Full Length Article

Efficient learning of Scale-Adaptive Nearly Affine Invariant Networks

Zhengyang Shen, Yeqing Qiu, Jialun Liu, Lingshen He, Zhouchen Lin

A B S T R A C T

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 $U(1)$, handling invariance individually. We employ an $E(n)$-invariant model for $E(n)$-invariance and average model outputs over data augmented from a $U(1)$ distribution for $U(1)$-invariance. Our method maintains a favorable computational complexity of $O(N^2)$ in 2D and $O(N^4)$ in 3D scenarios, in contrast to the $O(N^4)$ (2D) and $O(N^{12})$ (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 affine classification tasks 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, Szмaja, & Smueuders, 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: zi@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 Elsevier Ltd. All rights reserved.
However, their method does not support invariance to affine transformations. MacDonald, Ramasimghe, and Lacey (2022) proposed a method based on group convolutions to achieve affine equivariance by sampling from the Haar measure of the group using the Metropolis method based on group convolutions to achieve affine equivariance by rendering them computationally expensive for the large affine group. Furthermore, these model sizes also scale with the group sizes, consequently we address affine invariance by separately handling the

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 Invariant CNNs (RI-CNNs) (Shen et al., 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 computing invariant filters. However, it is important to note that these works can decouple computational costs from group size by analytically computing invariant filters. Notably, 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.

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 \(O(N^6)\) in 2D and \(O(N^9)\) in 3D, significantly more efficient than the \(O(N^8)\) and \(O(N^{12})\) complexities of averaged models (Benton et al., 2020; Immer et al., 2022) in 2D and 3D, not to mention the staggering \(O(N^{16})\) and \(O(N^{24})\) 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.,

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Symbol Description</th>
<th>Symbol Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>Transformation group</td>
<td>(\pi_g) Group action of (g)</td>
</tr>
<tr>
<td>(g)</td>
<td>Transformation in (G)</td>
<td>(f) Input data</td>
</tr>
<tr>
<td>(U(\mathbb{S}))</td>
<td>Uni-axial scaling group</td>
<td>(\Psi) Network model</td>
</tr>
<tr>
<td>(I(\mathbb{S}))</td>
<td>Isotropic scaling group</td>
<td>(\mu_0) Parameterized distribution of group</td>
</tr>
<tr>
<td>(S(\mathbb{O}))</td>
<td>Special orthogonal group</td>
<td>(\theta) Parameter of the distribution</td>
</tr>
<tr>
<td>(T(\alpha))</td>
<td>Translation group</td>
<td>(c_i) Learnable parameters</td>
</tr>
<tr>
<td>(E(n))</td>
<td>Euclidean group</td>
<td>(S) Transformation set</td>
</tr>
<tr>
<td>(Aff(n))</td>
<td>Affine group</td>
<td>(\mathcal{S}) Transformation in (S)</td>
</tr>
<tr>
<td>(G_i)</td>
<td>Subgroup of (G)</td>
<td>(U) Uniform distribution</td>
</tr>
<tr>
<td>(g_i)</td>
<td>Elements in (G_i)</td>
<td>(N) Number of samples</td>
</tr>
<tr>
<td>(\epsilon_{CE})</td>
<td>Classification loss</td>
<td>(\text{dim}) Dimensions of the group</td>
</tr>
<tr>
<td>(\epsilon_{R})</td>
<td>Regularization loss</td>
<td>(\sigma) Parameter of the uni-axial scaling</td>
</tr>
<tr>
<td>(\epsilon_{scal})</td>
<td>Variance penalty loss</td>
<td>(\Sigma) Transformation matrix of uni-axial scaling</td>
</tr>
</tbody>
</table>
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 Sauer, Froost, and Hinton (2017) and Hinton, Sauer, and Froost (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 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 $\text{Aff}(n)$ 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:

$$gx = Ax + t,$$

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

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 $I.S(n)$ and the uni-axial scaling group $U.S(n)$. 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 increase, 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 $\Psi$ to be considered invariant, it should satisfy the condition that for all transformations $g$ within the set $\mathcal{G}$, the following holds:

$$\Psi[g(x)] = \Psi[f].$$

Table 2

<table>
<thead>
<tr>
<th>$n$ = 2</th>
<th>$n$ = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>U(S(n))</td>
<td>1</td>
</tr>
<tr>
<td>I(S(n))</td>
<td>1</td>
</tr>
<tr>
<td>SO(n)</td>
<td>2</td>
</tr>
<tr>
<td>T(n)</td>
<td>3</td>
</tr>
<tr>
<td>E(n)</td>
<td>6</td>
</tr>
</tbody>
</table>

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 $\mathcal{G}$. Given a neural network $\Psi$, we can create a new model $\bar{\Psi}$, which approximates invariance to transformations in $\mathcal{G}$ by averaging the outputs using a parameterized distribution $\mu_\theta$ over the transformations $g \in \mathcal{G}$. In other words,

$$\bar{\Psi}[f] = \mathbb{E}_{g \sim \mu_\theta}[\Psi[g(x)]]$$

(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:

$$g_\theta = \exp \left( \sum_{i=1}^{6} c_i G_i \right)$$

(5)
where $\exp$ represents the matrix exponential function, $e_i \sim U[-1, 1]$, and $U[-1, 1]$ signifies the uniform distribution within the range of $-1$ to $1$. The terms $G_1, \ldots, G_6$ 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 $\theta_j$ 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 $\Psi[x_i/f]$ 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 $e_i \in U[-1, 1]$. 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 $O(N^{12})$.

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-factorized 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 the 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 functions. For rigid transformations, we sample multiple uni-axial scale augmentations of the other form of invariance, we employ RI-CNNs as the backbones of the parallel Siamese network, with their outputs being averaged.

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 $G_1, G_2, \ldots, G_6$. This implies that any element $g \in \mathcal{G}$ can be represented as:

$$g = g_1 g_2 \cdots g_6,$$

where for any $i$, there exists a corresponding $j$ such that $g_i \in G_j$.

To attain $\mathcal{G}$ invariance, we can decompose it into the individual invariances of $G_1, G_2, \ldots, G_6$. Suppose there exists a model $\Psi$ that maintains invariance over $G_1, G_2, \ldots, G_6$, which implies:

$$\forall g_i \in G_i, \quad \Psi[x_i/f] = \Psi[f]$$

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

$$\Psi[x_i/f] = \Psi[x_1 x_2 \cdots x_6/f] = \Psi[x_1 x_2 \cdots x_6 x_7 x_8 x_9/f] = \cdots = \Psi[f].$$

Thus, achieving $\mathcal{G}$ invariance is realized through the invariance of $G_1, \ldots, G_6$.

Intuitively, when considering the complex group $\mathcal{G}$ and its components $G_1, G_2, \ldots, G_6$, if the following inequality holds:

$$\dim(G_1) + \dim(G_2) + \cdots + \dim(G_6) < \dim(G),$$

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 $g \in Aff(n)$, it can be decomposed as

$$g = e_1 e_2 e_3 \cdots e_n,$$

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

$$\forall \nu \in E(n), s \in U S(n), \quad \Psi[x_i/f] = \Psi[x_i/f] = \Psi[f].$$

(11)

it is easy to deduce that

$$\Psi[x_i/f] = \Psi[x_1 x_2 \cdots x_6 x_7 x_8 x_9/f] = \Psi[f].$$

(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 & 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 $I S(2)$. This decomposition results in:

$$\dim(Sim(2)) = \dim(T(2)) + \dim(SO(2)) + \dim(IS(2)) = 2 + 1 + 1 = 4.$$

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.
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:

$$\hat{\Psi}[f] = \mathbb{E}[\hat{\Psi}[\sigma_s f]]$$  \hspace{1cm} (14)$$

where $s$ denotes the uni-axial scale transformation. It is evident from Eq. (14) that $\hat{\Psi}[f]$ 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, 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:

$$\Sigma_s = \begin{bmatrix} \sigma & 0 \\ 0 & 1 \end{bmatrix}$$\hspace{1cm} (15)$$

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

$$\hat{\Psi}[f] = \mathbb{E}[\Psi[\sigma_s f]] = \mathbb{E}_{\epsilon \sim U[-1,1]}[\Psi[\exp(\tilde{\theta})_s f]]$$\hspace{1cm} (16)$$

where $\Psi$ 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:

$$\mathcal{L}_R = -\|\theta\|_2$$\hspace{1cm} (17)$$

To tackle the challenge of achieving rigid transformation invariance, we adopt RI-CNNs as the backbone networks $\Psi$ 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 $\Psi$ for its rigid transformation invariance. We assume that the inputs $f$ follow a rotation-invariant distribution, which leads to the following analysis:

$$\mathbb{E}_f[\mathbb{E}_\theta[\Psi[f] - \Psi[s\theta f]]] = \mathbb{E}_f[\mathbb{E}_\theta[\Psi[f] - \Psi[f] + \Psi[s\theta f] - \Psi[s\theta f]]] \\
\leq \mathbb{E}_f[\mathbb{E}_\theta[\Psi[f] - \Psi[f]] + \mathbb{E}_\theta[\mathbb{E}_\theta[\Psi[s\theta f]] - \Psi[s\theta f]]] \\
= \mathbb{E}_f[\mathbb{E}_\theta[\Psi[s\theta f]] - \mathbb{E}_\theta[\Psi[s\theta f]]] = 2\mathbb{E}_f[\mathbb{E}_\theta[\Psi[f] - \mathbb{E}_\theta[\Psi[s\theta f]]]].$$

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

$$Pr(\mathbb{E}_\theta[\Psi[f]] - \mathbb{E}_\theta[\Psi[s\theta f]] > \delta) < \frac{\mathbb{V}ar(\mathbb{E}_\theta[\Psi[s\theta f]])}{\delta^2}$$\hspace{1cm} (18)$$

where $\mathbb{V}ar(\cdot)$ 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:

$$Loss = \mathcal{L}_{CE} + \lambda \mathcal{L}_R + \mu \mathcal{L}_{var}$$\hspace{1cm} (19)$$

where $\lambda$ and $\mu$ 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).
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 $28 \times 28$ 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 $\gamma$ and calculate the invariance error using the following formula:

$$\text{Error} = \frac{1}{n} \sum_{i=1}^{n} \frac{||\Psi[I_i]| - |\Psi[I_\gamma(i)]||_2}{|||I_i||_2},$$

where $I_i$ represents the test samples, and $\Psi$ 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 $\Psi$ was $\text{SO}(2)$-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 $L_{cur}$ 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 $L_{cur}$ in our approach.

5.2. MNIST variants

The AffiMNIST dataset is created by applying various reasonable affine transformations to images from the MNIST dataset. The resulting images are resized to $40 \times 40$ 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 $\text{SO}(2)$-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 invariance
with just two copies, a notably lower number when compared to
performance improvement. Notably, this outperformance is achieved
error rate of 0.56% on the affNIST dataset, signifying a remarkable
channel count.

Table 4
Test error rates on affNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test err. (%)</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacDonald et al. (w/ DA) (2022)</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>CNN (w/ DA) (Jin et al., 2017)</td>
<td>1.54</td>
<td></td>
</tr>
<tr>
<td>WINN (w/ DA) (Lee et al., 2018)</td>
<td>1.48</td>
<td></td>
</tr>
<tr>
<td>ITN (w/ DA) (Zhou, Tian, Bowlen, Shen, &amp; Vuille, 2020)</td>
<td>1.09</td>
<td></td>
</tr>
<tr>
<td>SO(2)-steerable CNN (Weiler &amp; Ces, 2019)</td>
<td>1.01</td>
<td></td>
</tr>
<tr>
<td>(Xu et al., 2020)</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Augerino (Benton et al., 2020)</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>Lila (Immer et al., 2022)</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>SANAIN (ours)</td>
<td>0.56</td>
<td></td>
</tr>
</tbody>
</table>

Table 5
Test error rates on SIM2MNIST.

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

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 $E_{var}$.

Table 4
Test error rates on affNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test err. (%)</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macdonald et al. (w/ DA) (2022)</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>CNN (w/ DA) (Jin et al., 2017)</td>
<td>1.54</td>
<td></td>
</tr>
<tr>
<td>WINN (w/ DA) (Lee et al., 2018)</td>
<td>1.48</td>
<td></td>
</tr>
<tr>
<td>ITN (w/ DA) (Zhou, Tian, Bowlen, Shen, &amp; Vuille, 2020)</td>
<td>1.09</td>
<td></td>
</tr>
<tr>
<td>SO(2)-steerable CNN (Weiler &amp; Ces, 2019)</td>
<td>1.01</td>
<td></td>
</tr>
<tr>
<td>(Xu et al., 2020)</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Augerino (Benton et al., 2020)</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>Lila (Immer et al., 2022)</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>SANAIN (ours)</td>
<td>0.56</td>
<td></td>
</tr>
</tbody>
</table>

Table 5
Test error rates on SIM2MNIST.

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

For this demanding task, we leverage a more expressive $SO(2)$-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 & 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 $[0, 2 \times 2.4]$, 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 & Daniilidis, 2018) (3.98% vs. 5.03%) and establishing a new state-of-the-art performance.

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 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 $96 \times 96$ 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.
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., 2009) and ImageNet-1k (Deng et al., 2009).

**CIFAR-10**, comprises colored natural images, each measuring 32 x 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 et al., 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test err. (%)</th>
<th>Params</th>
<th>Time (ms)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WideResNet</td>
<td>14.83</td>
<td>2.7M</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Augerino (Benton et al., 2020)</td>
<td>12.33</td>
<td>2.7M</td>
<td>52</td>
<td>10,517</td>
</tr>
<tr>
<td>Lila (Immer et al., 2022)</td>
<td>8.02</td>
<td>2.7M</td>
<td>52</td>
<td>10,517</td>
</tr>
<tr>
<td><strong>SANAIN</strong></td>
<td><strong>6.72</strong></td>
<td><strong>1.2M</strong></td>
<td>25</td>
<td><strong>3,893</strong></td>
</tr>
</tbody>
</table>

### 5.4. 3D shape retrieval

We also assess the effectiveness of SANAIN in the SHREC17 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 $\theta$-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 6

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Test err. (%)</th>
<th>Params</th>
<th>Time (ms)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WideResNet</td>
<td>14.83</td>
<td>2.7M</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Augerino (Benton et al., 2020)</td>
<td>12.33</td>
<td>2.7M</td>
<td>52</td>
<td>10,517</td>
</tr>
<tr>
<td>Lila (Immer et al., 2022)</td>
<td>8.02</td>
<td>2.7M</td>
<td>52</td>
<td>10,517</td>
</tr>
<tr>
<td><strong>SANAIN</strong></td>
<td><strong>6.72</strong></td>
<td><strong>1.2M</strong></td>
<td>25</td>
<td><strong>3,893</strong></td>
</tr>
</tbody>
</table>

### Table 7

Test error rates on ImageNet-1k.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 err. (%)</th>
<th>Top-5 err. (%)</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet (He, Zhang, Ren, &amp; Sun, 2016)</td>
<td>23.91</td>
<td>6.99</td>
<td>25M</td>
</tr>
<tr>
<td>Augerino (Benton et al., 2020)</td>
<td>23.54</td>
<td>6.68</td>
<td>25M</td>
</tr>
<tr>
<td>$C_8$-ResNet</td>
<td>21.96</td>
<td>5.88</td>
<td>27M</td>
</tr>
<tr>
<td><strong>SANAIN (ours)</strong></td>
<td><strong>20.44</strong></td>
<td><strong>5.15</strong></td>
<td><strong>22M</strong></td>
</tr>
</tbody>
</table>
By contrast, when \( \theta \) is discarded, the rotation invariance cannot be ensured, and the affine invariance is weak therefore, which degrades the performance. Importantly, our model has the capability to adaptively learn rotation invariance within our method. As displayed in Fig. 2(c), the omission of \( \theta \) 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 \( \theta \) 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.
References