Conic Activation Functions

Changqing Fu
CEREMADE
PSL Research University
Paris, France
cfu@ceremade.dauphine.fr

Laurent D. Cohen
CEREMADE
PSL Research University
Paris, France
cohen@ceremade.dauphine.fr

Editors: Marco Fumero, Clementine Domine, Zorah Lähner, Donato Crisostomi, Luca Moschella, Kimberly Stachenfeld

Abstract

Most activation functions operate component-wise, which restricts the equivariance of neural networks to permutations. We introduce Conic Linear Units (CoLU) and generalize the symmetry of neural networks to continuous orthogonal groups. By interpreting ReLU as a projection onto its invariant set—the positive orthant—we propose a conic activation function that uses a Lorentz cone instead. Its performance can be further improved by considering multi-head structures, soft scaling, and axis sharing. CoLU associated with low-dimensional cones maintains state-of-the-art performance compared with component-wise ReLU in a wide range of models—including MLP, ResNet, and UNet, etc., achieving better loss values and faster convergence. It significantly improves the training and performance of diffusion models. CoLU originates from a first-principles approach to various forms of neural networks and fundamentally changes their algebraic structure.

1 Introduction

Recurrent neural networks (RNNs), convolutional neural networks (CNNs) and Transformers (Vaswani et al., 2017) are examples of a symmetry principle in neural network architectures: they capture local patterns and uniformly apply them across the entire space. These architectures have laid a solid foundation for modern machine learning systems. RNNs repeatedly apply the same weights to the hidden states. This autoregressive form also inspires diffusion models (Sohl-Dickstein et al., 2015)—the patterns are uniform across intermediate states. Convolutional layers share the same weights in a small local window to slide across a large domain—the patterns are uniform at arbitrary spatial positions. In Transformers, the self-attention function applies its weights homogeneously to the word or pixel embedding space—the patterns are uniform in arbitrary directions since a simultaneous rotation on both the embedded query and key vectors does not change the attention dot product. Different kinds of pattern uniformity are consequences of the associated space homogeneity. These homogeneities (symmetries) have been a principle that continually inspires new designs of model architectures. Recent works continue to push the limit of model performance in vision or language tasks with reduced complexity and different types of symmetry, such as state space models (Gu & Dao, 2023) and more efficient Transformers (Liu et al., 2023).

The convolution and self-attention functions' symmetries are characterized by the equivariance under spatial translation and vector rotation—a function λ is equivariant under a group $\mathcal{G}$ if and only if $\forall P \in \mathcal{G}, P\lambda = \lambda P$, where the operation between them is the composition of functions. The same principle already applies to a basic multi-layer perceptron (MLP). First, the same function is used

Proceedings of the II edition of the Workshop on Unifying Representations in Neural Models (UniReps 2024).

recurrently throughout the forward propagation of the network up to an affine embedding; second, the function applies uniformly to each vector component (neuron). The first property is the foundation of deep models using the same activation functions. The second one results in permutation symmetry: a component-wise activation function is equivariant under the permutation group $\mathcal{G}$. The symmetry of activation functions induces the symmetry in model parameters: by substituting the equality $\lambda = P^{-1}\lambda P, \forall P \in \mathcal{G}$ into a two-layer neural network $f(x) = w\lambda(w'x)$, the network stays the same except that the permutation group acts on the weights such that adjacent weights (w, w') are permuted into (wP^{-1}, Pw'), which means the order of rows and columns of the weight matrices are exchanged. While permutation symmetry has been a fundamental assumption leading to component-wise activation functions, we take another path to reflect on this axiomatic assumption and raise the question:

Can symmetry beyond permutation improve neural networks?

The self-attention function in Transformers positively answers this question. We give a second answer and let activation functions be another solution. To further motivate this activation function, in Appendix G we start from symmetry principles to axiomatically derive the forms of any neural network structures from scratch, where we essentially modify the hypothesis that activation functions are component-wise, changing the Euclidean geometry to the hyperbolic geometry described in Appendix A. We further show in Appendix B that the proposed activation function and the self-attention function share the same type of symmetry, associated with Noether's Theorem. The symmetry group is related to linear mode connectivity explained in Appendix C, which means that the loss landscape of neural networks is empirically convex modulo the group, or the convex hull of the local minima is flattened by group alignment. Generalizing the group to infinite order fundamentally enlarges the algebraic structure of neural networks.

Contributions We propose an activation function called Conic Linear Units (CoLU), which introduces orthogonal group symmetry to neural networks. CoLU outperforms ReLU on the minimal example, and maintains state-of-the-art training and inference performance of various models including ResNet, UNet, and Transformers for recognition and generation. It achieves remarkable gains in training diffusion models.

2 Background

Component-Wise Activations The Rectified Linear Units (ReLU) (Nair & Hinton, 2010) is among the most commonly used activation functions, which induces sparse representations and is applied to sparse autoencoders (Gao et al., 2024). ReLU is mathematically fully characterized by axis homogeneity ($\forall i, \lambda\pi_i = \pi_i\lambda$ where π_i are axis projections), positive homogeneity ($\forall t \geq 0, \lambda t = t\lambda$), and idempotence ($\lambda\lambda = \lambda$). There are also bounded activation functions, such as the sigmoid function or the hyperbolic tangent function used in the long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997). In state-of-the-art vision and language models, smooth approximations of ReLU are preferred for their better performance, such as Leaky ReLU and Exponential Linear Units (ELU) (Clevert et al., 2015), Gaussian Error Linear Units (GELU) (Hendrycks & Gimpel, 2016), Sigmoid-Weighted Linear Units (SiLU) (Elfwing et al., 2018), etc.

Non-Component-Wise Activations Previous non-component-wise activation functions are essentially different from CoLU, such as using layer normalizations (Ni et al., 2024) or multiplying the input by a radial function (Ganev et al., 2021). In comparison, CoLU maintains ReLU's favorable conic projective property. In the previous version of CoLU (Fu & Cohen, 2024), it had not yet achieved universal improvement on all types of models, since its variants had not been developed.

Equivariance in Linear Layers For symmetries in the *linear* part of the model, recent works repeatedly confirm the potential benefits of guaranteeing and generalizing the symmetry principle, such as ensuring the convolution shift-invariance in recognition ResNet (Zhang, 2019) and in generative StyleGAN (Karras et al., 2021). Generalizing spatial symmetry groups to rotations leads to group equivariant convolutional neural networks (GCNN) (Cohen & Welling, 2016), and variants on data beyond pixels (Bronstein et al., 2021). In contrast, CoLU's symmetry is tangent to the spatial symmetry of convolutions. Leveraging number field extension beyond $\mathbb{R}$ results in linear layers with reduced parameters with restricted symmetry structures, such as complex networks (Trabelsi et al.,

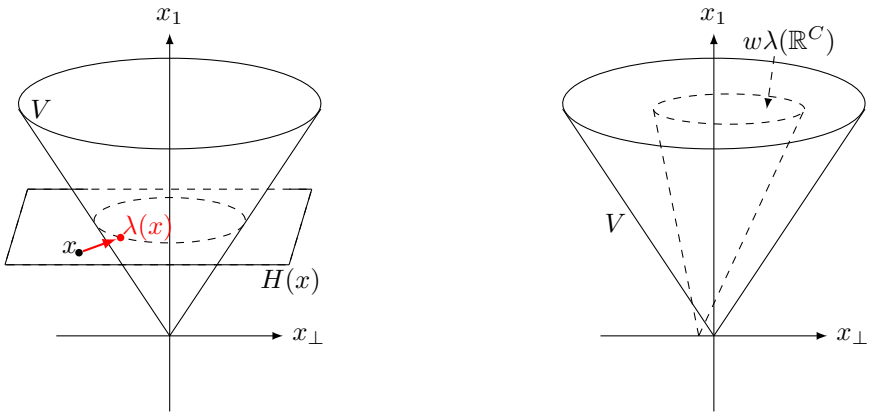
2018), symplectic networks (Chen et al., 2020), and even hypercomplex networks (Parcollet et al., 2019; Zhang et al., 2021), which are compatible with CoLU of associated dimensions.

Spatial versus Channel Correlations Invariant scattering convolutional networks (Bruna & Mallat, 2013) use wavelet bases as deterministic spatial correlations and only learn the pixel-wise linear layer or 1×1 convolution. It indicates that learning channel correlation plays a primary role in representing data patterns compared to spatial connections, and it motivates further investigations into general symmetries in the channel dimensions—the embedding space. Low-rank adaptation (Hu et al., 2022) and the low-rank attention scores in the self-attention function (Choromanski et al., 2020) are examples of putting low-rank assumptions in the embedding space to represent patterns efficiently. CoLU considers another assumption: it assumes potential subspace orthogonalities.

Orthogonality in the Embedding Space To ensure the orthogonality of the embedding space in the linear layers, the hard constraints are either reparameterizing the orthogonal manifold (Arjovsky et al., 2016) or using a projection onto the manifold during training (Li et al., 2019). The soft constraint method adds a regularization term to the loss function (Wang et al., 2020) and approximately learns the orthogonality. Orthogonal CNNs outperform conventional CNNs, suggesting that the orthogonality property helps neural networks gain robustness and generalization ability. The self-attention function in Transformers is also orthogonal-equivariant, compatible with CoLU's symmetry. Activating query and key embeddings with CoLU improves the training of toy Transformers, and we leave this for future work.

Other Constructions of Nonlinearities Weiler & Cesa (2019) survey nonlinear functions for equivariant networks, which does not cover the form of CoLU. Liu et al. (2024); Mantri et al. (2024) propose weighted activation functions that are essentially component-wise by positing other properties, where the equivariance is still restricted to permutations.

3 Conic Activation Functions



(a) Conic projection in 3D

(b) Affine transform of a cone

Figure 1: Illustration of a CoLU function λ and an affine transform w of a cone V.

A basic conic activation function is defined as $\lambda : \mathbb{R}^C \to \mathbb{R}^C$

$$\lambda(x)_i = \begin{cases} x_1, & i = 1 \\ \min\{\max\{x_1/(|x_\perp| + \varepsilon), 0\}, 1\} x_i, & i = 2, \ldots, C \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_C)$ is the input vector, $|\cdot|$ is the Euclidean norm, ε is a small constant taken as 10^{-7} for numerical stability, and $x_\perp$ denotes the normal vector $x_\perp = (0, x_2, x_3, \ldots, x_C)$, so that $x = x_1 e_1 + x_\perp$ holds. Here $e_1 = (1, 0, \ldots, 0) \in \mathbb{R}^C$ is a unit vector. Figure 1a visualizes a CoLU function with a red arrow and Figure 1b visualizes a transformed cone with a linear layer after CoLU. Figure 2 visualizes the connections between neurons of the basic CoLU and its variants to be defined in the sequel. The complexity of CoLU is $O(C)$, which is of the order of component-wise functions and is negligible compared to matrix multiplications. The design of CoLU is irrelevant to the choice of the first axis or another one since its adjacent linear layers are permutation equivariant.

3.1 Soft Scaling

The sigmoid-weighted conic activation function is defined as

$$\lambda(x)_i = \begin{cases} x_1, & i = 1 \\ \text{sigmoid}(x_1/(|x_\perp| + \varepsilon) - 1/2)x_i, & i = 2, \dots, C \end{cases} \tag{2}$$

where $\text{sigmoid}(x) = 1/(1 + \exp(-x))$. Compared with Equation (1), the weighting function $\min\{\max\{r, 0\}, 1\}$ is replaced by $\text{sigmoid}(r - 1/2)$, where $r = x_1/(|x_\perp| + \varepsilon)$ is the cotangent value of the cone's opening angle α, $r \to 1/\tan(\alpha)$ as $\varepsilon \to 0$.

The soft projection is inspired by the better performance of smooth functions such as SiLU $\lambda(x) = \text{sigmoid}(x)x$, compared to the piecewise-linear ReLU $\lambda(x) = \mathbb{1}_{\mathbb{R}_{\geq 0}}(x)x$. Figure 3 compares ReLU weighting with its sigmoid-weighted variant SiLU. Figure 4 compares the hard projection in Equation (1), firm projection weighted by $\text{sigmoid}(4r - 2)$ and sigmoid-weighted soft projection in Equation (2).

3.2 Multi-Head Structure

Inspired by group normalization (Wu & He, 2018), group convolution (Krizhevsky et al., 2012), etc., the channel dimension can be divided into G heads of dimension $S = C/G$. The group-conic activation function is defined as a group-wise application of the conic activation function. Suppose $\lambda : \mathbb{R}^S \to \mathbb{R}^S$ is defined in Equation (1) or (2), and $\pi_i^G : \mathbb{R}^C \to \mathbb{R}^S, i = 1, 2, \dots, G$ are the G-partition subspace projections, then λ in higher dimension C is uniquely characterized by $\pi_i^G \lambda = \lambda \pi_i^G$, or explicitly,

$$\lambda(x) = (\lambda(\pi_1^G(x)), \lambda(\pi_2^G(x)), \dots, \lambda(\pi_G^G(x))) \tag{3}$$

In the trivial case $G = 0$, there is no axis to project towards, and we specify that the activation function coincides with the identity function. In the special case $S = 2$ or when the cones are in a 2D space, the 1D cone section degenerates to a line segment with no rotationality, so we specify that the CoLU coincides with the component-wise activation function.

3.3 Axis Sharing

The shared-axis group CoLU is also uniquely defined by $\pi_i^G \lambda = \lambda \pi_i^G, i = 1, 2, \dots, G$ but with the G-partition subspace projections defined differently:

$$\pi_i^G = (\pi_1, \pi_{(S-1)(i-1)+2}, \pi_{(S-1)(i-1)+3}, \dots, \pi_{(S-1)i+1}), \quad i = 1, 2, \dots, G \tag{4}$$

where $\pi_j, j = 1, 2, \dots, C$ are projections to each axis. π_i^G is a projection onto the first dimension (the cone axis) and $S - 1$ other consecutive dimensions (the cone section). Therefore the relation among the dimension formula among (C, G, S) is $C - 1 = G(S - 1)$ in the shared-axis case.

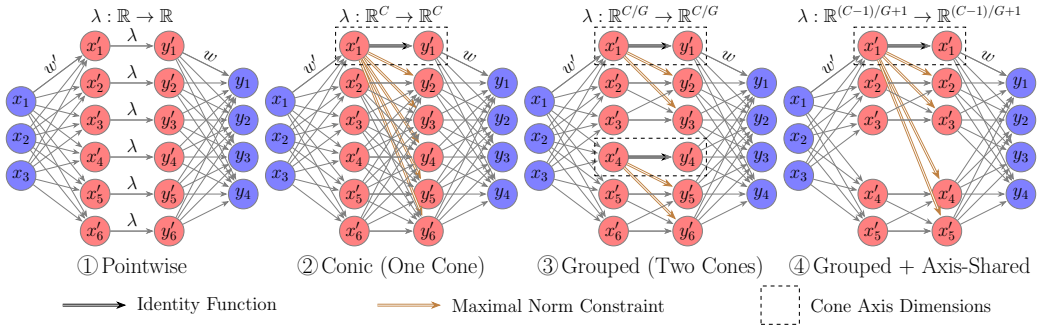


Figure 2: Connections between neurons in a two-layer neural network $y = w\lambda(w'x)$ with component-wise / conic / group-conic / shared-axis group-conic activation functions. In this illustrative example, the network width is $C = 6$ except that in the last shared-axis case $C = 5$. The number of cones is $G = 1$ when there is one cone and $G = 2$ in the grouped case. The yellow arrows mean the neuron thresholds the norm of the output vector, and the dashed frames denote the cones' axis dimensions.

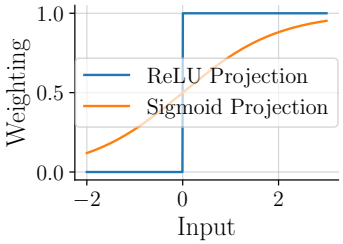


Figure 3: Weighting of hard-projected ReLU and sigmoid-weighted SiLU.

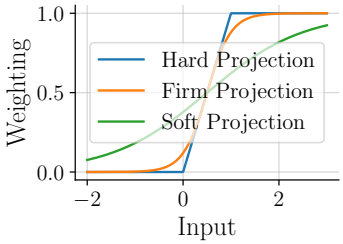


Figure 4: Weighting of the hard-, firm- and soft-projected conic activation functions.

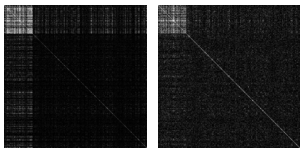


Figure 5: The correlations between weights $\mathrm{cov}(w')$ and between states $\mathrm{cov}(x')$. The bright areas on the top-left corners are the correlated axes.

Figure 5 illustrates the motivation of axis sharing: the colinear effect in the hidden states. In this example, w' is the first linear layer of a VAE's encoder $x \in \mathbb{R}^{784} \mapsto w\lambda(w'x) \in \mathbb{R}^{20}$ pretrained on the MNIST dataset, and x' is the first hidden state $x' = w'x \in \mathbb{R}^{500}$ where the 100 cone axes are permuted together for visualization. Therefore, the hidden dimension is $C = 500$, the number of groups is $G = 100$, the number of test examples is 10000, $w' \in \mathbb{R}^{784\times500}$, $x' \in \mathbb{R}^{10000\times500}$ and $\mathrm{cov}(w'), \mathrm{cov}(x') \in \mathbb{R}^{500\times500}$. The upper-left parts of the matrices are very bright, meaning that the axis dimensions are highly colinear.

3.4 Homogeneous Axes

An alternative form of CoLU ensures component homogeneity, by rotating the standard Lorentz Cone towards the all-one vector, and we call it a rotated conic activation function (RCoLU)

$$\lambda(x) = x_e + \max\{\min\{|x_e|/(|x_r| + \varepsilon), 0\}, 1\}x_r \tag{5}$$

where $x_e = x \cdot e$, $x_r = x - x_e$ and $e = (1/\sqrt{S}, \ldots, 1/\sqrt{S})$. The axis-homogeneous cone avoids splitting operations in the calculation. It can also be combined with grouping using Equation (4), and with axis sharing by setting $e = (1/\sqrt{C}, \ldots, 1/\sqrt{C})$ in Equation (5) instead of using Equation (4). RCoLU's performance boost over ReLU is similar to standard CoLU, so we omit it in the experiment section.

4 Why Conic Activation Functions

CoLU is motivated by modifying the axiom of ReLU so that the Euclidean (flat) geometry is generalized to the Minkowski (hyperbolic) geometry, The mathematical characterization of ReLU is the component-wise conic-projective property. A function is ReLU if and only if it is positive homogeneous, idempotent, and component-wise.

The proofs are provided in Appendix E.

4.1 Conic Projection

A cone's structure is characterized by the hyperbolic geometry detailed in Appendix A. We define the Lorentz cone $V = \{x \in \mathbb{R}^C : x_1^2 - x_2^2 - \ldots - x_C^2 \geq 0, x_1 \geq 0\}$ and the section hyperplane $H(x) = \{y \in \mathbb{R}^C : y_1 = x_1\}$, and denote $\widetilde{V} = \mathbb{R}_{\leq 0}e_1 \cup V$, where $\mathbb{R}_{\leq 0}e_1 = \{(t, 0, \ldots, 0) \in \mathbb{R}^C : t \leq 0\}$.

Definition 4.1 (Conic Projection)**.** *The conic projection is defined as* $x \in \mathbb{R}^C \mapsto \pi_{\widetilde{V}\cap H(x)}(x)$ *where* π *is the nearest point projection,* $\pi_A(x) = \mathrm{argmin}_{y\in A}\,|y - x|$.

The restriction of the projection on its image $\widetilde{V}$ is the identity function, which means it satisfies the idempotent property $\lambda^2 = \lambda$. Constraining the projection in $H(x)$ simplifies the computation while maintaining essential equivariance properties—it guarantees that the projection is always towards the cone axis. Since $V \cap H(x) = \varnothing$ when $x_1 < 0$, the projection is not feasible in the negative half-space, so V is extended to $\widetilde{V}$ for the well-definedness—on the negative half-space, the projection is

degenerate, $\pi_{\widetilde{V}\cap H(x)}(x) = (x_1, 0, \ldots, 0)$. In other words, the past Light Cone has zero light speed and thus zero opening angle.

Lemma 4.2 (CoLU is Conic Projection)**.** *Suppose λ is defined in Equation (1), then it coincides with a conic projection.*

$$\lim_{\varepsilon\to 0}\lambda(x) = \pi_{\widetilde{V}\cap H(x)}(x) = \pi_{\max\{x_1,0\}D+\min\{x_1,0\}e_1}(x) \tag{6}$$

where $D = \{x \in \mathbb{R}^C : x_1 = 1, \sum_{i=2}^{C} x_i \leq 1\}$ is the $(C-1)$-dimensional disk.

We note that V is the conic hull of D, and D is isometric to a hyperball in dimension $C-1$, and therefore it has the symmetry group $\mathcal{O}(C-1)$. In comparison, the invariant set of ReLU is the convex hull of the $C-1$ simplex Δ^{C-1}, defined as the convex hull of the unit vectors $\{e_i \in \mathbb{R}^C : i = 1, 2, \ldots, C\}$. Next, we discuss the general link between algebraic and geometric symmetry.

4.2 Generalized Symmetry Group

Inspired by classical geometry (Klein, 1893), bridging algebraic groups and geometric forms, the equivariant groups of neural networks are more intuitively motivated by the symmetry of the projections' invariant sets.

Definition 4.3 (Invariant Set)**.** *The invariant set of a function $\lambda : \mathbb{R}^C \to \mathbb{R}^C$ is defined as*

$$\mathcal{I}_\lambda = \{x \in \mathbb{R}^C : \lambda(x) = x\}$$

Moreover, the symmetry group $\mathcal{G}$ and the isometric symmetry group $\mathcal{G}^*$ of a set A is the group of affine and rigid functions that preserves the set:

$$\mathcal{G}_A = \{P \in \mathrm{GA}(C) : P(A) = A\}, \quad \mathcal{G}_A^* = \mathcal{G}_A \cap \mathcal{E}(n)$$

where $\mathrm{GA}(C)$ is the general affine group, and $\mathcal{E}(n) = \{P \in \mathrm{Map}(\mathbb{R}^C) : |P(x) - P(y)| = |x - y|, \forall x, y \in \mathbb{R}^C\}$ denotes the Euclidean group.

Definition 4.4 (Symmetry Group)**.** *The equivariance group and the isometric equivariance group of a function $\lambda : \mathbb{R}^C \to \mathbb{R}^C$ is defined as*

$$\mathcal{G}_\lambda = \{P \in \mathrm{GA}(C) : P\lambda = \lambda P\}, \quad \mathcal{G}_\lambda^* = \mathcal{G}_\lambda \cap \mathcal{E}(n)$$

Lemma 4.5 (Projective-Type Operators)**.** *If λ is either ReLU or CoLU, then $\mathcal{G}_\lambda = \mathcal{G}_{\mathcal{I}_\lambda}$, and $\mathcal{G}_\lambda^* = \mathcal{G}_{\mathcal{I}_\lambda}^*$.*

This algebra-geometry duality applies to more general neural network architectures, such as the self-attention function. The relation with Noether's theorem is discussed in Appendix B.

Corollary 4.6 (Permutation Symmetry)**.** *Suppose λ is the component-wise ReLU, then $\mathcal{I}_\lambda = \mathbb{R}_+^C$, $\mathcal{G}_\lambda = \mathcal{G}_{\mathcal{I}_\lambda} = \mathcal{S}(C)$ and $\mathcal{G}_\lambda^* = \mathcal{G}_{\mathcal{I}_\lambda}^* = \mathrm{Perm}(C)$, where $\mathbb{R}_+^C = \{x \in \mathbb{R}^C : x_i \geq 0, i = 1, 2, \ldots, C\}$ is the positive orthant, and $\mathcal{S}(C) = \{P\Lambda \in \mathrm{GL}(C) : P \in \mathrm{Perm}(C), \Lambda \in \mathrm{Diag}(C)\}$ is the scaled permutation group in dimension C, where Perm is the permutation group and Diag is the group of diagonal matrices with non-negative entries.*

Theorem 4.7 (Conic Symmetry)**.** *The symmetry groups of CoLU defined by Equation (3) or (4) are*

$$\mathcal{G}_\lambda = \mathcal{G}_{\mathcal{I}_\lambda} = \mathcal{S}(G) \times \mathcal{O}^G(S-1), \quad \mathcal{G}_\lambda^* = \mathcal{G}_{\mathcal{I}_\lambda}^* = \mathrm{Perm}(G) \times \mathcal{O}^G(S-1) \tag{7}$$

where $\mathcal{I}_\lambda = \widetilde{V}^G$. In the shared-axis case, $\mathcal{I}_\lambda = \widetilde{V}^G/\sim$ where the relation $\sim$ is defined as $x \sim y$ if and only if $\exists i, j \in \{1, 2, \ldots, G\}$ such that $\pi_i^G(x)_1 = \pi_j^G(y)_1$ and $\forall k \in \{2, 3, \ldots, S\}, \pi_i^G(x)_k = \pi_j^G(y)_k = 0$.

In Equation (7), $\mathcal{S}(G)$ represents the permutations among different cones and $\mathcal{O}(S-1)$ represents rotations or reflections within each cone. The motivation is that matrix conjugation modulo permutations reduce to block diagonal form, and we assume there are low-dimensional block sub-spaces that can hold orthogonal equivariance. The symmetry group is continuous and thus of order infinity, unprecedented in component-wise activations. We use the following construction to illustrate that it improves neural networks' generalization ability since component-wise activations fail to hold orthogonal equivariance whereas conic activations do.

Lemma 4.8 (Layerwise Orthogornal Equivariance)**.** *Assume a two-layer neural network* $y = f_\theta(x) = w\lambda(w'x)$ *with fixed width* C *and the training data* D *satisfies subspace orthogonal symmetry:* $\forall (x,y) \in \mathrm{D}, \forall P \in \mathcal{G}, (Px, Py) \in \mathcal{G}$*, where* $\mathcal{G} = \{P \in \mathrm{GL}(C) : P[1,2;1,2] \in \mathcal{O}(2), P[3,\dots,C;3,\dots,C] = \mathbf{I}_{C-2}, P[1,2;3,\dots,C] = P[3,\dots,C;1,2]^\top = 0\} \simeq \mathcal{O}(2)$*. Then,*

(1) (ReLU excludes orthogonal equivariance) If λ *is component-wise activation function, then* $\forall \theta \in (\mathbb{R}^{C^2} \backslash \{0\})^2$*,* $\exists x \in \mathbb{R}^C$ *and* $P \in \mathcal{G}$ *such that* $Pf_\theta(x) \neq f_\theta(Px)$*.*

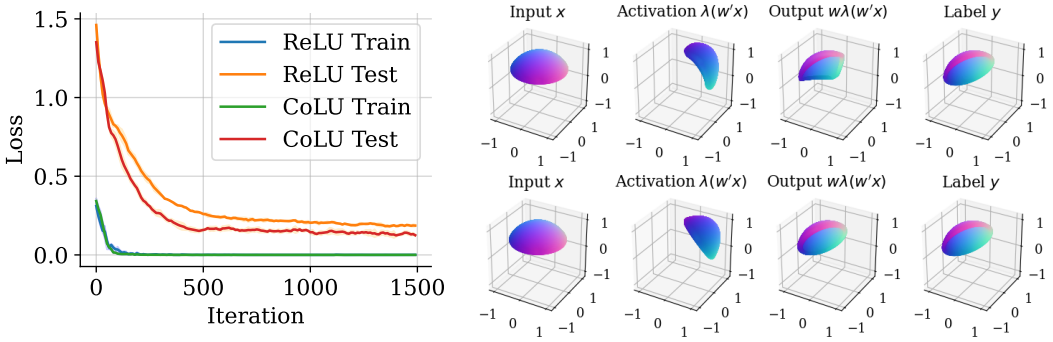
(2) (CoLU holds orthogonal equivariance) If λ *is that of Equation (1), then* $\exists \theta^\dagger = (w^\dagger, w'^\dagger)$ *such that* $\forall x \in \mathbb{R}^C, \forall P \in \mathcal{G}, Pf_{\theta^\dagger}(x) = f_{\theta^\dagger}(Px)$*.*

As a remark, we explain the sufficiency of rigid alignments with a compact group $\mathcal{G}^*$ (without the unbounded rescaling part) by adding a regularization term (or weight decay), to justify the common practice in the literature, which answers the open imitation of the sufficiency of permutation-only alignments in previous works, described in for example Bökman & Kahl (2024).

Remark 4.9 (Sufficiency of Compact Group Alignments)**.** *Suppose* L *is the alignment objective defined in the algorithms in Appendix F, then* $\exists \eta > 0$ *such that the regularized alignment coincides with isometric alignment:* $\operatorname{argmin}_{P \in \mathcal{G}_\lambda}(L(P) - \eta\|P\|) = \operatorname{argmin}_{P \in \mathcal{G}_\lambda^*} L(P)$*, where* $\|\theta\| = -\sum_w (\sum_i w_i^p)^{1/p}$ *is some norm of order* $p \geq 1$*.*

5 Experiments

5.1 Synthetic Data



(a) Train and test loss curves

(b) Forward pass of MLP. Top: ReLU. Bottom: RCoLU

Figure 6: Input, activations, output, and ground truth of a learned hemisphere rotation.

To demonstrate the advantage of the generalized symmetry of CoLU in the embedding space, we use a two-layer MLP to learn the rotation of a 2D hemisphere. The MLP is defined as $x \in \mathbb{R}^3 \mapsto w\lambda(w'x)$, where $w, w' \in \mathbb{R}^{3\times 3}$. The dataset D consists of 40K polar grid points and their rotated counterparts $(x, y = Rx)$ randomly split into train and test sets, where R represents a rotation of $45°$ around each of the three coordinate axes. Figure 6a shows the learning curve and maximum error in the test set. As shown in Figure 6b, ReLU does not capture orthogonal equivariance (rotation around the hemisphere axis) near the equator, instead projecting the boundary onto a triangle. In contrast, RCoLU successfully preserves rotational symmetry at every latitude, including the boundary. This is due to the geometry of the projection boundary: ReLU cuts the hemisphere with the positive orthant and produces a boundary of the 2-simplex Δ^2, while CoLU projects onto a cone that naturally preserves the circular pattern.

5.2 Toy VAE

The toy generative model is a VAE with a two-layer encoder and a two-layer decoder, trained on the binarized MNIST dataset. The test loss is compared since CoLU is hypothesized to increase the model's generalization ability.

Experimental Settings We use the Adam optimizer with a weight decay of 10^{-2} and train 10 epochs for each run. The global batch size is set to 128 and the learning rate is set to 10^{-3}. Each configuration is trained for 10 times with different random seeds. More detailed settings are provided in Appendix D.

Table 1: Comparisons of CoLU model with soft and hard projections with axis sharing. Unstable means some of the initializations do not converge.

Width C	**Group** G	**Dim** C	**Soft?**	**Train Loss** ($\times 10^2$)	**Test Loss** ($\times 10^2$)
2401	0	∞	Identity	1.1086 ± 0.0060	1.1982 ± 0.0011
			Identity	1.1072 ± 0.0031	1.1981 ± 0.0010
2401	1	2401	✓	1.0804 ± 0.0108	1.1740 ± 0.0009
			✗	1.0835 ± 0.0048	1.1656 ± 0.0013
2401	2	1201	✓	1.0302 ± 0.0065	1.1216 ± 0.0016
			✗	1.0226 ± 0.0057	1.1137 ± 0.0026
2401	10	241	✓	0.9181 ± 0.0060	1.0106 ± 0.0017
			✗	0.9166 ± 0.0041	1.0073 ± 0.0015
2401	50	49	✓	0.8698 ± 0.0055	$\mathbf{0.9688} \pm 0.0016$
			✗	0.8736 ± 0.0040	$\mathbf{0.9742} \pm 0.0024$
2401	200	13	✓	0.8424 ± 0.0084	$\mathbf{0.9643} \pm 0.0015$
			✗	0.8430 ± 0.0052	$\mathbf{0.9742} \pm 0.0019$
2401	800	4	✓	0.8388 ± 0.0268	$\mathbf{0.9764} \pm 0.013$
			✗	Unstable	Unstable
2401	1200	3	✓	0.8334 ± 0.0232	$\mathbf{0.9765} \pm 0.0071$
			✗	Unstable	Unstable
2401	-	-	SiLU	0.8429 ± 0.0034	0.9814 ± 0.0007
			ReLU	0.8195 ± 0.0039	0.9892 ± 0.0011

Table 2: Comparisons of soft-projected CoLU with or without axis sharing.

Width C	**Group** G	**Dim** C	**Share Axis?**	**Train Loss** ($\times 10^2$)	**Test Loss** ($\times 10^2$)
2401	0	∞	Identity	1.1086 ± 0.0060	1.1982 ± 0.0011
2400			Identity	1.1098 ± 0.0129	1.1985 ± 0.0015
2401	1	2401	✓	1.0804 ± 0.0108	1.1740 ± 0.0009
2401			✗	1.0828 ± 0.0080	1.1733 ± 0.0008
2401	2	1201	✓	1.0302 ± 0.0065	1.1216 ± 0.0016
2402			✗	1.0207 ± 0.0088	1.1179 ± 0.0029
2401	10	241	✓	0.9181 ± 0.0060	1.0106 ± 0.0017
2410			✗	0.9111 ± 0.0041	1.0096 ± 0.0013
2401	50	49	✓	0.8698 ± 0.0055	$\mathbf{0.9688} \pm 0.0016$
2450			✗	0.8783 ± 0.0045	0.9864 ± 0.0015
2401	200	13	✓	0.8424 ± 0.0084	$\mathbf{0.9643} \pm 0.0015$
2600			✗	0.8718 ± 0.0062	0.9833 ± 0.0021
2401	800	4	✓	0.8388 ± 0.0268	$\mathbf{0.9764} \pm 0.0139$
3200			✗	0.8801 ± 0.0073	0.9893 ± 0.0021
2401	1200	3	✓	0.8334 ± 0.0232	$\mathbf{0.9765} \pm 0.0071$
3600			✗	0.8808 ± 0.0099	0.9930 ± 0.0018
2401	-	-	SiLU	0.8429 ± 0.0034	0.9814 ± 0.0007
4800			SiLU	0.8402 ± 0.0041	0.9856 ± 0.0008

Results Table 1 compares hard-projected or soft-projected CoLU with ReLU or CoLU when the axes are shared. Table 2 compares the improvement from adding axis sharing in the soft projection case. The test losses at the best early-stopping steps are reported. The highlighted cases correspond to the hyperparameters where CoLU outperforms component-wise activation functions. Furthermore, Appendix D complements the learning curves of these hyperparameters. Combining axis sharing and soft projection effectively stabilizes the training when cone dimensions are low in the VAE experiments.

5.3 Toy MLP

According to the hyperparameter search above, we set the cone dimensions to $S = 4$, which complies with the number of chips in hardware platforms. We compare test accuracies in the MNIST recognition tasks to test the hypothesis of CoLU's generalization ability.

Experimental Settings We set the global batch size to 1024 and the learning rate to 10^{-3}. Each configuration is trained 7 times with different random seeds. More detailed settings are provided in Appendix D.

Table 3: Comparisons between ReLU and CoLU in two-layer MLP.

Activation	Width C	Dim S	Axis Sharing	Soft Projection	Train Loss	Test Accuracy
ReLU	512	-	-	✗	0.0000 ± 0.0000	0.9576 ± 0.0017
CoLU	512	4	✗	✗	0.0000 ± 0.0000	**0.9644** ± 0.0010
CoLU	511	4	✓	✓	0.0000 ± 0.0000	**0.9652** ± 0.0013

Results Table 3 compares ReLU with CoLU of low-dimensional orthogonal subspaces and shows the improvement from using axis sharing combined with soft projection. The network width C is set to be as close as possible to control the number of parameters under the constraints of dimension formula among C, G, S.

5.4 ResNet

To test the performance of CoLU in deeper models, we scale up the network to ResNet-56 and train them on the CIFAR10 dataset. Axis sharing and soft projection are omitted for clean comparisons with ReLU in the sequel.

Experimental Settings The ResNet architecture and the training recipe follow He et al. (2016). The runs are repeated for 10 times with different random seeds each lasting 180 epochs, and use the Adam optimizer with a batch size of 128, a learning rate of 10^{-3}, and a weight decay coefficient of 10^{-2}. Finer training settings will achieve better baselines, and CoLU remains superior to ReLU.

Table 4: Comparisons between ReLU and CoLU in ResNet-56.

Activation	Test Accuracy	Test Loss	Train Accuracy	Train Loss
ReLU	**93.5851** ± 0.442	**0.2672** ± 0.0161	99.8351 ± 0.0194	0.0072 ± 0.0003
CoLU	92.7282 ± 0.357	0.3176 ± 0.0175	**99.8817** ± 0.0159	**0.0066** ± 0.0002

Results Table 4 shows that CoLU is on par with ReLU and the training is stable across different initializations.

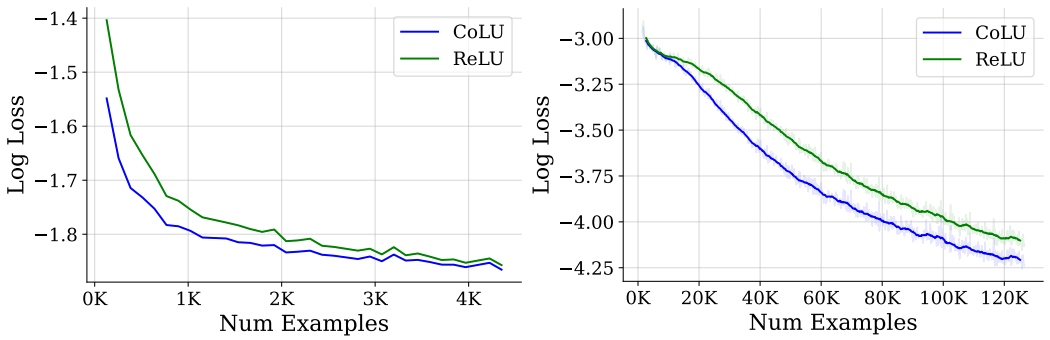
5.5 Diffusion Models

We compare CoLU and ReLU in unconditional generation with diffusion models (Sohl-Dickstein et al., 2015) trained on the CIFAR10 and Flowers datasets. Then we show the possibility of borrowing a pretrained text-to-image model (Rombach et al., 2022) and fine-tuning it to a CoLU model. Detailed settings are in Appendix D.

Training Results Figure 7 shows that CoLU-based UNets converge faster and achieve lower losses than the ReLU-based baselines. On the small dataset CIFAR10, the convergence is observed to be much faster. On the larger Flowers dataset, the loss of the CoLU model is significantly

Table 5: Comparisons in diffusion UNet.

Activation	Train Loss (CIFAR10)	Train Loss (Flowers)
ReLU	0.1606	0.01653
CoLU	**0.1593**	**0.01458**



(a) CIFAR10 (b) Flowers

Figure 7: Learning curves of ReLU and CoLU diffusion models.

lower than the ReLU model throughout the training. Table 5 shows quantitative improvement of CoLU in diffusion UNets. Appendix D shows generated samples on the Flowers dataset.

Fine-Tuning Results We replace all activation functions in the UNet with soft-projected conic activation functions of $G = 32$ without axis sharing. Appendix D shows generated samples from the fine-tuned model and visually compares the original activation and CoLU models.

5.6 MLP in GPT2

CoLU works on par with ReLU in the MLP part of a Generative Pretrained Transformer (GPT2) trained on Shakespeare's play corpus. Appendix D specifies detailed experimental settings. We also observe that CoLU achieves slower overfitting and lower test loss with the same training loss.

Table 6: Comparisons between ReLU and CoLU on GPT's MLP.

Activation	Train Loss (Large)	Steps/s (Large)	Train Loss (Small)	Steps/s (Small)
ReLU	**0.21068** ± 0.00239	24.507 ± 2.656	1.21608 ± 0.00685	65.238 ± 1.765
CoLU	0.21782 ± 0.00198	24.336 ± 1.220	**1.21155** ± 0.00493	65.296 ± 0.594

5.7 Linear Mode Connectivity

CoLU enlarges the group of neural networks' linear mode connectivity, explained in Appendix C.

Convolution Filter Symmetry Diffusion models with ReLU and CoLU have different symmetry patterns in the convolution filters. We show in Appendix D that between the last layer of two diffusion UNets trained with different initialization on CIFAR10, a ReLU model's convolution filters can be permuted to match each other, whereas a CoLU model cannot since the orthogonal symmetry relaxes to additional color space rotations.

Generative Model Alignment For completeness, we show alignment results on the ReLU and CoLU-based models in Appendix D. In the literature on linear mode connectivity, few works study generative models, and we show that the generative VAEs also reveal linear mode connectivity under the equivariance groups of activation functions.

6 Conclusion

In this work, we introduced Conic Linear Units (CoLU) to let neural networks hold layerwise orthogonal equivariance. CoLU works on par with ReLU and scales to a broad range of large models. The code is available at `https://github.com/EvergreenTree/di-f-fu-sion`.

Acknowledgments and Disclosure of Funding

This research is supported with Cloud TPUs from Google's TPU Research Cloud (TRC), and is funded by the PRAIRIE 3IA Institute of the French ANR-19-P3IA-0001 program. We would like to thank anonymous reviewers, Irène Waldspurger and Jamal Atif for their helpful suggestions. Changqing Fu would like to thank Antonin Chambolle, Yann Chevaleyre, and Erwan Fagnou for helpful discussions.

References

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128. PMLR, 2016.

Georg Bökman and Fredrik Kahl. Investigating how ReLU-networks encode symmetries. *Advances in Neural Information Processing Systems*, 36, 2024.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkgYPREtPr.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Taco S Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999. PMLR, 2016.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Changqing Fu and Laurent D Cohen. Conic linear units: Improved model fusion and rotational-symmetric generative model. In *International Conference on Computer Vision Theory and Applications*, 2024.

Iordan Ganev, Twan van Laarhoven, and Robin Walters. Universal approximation and model compression for radial neural networks. *arXiv preprint arXiv:2107.02550*, 2021.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in Neural Information Processing Systems*, 31, 2018.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Uncertainty in Artificial Intelligence*, pp. 876–885, 2018.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: Renormalizing permuted activations for interpolation repair. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gU5sJ6ZggcX.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2021.

Felix Klein. A comparative review of recent researches in geometry. *Bulletin of the American Mathematical Society*, 2(10):215–249, 1893.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Shuai Li, Kui Jia, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.

Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. EfficientViT: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14420–14430, 2023.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

Krishna Sri Ipsit Mantri, Xinzhi Wang, Carola-Bibiane Schönlieb, Bruno Ribeiro, Beatrice Bevilacqua, and Moshe Eliasof. Digraf: Diffeomorphic graph-adaptive activation function. *CoRR*, 2024.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

Yunhao Ni, Yuxin Guo, Junlong Jia, and Lei Huang. On the nonlinearity of layer normalization. In *International Conference on Machine Learning*, volume 235, pp. 37957–37998. PMLR, 21–27 Jul 2024.

Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. Quaternion recurrent neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByMHvs0cFQ.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, June 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1T2hmZAb.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, Steven Liu, William Berman, Yiyi Xu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. URL https://github.com/huggingface/diffusers.

Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11505–11515, 2020.

Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. *arXiv preprint arXiv:2102.08597*, 2021.

Richard Zhang. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning*, pp. 7324–7334. PMLR, 2019.