performance, with polar CNNs performing worse than standard CNNs. Moreover, their methods theoretically maintain invariance over arbitrary dilations without imposed bounds. However, excessive invariance can introduce unintended biases into the network, negatively impacting performance.

In contrast, our SANAIN method dynamically constrains the dilation range to a sensible interval, specifically $[0.67, 1.48]$, with a maximum scale factor of 2.21, closely aligning with the actual scale factor of 2.4. This adaptive approach yields superior results, with SANAIN surpassing PTN (Esteves, Allenblanchette, Zhou & and Daniilidis, 2018) (3.98% vs. 5.03%) and establishing a new state-of-the-art performance.

Table 6
Test error rates on CIFAR-10 without data augmentation. Time per batch and GPU memory occupation are measured during inference.

Method	Test err. (%)	Params	Time (ms)	Memory (MB)
WideResNet	14.83	2.7M	–	–
Augerino (Benton et al., 2020)	12.33	2.7M	52	10,517
Lila (Immer et al., 2022)	8.02	2.7M	52	10,517
SANAIN	**6.72**	**1.2M**	**25**	**3,893**

Table 7
Test error rates on ImageNet-1k.

Method	Top-1 err. (%)	Top-5 err. (%)	Params
ResNet (He, Zhang, Ren, & Sun, 2016)	23.91	6.99	25M
Augerino (Benton et al., 2020)	23.54	6.68	25M
C_8-ResNet	21.96	5.88	27M
SANAIN (ours)	**20.44**	**5.15**	22M

In terms of efficiency, SANAIN demonstrates remarkable advantages. It processes each batch in just 9 ms and consumes 2407MB of GPU memory. In contrast, both Lila and Augerino require 63 ms and a substantial 5814MB of GPU memory for the same inference tasks.

5.3. Natural image classification

In the context of natural images, affine transformations often occur due to varying camera positions and distances. To evaluate our method and stress its practical value, we perform comparisons with its counterparts, Augerino and Lila, using the CIFAR-10 dataset (Krizhevsky et al., 2017) and ImageNet-1k (Deng et al., 2009).

CIFAR-10, comprises colored natural images, each measuring 32 × 32 pixels, drawn from 10 distinct classes. The dataset consists of 50,000 training images, 10,000 test images, and an additional 5000 training images for validation.

For CIFAR-10, Augerino and Lila employed WideResNet as their backbones in previous work (Immer et al., 2022). In our approach, we replace the standard convolutional layers with C_8-steerable layers, and then the C_8-WideResNet serves as the backbone for SANAIN. Despite increasing the number of channels in the C_8-WideResNet compared to the standard WideResNet, our models still maintain lower parameter counts and demonstrate improved efficiency. We employ 2 copies for uni-axial scale invariance. Our model is trained using stochastic gradient descent (SGD) with a Nesterov momentum (Sutskever, Martens, Dahl, & Hinton, 2013) of 0.9 without dampening. The training process runs for 200 epochs with a batch size of 128 and a weight decay of 0.0005. The initial learning rate is set to 0.1 and decays by a factor of 0.2 every 70 epochs.

The results in Table 6 demonstrate that SANAIN achieves the most favorable test error rate (6.72%) compared to Augerino (12.33%) and Lila (8.02%), all while using significantly fewer parameters and copies. Compared with the WideResNet, SANAIN achieves a remarkable improvement with an acceptable computational cost of only 2 copies.

Additionally, to evaluate the efficiency of our image classification, we measure the inference time and GPU memory usage. Notably, SANAIN requires less than half the inference time and GPU memory compared to the other two averaged models, while achieving significantly lower test errors.

ImageNet-1k, is the most commonly used subset of ImageNet (Deng et al., 2009). This dataset spans 1000 object classes and contains 1,281,167 training images and 50,000 validation images. We evaluate the model performance on the validation images.

We employ ResNet-50 as the base model. The vanilla Augerino is implemented by utilizing 2 affine augmentations with ResNet-50 as the subnetworks. C_8-ResNet is implemented by replacing the convolutional layers of ResNet-50 by C_8-steerable ones (Weiler & Cesa, 2019). Our SANAIN is implemented by employing C_8-ResNet as subnetworks with 2 uni-axial scale augmentations. We train all models using SGD and a Nesterov momentum (Sutskever et al., 2013) of 0.9 without dampening, with an initial learning rate of 0.1. The models are trained for 100 epochs and the learning rate is divided by 10 at 30, 60 and 90 epochs. The batch size is set to 256. The weight factors λ and μ are 0.1 and 0.001, respectively.

As shown in Table 7, our SANAIN outperforms affine invariant Augerino (2 copies) and C_8-ResNet with fewer parameters, because of better affine invariance. For Lila (Immer et al., 2022), the computational complexity and memory complexity during training are $\mathcal{O}(NPC + P^{1.5})$ and $\mathcal{O}(MPC)$, respectively (Immer et al., 2022), where P represents the number of parameters, M denotes the batch size, N signifies the data points and C indicates the neural network outputs. The intractable costs make it challenging to be evaluated on large and practical datasets, such as ImageNet datasets. In contrast, our method can be evaluated on ImageNet. Regarding MacDonald et al.'s approach (MacDonald et al., 2022), as discussed in Section 4.4, the extensive sampling complicates the construction of a deep neural network, let alone testing it in practical scenarios, where large models are required to address large datasets.

Accordingly, the above results indicate that incorporating affine invariance into networks, as SANAIN does, is helpful for natural image classification.

5.4. 3D shape retrieval

We also assess the effectiveness of SANAIN in the SHREC'17 retrieval task (Savva et al., 2017), a complex 3D vision task involving transformed shapes. The retrieval task features 51,162 models of 3D shapes spanning 55 classes. This dataset is partitioned into 35,764 training samples, 5133 validation samples, and 10,265 test samples. Our focus is on the "perturbed" version of the dataset, where models undergo various transformations. The retrieval performance is quantified using the average mean average precision (mAP), measured for both the micro-average and macro-average versions, collectively referred to as the "score".

We adopt $\mathcal{O}$-steerable CNNs as used in Shen et al. (2022) for our backbone. These models exhibit invariance over the cubic group, and we incorporate 2 copies for uni-axial scale invariance. Our model is trained using the Adam optimizer (Kingma & Ba, 2015). Training extends for 2000 epochs with a batch size of 32. The initial learning rate is set to 0.01 and is reduced by a factor of 10 at 700 and 1,400 epochs.

Many existing methods (Banerjee et al., 2020; Cobb et al., 2020; Esteves, Allen-Blanchette, Makadia, & Daniilidis, 2018; Li et al., 2021; Weiler, Geiger, et al., 2018) tackle this dataset by employing 3D rotation invariant models. However, these models do not account for general affine transformations, which often stem from variations in sampling poses or data itself. The results presented in Table 8 reveal a significant performance boost (from 58.6% to 60.9%) upon incorporating affine invariance into the networks through our SANAIN method. This outcome underscores the efficacy of applying SANAIN in the context of 3D vision tasks.

5.5. Efficiency evaluation

To assess the efficacy of SANAIN, we present a comparison between our model and its counterparts, the vanilla affine invariant Augerino (Benton et al., 2020) and Lila (Immer et al., 2022), using different numbers of copies on the affNIST dataset. As depicted in Table 9, SANAIN achieves a remarkable test error of only 0.56% with just two copies, significantly outperforming vanilla Augerino (1.27%) and Lila (1.13%) when employing the same number of copies.

To further highlight the superiority of SANAIN, we increase the number of copies for both vanilla Augerino and Lila. Although their results improve with more copies, they still fall short of SANAIN's

Table 8
Performance comparison on the SHREC'17 perturbed dataset. Retrieval performance is evaluated using the average mean average precisions (mAP) for both micro-average and macro-average versions, denoted as the "score.".

Method	Score	micro			macro			Params
		P@N	R@N	mAP	P@N	R@N	mAP	
SE3CNN (Weiler, Geiger, et al., 2018)	55.5	70.4	70.6	66.1	49.0	54.9	44.9	0.14M
(Li, Fujiwara, Okura, & Matsushita, 2021)	56.5	69.4	69.4	65.8	48.1	56.0	47.2	2.9M
VolterraNet (Banerjee, Chakraborty, Bouza, & Vemuri, 2020)	–	71.0	70.0	67.0	–	–	–	0.4M
(Esteves, Allen-Blanchette, Makadia, & Daniilidis, 2018)	56.5	71.7	73.7	68.5	45.0	55.0	44.4	0.5M
(Cobb et al., 2020)	–	71.9	71.0	67.9	–	–	–	0.25M
PDO-s3DCNN (Shen et al., 2022)	58.6	**72.9**	73.0	68.8	**51.9**	57.7	48.3	0.15M
SANAIN (ours)	**60.9**	72.8	**73.7**	**69.8**	51.0	**60.0**	**52.0**	0.15M

Table 9
The performance of Augerino, Lila, and SANAIN on affNIST. The running time per batch and the memory occupation is measured during inference.

Method	Copies	Test err. (%)	Time (ms)	Memory (MB)
Augerino (Benton et al., 2020)	2	1.27 ± 0.07	8	1,801
	4	0.95 ± 0.03	12	1,891
	8	0.83 ± 0.05	16	2,077
	16	0.83 ± 0.04	21	2,841
	31	0.81 ± 0.04	49	4,213
Lila (Immer et al., 2022)	2	1.13 ± 0.09	8	1,801
	4	0.89 ± 0.04	12	1,891
	8	0.79 ± 0.02	16	2,077
	16	0.75 ± 0.02	21	2,841
	31	0.72 ± 0.03	49	4,213
SANAIN (ours)	2	**0.56 ± 0.04**	**6**	**1,871**

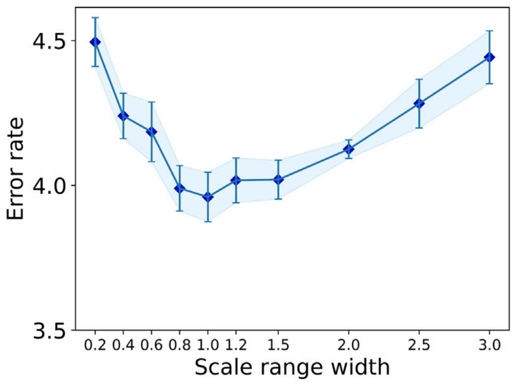


Fig. 3. Test error rates on SIM2MNIST with varying scale range widths.

Table 10
Results on affNIST and SIM2MNIST with different combinations of loss terms.

$\mathcal{L}_{CE}$	$\mathcal{L}_{var}$	$\mathcal{L}_{R}$	Aff. err. (%)	SIM. err. (%)
✓			1.34	5.79
✓	✓		1.01	5.11
✓		✓	0.86	4.92
✓	✓	✓	**0.56**	**3.98**

performance. It is important to note that the architectures and channel counts of their backbones remain similar to ours. Therefore, their reliance on more copies incurs significantly higher computational costs and memory usage, underscoring the efficiency and effectiveness of our approach.

In terms of inference time and GPU resource consumption, SANAIN stands out as the most efficient method for handling affine invariance. It not only achieves superior results but also requires the least time and nearly the lowest GPU resources compared to the two averaged models, further confirming the overall efficiency of SANAIN.

5.6. Ablation study

To verify the effectiveness of adaptive scale invariance, we conduct experiments by fixing the scale range width, represented as $\exp(\theta) - \exp(-\theta)$, and evaluate the performance on the SIM2MNIST dataset. The results, illustrated in Fig. 3, demonstrate that our model excels when the scale range width falls within an appropriate scope. Importantly, our model exhibits the capability to adaptively recover the optimal scale range width (as seen in Section 5.2). Notably, both insufficient and excessive scale invariance adversely affect the performance, underscoring the significance of adaptive scaling in achieving optimal results.

We then justify the effectiveness of $\mathcal{L}_{var}$ and $\mathcal{L}_{R}$. As shown in Table 10, employing both $\mathcal{L}_{var}$ and $\mathcal{L}_{R}$ results in the best performance. By contrast, when $\mathcal{L}_{R}$ is discarded, the models tend to be unconstrained and the learned scale invariance extent is insufficient. When $\mathcal{L}_{var}$ is discarded, the rotation invariance cannot be ensured, and the affine invariance is weak therefore, which degrades the performance.

Moreover, we meticulously examine the impact of $\mathcal{L}_{var}$ on affine invariance within our method. As displayed in Fig. 2(c), the omission of $\mathcal{L}var$ significantly deteriorates affine invariance, mainly due to the lack of robust rotation invariance in this context (as evidenced by Fig. 2(b)). This emphasizes the indispensability and effectiveness of incorporating $\mathcal{L}_{var}$ in our approach.

6. Conclusions

In our study, we have demonstrated that an affine transformation can be elegantly decomposed into the combination of rigid transformations and uni-axial scalings. Leveraging this insight, we have taken a divide-and-conquer approach, addressing these simpler invariances separately, leading to an efficient framework for achieving affine invariance. Importantly, our model has the capability to adaptively learn a reasonable range of scales. Through extensive experimentation, we have validated the remarkable efficiency of our approach in preserving affine invariance, showcasing significant improvements over prior methods in both 2D and 3D contexts.

However, it is important to note that our method only achieves approximate affine invariance, not exact, and it focuses on invariance rather than equivariance, which is a more general concept with applications like image segmentation. In our future work, we plan to explore these aspects further, seeking to broaden the horizons of our approach.

CRediT authorship contribution statement

Zhengyang Shen: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yeqing Qiu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jialun Liu:** Visualization. **Lingshen He:** Visualization. **Zhouchen Lin:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Z. Lin was supported by National Key R&D Program of China (2022ZD0160302), the NSF China (No. 62276004), and the major key project of PCL, China (No. PCL2021A12).

Appendix. Detailed Proof of Theorems

Proof of Proposition 1. As for the affine transformation

$$gx = Ax + t, \tag{A.1}$$

it could be decomposed as follows using the singular value decomposition (SVD):

$$gx = U\Sigma V x + t, \tag{A.2}$$

where $U, V \in O(n)$ and Σ is a diagonal matrix with its diagonal elements positive. Furthermore, we can derive that

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} = \begin{bmatrix} \sigma_1 & \\ & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & & \\ & \sigma_2 & \\ & & I_{n-2} \end{bmatrix} \cdots \begin{bmatrix} I_{n-1} & \\ & \sigma_n \end{bmatrix}. \tag{A.3}$$

We further derive that

$$\begin{bmatrix} I_{i-1} & & \\ & \sigma_i & \\ & & I_{n-i} \end{bmatrix} = \begin{bmatrix} & & -1 & \\ & I_{i-2} & & \\ 1 & & & \\ & & & I_{n-i} \end{bmatrix} \begin{bmatrix} \sigma_i & \\ & I_{n-1} \end{bmatrix} \times \begin{bmatrix} & & 1 & \\ & I_{i-2} & & \\ -1 & & & \\ & & & I_{n-i} \end{bmatrix} \tag{A.4}$$

and the above formula can be further simply denoted as $P_i \Sigma_{\sigma_i} Q_i$. As a result, we have that

$$\Sigma = P_1 \Sigma_{\sigma_1} Q_1 \cdots P_n \Sigma_{\sigma_n} Q_n,$$

where $\Sigma_{\sigma_i} \in US(n)$ and $P_i, Q_i \in SO(n)$. Noting that $SO(n), O(n)$ and $T(n)$ are all subgroups of $E(n)$, the affine transformation can be decomposed as the combinations of rigid transformations and uni-scale transformations, i.e.,

$$gx = UP_1 \Sigma_{\sigma_1} Q_1 \cdots P_n \Sigma_{\sigma_n} Q_n V x + t. \quad \square \tag{A.5}$$

References

Banerjee, M., Chakraborty, R., Bouza, J., & Vemuri, B. C. (2020). VolterraNet: A higher order convolutionalnetwork with group equivariance forhomogeneous manifolds. In *IEEE TPAMI*.

Benton, G., Finzi, M., Izmailov, P., & Wilson, A. G. (2020). Learning invariances in neural networks from training data. In *NeurIPS*.

Cesa, G., Lang, L., & Weiler, M. (2021). A program to build E(N)-equivariant steerable CNNs. In *ICLR*.

Cobb, O. J., Wallis, C. G., Mavor-Parker, A. N., Marignier, A., Price, M. A., d'Avezac, M., et al. (2020). Efficient generalized spherical CNNs. In *ICLR*.

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. In *ICML* (pp. 2990–2999).

Cohen, T. S., & Welling, M. (2017). Steerable CNNs. In *ICLR*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR* (pp. 248–255).

Esteves, C., Allen-Blanchette, C., Makadia, A., & Daniilidis, K. (2018). Learning SO(3) equivariant representations with spherical CNNs. In *ECCV* (pp. 52–68).

Esteves, C., Allenblanchette, C., Zhou, X., & Daniilidis, K. (2018). Polar transformer networks. In *ICLR*.

Finzi, M., Stanton, S., Izmailov, P., & Wilson, A. (2020). Generalizing convolutional networks for equivariance to Lie groups on arbitrary continuous data. In *ICML* (pp. 3165–3176).

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *ICML* (pp. 1319–1327).

Guo, X., Zhu, E., Liu, X., & Yin, J. (2019). Affine equivariant autoencoder. In *IJCAI* (pp. 2413–2419).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770–778).

Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing. In *ICLR*.

Hoogeboom, E., Peters, J. W., Cohen, T. S., & Welling, M. (2018). HexaConv. In *ICLR*.

Immer, A., van der Ouderaa, T., Rätsch, G., Fortuin, V., & van der Wilk, M. (2022). Invariance learning in deep neural networks with differentiable Laplace approximations. In *NeurIPS*.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *NeurIPS*.

Jenner, E., & Weiler, M. (2021). Steerable partial differential operators for equivariant neural networks. In *ICLR*.

Jin, L., Lazarow, J., & Tu, Z. (2017). Introspective classification with convolutional nets. In *NeurIPS*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. In *Communications of the ACM* (pp. 84–90).

Laptev, D., Savinov, N., Buhmann, J. M., & Pollefeys, M. (2016). TI-Pooling: Transformation-invariant pooling for feature learning in convolutional neural networks. In *CVPR* (pp. 289–297).

Lee, K., Xu, W., Fan, F., & Tu, Z. (2018). Wasserstein introspective neural networks. In *CVPR* (pp. 3702–3711).

Li, F., Fujiwara, K., Okura, F., & Matsushita, Y. (2021). A closer look at rotation-invariant deep point cloud analysis. In *ICCV* (pp. 16218–16227).

Luo, Z., Zhou, L., Bai, X., Chen, H., Zhang, J., Yao, Y., et al. (2020). ASLFeat: Learning local features of accurate shape and localization. In *CVPR* (pp. 6589–6598).

MacDonald, L. E., Ramasinghe, S., & Lucey, S. (2022). Enabling equivariance for arbitrary lie groups. In *CVPR* (pp. 8183–8192).

Marcos, D., Volpi, M., Komodakis, N., & Tuia, D. (2017). Rotation equivariant vector field networks. In *ICCV* (pp. 5048–5057).

Miao, N., Rainforth, T., Mathieu, E., Dubois, Y., Teh, Y. W., Foster, A., et al. (2023). Learning instance-specific augmentations by capturing local invariances. In *ICML* (pp. 24720–24736).

Mishkin, D., Radenovic, F., & Matas, J. (2018). Repeatability is not enough: Learning affine regions via discriminability. In *ECCV* (pp. 284–300).

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *NeurIPS*.

Savva, M., Yu, F., Su, H., Kanezaki, A., Furuya, T., Ohbuchi, R., et al. (2017). Large-scale 3D shape retrieval from ShapeNet Core55: SHREC'17 track. In *Proceedings of the workshop on 3D object retrieval* (pp. 39–50). Eurographics Association.

Shen, Z., He, L., Lin, Z., & Ma, J. (2020). PDO-eConvs: Partial differential operator based equivariant convolutions. In *ICML* (pp. 8697–8706).

Shen, Z., Hong, T., She, Q., Ma, J., & Lin, Z. (2022). PDO-s3DCNNs: Partial differential operator based steerable 3D CNNs. In *ICML* (pp. 19827–19846).

Sosnovik, I., Szmaja, M., & Smeulders, A. (2020). Scale-equivariant steerable networks. In *ICLR*.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *ICML* (pp. 1139–1147).

Weiler, M., & Cesa, G. (2019). General E(2)-equivariant steerable CNNs. In *NeurIPS*.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., & Cohen, T. S. (2018). 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In *NeurIPS*.

Weiler, M., Hamprecht, F. A., & Storath, M. (2018). Learning steerable filters for rotation equivariant CNNs. In *CVPR* (pp. 849–858).

Winkels, M., & Cohen, T. S. (2019). Pulmonary nodule detection in CT scans with equivariant CNNs. In *Medical image analysis* (pp. 15–26).

Worrall, D., & Brostow, G. (2018). CubeNet: Equivariance to 3D rotation and translation. In *ECCV* (pp. 567–584).

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *CVPR* (pp. 5028–5037).

Xu, W., Wang, G., Sullivan, A., & Zhang, Z. (2020). Towards learning affine-invariant representations via data-efficient CNNs. In *WACV* (pp. 904–913).

Zhao, Y., Tian, Y., Fowlkes, C., Shen, W., & Yuille, A. (2020). Resisting large data variations via introspective transformation network. In *WACV* (pp. 3080–3089).

Zhou, Y., Ye, Q., Qiu, Q., & Jiao, J. (2017). Oriented response networks. In *CVPR* (pp. 519–528).