LEARNING POLYNOMIAL PROBLEMS WITH $SL(2,\mathbb{R})$-EQUIVARIANCE

Hannah Lawrence* **& Mitchell Tong Harris***
Massachusetts Institute of Technology

ABSTRACT

Optimizing and certifying the positivity of polynomials are fundamental primitives across mathematics and engineering applications, from dynamical systems to operations research. However, solving these problems in practice requires large semidefinite programs, with poor scaling in dimension and degree. In this work, we demonstrate for the first time that neural networks can effectively solve such problems in a data-driven fashion, achieving tenfold speedups while retaining high accuracy. Moreover, we observe that these polynomial learning problems are equivariant to the non-compact group $SL(2,\mathbb{R})$, which consists of area-preserving linear transformations. We therefore adapt our learning pipelines to accommodate this structure, including data augmentation, a new $SL(2,\mathbb{R})$-equivariant architecture, and an architecture equivariant with respect to its maximal compact subgroup, $SO(2,\mathbb{R})$. Surprisingly, the most successful approaches in practice do not enforce equivariance to the entire group, which we prove arises from an unusual lack of architecture universality for $SL(2,\mathbb{R})$ in particular. A consequence of this result, which is of independent interest, is that there exists an equivariant function for which there is no sequence of equivariant polynomials multiplied by arbitrary invariants that approximates the original function. This is a rare example of a symmetric problem where data augmentation outperforms a fully equivariant architecture, and provides interesting lessons in both theory and practice for other problems with non-compact symmetries.

1 INTRODUCTION

Machine learning has emerged as a powerful tool for accelerating classical but time-consuming algorithms in scientific domains. In PDE time-stepping and molecular dynamics, for instance, productive lines of work have arisen around learning to forward-step in time or regress unknown parameters (Batzner et al., 2022; Han et al., 2018). These problems are solvable by existing methods, but can be significantly sped-up by training a neural net on a dataset of interest. Machine learning is especially valuable for problems for which, once a solution has been found, its correctness can be verified efficiently; this is the case for e.g. modeling catalysts, where a candidate output conformation can be confirmed with a few DFT steps (Chanussot et al., 2021).

Inspired by this line of research, we hypothesize that machine learning may allow for learning from specialized datasets of *polynomials*. Polynomials are ubiquitous in mathematics and engineering, and primitives such as minimizing a polynomial (Parrilo & Sturmfels, 2003; Jiang, 2013; Passy & Wilde, 1967; Nie, 2013; Shor, 1998) or certifying its positivity (Lasserre, 2001; Prestel & Delzell, 2013; Parrilo, 2000b), have attracted attention due to their direct applications to diverse problems ranging from operations research to control theory (Henrion & Garulli, 2005; Tedrake et al., 2010). Examples include certifying the stability of a dynamical system with a Lyapunov function, robot path planning, and designing nonlinear controllers (Ahmadi & Majumdar, 2016). Although there exist classical methods (ApS, 2022) for solving these problems, they are often computationally expensive.

A particularly attractive polynomial task for machine learning, which we discuss in further detail in Section 3, is predicting a matrix certificate of nonnegativity. This certificate can be efficiently checked independently of the neural network, and if the matrix is positive semidefinite (PSD), it *guarantees*

*Author order determined by coin flip.

that the original polynomial is nonnegative. Thus, one does not need to trust the trained network's accuracy; the utility of a certificate, once produced by any means, can be verified efficiently.

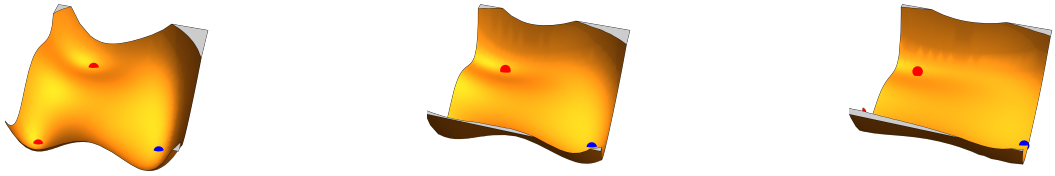
A trained neural network could not hope to outperform an existing semidefinite programming (SDP)-based technique on all instances; rather, we assume that there is an application-specific *distribution* of inputs (polynomials), and try to learn the shared attributes of such instances that may make a more efficient solution possible. This is similar to the assumption made when searching for sum of squares (SOS) certificates for nonnegativity: in general, it is NP hard to certify that a polynomial is nonnegative (Murty & Kabadi, 1985), but one could still search for a SOS certificate of low degree via SDPs, as we discuss in Section 3. One simple example of a class with exploitable structure could be polynomials all of the form $p_i(x) = s(x)^2 + c_i$, for $s(x)$ a real-rooted polynomial and constants c_i. If consistently present in a dataset, this structure can be leveraged to quickly compute global optima. Instead of discovering such features by hand, we *learn* them with a neural solver — ideally more quickly than off-the-shelf algorithms for minimization, positivity certification, and beyond.

In other contexts of structured inputs, it is now common practice to adapt neural architectures to the invariances of the input and output data types (Cohen & Welling, 2016). For example, architectures for point clouds are usually agnostic to translation, rotation, and permutation (Thomas et al., 2018a; Anderson et al., 2019), imparting improved sample efficiency and generalization (Elesedy & Zaidi, 2021). How can we design learning pipelines which take into account the underlying symmetries of *polynomial* learning tasks? In particular, the majority of polynomial tasks transform predictably under any linear change of variables. For example, the minimizer of a polynomial tranforms equivariantly with respect to linear transformations of the input variables, while the minimum value itself is invariant (Figure 1). Instead of working with the entire general linear group, we restrict our attention to the subgroup $SL(2, \mathbb{R})$. The special linear group $SL(2, \mathbb{R})$ consists of 2×2 matrices with determinant 1, which are sometimes called "area-preserving". Although the group is non-compact, which entails many difficulties (e.g. there is no finite invariant measure over a non-compact group, precluding the "lifting and convolving" approach of Finzi et al. (2020); see also Appendix C.1), the irreducible representations and the Clebsch-Gordan decomposition (defined in Section 4.2) provide building blocks for an equivariant architecture (Bogatskiy et al., 2020). In the case of $SL(2, \mathbb{R})$, they are both well-understood and naturally relate to polynomials. As a result, we can define an architecture that achieves *exact* equivariance to area-preserving linear transformations — subject only to numerical error from finite precision, and not to Monte Carlo integral approximations as in most previous work on Lie group symmetries (Finzi et al., 2020; MacDonald et al., 2021). To the best of our knowledge, this is the *only* known architectural technique to achieve exact $SL(2, \mathbb{R})$-equivariance.

However, we make a surprising discovery: although this well-established method (taking tensor products and linear combinations of irreps) can approximate any equivariant *polynomial*, it cannot approximate any equivariant *function*. In other words, we prove that there exists an $SL(2, \mathbb{R})$-equivariant function for which there is no converging sequence of equivariant polynomials multiplied by arbitrary invariants (Corollary 1), a result which may be of independent interest. Moreover, the function (equation 1) may be the very function one wants to approximate in positivity verification applications. Because of this impossibility result, we empirically explore other equivariance-inspired techniques, which obtain promising experimental results: data augmentation over well-conditioned subsets of $SL(2, \mathbb{R})$, as well as an equivariant architecture with respect to the subgroup $SO(2, \mathbb{R})$.

1.1 Our Contributions

In this work, our main contributions are threefold. First, we apply machine learning methods to certain polynomial problems that interest the optimization and control communities: positivity verification and minimization. To the best of our knowledge, this is the first application of machine learning to these problems. Second, we implement the first exactly $SL(2, \mathbb{R})$-equivariant architecture, which is universal over equivariant polynomials. Third, and perhaps most surprisingly, we prove that there exists an $SL(2, \mathbb{R})$-equivariant function for which there is no converging sequence of $SL(2, \mathbb{R})$-equivariant polynomials. This result indicates that no tensor product-based architecture can attain universality over $SL(2, \mathbb{R})$-equivariant functions. We therefore compare the performance of other equivariance-inspired techniques that do not enforce strict equivariance to the entire symmetry group. By closely evaluating equivariance with respect to $SL(2, \mathbb{R})$ on a concrete, richly-understood set of application problems, we provide a cautionary note on the incorporation of non-compact symmetries, and an invitation to lean more heavily on data augmentation in such cases.



(a) $g = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ (b) $g = \begin{pmatrix} 1.2 & 0 \\ 0 & \frac{1}{1.2} \end{pmatrix}$ (c) $g = \begin{pmatrix} 1.4 & 0 \\ 0 & \frac{1}{1.4} \end{pmatrix}$

Figure 1: **The action of $SL(2,\mathbb{R})$ on polynomials.** Each plot is the result of applying some $g \in SL(2,\mathbb{R})$ to the degree 6 bivariate polynomial $\left((x-3)^2+(y-3)^2\right)\left((x+2)^2+\frac{1}{4}(y-2)^2\right)\left(\frac{1}{2}(x-2)^2+(y+2)^2\right)-8\left(x^2+y^2\right)$, stretching or compressing the polynomial along each coordinate axis. The local minima are marked in red, and the global minimum in blue. While the minimum value is invariant, the minimizer transforms equivariantly under $SL(2,\mathbb{R})$.

1.2 Related Work

Equivariant Learning Architectures enforcing equivariance to *compact* Lie groups abound in both theory and practice, often focusing on $SO(3)$ as it arises widely in practice. To name just a few, Cohen et al. (2018) enforces spherical equivariance using real-space nonlinearities, while Kondor et al. (2018) uses the tensor product nonlinearity. Several architectures designed for rotation-equivariant point cloud inputs also use tensor product nonlinearities (Thomas et al., 2018a; Anderson et al., 2019). Methods for more general Lie groups based on Monte Carlo approximations of convolution integrals include Finzi et al. (2020) and MacDonald et al. (2021), while Finzi et al. (2021) presents a computational method for solving for an equivariant linear layer of an arbitrary Lie group. Most relevant to our architecture is Bogatskiy et al. (2020), which introduces a Lorentz-group equivariant architecture closely related to ours; we discuss the relationship in more detail in Section 4, but note that our application area (and therefore the output layer), empirical findings, and universality properties are distinct. Finally, Gerken et al. (2022) provides one other empirical example where augmentation outperforms equivariance. However, they observe this phenomenon only for invariant tasks with respect to the compact group of rotations, whereas we work with a task equivariant to a non-compact group, and therefore hypothesize that the underlying mechanisms are quite different.

Polynomial problems Interest in polynomial problems is driven in part by the ubiquity of their applications. One application is to find a Lyapunov function to certify stability of a system, which can be accomplished with a semidefinite programming (SDP) solver (Parrilo, 2000a;b). SDP solvers are effective when the dimensions are not too large (Mittelmann, 2003). A different method to prove polynomial nonnegativity on an interval is to write the polynomial in the Bernstein basis (Farouki, 2012; Harris & Parrilo, 2023), which has been used to solve problems of robust stability (Garloff, 2000; Garloff et al., 1997; Malan et al., 1992; Zettler & Garloff, 1998). When the interval is not compact, however, the results are sensitive to the particular nonlinear map of the original interval to $[0,1]$. For global polynomial minimization, a comparison of abstract approaches, including sum of squares-based methods, are given in Parrilo & Sturmfels (2003). Methods such as grid refinement and cutting plane algorithms can be used for many applications (Hettich & Kortanek, 1993), but the results are not certifiably feasible. One such application is in designing filter banks for signal processing. Perfect reconstruction is a desirable property of a filter bank, and is known to be equivalent to nonnegativity of a univariate trigonometric polynomial (Kortanek & Moulin, 1998; Moulin et al., 1997).

2 Preliminaries

Special Linear Group Recall that a group G is a set closed under an associative binary operation, for which an identity element e and inverses exist. The group $SL(2,\mathbb{R})$ consists of 2×2 real-valued matrices of determinant 1. The condition number of a matrix M is the ratio of its maximum to its minimum singluar values. (Note that $SL(2,\mathbb{R})$ contains arbitrarily poorly-conditioned matrices, as

e.g. diagonal matrices with entries a and $\frac{1}{a}$ have condition number a^2.) G acts on a set $\mathcal{X}$ if for every $g, h \in G$ and $x \in \mathcal{X}$, $g(hx) = (gh)x$ and $ex = x$. In this case, the orbit of $x_0 \in \mathcal{X}$ is $\{gx_0 | g \in G\}$. A homogeneous polynomial, or *form*, is one in which every term has the same degree. We are concerned with homogeneous polynomials in two real variables, or *binary forms*. Let $V(d)$ denote the vector space of binary forms of degree d, which can be identified with $\mathbb{R}^{d+1}$. For an element $p \in V(d)$, we may write it as p or $p(\vec{x})$. The action of $g \in SL(2, \mathbb{R}) \subset \mathbb{R}^{2\times 2}$ on a binary form p is defined in Section 4.1 and shown in Figure 1.

Representation Theory A *representation* of a group G is a vector space V together with a map $\rho : G \to GL(V)$ satisfying $\rho(g_1)\rho(g_2) = \rho(g_1 g_2)$ for all $g_1, g_2 \in G$. An *irreducible representation*, or irrep, is a representation for which there does not exist a subspace $W \subseteq V$ satisfying $\rho(g)w \in W$ for all $g \in G, w \in W$. The irreps are crucial tools for understanding a group's other representations. The tensor product of representations (V_1, ρ_1) and (V_2, ρ_2) is given by $\rho : G \to GL(V_1 \otimes V_2)$, $\rho(g) = \rho_1(g) \otimes \rho_2(g)$, and is itself a representation. A reducible representation V_3 may be decomposed into a direct sum of irreps V_1 and V_2. When V_3 is itself a tensor product of irreps, this is known as the Clebsch-Gordan decomposition. The relation $V_3 \simeq V_1 \oplus V_2$ means that there exists an invertible, equivariant, linear map $T : V_3 \to V_1 \oplus V_2$. An *equivariant* map with respect to a group G is a map $f : V_1 \to V_2$ between representations (V_1, ρ_1) and (V_2, ρ_2) satisfying $f(\rho_1(g)v_1) = \rho_2(g)f(v_1)$ $\forall\, g \in G$, $v_1 \in V_1$.

3 TASK: SUM OF SQUARES CERTIFICATE OF POLYNOMIAL NONNEGATIVITY

For the first time, we propose machine learning for learning properties of a polynomial input. Many questions about polynomials, as discussed in Section 1, are invariant under a change of coordinates given by $SL(2, \mathbb{R})$. In this section, we introduce a widely applicable example, positivity verification, and we defer polynomial minimization to Appendix B. [1]

We discuss the following problem in depth because even though the problem can be solved with convex optimization, classical methods are slow for even moderately high degree or dimension polynomials, so an efficient learned solution could be useful in practice. Moreover, many problems about polynomials can be reduced to certifying nonnegativity (e.g. proving α is a lower bound on p is equivalent to showing $p - \alpha \geq 0$).

One method to prove that a polynomial is nonnegative is by the semidefinite programming (SDP) method described in Parrilo (2000b); Lasserre (2001). Define the $d-$lift as $\vec{x}^{[d]} = \begin{pmatrix} y^d & xy^{d-1} & \cdots & x^d \end{pmatrix}^T$. Suppose $p(\vec{x}) = \vec{x}^{[d]^T} Q \vec{x}^{[d]}$ for some positive semidefinite $(d+1) \times (d+1)$ symmetric matrix Q. $\vec{x}^{[d]^T} Q \vec{x}^{[d]}$ is nonnegative by definition, so the equality to $p(\vec{x})$ implies p is nonnegative for all x and y. The problem of finding such a Q is traditionally relegated to a convex optimization interior point solver. One convenient formulation given to a solver is

$$f(p) = \text{argmax} \log\det Q \text{ such that } p(\vec{x}) = \vec{x}^{[d]^T} Q \vec{x}^{[d]} \text{ and } Q \succeq 0, \tag{1}$$

because this objective finds the analytic center (Boyd & Vandenberghe, 2004, §8.5) of the feasible region, which conveniently avoids the boundary of the feasible region. If p is positive on $\mathbb{R}^2 \setminus \{(0,0)\}$, then f is well-defined as the optimization problem has a unique maximizer. The analytic center is a good choice precisely because of its scale-invariance and the following equivariance property.

For $A \in SL(2, \mathbb{R})$, define the *induced* matrix $A^{[d]}$ as the $(d+1) \times (d+1)$ matrix for which $(A\vec{x})^{[d]} = A^{[d]}\vec{x}^{[d]}$, as discussed in Marcus & Minc (1992); Marcus (1973); Parrilo & Jadbabaie (2008). The induced matrices describe how the irreps act; therefore, they are analogous to the Wigner d-matrices for $SO(3)$. Viewing a polynomial as an inner product between a basis and a coefficient vector, we get the important relation that $(A\vec{x})^T p = \vec{x}^{[d]^T}(A^{[d]^T} p)$. Therefore, the optimizer $f(p)$ has the equivariance property $f(p(A\vec{x})) = A^{[d]^T} f(p(\vec{x}))A^{[d]}$.

This application is well-suited for machine learning because we often only care if there exists a positive semidefinite Q. If the model predicts some matrix satisfying $p(\vec{x}) = \vec{x}^{[d]^T} Q \vec{x}^{[d]}$ and $Q \succeq 0$, then p is automatically nonnegative; it is simple to validate this certificate of nonnegativity. This

[1] While this problem makes sense in higher dimensions (with $SL(d, \mathbb{R})$), we restrict to $d = 2$ for simplicity and defer higher dimensions to future work.

feature is unusual for machine learning applications, but highly advantageous. Even in safety-critical systems, a pipeline could include a neural network whose output is then certified.

4 $SL(2,\mathbb{R})$-EQUIVARIANT ARCHITECTURE

The equivariance property of f in equation 1 suggests using an equivariant learning technique. A fully equivariant architecture for $SL(2,\mathbb{R})$ has not been previously presented. Indeed, since $SL(2,\mathbb{R})$ is a noncompact group, forming invariants by group averaging is impossible, as there is no finite invariant measure over $SL(2,\mathbb{R})$(see Appendix C.1 for details). However, there is nonetheless a straightforward strategy for equivariance. We adopt the architecture for noncompact groups given by Bogatskiy et al. (2020), which follows the established framework of taking tensor products between irreps and then applying the Clebsch-Gordan transformation (Thomas et al., 2018b; Kondor et al., 2018). Our fully equivariant $SL(2,\mathbb{R})$ architecture can be viewed as a natural specialization of their insightful Lorentz group architecture. In fact, the exposition simplifies dramatically because our binary form inputs lie precisely in the group's finite-dimensional irreps. Moreover, the Clebsch-Gordan decomposition is given by the classical transvectant, which has been well-studied in the invariant theory literature (see Section 4.1).

In Section 4.1, we provide the necessary background on the representation theory of $SL(2,\mathbb{R})$. In Section 4.2, we describe the $SL(2,\mathbb{R})$-equivariant architecture. In Section 4.3, however, we prove that **universality over equivariant polynomials does not provide universality over arbitrary smooth equivariant functions for $SL(2,\mathbb{R})$.** This is a surprising and general impossibility result, which contrasts with Bogatskiy et al. (2020) and casts doubt on the possibility of any universal $SL(2,\mathbb{R})$-equivariant architecture.

4.1 REPRESENTATION THEORY OF $SL(2,\mathbb{R})$

For the purposes of this paper, we are interested in finite-dimensional representations of $SL(2,\mathbb{R})$. For every positive integer d, the $(d+1)$-dimensional vector space of degree d binary forms ($V(d)$), is an irreducible representation of $SL(2,\mathbb{R})$. Throughout, we assume a monomial basis for $V(d)$ (i.e. x^d, $x^{d-1}y, \ldots, y^d$) and represent elements of $V(d)$ as vectors of coefficients in $\mathbb{R}^{d+1}$. Let $p \in V(d)$, and define the corresponding action on $V(d)$ by $\rho_d(g)p(\vec{x}) = p(g^{-1}\vec{x})$. These polynomial representations are all of $SL(2,\mathbb{R})$'s finite-dimensional irreps. Polynomial inputs are particularly well-suited to $SL(2,\mathbb{R})$-equivariance because they lie in the finite-dimensional irreducible representations.

Given two of these finite dimensional representations, $V(d_1)$ and $V(d_2)$, the tensor product representation decomposes as $V(d_1) \otimes V(d_2) \simeq \bigoplus_{n=0}^{\min(d_1,d_2)} V(d_1+d_2-2n)$. The linear map verifying this isomorphism is an invertible, equivariant map we call T, which is the Clebsch-Gordan decomposition for $SL(2,\mathbb{R})$ (e.g. (Böhning & Bothmer, 2010)). A linear map is fully specified when defined on a basis, which we now do for T. Let $p(x_1,y_1) \in V(d_1)$ and $q(x_2,y_2) \in V(d_2)$. The d_1+d_2-2n component of $T(p \otimes q)$ is given by the classical nth order *transvectant* (see e.g. Olver (1999, §5)):

$$\psi_n(p,q) := \left(\left(\frac{\partial^2}{\partial x_1 \partial y_2} - \frac{\partial^2}{\partial x_2 \partial y_1}\right)^n p \cdot q\right)\Bigg|_{\substack{x=x_1=x_2 \\ y=y_1=y_2}} = \sum_{m=0}^{n} (-1)^m \binom{n}{m} \frac{\partial^n p}{\partial x^{n-m} \partial y^m} \frac{\partial^n q}{\partial x^m y^{n-m}} \tag{2}$$

where other authors, e.g. Böhning & Bothmer (2010), sometimes include a constant factor depending on the degrees and n. This is the isomorphism between the tensor product space and the ordinary spaces of binary forms.

4.2 ARCHITECTURE DETAILS

Our equivariant architecture is described in Algorithm 1 and in Figure 3. Each layer can be described by the nonlinear part and the linear part, and we describe each below. The inputs are elements of the finite dimensional irrep spaces of $SL(2,\mathbb{R})$.

Nonlinearity Each input channel has a list of polynomials, which we represent with vectors in $\mathbb{R}^d$. We compute all pairwise tensor products and decompose these tensor products with T from Section 4.1, which gives a list of polynomials. We additionally apply a multi-layer perceptron ("LearnedMLP" in Algorithm 1) to any degree 0 polynomials.

Algorithm 1 $SL(2,\mathbb{R})$-equivariant architecture

Input: $p_{init} \in \mathbb{R}^{d+1}$ where $\mathbb{R}^{d+1} \cong V(d)$, $d_1, \ldots, d_L$
Output: $M \in S^{\frac{d}{2}+1}$

$p_d = p_{init}$; `inputDegrees` $= \{d\}$; `outputDegrees` $= \{\}$ ▷ Irrep bookkeeping
for layer $\ell = 0, \ldots, L-1$ **do**
  Init $Q_r = [\,]$ for all r
  for i, j in `inputDegrees` **do**
    for $n = 0, \ldots, \min(i,j)$ **do**
      Append $\psi_n(p_i, p_j) \in \mathbb{R}^{i+j-2n}$ to Q_{i+j-2n} ▷ Nonlinearity and CG Decomp.
      Append $i + j - 2n$ to `outputDegrees`
  $p_0 = \text{LearnedMLP}([Q_0, p_0])$ ▷ MLP on Invariants + Skip Connection
  for i in `outputDegrees` $\setminus \{0\}$ **do**
    $p_i = \text{LearnedLinearCombination}(Q_i, p_i)$ ▷ Linear Layer + Skip Connection
  `inputDegrees`=`outputDegrees`; `outputDegrees`={}
Return $L(p_0, p_1, \ldots, p_{d-1}, p_{init})$, as defined below ▷ Final Linear Layer

Learned linear layer After the Clebsch-Gordan (CG) decomposition T, we have a collection of polynomials ranging from degree 0 to degree $2d$. Gather all resulting polynomials of degree k (from any tensor product and across any channel) into the vector of polynomials v. Let $\#k$ be the number of such polynomials. Let c be the number of output channels. W is a $c \times \#k$ learnable matrix. Then Wv are the inputs of degree k for the next layer. This operation is denoted by "LearnedLinearCombination" in Algorithm 1, where we additionally append to v the input to the layer of degree k as a skip connection.

Last layer (general) The inputs to the last layer L are elements of the irreducible representation spaces, and the output lies in the vector space corresponding to some finite-dimensional representation. In other words, $L : \bigoplus_{n=0}^{d} V(n) \to V$ for some reducible representation ρ with associated vector space V. Informally, Schur's Lemma describes the set of equivariant linear maps L with this property as those which map linearly from the input irrep space to only matching irreps in V; see e.g. Stiefel & Fässler (2012, §2) for details. Although this methodology on its own is quite standard, in the next paragraph, we show a convenient way of choosing such an L for positivity verification.

Last layer (positivity verification) Suppose (for this paragraph) the input polynomial is degree $2d$. The output space is $V = S^{d+1}$, symmetric $(d+1) \times (d+1)$ matrices, which decomposes into irreps as $\bigoplus_{k=0}^{d} V(2k)$ (the odd irreps do not contribute to symmetric matrices). An equivariant, linear map L from $\bigoplus_{k=0}^{d} V(2k)$ to S^{d+1} (which is unique up to a linear combination of channels by a real version of Schur's Lemma (Behboodi et al., 2022, §8)) can be conveniently precomputed using the transvectant; see Section A.1. For our application, we need $p(\vec{x}) = \vec{x}^{[d]^T} Q \vec{x}^{[d]}$. Although this may seem difficult to enforce, there is an elegant solution: we overwrite the degree $2d$ irrep with the input polynomial before computing L. This yields the desired property, because it can be shown (see Lemma 3) that $x^{[d]^T} L(0, \ldots, p_i, \ldots, 0) x^{[d]} = 0$ for any $i < 2d$, and $x^{[d]^T} L(0, \ldots, 0, p_{2d}) x^{[d]} = p_{2d}$.

This architecture can represent any equivariant polynomial in the following sense, which is a restatement of Lemma D.1 of Bogatskiy et al. (2020) (and a common primitive towards universality).

Lemma 1. *Let $p : V \to U$ be a polynomial that is equivariant with respect to $SL(2,\mathbb{R})$, where V and U are finite-dimensional representations of $SL(2,\mathbb{R})$. Then there exists a fixed choice of architectural parameters such that Algorithm 1 exactly computes p.*

4.3 Lack of universality

In this section, we prove that the architecture in Section 4.2 is not universal, in spite of Lemma 1. In particular, there are no parameter settings (representing an equivariant polynomial times an arbitrary invariant) that uniformly approximate the equivariant function f described in equation 1.

Theorem 1. *Let $\mathcal{N}_W(p)$ denote the output of Algorithm 1 with (learned) parameters W applied to the input p. Let f be the continuous, $SL(2,\mathbb{R})$-equivariant function defined in equation 1. There exists*

an input polynomial p, and an absolute constant $\epsilon > 0$ such that for any W, $|f(p) - \mathcal{N}_W(p)| > \epsilon$. Therefore, the architecture of Section 4.2 is not universal.

This demonstrates more broadly that the function f is not approximable by any sequence of equivariant polynomials (independent of a particular neural architecture). We highlight that this particular function is not an unnatural one dreamt up for the sake of this proof; rather, it is the very function we seek to approximate for positivity verification applications. We show that when $p = x^8 + y^8$, there is an algebraic obstruction to predicting $f(p)$ (see also Figure 4).

Definition 1. *Call the monomial $x^k y^r$* balanced mod b *if $k \equiv r \pmod b$. The polynomial p is* balanced mod b *if it is the weighted sum of monomials that are all balanced mod b.*

Lemma 2. *If f and g are balanced mod b, then for every n, $\psi_n(f, g)$ is balanced mod b.*

Proof. Suppose $f = \sum_i f_i$ and $g = \sum_j g_j$, where all f_i and g_j are monomoials balanced mod b. By bilinearity, $\psi_n(f, g) = \sum_{i,j} \psi_n(f_i, g_j)$. Therefore, it is enough to prove that $\psi_n(f_i, g_j)$ is the sum of balanced monomials. Let $f_i = x^{k_i} y^{r_i}$ and $g_j = x^{k_j} y^{r_j}$. A single term in the second definition of ψ_n in equation 2 gives a monomial proportional to

$$\frac{\partial^n (x^{k_i} y^{r_i})}{\partial x^{n-m} \partial y^m} \frac{\partial^n (x^{k_j} y^{r_j})}{\partial x^m \partial y^{n-m}} \propto x^{k_i + k_j - n} y^{r_i + r_j - n}. \tag{3}$$

Since $k_i \equiv r_i$ and $k_j \equiv r_j$, then $k_i + k_j - n \equiv r_i + r_j - n \pmod b$. □

Proof of Theorem 1. Since $p = x^8 + y^8$ is balanced mod 8, every polynomial output from every layer will be as well by Lemma 2. (Note that neither the MLP applied to the invariants nor linear combinations of existing monomials will introduce new monomials.) Therefore, the final degree 4 output is proportional to $x^2 y^2$, implying there is no x^4 or y^4 term. For the L described in Section 4.2, any linear combination of $L(x^8)$, $L(y^8)$, $L(x^4 y^4)$, $L(x^2 y^2)$, and $L(1)$ has a sparsity pattern of

$$\begin{pmatrix} \times & 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & \times & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ \times & 0 & 0 & 0 & \times \end{pmatrix}; \text{ however, } f(p) \approx \begin{pmatrix} 1 & 0 & \boxed{-1.563} & 0 & 1/3 \\ 0 & \boxed{3.126} & 0 & -8/3 & 0 \\ \boxed{-1.563} & 0 & 14/3 & 0 & \boxed{-1.563} \\ 0 & -8/3 & 0 & \boxed{3.126} & 0 \\ 1/3 & 0 & \boxed{-1.563} & 0 & 1 \end{pmatrix},$$

which is a rounded/truncated result from ApS (2022). The sparsity pattern can be checked via the transvectant definition of L. Notice that the boxed numbers in the true solution (right) must be zero for any output of $\mathcal{N}_W(x^8 + y^8)$, regardless of W. Therefore, no output of the neural net can have the correct sparsity pattern. □

In light of Lemma 1, the problem is not inherent to our specific architecture, because our architecture can represent any equivariant polynomial. Instead, the problem must fundamentally be that f cannot be approximated by equivariant polynomials. Therefore, we have the following corollary of potential independent interest, even outside of the machine learning community.

Corollary 1. *There exists a continuous, $SL(2, \mathbb{R})$-equivariant function and a compact subset of its domain for which no sequence of $SL(2, \mathbb{R})$-equivariant polynomials times $SL(2, \mathbb{R})$-invariant functions converge pointwise to the function on the subset.*

What is different? There are a few major differences between this setup and other related universality results. First, the function f is only defined for a subset of binary forms. Another difference is the equivariance to the real group $SL(2, \mathbb{R})$ and not $SL(2, \mathbb{C})$. Theorem 4.1 of Bogatskiy et al. (2020) states that the analog of the architecture in Section 4.2 for $SL(2, \mathbb{C})$ is universal over equivariant functions. More generally, it is common to prove universality via polynomial approximation (Yarotsky, 2018; Dym & Maron, 2021). What goes wrong with such proofs in the real case? One problem is that the algebra of invariants with respect to the group, and the algebra of invariants with respect to a *maximally compact subgroup*, may differ for the real-valued case, as noted by Haddadin (2021).[2]

[2]For example, $SO(2, \mathbb{R})$ is the maximal compact subgroup of $SL(2, \mathbb{R})$ (Lang, 1985, p. 19). However, $a + c$ is an invariant of the polynomial $ax^2 + bxy + cy^2$ under an $SO(2, \mathbb{R})$ action, but is not an invariant under an $SL(2, \mathbb{R})$ action.

Model Name	Description
mlp-aug-rots	MLP with rotation augmentations
mlp-aug-a-b	MLP with augmentations by $g^{[d]}$ and g has condition number κ s.t. $a \leq \kappa \leq b$
SO2Net-aug-a-b	$SO(2,\mathbb{R})$ equivariant network with data augmented by $g^{[d]}$ as above
SO2Net	$SO(2,\mathbb{R})$ equivariant architecture with no data augmentation
SL2Net	$SL(2,\mathbb{R})$ equivariant architecture with no data augmentation
MLP	multilayer perceptron with no data augmentation

Table 1: Model names and meanings

Implications of these results. Nearly all equivariant universality results of which we are aware rely on polynomial approximation as a crucial intermediate step (Yarotsky, 2018; Dym & Maron, 2021; Bogatskiy et al., 2020). Moreover, the exhibited hard function is exactly the one we wish to approximate in many applications. Therefore, it is unclear whether any $SL(2,\mathbb{R})$-equivariant architecture holds promise for this class of problems. Instead, we propose returning to *data augmentation*, as well as an architecture equivariant to $SL(2,\mathbb{R})$'s well-studied maximal compact subgroup, $SO(2,\mathbb{R})$. One might hope to use a clever combination of (non-polynomial) renormalization and $SO(2,\mathbb{R})$-equivariance to circumvent our impossibility result and recover full $SL(2,\mathbb{R})$-equivariance; however, we prove in Proposition 2 of Appendix A.2 that this is not possible. For the sake of completeness, we include the SL2Net architecture in our experiments, but find that it underperforms alternatives.

5 Experiments

Since learning on polynomial input data is a novel contribution, there do not yet exist standardized benchmarks. Therefore, we design our own tasks: polynomial minimization and positivity verification. We have released all data generation (as well as training) code, so that future research may build on these preliminary benchmarks. We defer an exploration of polynomial minimization to Appendix B.

Our experiments are divided based on the underlying distribution of polynomials used to generate synthetic data, described in further detail in Appendix B.2. The second distribution is a structured class of polynomials that are optimal for the Delsarte linear programming bound for the cardinality of a spherical code (Delsarte et al., 1977), providing a realistic distribution of "natural" polynomials. An interior point method (ApS, 2022) is used to generate the values of $f(p)$ in equation 1.

There are several practical considerations when augmenting by elements of $SL(2,\mathbb{R})$. First, since $SL(2,\mathbb{R})$ is not a compact group, *any data augmentation induces a distribution shift* (see Appendix C for a proof). This is in contrast to compact groups, where any datapoint in an orbit is just as likely as any other datapoint in that orbit. Therefore, $SL(2,\mathbb{R})$-equivariant methods aid primarily in out-of-distribution generalization, rather than in-distribution sample complexity. Further challenges associated with the non-compactness of $SL(2,\mathbb{R})$ are discussed in Appendix C.

In our experiments, we compare several instantiations of equivariant learning. The first is the $SL(2,\mathbb{R})$ equivariant architecture described in Section 4.2. The second is a class of multilayer perceptrons (MLPs). We train the MLPs on an *augmented* training distribution: starting with the same training set, we apply random $SL(2,\mathbb{R})$ transformations of the training polynomials. We control the conditioning of the augmentation matrices both to avoid numerical issues and to induce more or less dramatic distribution shifts to obtain several different trained MLPs. The third architecture is an $SO(2,\mathbb{R})$ equivariant model, whose details are described in Appendix A.3. We train additional versions of the $SO(2,\mathbb{R})$ equivariant architecture by including data augmentations of various condition numbers. A description of the model names are given in Table 1. Experiment details for all experiments, as well as equivariance tests for the various models, are included in Appendix B.

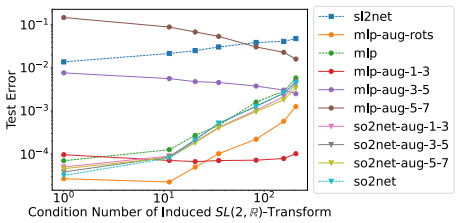
Timing Comparison: Trained Network vs Solver An impetus for turning to machine learning was to find faster ways of solving polynomial problems in practice. We compare the run time of a first order SDP solver with the wall time of a trained MLP on a CPU in Table 2. Since Mosek (ApS, 2022) was used to generate the data distribution, those values are used as ground truth. The MLP is about two orders of magnitude faster, while still very accurate, as reported in Table 2. While the true test will be for much higher degrees, where traditional SDP solvers become prohibitive, these results are strong experimental motivation to pursue machine learning for this task.

Degree	6	8	10	12	14
MLP NMSE	9.5e-6	6.0e-5	2.9e-5	2.3e-5	1.1e-5
MLP times (min)	0.062	0.082	0.17	0.22	0.29
SCS NMSE	2.7e-5	6.2e-5	1.2e-4	2.7e-4	1.2e-3
SCS times (min)	3.50	4.94	9.11	18.8	37.4

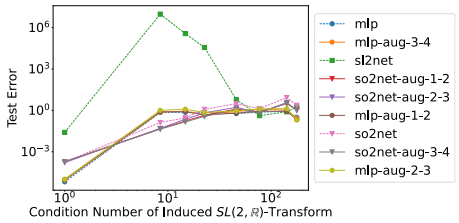
Table 2: Comparison of the MLP to the first order solver SCS (O'Donoghue et al., 2016). The mean squared errors are with respect to the ground truth labels computed by the second order solver, Mosek (ApS, 2022). The times are estimates for 5,000 test examples on a single CPU.

Comparison of Equivariance and Augmentation for Out-of-Distribution Generalization In Figure 2, we report the test errors for each of the models. We did not only test on the test dataset; we also randomly augmented the test dataset with $A \in SL(2, \mathbb{R})$ transformations. The average value of the condition number of $A^{[d]}$ for the transformations on the test set is labeled on the horizontal axis. The leftmost points are the test errors on the original test dataset. The rightmost points are the test errors resulting from transforming the test dataset with transformations with large condition numbers.

Observations On the untransformed test set, the best results are the MLP trained with augmentations close to rotations, followed by the SO2Net and unaugmented MLP. An advantage of the SO2Net is that it has about half as many parameters (see Appendix B.2 for model parameter counts). Using high condition numbers for the *training* augmentation of an MLP increases the test error of the untransformed dataset, validating our assertions in Appendix C.1 about distribution shifts induced by poorly conditioned matrices. Similarly, some models have this behavior when they encounter distribution shifts in the *test* set: the MLP, MLP augmented with rotations, and SO2Nets share the trend that the test error increases as the test distribution undergoes more transformations.



(a) Test errors for max det on random polynomials



(b) Test errors for max det on polynomials arising from spherical code bounds

Figure 2: **Test errors.** The test dataset is transformed by random $A \in SL(2, \mathbb{R})$. The average value of the condition numbers of $A^{[d]}$ for the transformations is labeled on the horizontal axis. The computed loss is the MSE normalized by the ground-truth labels' norm (from the SDP) and with an additive term to avoid dividing by zero, $\frac{\|\mathcal{N}(p)-f(p)\|_{\text{Fro}}^2}{1+\|f(p)\|_{\text{Fro}}^2}$, and averaged over 3 independent runs.

6 Conclusion

In this work, we demonstrated the promise of machine learning for polynomial problems, particularly with equivariant learning techniques. Even a simple MLP can vastly outperform an SDP solver in terms of speed, while equivariance-based approaches such as data augmentation and rotation-equivariance can improve out-of-distribution generalization. Surprisingly, although we construct an exactly $SL(2, \mathbb{R})$-equivariant architecture capable of representing any $SL(2, \mathbb{R})$-equivariant polynomial (demonstrating that one can overcome some of the superficial difficulties of equivariance to noncompact groups), it turns out that this does not suffice for universality. Therefore, any approach to equivariant architectures based on tensor products or polynomial approximation likely fails.

As future work, it would be interesting to consider paradigms for equivariance that sidestep the fundamental roadblock of polynomial approximation. One promising direction is frame averaging (Puny et al., 2021) for $SL(2, \mathbb{R})$. From the perspective of applications, our proof-of-concept experiments provide a strong signal to pursue the acceleration of polynomial optimization with machine learning.

ACKNOWLEDGMENTS

We thank Pablo Parrilo for inspiring the project and many productive conversations. HL is supported by the Fannie and John Hertz Foundation and the NSF Graduate Fellowship under Grant No. 1745302. MH is supported by the NSF Graduate Fellowhip under Grant No. 2141064.

REFERENCES

Amir Ali Ahmadi and Anirudha Majumdar. Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters*, 10:709–729, 2016.

Brandon Anderson, Truong-Son Hy, and Risi Kondor. *Cormorant: Covariant Molecular Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

MOSEK ApS. Mosek optimizer api for python. *Version*, 9(17):6–4, 2022.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1), 2022. doi: 10.1038/s41467-022-29939-5. URL https://par.nsf.gov/biblio/10381731.

Arash Behboodi, Gabriele Cesa, and Taco S Cohen. A pac-bayesian generalization bound for equivariant networks. *Advances in Neural Information Processing Systems*, 35:5654–5668, 2022.

Alexander Bogatskiy, Brandon Anderson, Jan T Offermann, Marwah Roussi, David W Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Christian Böhning and Hans-Christian Graf v Bothmer. A clebsch–gordan formula for sl3 (c) and applications to rationality. *Advances in Mathematics*, 224(1):246–259, 2010.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 11(10):6059–6072, 4 2021. doi: 10.1021/acscatal.0c04525.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2990–2999. JMLR.org, 2016.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *International Conference on Learning Representations (ICLR)*, 2018.

P. Delsarte, J. M. Goethals, and J. J. Seidel. Spherical codes and designs. *Geometriae Dedicata*, 6(3):363–388, 1977. doi: 10.1007/BF03187604. URL https://doi.org/10.1007/BF03187604.

Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. *ArXiv*, abs/2010.02449, 2021.

Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. *arXiv preprint arXiv:2102.10333*, 2021.

Rida T Farouki. The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3165–3176. PMLR, 2020.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3318–3328. PMLR, 2021. URL http://proceedings.mlr.press/v139/finzi21a.html.

Juergen Garloff, Birgit Graf, and Michael Zettler. Speeding up an algorithm for checking robust stability of polynomials. *IFAC Proceedings Volumes*, 30(16):183–188, 1997.

Jürgen Garloff. Application of bernstein expansion to the solution of control problems. *Reliable computing*, 6(3):303–320, 2000.

Jan Gerken, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. Equivariance versus augmentation for spherical images. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7404–7421. PMLR, 17–23 Jul 2022.

Ward Haddadin. Invariant polynomials and machine learning. *arXiv preprint arXiv:2104.12733*, 2021.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Mitchell Tong Harris and Pablo A Parrilo. Improved Nonnegativity Testing in the Bernstein Basis via Geometric Means. *arXiv preprint arXiv:2309.10675*, 2023.

Didier Henrion and Andrea Garulli. *Positive polynomials in control*, volume 312. Springer Science & Business Media, 2005.

Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM review*, 35(3):380–429, 1993.

Bo Jiang. *Polynomial optimization: structures, algorithms, and engineering applications*. University of Minnesota, 2013.

Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. *Advances in Neural Information Processing Systems*, 31:10117–10126, 2018.

KO Kortanek and Pierre Moulin. Semi-infinite programming in orthogonal wavelet filter design. *Semi-Infinite Programming*, pp. 323–360, 1998.

Serge Lang. *Sl 2 (r)*, volume 105. Springer Science & Business Media, 1985.

Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

Lachlan MacDonald, Sameera Ramasinghe, and Simon Lucey. Enabling equivariance for arbitrary lie groups. In *CVPR*, 11 2021.

Stefano Malan, Mario Milanese, Michele Taragna, and Jürgen Garloff. b^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic perturbations. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pp. 128–133. IEEE, 1992.

Marvin Marcus. *Finite dimensional multilinear algebra*, volume 23. M. Dekker, 1973.

Marvin Marcus and Henryk Minc. *A survey of matrix theory and matrix inequalities*, volume 14. Courier Corporation, 1992.

Hans D Mittelmann. An independent benchmarking of sdp and socp solvers. *Mathematical Programming*, 95(2):407–430, 2003.

Pierre Moulin, Mihai Anitescu, Kenneth O Kortanek, and Florian A Potra. The role of linear semi-infinite programming in signal-adapted qmf bank design. *IEEE Transactions on Signal Processing*, 45(9):2160–2174, 1997.

Katta G Murty and Santosh N Kabadi. Some np-complete problems in quadratic and nonlinear programming. Technical report, 1985.

Jiawang Nie. Polynomial optimization with real varieties. *SIAM Journal on Optimization*, 23(3): 1634–1646, 2013.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016. URL http://stanford.edu/~boyd/papers/scs.html.

Peter J Olver. *Classical invariant theory*. Number 44. Cambridge University Press, 1999.

Pablo A Parrilo. On a decomposition of multivariable forms via lmi methods. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, volume 1, pp. 322–326. IEEE, 2000a.

Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000b.

Pablo A Parrilo and Ali Jadbabaie. Approximation of the joint spectral radius using sum of squares. *Linear Algebra and its Applications*, 428(10):2385–2402, 2008.

Pablo A Parrilo and Bernd Sturmfels. Minimizing polynomial functions. *Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 60:83–99, 2003.

Ury Passy and DJ Wilde. Generalized polynomial optimization. *SIAM Journal on Applied Mathematics*, 15(5):1344–1356, 1967.

Alexander Prestel and Charles Delzell. *Positive polynomials: from Hilbert's 17th problem to real algebra*. Springer Science & Business Media, 2013.

Omri Puny, Matan Atzmon, Edward J. Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2021.

Naum Zuselevich Shor. *Nondifferentiable optimization and polynomial problems*, volume 24. Springer Science & Business Media, 1998.

E Stiefel and A Fässler. *Group theoretical methods and their applications*. Springer Science & Business Media, 2012.

Gabor Szegő. *Orthogonal polynomials*, volume 23. American Mathematical Soc., 1939.

Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8): 1038–1052, 2010.

Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018a. URL http://dblp.uni-trier.de/db/journals/corr/corr1802.html#abs-1802-08219.

Nathaniel Thomas, Tess E. Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018b. URL http://arxiv.org/abs/1802.08219.

Dian Wang, Jung Yeon Park, Neel Sortur, Lawson L. S. Wong, Robin Walters, and Robert Platt. The surprising effectiveness of equivariant models in domains with latent symmetry. *CoRR*, abs/2211.09231, 2022. doi: 10.48550/arXiv.2211.09231.

Dian Wang, Xupeng Zhu, Jung Yeon Park, Robert Platt, and Robin Walters. A general theory of correct, incorrect, and extrinsic equivariance. *CoRR*, abs/2303.04745, 2023. doi: 10.48550/arXiv.2303.04745.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55:407–474, 2018.

Michael Zettler and Jürgen Garloff. Robustness analysis of polynomials with polynomial parameter dependency using bernstein expansion. *IEEE Transactions on Automatic Control*, 43(3):425–431, 1998.