Quick ViTs: Speeding up Vision Transformers through Equivariance

David Nordström[1], Johan Edstedt[2], Fredrik Kahl[1], and Georg Bökman[3]

[1] Chalmers University of Technology
[2] Linköping University
[3] University of Amsterdam
https://github.com/davnords/octic-vits

Abstract. Natural images exhibit strong geometric regularities: local structures, such as edges, corners, and textures, appear in many orientations and mirror configurations. Since Vision Transformers (ViTs) operate on square image patches, these transformations naturally correspond to the dihedral symmetry group D_8, also known as the octic group. Recent work has shown that ViTs can be made reflection equivariant and more efficient than standard ViTs simultaneously by implementing the linear layers in the Fourier domain of the reflection group. In this work, we extend the equivariance to reflections and rotations and analyze the scalability of the resulting networks. Our Quick ViTs, based on octic equivariant linear layers, achieve 5.33x reductions in FLOPs and up to 8x reductions in memory compared to ordinary linear layers. By analyzing the arithmetic intensity of these layers, we identify theoretical limits on how much the FLOP savings translate into throughput improvements on modern GPUs. However, these limitations disappear as the embedding dimensions increase. Enabled by their computational efficiency, we conduct a broader empirical evaluation of equivariant ViTs than in previous work. Upon training supervised (DeiT-III) and self-supervised (DINOv2) on ImageNet-1K, we find that our Quick ViTs match or exceed baseline accuracy while at the same time providing substantial efficiency gains.

Keywords: Equivariance · Vision Transformer · Efficiency

1 Introduction

In the pursuit of flexible yet scalable models, Vision Transformers (ViTs) [22] have emerged as the dominant architecture in modern computer vision. While ViTs are sometimes heralded as having minimal inductive biases, one key to their success is the construction of visual tokens from image patches and the weight-sharing over the tokens implemented by transformer layers. This weight-sharing ensures permutation equivariance and yields better performing models than allowing arbitrary interactions over tokens [5].

In general, equivariance can provide a powerful inductive bias in neural networks by enforcing structured responses to transformations such as permutations, translations, rotations, or reflections. However, implementations of equivariant

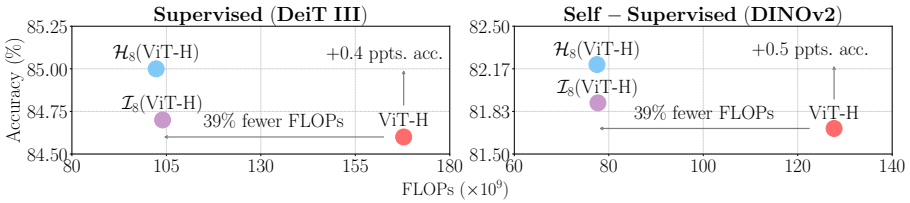


Fig. 1: Computational savings. Using our equivariant layers in ViTs significantly reduces the computational complexity without sacrificing accuracy on ImageNet-1K, for both supervised and self-supervised training. Detailed results can be found in Section 5.

networks often lack in computational efficiency compared to non-equivariant counterparts [35], limiting their popularity in large scale settings. The permutation equivariance in transformers is a clear exception to this rule, as it instead improves the computational efficiency of the model. Recent work has furthermore demonstrated how equivariance under reflections can be implemented efficiently in ViTs, obtaining equivariant ViTs that are more computationally efficient than standard ViTs [11]. In this work, we extend the equivariance to rotations and reflections and introduce Quick ViTs, a family of computationally efficient equivariant vision transformers. Our central observation is that implementing equivariance in the Fourier domain of the symmetry group allows the linear layers to be computed significantly more efficiently. While [11] only present proof-of-concept experiments, we provide a more extensive evaluation using a self-supervised DINOv2 pipeline and evaluating on both segmentation and classification. We find that rotation and reflection equivariant layers can effectively preserve or improve performance of baseline models while requiring significantly fewer FLOPs, see Figure 1.

The analysis of computational complexity in [11] is limited to counting FLOPs. We also consider the arithmetic intensity of the equivariant layers, which is crucial to obtain optimal GPU throughput. The analysis of arithmetic intensity shows how the combination of hardware and datatype determines how much of the reduction in FLOPs can be expected to translate into throughput improvements.

In summary, our contributions are as follows:

(a) We introduce Quick ViTs, a family of vision transformers with octic equivariant layers (Section 3). We propose two variants that are either fully equivariant ($\mathcal{I}_8$) or break equivariance in late layers of the network ($\mathcal{H}_8$). The equivariant linear layers use 5.33 times fewer FLOPs and 8 times less memory per feature dimension than ordinary linear layers.

(b) We analyze the arithmetic intensity of the equivariant linear layers and relate this analysis to the observed throughput improvements (Section 4).

(c) We demonstrate empirically that our ViTs can be integrated with state-of-the-art training recipes (DeiT III and DINOv2) without re-tuning hyperparameters (Section 5). We achieve a 40% FLOP saving with our $\mathcal{I}_8$(ViT-H) and $\mathcal{H}_8$(ViT-H) models while matching baseline performance (Figure 1).

(d) We study the effects of different invariantization methods including symmetry breaking and varying number of equivariant layers (Section 5.3). These ablations help guide future research on equivariant architectures at scale.

Our approach seamlessly integrates into existing ViT architectures and leads to significant gains in throughput and memory efficiency, without sacrificing accuracy. In addition to introducing new state-of-the-art ViTs, from a broader perspective, our contributions give clear evidence that equivariance can matter at scale, which has been a subject of debate in recent literature [1,11,12,60].

2 Related Work

Vision Transformers. The ViT was introduced by [22] and has subsequently achieved state-of-the-art results in many domains of computer vision [14,23,34,40, 46,58]. Significant efforts have been made to scale ViTs [19,64], alongside strategies to do so efficiently [2]. Our approach to making ViTs faster via equivariance is orthogonal to other approaches such as pruning or quantizing. We view it as interesting future work to combine our method with such other efficiency improvements.

Equivariant Networks. The equivariance of CNNs to (cyclic) image translations can be extended to incorporate larger symmetry groups such as rotations and reflections, as shown by [15,21] using Group Equivariant CNNs (G-CNNs). [16,61] generalized G-CNNs to steerable CNNs, where the features transform according to general group representations. Our ViTs can be seen as a more scalable ViT analogue of the octic steerable CNNs by [16]. There have also been prior efforts on attention- and transformer-based equivariant architectures, for both point clouds [3,26,30,38] and, more closely to ours, images [37,48,49,63]. Even when equivariance is not explicitly built into the architecture, neural networks may still learn approximately equivariant representations when trained on data with the corresponding symmetries [10]. However, these prior works do not obtain computational benefits over non-equivariant networks, in contrast to our ViTs.

Our work is part of an ongoing research direction of studying and improving the scalability of equivariant networks [6,7,12,43,56]. Prior work in this direction mostly focuses on point cloud data, with the notable exception of [28] and [11] who consider images. [28] increase the computational efficiency of G-CNNs but in contrast to our work do not achieve benefits over standard networks.

Directly inspiring our work, [11] demonstrate that incorporating reflection equivariance into modern image classifiers increases computational efficiency while maintaining classification performance. In this work, we use both rotation and reflection equivariance, to obtain larger computational savings. We evaluate our equivariant ViTs on a larger set of tasks than prior literature on equivariant image models and we extend the analysis of computational efficiency from counting FLOPs to looking at arithmetic intensity. We also analyze the optimal number of equivariant layers and invariantization.

3 Method

In this section, we explain the construction of rotation and reflection equivariant ViT layers. We begin with preliminaries in Section 3.1, followed by the introduction of equivariant ViTs in Section 3.2, specifics of the transformer layers in Section 3.3. We summarize the most important notation at the end of Section 3.1.

3.1 Preliminaries on Octic Equivariance

In this work, we focus on the dihedral group with eight elements,

$$\mathrm{D}_8 = \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}, \tag{1}$$

such that $r^4 = s^2 = e$ is the identity element and $r^3 = srs$.[4] D_8 acts on images by horizontal reflections s and 90° anti-clockwise rotations r (the reflection axis and rotation direction are arbitrary choices). D_8 is also called the octic group, and we opt for this shorter name throughout. We discuss the generalization to larger dihedral groups in Appendix C.

In equivariant network layers, we need to keep track how the output transforms when the input transforms under D_8. As mentioned in the introduction, transformer layers are generally equivariant under permutations of the tokens. When we reflect and rotate an image, the image patches get permuted but also reflected and rotated. In our equivariant ViTs, we will specify a transformation of the channel dimension of the visual tokens that should correspond to the internal reflections/rotations of the image patches. Then we will construct our layers such that they preserve equivariance with respect to both permutations (obtained by default in token-wise transformer layers) as well as the additional internal transformation, and the internal transformation will be formalized using a real group representation of D_8 as explained next.

In our setting, a real group representation is a vector space $\mathbb{R}^n$ equipped with a group homomorphism ρ from D_8 to the group of $n \times n$ invertible real matrices. In other words for every $g \in \mathrm{D}_8$, $\rho(g)$ is an invertible matrix and for every $g, h \in \mathrm{D}_8$, $\rho(gh) = \rho(g)\rho(h)$. ρ specifies how a vector in $\mathbb{R}^n$ transforms under D_8. There are only a few different representations of D_8 used in this work, we list them below as examples. Finally, it is worth mentioning that all representations considered here are orthogonal, i.e., satisfy $\rho(g)^{-1} = \rho(g)^\mathsf{T}$.

The atomic building blocks of group representations are the so-called irreducible representations.

Example 1 (Irreducible representations). The five irreducible representations, short irreps, of D_8 are defined by

$$\begin{aligned} &\rho_{\mathrm{A1}}(r) = \rho_{\mathrm{A1}}(s) = 1; \qquad \rho_{\mathrm{A2}}(r) = 1,\ \rho_{\mathrm{A2}}(s) = -1; \\ &\rho_{\mathrm{B1}}(r) = -1,\ \rho_{\mathrm{B1}}(s) = 1; \quad \rho_{\mathrm{B2}}(r) = \rho_{\mathrm{B2}}(s) = -1; \\ &\text{and} \quad \rho_{\mathrm{E}}(r) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},\ \rho_{\mathrm{E}}(s) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{2}$$

[4] The reader is cautioned that D_8 is sometimes alternatively denoted D_4.

We use the same notation as the original work on steerable CNNs [16] for these irreps, but choose a different basis for ρ_E. It is known from elementary representation theory that any representation of D_8 can be decomposed into irreps as

$$\rho(g) = Q \left(\bigoplus_{i \in \{\mathrm{A1},\mathrm{A2},\mathrm{B1},\mathrm{B2},\mathrm{E}\}} m_i \rho_i(g) \right) Q^{-1} \tag{3}$$

where $\oplus$ denotes direct sum of representations, or stacking matrices in a block diagonal, and we write $m_i\rho_i(g)$ for $\oplus$'ing $\rho_i(g)$ with itself m_i times. Here Q is an invertible matrix and the m_i are integers specifying the multiplicity of each irrep.

Example 2 (Regular representation). The regular representation ρ_{reg} can be thought of as D_8 acting canonically on the vector space of functions $\phi : \mathrm{D}_8 \to \mathbb{R}$:

$$[\rho_{\mathrm{reg}}(g)\phi](h) = \phi(g^{-1}h). \tag{4}$$

We identify each $\phi : \mathrm{D}_8 \to \mathbb{R}$ with the vector

$$\big(\phi(e)\ \phi(r^3)\ \phi(r^2)\ \phi(r)\ \phi(s)\ \phi(sr^3)\ \phi(sr^2)\ \phi(sr)\big)^{\mathsf{T}} \in \mathbb{R}^8 \tag{5}$$

so that $\rho_{\mathrm{reg}}(g)$ can be written as a permutation matrix. Importantly, ρ_{reg} commutes with pointwise activation functions such as GELU [29].

Example 3 (Isotypical decomposition / Fourier transform). The regular representation ρ_{reg} can be block-diagonalized to its isotypical decomposition ρ_{iso} through (3) as $\rho_{\mathrm{reg}}(g) = Q_{\mathrm{reg}}\rho_{\mathrm{iso}}(g)Q_{\mathrm{reg}}^{-1}$ with

$$\rho_{\mathrm{iso}}(g) = \rho_{\mathrm{A1}}(g) \oplus \rho_{\mathrm{A2}}(g) \oplus \rho_{\mathrm{B1}}(g) \oplus \rho_{\mathrm{B2}}(g) \oplus 2\rho_{\mathrm{E}}(g). \tag{6}$$

The change of basis Q_{reg} (written out in full in Section A) is the inverse Fourier transform of D_8, with $Q_{\mathrm{reg}}^{-1} = Q_{\mathrm{reg}}^{\mathsf{T}}$ being the Fourier transform.

Equivariant networks use equivariant linear layers, which map between representations. A linear G-equivariant map or intertwiner, $W \in \mathbb{R}^{n\times n}$, between representations ρ_1, ρ_2, commutes with their action, i.e., $\rho_2(g)W = W\rho_1(g)$. It follows from Schur's lemma that $W = \lambda I$ for some scalar λ if ρ_1, ρ_2 are irreps, and that $\lambda = 0$ if ρ_1, ρ_2 are not isomorphic [52, Section I.2.2][5]. As ρ_{iso} is just a stack of irreducible representations, any intertwiner between such representations will be sparse, which is why they are less computationally expensive than ordinary linear layers. The naive computational complexity for a linear map $W : \mathbb{R}^{|\mathrm{D}_8|} \to \mathbb{R}^{|\mathrm{D}_8|}$ is $|\mathrm{D}_8|^2 = 64$ multiplications[6]. In contrast, intertwiners $\rho_{\mathrm{iso}} \to \rho_{\mathrm{iso}}$ require a total of $\sum_i^k m_i^2 d_i = 1 + 1 + 1 + 1 + 2^2 \cdot 2 = 12$ multiplications for D_8. From a signal processing perspective, this is analogous to convolution being point-wise multiplications in frequency space. We visualize the computational savings obtained by working in the Fourier domain of D_8 in Figure 2.

[5] We can apply Schur's lemma for complex representations here since the irreps listed in Example 1 are irreducible over the complex numbers. We will however only use real-valued linear layers, i.e. $\lambda \in \mathbb{R}$.

[6] We ignore the additions for simplicity.

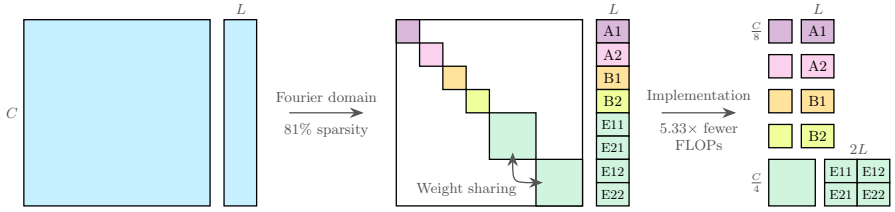


Fig. 2: D_8 **Linear layers.** Implementing equivariant linear layers in the Fourier domain of D_8 gives a major computational benefit. **Left:** A $C \times C$ weight matrix being multiplied by L tokens of feature dimension C. **Center:** The block-diagonalization that happens when enforcing the layer to be D_8-equivariant in the Fourier domain. More precisely, we enforce equivariance with respect to the representation $\frac{C}{8}\rho_{\text{iso}}$ that splits into irreps $\rho_{\text{A1}}, \rho_{\text{A2}}, \rho_{\text{B1}}, \rho_{\text{B2}}$ and ρ_{E} as detailed in Section 3. There is no mixing between different irreps and the weight sharing in the block-diagonal stems from the fact that ρ_{E} is a two-dimensional irrep. **Right:** An efficient implementation of the original $C \times C$ by $C \times L$ matrix multiplication as four $\frac{C}{8} \times \frac{C}{8}$ by $\frac{C}{8} \times L$ and one $\frac{C}{4} \times \frac{C}{4}$ by $\frac{C}{4} \times 2L$ matrix multiplication. An equivariant linear layer of this type requires $16/3 \approx 5.33$ times fewer FLOPs to compute and has 8 times fewer parameters than the ordinary linear layer shown to the left.

Example 4 (Images). Square images can be considered as elements of $\mathbb{R}^{3\times M\times M}$ where 3 is the number of color channels and M is the image height/width in pixels. There is a natural D_8-representation ρ_{image} associated with square images, where $\rho_{\text{image}}(r)$ is the pixel permutation rotating the images anti-clockwise by $90°$ and $\rho_{\text{image}}(s)$ is the permutation reflecting the images left-to-right.

Example 5 (ViT features). In ViTs, features are elements of $\mathbb{R}^{C\times N\times N}$, which we will think of as $C \times N^2$ matrices. N is the number of image tokens along the height/width of the image, so $N = M/P$ where P is the patch size (typically $P = 14$ or $P = 16$) and C is the channel dimension. The simplest representation that we consider on features is the permutation representation ρ_{token} that, analogously to ρ_{image}, permutes the tokens according to elements of D_8.

Example 6 (Steerable ViT features). We can equip the channel dimension of ViT features with a group representation ρ_{chan} to obtain "steerable" features. If C is divisible by 8, we can consider multiples of ρ_{reg} or ρ_{iso} as ρ_{chan}. The complete representation ρ acting on the $C \times N^2$-matrix $\mathbf{x}$, is then permuting the tokens according to ρ_{token} and modifying the channels according to ρ_{chan}. Concretely,

$$\rho(g)\text{Vec}(\mathbf{x}) = \text{Vec}\left(\rho_{\text{chan}}(g)\mathbf{x}\rho_{\text{token}}(g)^{\mathsf{T}}\right) = \left(\rho_{\text{token}}(g) \otimes \rho_{\text{chan}}(g)\right)\text{Vec}(\mathbf{x}), \quad (7)$$

where $\text{Vec}(\mathbf{x})$ is the column-wise vectorization of the matrix $\mathbf{x}$ and $\otimes$ is the tensor product of representations or (equivalently) the Kronecker product of matrices. We refer to features transforming according to (7) as ρ_{chan}-steerable, or features of type ρ_{chan}. This is a simpler form of the induced representations typically considered in steerable CNNs [16,61], the simplification coming from the fact that

we don't enforce translation equivariance. Steerable ViT-features are illustrated in the Appendix, Figure 5.

Example 7 (Patchification of images). One can consider a patchified image as a steerable ViT feature in the following way. By patchification we mean the operation of reshaping a $3 \times M \times M$ image first to N^2 patches of size $P \times P$, with $NP = M$ and then to a $3P^2 \times N^2$ matrix. When we transform the original image by ρ_{image}, the patchified image is rotated by ρ_{token} and ρ_{chan} as in (7). Now $\rho_{\text{chan}}(g)$ is a permutation matrix that rotates or mirrors a patch and we will denote this particular ρ_{chan} by ρ_{patch}.

Notation. For convenience of the reader, we collect the most important notation in this section. We use the bold letter $\mathbf{x}$ for ViT features, which have shape $C \times L$, where L is the number of tokens and C the channel dimension. Typically, $L = N^2$, where N is the image height/width in tokens, or $L = N^2 + 1$ with a class token. For an individual C-dimensional token we use the letter x which is often acted on by $\rho_{\text{chan}} = \frac{C}{8}\rho_{\text{iso}}$, in which case we can split x into $C/8$-dimensional subtokens $x_{\text{A1}}, x_{\text{A2}}, x_{\text{B1}}, x_{\text{B2}}, x_{\text{E11}}, x_{\text{E12}}, x_{\text{E21}}, x_{\text{E22}}$, where the first four transform according to the irreps $\rho_{\text{A1}}, \rho_{\text{A2}}, \rho_{\text{B1}}, \rho_{\text{B2}}$ respectively while the $2 \times \frac{C}{8}$-matrices $(x_{\text{E11}} \quad x_{\text{E12}})^\mathsf{T}$ and $(x_{\text{E21}} \quad x_{\text{E22}})^\mathsf{T}$ both transform according to ρ_{E}.

3.2 Quick ViTs

We construct ViT versions that map images to steerable ViT features $\mathbf{x} \in \mathbb{R}^{C \times L}$ in the first layer (PatchEmbed) and then process steerable ViT features in subsequent layers. We choose ρ_{chan} to be a multiple of ρ_{iso}, enabling efficient linear layers. Typically, we write $\rho_{\text{chan}} = \frac{C}{8}\rho_{\text{iso}}$ where C is the embedding dimension of the ViT. For classification tasks, we map the steerable ViT features to D_8 invariant features fed into a classification head. We will also consider networks that use ρ_{chan}-steerable features for the first layers and then map either to $D\rho_{\text{A1}}$-steerable features, these networks are denoted $\mathcal{I}_8(\text{ViT})$, or break equivariance, denoted $\mathcal{H}_8(\text{ViT})$ (here $\mathcal{I}$ is short for invariant and $\mathcal{H}$ for hybrid). We refer to ViTs that fall into these three families broadly as *Quick ViTs* or, as they in our current implementation are specialized for the octic group, *octic ViTs*. In Section 5.3, we study the effect of varying the number of ρ_{chan}-equivariant layers.

A commonly appreciated fact is that a transformer component b such as an MLP or Attention layer, is permutation equivariant over tokens, which implies

$$b(\mathbf{x}\rho_{\text{token}}(g)^\mathsf{T}) = b(\mathbf{x})\rho_{\text{token}}(g)^\mathsf{T}. \tag{8}$$

For b to be fully equivariant it also needs to be so in the channel dimension

$$b\left(\rho_{\text{chan}}(g)\mathbf{x}\rho_{\text{token}}(g)^\mathsf{T}\right) = \rho_{\text{chan}}(g)b(\mathbf{x})\rho_{\text{token}}(g)^\mathsf{T}. \tag{9}$$

Designing the components of ViT blocks so that (9) holds is the topic of Section 3.3.

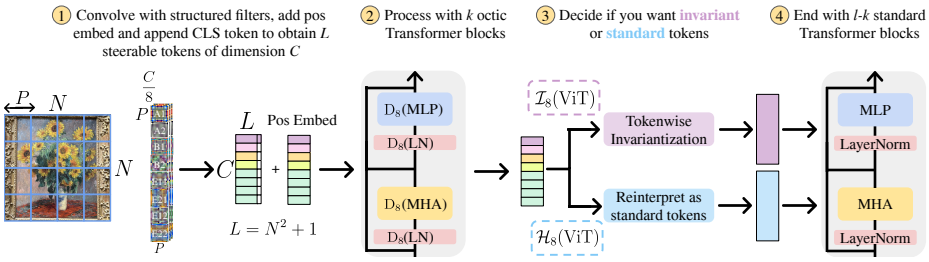


Fig. 3: Quick ViT architecture. Patches are first extracted from an image using specialized octic equivariant filters and the resulting features are processed by k octic equivariant ViT blocks. The final embeddings can be fed to $l - k$ standard transformer blocks (as demonstrated by our $\mathcal{H}_8$ and $\mathcal{I}_8$ ViTs).

3.3 Octic Equivariant Layers

In this section we detail our implementations of D_8 equivariant transformer layers. Together, these pieces can be combined into a ViT block $b = b_2 \circ b_1$ where

$$b_1(\mathbf{x}) = \mathbf{x} + \mathrm{MHA}(\mathrm{LN}(\mathbf{x})), \quad b_2(\mathbf{x}) = \mathbf{x} + \mathrm{MLP}(\mathrm{LN}(\mathbf{x})). \tag{10}$$

LN is layer normalization, MHA multi-head self-attention and MLP a $C \to 4C$ linear map, GELU, and $4C \to C$ linear map. The blocks can subsequently be stacked, as illustrated in Figure 3.

The Patch Embedding Layer. The first layer in a ViT, following [22], is the patch embedding, short PatchEmbed. In our case, it can be viewed as a mapping from steerable features of type ρ_{patch} to steerable features of type $\frac{C}{8}\rho_{\text{iso}}$ where C is the embedding dimension of the ViT. It is implemented as a convolution over the input image with kernel size and stride equal to the patch size P. The convolution kernels are weight sharing constrained to map to features of the different irreps-types in $\frac{C}{8}\rho_{\text{iso}}$, see Appendices D and D.1 for kernel visualizations.

Directly after PatchEmbed, we add a learnable positional encoding $\mathbf{e} \in \mathbb{R}^{C \times L}$ to the features. The positional encoding is not constant over tokens, thereby breaking translation equivariance. To be D_8 equivariant $\mathbf{e}$ must satisfy

$$\begin{aligned} \forall \mathbf{x} : \rho_{\text{chan}}(g)\mathbf{x}\rho_{\text{token}}(g)^\mathsf{T} + \mathbf{e} &= \rho_{\text{chan}}(g)(\mathbf{x} + \mathbf{e})\rho_{\text{token}}(g)^\mathsf{T} \\ \iff \quad \mathbf{e} &= \rho_{\text{chan}}(g)\mathbf{e}\rho_{\text{token}}(g)^\mathsf{T}. \end{aligned} \tag{11}$$

In words, the positional encoding at a specific token position p must be a $\rho_{\text{chan}}(g)$-transformed version of the positional encoding at the token position that is permuted to p by $\rho_{\text{token}}(g)$. After adding the positional encoding, we append a learnable class token `[CLS]` $\in \mathbb{R}^{C \times 1}$ to the features. To ensure equivariance, we enforce it to be non-zero only in the A1 feature type.

Linear Layers. Linear layers appear in ViTs both in the MLP block and the MHA block. As mentioned in Section 3.1 and illustrated in Figure 2, equivariant linear layers map between irreps of the same type due to Schur's lemma. This fact was used to construct efficient reflection-equivariant neural networks by [11]. Here, we use the same approach for octic equivariance.

To re-iterate, for features of type $\rho_{\text{chan}} = \frac{C}{8}\rho_{\text{iso}}$ we consider each token $x \in \mathbb{R}^C$ split into $x_{\text{A1}}, x_{\text{A2}}, x_{\text{B1}}, x_{\text{B2}}, x_{\text{E}}$ of dimensions $C/8, C/8, C/8, C/8$ and $2 \times C/4$. Linear layers map each irrep type to itself, meaning that they are parameterised by four $C/8 \times C/8$ matrices and one $C/4 \times C/4$ matrix, yielding a factor 8 fewer parameters than a general linear layer.

In terms of FLOPs needed to compute the linear layer, $x_i \mapsto W_i x_i$ requires $C/8 \cdot C/8 = C^2/64$ FLOPs for $i \in \{\text{A1, A2, B1, B2}\}$ while due to the "weight-sharing" over the two dimensions in ρ_{E}, it requires $2 \cdot C/4 \cdot C/4 = C^2/8$ FLOPs for $i = \text{E}$. In total we therefore get $16/3 \approx 5.33$ times fewer FLOPs than the C^2 required for an ordinary linear layer.

Activation Functions, Layer Norm, Attention, and Invariantization. A pointwise activation function σ can be applied equivariantly after transforming the features from the Fourier domain (multiples of ρ_{iso}), to the spatial domain (multiples of ρ_{reg}), as discussed in Example 3. In the spatial domain σ can be applied point-wise as this commutes with the permutation representation ρ_{reg}. In Quick ViTs, following prior work, we apply the GELU activation function.

We implement (token-wise) equivariant layer normalization by transforming each irrep separately to mean 0 followed by division of norm over all the irreps.

If two tokens q and k transform according to the same orthogonal representation ρ_{chan}, then $q^\mathsf{T}k$ is invariant under D_8 since $(\rho_{\text{chan}}(g)q)^\mathsf{T}(\rho_{\text{chan}}(g)k) = q^\mathsf{T}\rho_{\text{chan}}(g)^\mathsf{T}\rho_{\text{chan}}(g)k = q^\mathsf{T}k$. This means that the computation of attention logits in ordinary scaled dot-product attention is invariant, so the subsequent weighted sum over value tokens is equivariant.

To output D_8 invariant classification predictions, we map from features of type $\frac{C}{8}\rho_{\text{iso}}$ to features of type $C\rho_{\text{A1}}$ and then extract the `[CLS]` token. We ablate different invariantizations to A1-tokens in Appendix E, including linear invariants, triple correlation [32, 51], max filtering [13], generators of the ring of invariant polynomials and canonisation of the signal [31]. We find that a power spectrum invariantization works well and settle on that for the the experiments.

4 Computational Efficiency

As transformers scale, the linear layers dominate the execution time [33]. Thus, as ViTs grow, the FLOPs savings will approach those of the linear layer, a reduction of 5.33 times. We plot the FLOPs savings of a ViT block as the embedding dimension increases in Figure 4a. The computational benefits of equivariant ViTs are more pronounced at scale, and we benchmark the throughput of various ViT sizes from the literature and their equivariant counterparts in Table 1. In this throughput benchmark we replace all of the ViT blocks with equivariant

Table 1: Compute scaling. We measure the scaling of D_8-equivariant ViTs. The model sizes are taken from [19] and we do not train the largest models as part of this work. The numbers ending in "x" describe the change from standard ViT statistics of the corresponding equivariant ViTs. We benchmark in half precision. In full precision speed gains are much more pronounced, especially at small scales, c.f. Table 2. In this table we show models where all linear layers are replaced by block-diagonal layers. In Section 5 we train models with only half of the layers block-diagonal. Hence the throughput improvements there are lower.

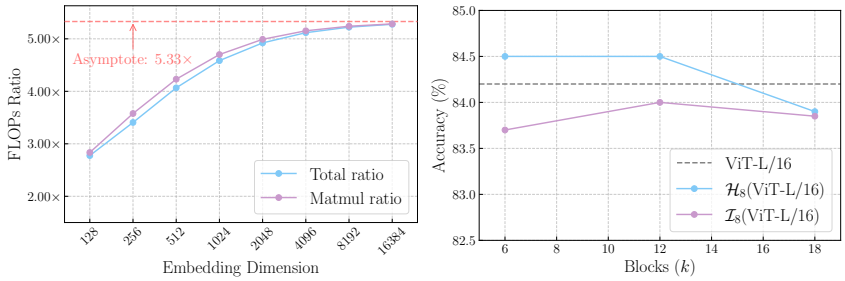
Model	Width	Depth	MLP-dim	Heads	Speed(bf16)	FLOPs	Peak Mem.
ViT-L	1024	24	4096	16	1.32x	1/4.58x	1/2.44x
ViT-H	1280	32	5120	16	1.47x	1/4.58x	1/2.88x
ViT-G	1664	48	8192	16	1.91x	1/4.88x	1/4.36x
ViT-e	1792	56	15360	16	2.37x	1/5.01x	1/5.30x
ViT-22B	6144	36	24576	48	3.54x	1/5.18x	1/5.80x

blocks, and therefore the models fall into the $\mathcal{I}_8$ family of networks, but with invariantization after the last block rather than half the blocks.

As shown in Table 1, savings in FLOPs do not translate one-to-one to improvements in throughput (images per second) in our current implementation. However, it is still the case that the throughput is greatly improved. Our Quick ViT models are pure PyTorch with `torch.compile`, except for the GELU nonlinearity. We implement a custom Triton [54] kernel that fuses the GELU nonlinearity with the Fourier and inverse Fourier transforms, limiting memory transfers and kernel invocation overhead. While ordinary GELU is pointwise, the new fused Triton kernel is eight to eight points. For extra efficiency, we implement the Fourier transforms by a FFT on D_8, described in Appendix A. In Appendix A.2 we show that the overhead of our nonlinearity compared to standard GELU is minimal when compared to the computational complexity of the linear layers. The gap between savings in FLOPs and speedup is in fact due to the linear layers not having sufficient arithmetic complexity at low channel dimensionality as demonstrated in the next section.

4.1 Arithmetic Intensity

The arithmetic intensity of an operation is the FLOPs per transferred byte to the GPU and it can be compared with the FLOPs per bandwidth of a given device to obtain a bound on the maximum achievable throughput [4]. The arithmetic intensities are $\frac{2BCF}{P(BC+CF+BF)}$ and $\frac{2BCF/5.33}{P(BC+CF/8+BF)}$ for the standard and octic linear layers, respectively, where B is the batch size in tokens, C is the input dimension, F is the output dimension and P is the precision in bytes. This means that the octic and ordinary layers scale differently. At large scale, not only FLOPs are improved by octic layers but also arithmetic intensity. For instance, for $B = 196$ (one image worth of tokens), $P = 2$ (half precision) and $F = 4C$ (a typical MLP expansion factor) one can calculate that ordinary linear layers



(a) FLOPs ratio vs. embedding dimension **(b)** Accuracy vs. number of octic blocks (k)

Fig. 4: (a) Reduction in FLOPs from a non-equivariant transformer block to an octic-equivariant block vs. embedding dimension. The matmul ratio reflects only matrix multiplications in linear layers and Attention; the total ratio includes all computations. **(b)** The effect of changing number of octic blocks (k) for ViT-L, out of $l = 24$ blocks.

have higher arithmetic intensity up to $C \approx 3200$, whereas octic linear layers have higher arithmetic intensity at thereafter. For the experiments in this paper, we are not able to scale to such large dimensions, but still get throughput benefits due to savings in FLOPs, as shown in Table 1.

For larger batch sizes, standard linear layers typically have larger arithmetic intensity than octic linear layers, because the terms without B in the denominator become negligible. However, at sufficiently large C both standard and octic linear layers are compute bound and hence what matters for throughput is the total number of FLOPs, making octic linear layers 5.33 times faster.

As mentioned, the main bottleneck to achieve throughput gains closer to the actual FLOP reductions in octic ViTs is that the linear layers are not performing sufficiently well for low feature dimensions C. To more clearly illustrate the bottleneck and tie it to the arithmetic intensity (AI) of the layers we conduct a simple throughput benchmark for only linear layers. We include standard linear layers, octic equivariant linear layers and block-diagonal layers with eight equal size blocks, all from C to $F = 4C$ channels. The block-diagonal layers have arithmetic intensity $\frac{2BCF/8}{P(BC+CF/8+BF)}$.

We benchmark with batch size $64 \cdot 196$ (i.e. 64 images worth of tokens) on an A100-80GB GPU. The performance in bf16 and fp16 precision is equal with an advertised maximum achievable performance of 312 TFLOPs/s. We use bf16 precision when training the networks in this paper. For reference we also include results in fp32 precision, where the advertised maximum achievable performance is 19.5 TFLOPs/s. The ridge point for when computations become theoretically compute bound is at 156 FLOPs/byte for the lower precision and 10 FLOPs/byte for the higher precision. We see in Table 2 that for non-compute bound layers such as octic and block-diagonal linears at $C = 1024$ in fp16, the performance in TFLOPs/s is far below the compute bound layers, leading to a lower speedup than predicted by the FLOP counts. However, in fp32 the speedup is drastically higher.

Table 2: Benchmarking linear layers. Performance of three different types of linear layers over different precisions and different amount of channels.

Linear layer	Precision	Channels	AI (FLOPs/byte)	Time (ms)	Perf. (TFLOPs/s)	Speedup
Standard	fp16	$1024 \to 4096$	770	0.47	230	1.0
D_8	fp16	$1024 \to 4096$	150	0.19	110	2.6
Eight blocks	fp16	$1024 \to 4096$	100	0.13	100	3.6
Standard	fp16	$8192 \to 32768$	4300	24	280	1.0
D_8	fp16	$8192 \to 32768$	1200	5.3	240	4.6
Eight blocks	fp16	$8192 \to 32768$	770	3.4	250	7.2
Standard	fp32	$1024 \to 4096$	380	5.6	19	1.0
D_8	fp32	$1024 \to 4096$	80	1.2	17	4.8
Eight blocks	fp32	$1024 \to 4096$	50	0.76	17	7.4
Standard	fp32	$8192 \to 32768$	2200	360	19	1.0
D_8	fp32	$8192 \to 32768$	580	67	19	5.33
Eight blocks	fp32	$8192 \to 32768$	380	45	19	7.95

5 Experiments

In this section, we evaluate our Quick ViTs on supervised (DeiT III) and self-supervised (DINOv2) training recipes and perform ablations. DeiT III [55] is a popular and highly tuned supervised training recipe for classification and DINOv2 [44] is a state-of-the-art self-supervised method to extract visual features at large scale. All models are trained on ImageNet-1K [20,47,50] following the official implementations. Code is available at https://github.com/davnords/octic-vits.

5.1 DeiT III

We train an array of networks on the supervised task of image classification and compare to the performance reported by [55] and [11]. We find, as illustrated in Table 3, that incorporating octic-equivariant layers provides significant computational savings without sacrificing accuracy. In particular, our $\mathcal{H}_8$(ViT-H/14) model achieves a classification performance of 85.0% compared to the baseline of 84.6% while using only 61% of the FLOPs and matching the performance of the $\mathcal{H}_2$(ViT-H/14) that incorporates flopping (D_2) equivariance while being more computationally efficient. Similar computational gains are obtained with the invariant model $\mathcal{I}_8$(ViT-H/14), which achieves a performance of 84.7%.

In the final column of Table 3, we study the effect of evaluating models on a randomly rotated validation set without training on such augmentations. We find that the invariant model performs equally well while the performance of the remaining models (including $\mathcal{H}_8$) significantly degrade.

Table 3: DeiT III evaluation. We measure the Top-1 classification accuracy on ImageNet-1K for different model sizes. Our networks are marked with †.

Model	params ($\times 10^6$)	throughput bf16 (im/s)	FLOPs ($\times 10^9$)	Peak Mem. (MB)	Top-1 Acc. ↑	OOD Rot. ΔAcc.↑
ViT-H/14	632.1	569	167.8	3285	84.6	-12.6
$\mathcal{H}_2$(ViT-H/14)	474.2	640	127.4	2734	**85.0**	-13.4
$\mathcal{I}_8$(ViT-H/14)†	362.3	657	104.0	2249	84.7	**0.0**
$\mathcal{H}_8$(ViT-H/14)†	355.8	660	102.3	2223	**85.0**	-13.4
ViT-L/16	304.4	1421	61.9	1557	84.2	-13.4
$\mathcal{H}_2$(ViT-L/16)	228.3	1610	46.9	1458	**84.5**	-13.9
$\mathcal{I}_8$(ViT-L/16)†	175.5	1618	38.5	1210	84.0	**0.0**
$\mathcal{H}_8$(ViT-L/16)†	171.3	1615	37.7	1194	**84.5**	-13.6

Table 4: DINOv2 evaluation. We evaluate the frozen DINOv2 features by classification accuracy on ImageNet-1K (IN1K) and segmentation mIoU on ADE20K [66,67] and VOC2012 [24]. Our networks are marked with †.

Model	FLOPs	IN1K (acc.) ↑		ADE20K (mIoU) ↑		VOC2012 (mIoU) ↑	
	($\times 10^9$)	linear	k-NN	linear	k-NN	linear	k-NN
ViT-H/16	127.7	81.7	81.0	34.7	30.6	70.7	60.9
$\mathcal{H}_2$(ViT-H/16)	97.0	81.9	80.9	34.8	30.8	70.7	61.2
$\mathcal{I}_8$(ViT-H/16)†	77.7	81.9	80.9	33.9	29.2	70.6	61.2
$\mathcal{H}_8$(ViT-H/16)†	77.5	**82.2**	**81.4**	**35.1**	**31.1**	**70.8**	**61.7**
ViT-L/16	61.9	80.9	80.5	33.2	28.4	69.1	58.1
$\mathcal{H}_2$(ViT-L/16)	46.9	**81.3**	80.7	33.4	29.0	69.3	58.0
$\mathcal{I}_8$(ViT-L/16)†	38.5	81.2	80.4	32.6	28.0	**70.0**	**59.6**
$\mathcal{H}_8$(ViT-L/16)†	37.7	**81.3**	**80.8**	**33.6**	**29.4**	69.1	59.5

5.2 DINOv2

As another pretraining task, we consider the DINOv2 recipe and train our own baselines.[7] Results are summarized in Tab. 4. We find that incorporating octic-equivariant layers maintains or improves downstream classification and segmentation performance while saving FLOPs. In particular, our invariant network $\mathcal{I}_8$(ViT-H/16) matches the downstream performance of the baseline while using only 61% of the FLOPs and $\mathcal{H}_8$(ViT-H/16) slightly improves performance with similar savings.

In Appendix F.3, we further investigate the performance of our invariant model and evaluate it on white blood cell classification, a task that lacks a canonical orientation. The invariant model $\mathcal{I}_8$(ViT-L/16) outperforms the baseline on most evaluated metrics.

[7] Note that the pretraining objectives for DINOv2 [44] and DINOv3 [53] are identical.

5.3 Ablations

Impact of Equivariance. We study the effect of equivariance by replacing the kernel constrained equivariant patch embed by an arbitrarily linear mapping while keeping the rest of the architecture the same as for $\mathcal{H}_8$. In principle, the model can learn to be equivariant. We evaluate ViT-B using the DeiT III recipe and achieve an accuracy of 82.4 compared to 83.0 for $\mathcal{H}_8$(ViT-B). The results suggest that equivariance yields higher accuracy than arbitrary mappings with the same block-diagonal structure (in addition to providing steerable features that can be useful in downstream applications).

Number of Octic Blocks. We experiment with incorporating non-equivariant blocks following [61]. We include networks where the first k of the ViT blocks are octic and the remaining $l - k$ are standard blocks. Figure 4b ablates different values of k for ViT-L ($l = 24$). We find that $k = \frac{l}{2}$ strikes a good balance between computational efficiency and representational power and use this throughout the paper. Note that $\mathcal{I}_8$ performs worse for small k due to early invariantization.

We find that the $\mathcal{H}_8$-networks typically outperform $\mathcal{I}_8$-networks. This usefulness of symmetry breaking corroborates findings in prior work, *e.g.* [56, 61]. The intuition is that ImageNet is not a rotationally invariant dataset, so it leads to improved performance to let the network use the fact that the images are upright as extra information for solving the classification task.

6 Limitations

In this work, we limit our scope to an extensive study of the D_8 group and leave larger dihedral groups for future work. A theoretical discussion regarding scaling to larger dihedral groups is given in Appendix C for reference. We follow baseline training recipes without hyperparameter tuning, and we do not conduct an extensive ablation study of the share of features per irrep or scale the size beyond ViT-H.

7 Conclusion

We introduced octic-equivariant ViT layers that, when incorporated, maintain accuracy while significantly reducing computational complexity. We validated our proposed architectures by their effectiveness in both supervised and self-supervised learning, and conducted ablation studies to isolate the effect of invariantization, equivariance, and the number of octic blocks. In particular, we achieved an approximate 40% reduction in FLOPs for ViT-H without sacrificing accuracy, positioning Quick ViTs as a strong addition to the catalog of vision architectures.

For future work, we intend to incorporate Quick ViTs into a broader range of computer vision applications, in particular in settings where geometric equivariance provides a natural inductive bias. Promising directions include 3D vision

tasks, such as multi-view masked image modeling [41] and feed-forward 3D reconstruction [58,59], as well as image matching, where robustness to rotations and other geometric transformations is central [8,9,42]. These directions would further clarify the role of equivariant architectures as efficient and geometry-aware backbones beyond image classification.

Acknowledgements

This work was supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and by the strategic research environment ELLIIT, funded by the Swedish government. The computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at C3SE, partially funded by the Swedish Research Council through grant agreement no. 2022-06725, and by the Berzelius resource, provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

References

1. Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A.J., Bambrick, J., et al.: Accurate structure prediction of biomolecular interactions with alphafold 3. Nature **630**(8016), 493–500 (2024)
2. Alabdulmohsin, I.M., Zhai, X., Kolesnikov, A., Beyer, L.: Getting vit in shape: Scaling laws for compute-optimal model design. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 16406–16425. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/3504a4fa45685d668ce92797fbbf1895-Paper-Conference.pdf
3. Assaad, S., Downey, C., Al-Rfou, R., Nayakanti, N., Sapp, B.: Vn-transformer: Rotation-equivariant attention for vector neurons. Transactions on Machine Learning Research (1 2023)
4. Austin, J., Douglas, S., Frostig, R., Levskaya, A., Chen, C., Vikram, S., Lebron, F., Choy, P., Ramasesh, V., Webson, A., Pope, R.: How to scale your model. Online (2025), retrieved from https://jax-ml.github.io/scaling-book/
5. Bachmann, G., Anagnostidis, S., Hofmann, T.: Scaling mlps: A tale of inductive bias. Advances in Neural Information Processing Systems **36**, 60821–60840 (2023)
6. Bekkers, E.J., Vadgama, S., Hesselink, R., der Linden, P.A.V., Romero, D.W.: Fast, expressive SE(n) equivariant networks through weight-sharing in position-orientation space. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=dPHLbUqGbr
7. Bharadwaj, V., Glover, A., Buluc, A., Demmel, J.: An Efficient Sparse Kernel Generator for O(3)-Equivariant Deep Networks. Society for Industrial and Applied Mathematics (2025), https://arxiv.org/abs/2501.13986
8. Bökman, G., Edstedt, J., Felsberg, M., Kahl, F.: Steerers: A framework for rotation equivariant keypoint descriptors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024)

9. Bökman, G., Kahl, F.: A case for using rotation invariant features in state of the art feature matchers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2022)
10. Bökman, G., Kahl, F.: Investigating how ReLU-networks encode symmetries. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)
11. Bökman, G., Nordström, D., Kahl, F.: Flopping for flops: Leveraging equivariance for computational efficiency. In: Forty-second International Conference on Machine Learning (2025)
12. Brehmer, J., Behrends, S., Haan, P.D., Cohen, T.: Does equivariance matter at scale? Transactions on Machine Learning Research (2025), https://openreview.net/forum?id=wilNute8Tn
13. Cahill, J., Iverson, J.W., Mixon, D.G., Packer, D.: Group-invariant max filtering. Foundations of Computational Mathematics pp. 1–38 (2024)
14. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I. p. 213–229. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-58452-8_13, https://doi.org/10.1007/978-3-030-58452-8_13
15. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: ICML (2016)
16. Cohen, T., Welling, M.: Steerable CNNs. In: ICLR (2017)
17. Dao, T.: FlashAttention-2: Faster attention with better parallelism and work partitioning. In: International Conference on Learning Representations (ICLR) (2024)
18. Darcet, T., Baldassarre, F., Oquab, M., Mairal, J., Bojanowski, P.: Cluster and predict latents patches for improved masked image modeling. Transactions on Machine Learning Research (feb 2025), published February 12, 2025
19. Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A.P., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme Ruiz, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., Steenkiste, S.V., Elsayed, G.F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M., Gritsenko, A.A., Birodkar, V., Vasconcelos, C.N., Tay, Y., Mensink, T., Kolesnikov, A., Pavetic, F., Tran, D., Kipf, T., Lucic, M., Zhai, X., Keysers, D., Harmsen, J.J., Houlsby, N.: Scaling vision transformers to 22 billion parameters. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 7480–7512. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/dehghani23a.html
20. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
21. Dieleman, S., De Fauw, J., Kavukcuoglu, K.: Exploiting cyclic symmetry in convolutional neural networks. In: International conference on machine learning. pp. 1889–1898. PMLR (2016)
22. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021)

23. Edstedt, J., Nordström, D., Zhang, Y., Bökman, G., Astermark, J., Larsson, V., Heyden, A., Kahl, F., Wadenbäck, M., Felsberg, M.: RoMa v2: Harder better faster denser feature matching. In: Proceedings of the European Conference on Computer Vision (ECCV) (2026)
24. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**, 303–338 (06 2010). https://doi.org/10.1007/s11263-009-0275-4
25. Ferraro, L., Galetto, F., Gandini, F., Huang, H., Mastroeni, M., Ni, X.: The invariantring package for macaulay2. Journal of Software for Algebra and Geometry **14**(1), 5–11 (2024)
26. Fuchs, F.B., Worrall, D.E., Fischer, V., Welling, M.: Se(3)-transformers: 3d roto-translation equivariant attention networks. In: Advances in Neural Information Processing Systems 34 (NeurIPS) (2020)
27. Grayson, D.R., Stillman, M.E.: Macaulay2, a software system for research in algebraic geometry. Available at http://www2.macaulay2.com
28. He, L., Chen, Y., shen, z., Dong, Y., Wang, Y., Lin, Z.: Efficient equivariant network. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 5290–5302. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper_files/paper/2021/file/2a79ea27c279e471f4d180b08d62b00a-Paper.pdf
29. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
30. Hutchinson, M.J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y.W., Kim, H.: Lietransformer: Equivariant self-attention for lie groups. In: Proceedings of the 38th International Conference on Machine Learning (ICML). pp. 4533–4543. PMLR (2021)
31. Kaba, S.O., Mondal, A.K., Zhang, Y., Bengio, Y., Ravanbakhsh, S.: Equivariance with learned canonicalization functions. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 15546–15566. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/kaba23a.html
32. Kakarala, R.: The bispectrum as a source of phase-sensitive invariants for fourier descriptors: a group-theoretic approach. Journal of Mathematical Imaging and Vision **44**, 341–353 (2012)
33. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020)
34. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., Girshick, R.: Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4015–4026 (October 2023)
35. Klee, D., Park, J.Y., Platt, R., Walters, R.: A comparison of equivariant vision models with imagenet pre-training. In: NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations (2023), https://openreview.net/forum?id=3ItzNHPov9
36. Koch, V., Wagner, S.J., Kazeminia, S., Sancar, E., Hehr, M., Schnabel, J.A., Peng, T., Marr, C.: Dinobloom: a foundation model for generalizable cell embeddings in hematology. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 520–530. Springer (2024)

37. Kundu, S., Kondor, R.: Steerable transformers for volumetric data. In: Forty-second International Conference on Machine Learning (2025)
38. Liao, Y.L., Smidt, T.: Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In: International Conference on Learning Representations (2023), https://openreview.net/forum?id=KwmPfARgOTD
39. Matek, C., Krappe, S., Münzenmayer, C., Haferlach, T., Marr, C.: An expert-annotated dataset of bone marrow cytology in hematologic malignancies. Data set (2021). https://doi.org/10.7937/TCIA.AXH3-T579, https://doi.org/10.7937/TCIA.AXH3-T579
40. Nordström, D., Edstedt, J., Bökman, G., Astermark, J., Heyden, A., Larsson, V., Wadenbäck, M., Felsberg, M., Kahl, F.: LoMa: Local feature matching revisited. In: Proceedings of the European Conference on Computer Vision (ECCV) (2026)
41. Nordström, D., Edstedt, J., Kahl, F., Bökman, G.: MuM: Multi-view masked image modeling for 3d vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2026)
42. Nordström, D., Edstedt, J., Kahl, F., Bökman, G.: Who handles orientation? Investigating invariance in feature matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2026)
43. NVIDIA: cuEquivariance: High-performance equivariant neural networks, https://docs.nvidia.com/cuda/cuequivariance/index.html
44. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.Y., Li, S.W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning robust visual features without supervision. Transactions on Machine Learning Research (2024), https://openreview.net/forum?id=a68SUt6zFt, featured Certification
45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
46. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning (2021), https://api.semanticscholar.org/CorpusID:231591445
47. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do ImageNet classifiers generalize to ImageNet? In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 5389–5400. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/recht19a.html
48. Rojas-Gomez, R.A., Lim, T.Y., Do, M.N., Yeh, R.A.: Making vision transformers truly shift-equivariant. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5568–5577 (June 2024)
49. Romero, D., Bekkers, E., Tomczak, J., Hoogendoorn, M.: Attentive group equivariant convolutional networks. In: International Conference on Machine Learning. pp. 8188–8199. PMLR (2020)

50. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)
51. Sanborn, S., Miolane, N.: A general framework for robust g-invariance in g-equivariant networks. Advances in Neural Information Processing Systems **36**, 67103–67124 (2023)
52. Serre, J.P.: Linear Representations of Finite Groups, Graduate Texts in Mathematics, vol. 42. Springer, New York, NY (1977). https://doi.org/10.1007/978-1-4684-9458-7, http://link.springer.com/10.1007/978-1-4684-9458-7
53. Siméoni, O., Vo, H.V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., Bojanowski, P.: DINOv3 (2025), https://arxiv.org/abs/2508.10104
54. Tillet, P., Kung, H.T., Cox, D.: Triton: an intermediate language and compiler for tiled neural network computations. In: Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages. p. 10–19. MAPL 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3315508.3329973, https://doi.org/10.1145/3315508.3329973
55. Touvron, H., Cord, M., Jégou, H.: Deit iii: Revenge of the vit. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 516–533. Springer Nature Switzerland, Cham (2022)
56. Vadgama, S., Islam, M.M., Buracus, D., Shewmake, C., Bekkers, E.: On the utility of equivariance and symmetry breaking in deep learning architectures on point clouds. arXiv preprint arXiv:2501.01999 (2025)
57. Van Horn, G., Cole, E., Beery, S., Wilber, K., Belongie, S., Mac Aodha, O.: Benchmarking representation learning for natural world image collections. In: Computer Vision and Pattern Recognition (2021)
58. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: Vggt: Visual geometry grounded transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2025)
59. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: Dust3r: Geometric 3d vision made easy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20697–20709 (June 2024)
60. Wang, Y., Elhag, A.A., Jaitly, N., Susskind, J.M., Bautista, M.Á.: Swallowing the bitter pill: Simplified scalable conformer generation. In: International Conference on Machine Learning. pp. 50400–50418. PMLR (2024)
61. Weiler, M., Cesa, G.: General $E(2)$-equivariant steerable CNNs. In: NeurIPS (2019), https://proceedings.neurips.cc/paper/2019/file/45d6637b718d0f24a237069fe41b0db4-Paper.pdf
62. Wightman, R.: Pytorch image models. https://github.com/rwightman/pytorch-image-models (2019). https://doi.org/10.5281/zenodo.4414861
63. Xu, R., Yang, K., Liu, K., He, F.: $e(2)$-equivariant vision transformer. In: Uncertainty in Artificial Intelligence. pp. 2356–2366. PMLR (2023)
64. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12104–12113 (June 2022)
65. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)

66. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
67. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. International Journal of Computer Vision **127**(3), 302–321 (2019)