Relevance of Rotationally Equivariant Convolutions for Predicting Molecular Properties

Benjamin Kurt Miller
Freie Universität Berlin
Berlin, Germany
Lawrence Berkeley National Laboratory
Berkeley, California, USA
b.k.miller@uva.nl

Mario Geiger
EPFL
Lausanne, Switzerland
Lawrence Berkeley National Laboratory
Berkeley, California, USA
mario.geiger@epfl.ch

Tess E. Smidt
Lawrence Berkeley National Laboratory
Berkeley, California, USA
tsmidt@lbl.gov

Frank Noé
Freie Universität Berlin
Berlin, Germany
Rice University
Houston, Texas, USA
frank.noe@fu-berlin.de

Abstract

Equivariant neural networks (ENNs) are graph neural networks embedded in $\mathbb{R}^3$ and are well suited for predicting molecular properties. The ENN library `e3nn` has customizable convolutions, which can be designed to depend only on distances between points, or also on angular features, making them rotationally invariant, or equivariant, respectively. This paper studies the practical value of including angular dependencies for molecular property prediction directly via an ablation study with `e3nn` and the QM9 data set. We find that, for fixed network depth and parameter count, adding angular features decreased test error by an average of 23%. Meanwhile, increasing network depth decreased test error by only 4% on average, implying that rotationally equivariant layers are comparatively parameter efficient. We present an explanation of the accuracy improvement on the dipole moment, the target which benefited most from the introduction of angular features.

1 Introduction

The discovery of novel molecules has been accelerated by advances in computational quantum chemistry and machine learning assisted exploration of chemical space [1, 2, 3]. The successes have been characterized by designing bespoke neural networks which have relevant properties "baked-in," such as parameter sharing across calculations on individual atoms, continuous convolutions, invariance to atomic indexing, and invariance to rotation and translation [4, 5]. Meanwhile, there has also been development on neural networks which are equivariant to group action [6], some with molecules in mind [7, 8]. Equivariant neural networks can be seen as a super-set of invariant ones because a group necessarily contains the identity element. The question considered in this paper can loosely be stated as: When doing regression on scalar molecular properties, what is missing when one employs only invariant layers in a neural network as opposed to including equivariant ones?

We explore this question using the QM9 benchmark [9, 10] by predicting quantum chemical properties of small molecules. While the molecules can rotate and translate, affecting the molecule's position vectors, the QM9 properties are all scalar and invariant to translation or rotation. Here we compare equivariant neural networks (ENNs) that predict rototranslationally invariant molecular properties but

Machine Learning for Molecules Workshop at NeurIPS 2020. https://ml4molecules.github.io

differ by whether their internal features are rotationally invariant (convolutions depend on distances) or equivariant (convolutions depend on distances and angles). We also investigate whether increasing depth in networks with rotationally invariant layers is comparatively effective at reducing test error. The networks are implemented in the PyTorch [11] library e3nn [12] using the $SE(3)$ equivariant point modules. QM9 data handling and training routines were borrowed from SchNetPack [13].

Given atomic positions $\mathbf{r} \in \mathbb{R}^{3\times N}$ and atomic features F^h, layer h of an `e3nn` produces atomic features F^{h+1} by $\mathcal{L}^h(\mathbf{r}, F^h) = F^{h+1}$. F^h is a collection of u_0^h scalars $F_{\ell=0}^h$ and u_1^h vectors $F_{\ell=1}^h$ flattened into a column by $F^h = \text{vec}(F_{\ell=0}^h \oplus F_{\ell=1}^h)$. The total multiplicity of features at layer h is $u^h = u_0^h + u_1^h$. The rotation matrix $\mathbf{R}$ acts on F^h in block matrix notation by

$$\mathbf{R}F^h = \begin{pmatrix} \mathbf{R}_{\ell=0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\ell=1} \end{pmatrix} \begin{pmatrix} F_{\ell=0}^h \\ F_{\ell=1}^h \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\ell=1} \end{pmatrix} \begin{pmatrix} F_{\ell=0}^h \\ F_{\ell=1}^h \end{pmatrix}. \tag{1}$$

In this paper, we consider the case of rotationally invariant layers, which produce features that *do not rotate*, i.e. $u_1 = 0$, and compare their performance to rotationally equivariant layers, which produce *rotating* features, i.e. $u_1 \neq 0$. In order to predict a rotationally invariant target value, the output features, $F^{h_{max}} = F_{\ell=0}^{h_{max}}$, do not rotate. The difference between networks lies in the equivariance or invariance of their internal layers. We call networks containing only features that do not rotate and rotationally invariant layers L0Nets, while networks containing rotating features and equivariant layers are called L1Nets. A more general framing in terms of spherical harmonics can be found in the `e3nn` library [12] . In said framing, features are seen as spherical harmonics of degree ℓ.

1.1 Related Work

Molecular properties, which depend only on the atomic distance graph, are commonly predicted by kernel methods or Gaussian process regression [14, 15, 16, 17] or graph neural networks [18, 19], where ENNs are usually employed for predicting physical properties, which depend on atomic displacement vectors [20, 21]. While kernel approaches are more data-efficient, graph neural networks scale to larger amounts of data. Inspiration for our study came from literature on invariant and equivariant ENNs for molecular property prediction. SchNet [4, 13] introduced atom-wise features with continuous convolution. Tensor Field Networks [7] and Cormorant [8] generalized the approach to angular-feature based rotation equivariant networks. In parallel, although aimed at voxelized data, se3cnn [6] developed the gated nonlinearity. The library under consideration, `e3nn`, represents a superset of Tensor Field Networks, SchNet, and se3cnn. Support for Cormorant's so-called two-body interaction has also been included in `e3nn` but is not considered in this experiment.

Although DimeNet [22] is the leading architecture on QM9 regression, it is not considered in our analysis. Their use of directional message passing, while effective, is not trivially compatible with SchNet, Cormorant, or `e3nn`. Their edge featurization using spherical Fourier-Bessel functions could be incorporated rather simply, but investigation is left for future work.

While QM9 remains the gold standard for most machine learning studies, a new data set called QM7-X [23] contains a wealth of tensor properties suitable for prediction with networks like `e3nn` or Cormorant. SchNet and DimeNet cannot predict tensor quantities in their current incarnations.

2 Methods

We employ both an L0Net and an L1Net to do regression on scalar target values from the QM9 data set given molecular input data. A molecule is an unordered set of $N \in \mathbb{N}$ atoms, each with position $\mathbf{r}_a \in \mathbb{R}^3$ and element Z_a which is represented as a one-hot scalar array. We parameterize a neural network such that $\{(\mathbf{r}_1, Z_1), ..., (\mathbf{r}_N, Z_N)\} \mapsto \mathcal{F}(\mathbf{r}_1, ..., \mathbf{r}_N, Z_1, ..., Z_N)$ where we restrict the image to be invariant to rotations and translations, as well as permutations in atomic indexing. Every layer uses parameter sharing across atoms and the final step accumulates the value of every atom with a symmetric function. A schematic of the entire architecture can be seen in Figure 1.

Atom-wise A dense layer applied to every atom with parameters shared across atoms. Given weights $W_{u'u}$, bias b_u, non-rotating, scalar features F on atom A at layer h with multiplicity u' we write $F_{u\,A}^{h+1} = \sum_{u'} F_{u'\,A}^h W_{u'u} + b_{u'}$. This layer is also used as a learned embedding of the atomic element.

Radial Basis Function (rbf) The radial basis $\phi : \mathbb{R} \to \mathbb{R}^{\mathcal{B}}$, expands $d = \|\mathbf{r}_b - \mathbf{r}_a\|$ by

$$\phi(d) = \begin{cases} cos^2(\frac{\pi}{2}\frac{d-\mu_{\mathcal{B}}}{\mu_{\mathcal{B}+1}-\mu_{\mathcal{B}}}) & -1 \leq \frac{d-\mu_{\mathcal{B}}}{\mu_{\mathcal{B}+1}-\mu_{\mathcal{B}}} \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $0\,\text{Å} \leq \mu_{\mathcal{B}} \leq 11.1\,\text{Å}$ is a sequence of $\mathcal{B} = 84$ equally spaced "radial basis centers."

Convolution The convolutional filter f consists of a learned scalar array radial function multiplied by a multiplicity of spherical harmonics of degree ℓ_f. The filter is assigned an atomic index based on the atom on which it was evaluated. The filter and the atomic features interact which necessitates a double atomic indexing of A and B. The degree indices $\ell_{out}, \ell_{in}, \ell_f$ correspond to order indices i, j, k respectively. u, v both represent multiplicity. Using the input features F, Clebsch-Gordan coefficients C, filter spherical harmonics $Y(\frac{\mathbf{r}_B - \mathbf{r}_A}{\|\mathbf{r}_B - \mathbf{r}_A\|})$, learned scalar radial coefficients $R(\phi(\|\mathbf{r}_B - \mathbf{r}_A\|))$, and normalization coefficients n, the convolutional output $\tilde{F}$ is defined, with the layer h index omitted,

$$\tilde{F}^{\ell_{out}}_{ui\,A} = \sum_{B\ \ell_f\ \ell_{in}\ vj\ k} C^{\ell_{out}\ell_{in}\ell_f}_{ijk}\ Y^{\ell_f}_{k\,AB}\ R^{\ell_{out}\ell_{in}\ell_f}_{uv\,AB}\ n^{\ell_{out}\ell_{in}}_{AB}\ F^{\ell_{in}}_{vj\,B}. \tag{3}$$

The customization between the invariant, scalar-only, distance-based L0Net and the equivariant, scalar-and-vector, distance-and-angle-based L1Net is determined by the degrees ℓ_{in}, ℓ_{out}. L0Nets only use $\ell_{in}, \ell_{out} = 0$ while L1Nets allow for $\ell_{in}, \ell_{out} \in \{0, 1\}$. The normalization is defined such that input features, with component-wise unity second moments, and component-wise normally distributed radial components produce features with component-wise unity second moments.

Gated Block This layer is used to provide a nonlinearity to the output of the convolution. Scalars are handled normally, i.e. $\mathcal{L}(F^{\ell=0}) = \text{Softplus}(\tilde{F}^{\ell=0})$, while vector, $\ell = 1$, features are multiplied by a scalar passed through an activation function. Specifically, $\mathcal{L}(F^{\ell=1}_u) = \text{Sigmoid}(\tilde{F}^{\ell=0}_{u+\mathcal{O}})\tilde{F}^{\ell=1}_u$. This introduces nonlinearity while maintaining equivariance. The previous layer produces extra learned scalar features, of multiplicity u_1 with index offset $\mathcal{O}$, in order to utilize this nonlinearity.

Final Atom-wise and Shift, Scale, Aggregate The last Convolution & Gated Block is restricted to output scalar, non-rotating features, facilitating an atom-wise layer on those features while retaining overall rotation invariance. The final atomic features are summed to produce a single scalar output, $P = \sum_{a=0}^{N} F^{h_{max}}_a$. In order to keep P near mean zero and variance one, it is shifted and scaled using statistics calculated from the training set and atomic references from the QM9 data set to finally output the target prediction, $\hat{target}$. We employ the MSE loss between $\hat{target}$ and $target$.

2.1 Experiment

Using QM9 and following the training procedure from SchNetPack, we selected random training, validation and test sets with 109,000, 1,000 and 23,885 molecules, respectively. Each network architecture was trained on each of the 12 QM9 properties. This procedure was repeated three times with an L1Net, an L0Net, and an L0Net Deep, where L0Net Deep has an additional Convolution & Gated Block. The specifics of the three network architectures were determined by hyperparameter search, as described in the supplementary material. The L0Net is the same as the L1Net, except that the $F^{\ell=1}_{u=1,\ldots,29}$ features are dropped and the multiplicity of $F^{\ell=0}$ features is increased by $3 \times 29 = 87$.

The adam optimizer [24] was employed with standard parameters and an initial learning rate of 6.53×10^{-3}. The learning rate was exponentially decayed by factor 0.5 on a loss plateau of 5 epochs to a minimum of 10^{-7}. Maximum training epochs was set at 200 with early stopping patience of 50.

We quantify the performance across L0Net, L1Net, and L0Net Deep in Table 1. If we average the $\%\mathcal{E}_{L1,L0}$ column across targets, we find that introducing rotating features improved performance by 23% on the mean absolute error. In other words, the ablation of rotating features, by changing L1Net to L0Net without altering the architecture otherwise, significantly reduced the parameter efficiency by decreasing the accuracy and keeping the number of parameters constant.

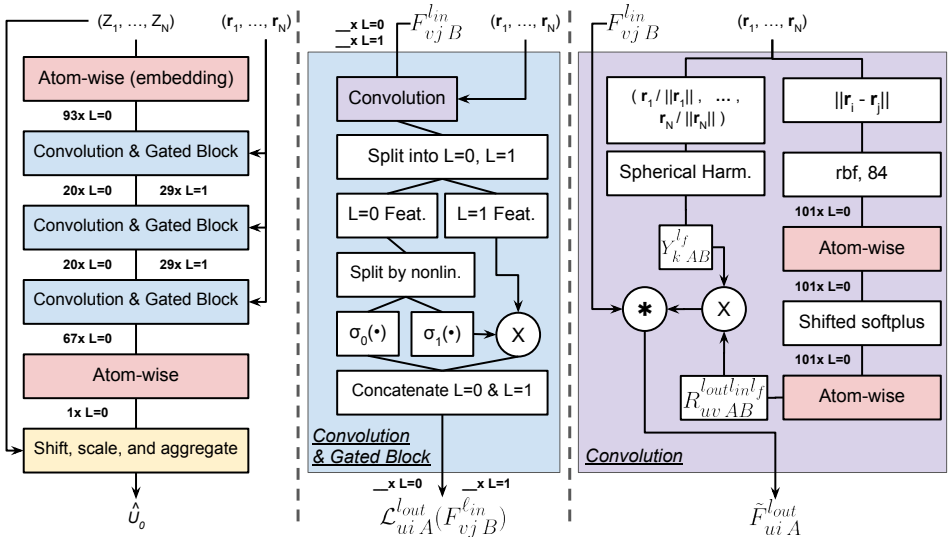

Figure 1: Illustration of L1Net with the architecture on the left, the convolution & gated block in the center, and the convolution on the right. The scalar activation function $\sigma_0(\cdot) = \text{Softplus}(\cdot)$, while the gated activation function $\sigma_1(\cdot) = \text{Sigmoid}(x)$. The notation "$T$ x L = D" implies that this connection contains a multiplicity T of features with degree D. The target output $\hat{U}_0$ is shown as an example.

Target	SchNet	SchNet Pack	Cormorant	L1Net	L0Net	L0Net Deep	$\%\mathcal{E}_{\text{L1,L0}}$	$\%\mathcal{E}_{\text{Deep,L0}}$	$\Delta\mathcal{E}_{\text{L1,Deep}}$	$\%\mathcal{E}_{\text{L1,Deep}}$
μ (D)	0.033	**0.021**	0.038	**0.043**	0.086	0.091	-0.501	0.055	-0.048	-0.556
α (a_0^3)	0.235	0.124	**0.085**	**0.088**	0.115	0.115	-0.235	0.000	-0.027	-0.235
ϵ_{HOMO} (meV)	41.000	47.000	**34.000**	46.015	47.069	**45.294**	-0.022	-0.038	0.721	0.015
ϵ_{LUMO} (meV)	**34.000**	39.000	38.000	**34.646**	39.947	37.217	-0.133	-0.068	-2.571	-0.064
ϵ_{gap} (meV)	63.000	74.000	**61.000**	**67.543**	70.344	67.873	-0.040	-0.035	-0.330	-0.005
$\langle R^2 \rangle$ (a_0^2)	**0.073**	0.158	0.961	**0.354**	0.579	0.382	-0.389	-0.340	-0.028	-0.048
zpve (meV)	1.700	**1.616**	2.027	**1.561**	1.804	1.800	-0.135	-0.002	-0.239	-0.132
U_0 (meV)	14.000	**12.000**	22.000	**13.464**	19.943	18.487	-0.325	-0.073	-5.023	-0.252
U (meV)	19.000	**12.000**	21.000	**13.834**	19.889	19.533	-0.304	-0.018	-5.699	-0.287
H (meV)	14.000	**12.000**	21.000	**14.358**	21.001	20.744	-0.316	-0.012	-6.386	-0.304
G (meV)	14.000	**13.000**	20.000	**13.989**	20.057	18.744	-0.303	-0.065	-4.755	-0.237
C_v ($\frac{\text{cal}}{\text{molK}}$)	0.033	0.034	**0.026**	**0.031**	0.035	0.037	-0.114	0.057	-0.006	-0.171

Table 1: This table quantifies the mean absolute error of relevant models on the QM9 regression targets over unseen test data. The L1 and L0Nets are compared to their closest relatives, SchNet and Cormorant as well as an L0Net with an extra Convolution & Gated Block layer called L0Net Deep. $\Delta\mathcal{E}_{X,Y}$ implies $X - Y$ where X, Y are mean absolute errors of models. $\%\mathcal{E}_{X,Y}$ is the same calculation divided by the L0Net mean absolute error on the same target. The size of train/validation/test sets differed across SchNet, SchNetPack, and Cormorant. The L·Nets were trained like the published version of SchNetPack [13] in this regard. Bold face indicates best performance within the sub-table.

We compare these results to an alternative modification of the architecture, namely, introducing another, rotationally invariant, Convolution & Gated Block to our L0Net. Averaging $\%\mathcal{E}_{Deep,L0}$ across targets reveals that L0Net Deep reduces the error by an average of 4%–significantly less effective while introducing another layer of parameters. Notably, L1Net improved on every target when compared to L0Net, while L0Net Deep worsened predictions on C_v and μ.

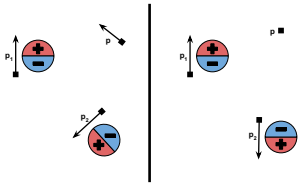

Figure 2: The magnitude of the total dipole moment depends on the orientation of the constituents, which L0Net convolutions do not consider.

In simple cases, predicting the magnitude of a rotating vector quantity, like μ, requires functional dependence on the angles between constituents in order to make unbiased predictions. Consider the case of an estimator $\mathcal{F}$ which predicts the magnitude squared total dipole moment of two constituent dipoles $p^2 = \|\mathbf{p}_1 + \mathbf{p}_2\|^2 = p_1^2 + 2p_1p_2\cos\theta_{12} + p_2^2$. $\mathcal{F}$ is restricted from functional dependence on θ_{12}, thus $\mathbb{E}[\mathcal{F}] = \mathcal{F}$. If we assume the best-case scenario, $\mathbb{E}[\cos\theta_{12}] = 0$, and the likely scenario, $\mathbb{E}[\cos^2\theta_{12}] > 0$, then the

expected squared error is

$$\begin{aligned}\min_{\mathcal{F}} \mathbb{E}[(\mathcal{F} - p^2)^2] &= \min_{\mathcal{F}} \mathcal{F}^2 - 2\mathcal{F}(p_1^2 + p_2^2) + (p_1^2 + p_2^2)^2 + 4p_1^2 p_2^2 \mathbb{E}[\cos^2 \theta_{12}] \\ &= 4p_1^2 p_2^2 \mathbb{E}[\cos^2 \theta_{12}] > 0;\end{aligned} \tag{4}$$

implying $\mathcal{F}$ is a biased estimator. By introducing rotating features through rotationally equivariant convolutions, the network is effectively introducing functional dependence angles between vector quantities, just like in this example. Given that SchNetPack does so well on the dipole moment, it remains an open question whether distances alone, in an average molecule in QM9, are enough to orient atom-wise contributions to the dipole moment. This investigation is left for future work.

L1Net is competitive in comparison with the presented architectures despite having fewer parameters and layers than the others. Still, it isn't clear if there is a class of targets which are fundamentally better suited to architectures with rotating features. The ablation study had the most impact on dipole moment μ, electronic spatial extent $\langle R^2 \rangle$, and energy at 0K U_0; however, SchNetPack, without rotating features, had the lowest error on those targets and currently holds the state-of-the-art prediction on dipole moment. The SchNet family of models includes 6 convolutions and 20 atom-wise layers (32 including filter generating networks), more than the L·Nets. Since our results imply that adding convolutions was not very efficient, it may be that more atom-wise layers are critical to gain expressivity with non-rotating features. We performed a small experiment in this direction within our framework using a network called L0Net Outdeep. See the supplementary material.

Cormorant and L1Net outperformed all architectures without rotating features on isotropic polarizability α and heat capacity C_v. L1Net includes the gated activation function, while Cormorant does not. Given that L1Net outperformed Cormorant on seven targets, while using fewer parameters, this is evidence that gated nonlinearities are worthwhile. Cormorant used an architecture which could be cast as an L3Net in our framework, by including spherical harmonic features up to degree 3. They applied 4 convolutional layers, "CGLayers," and do not have a clear equivalent to the atom-wise layer. Cormorant includes a so-called "two-body interaction" which no other network applies.

3 Conclusion

We performed an ablation study of the L1Net in order to determine the value of rotationally equivariant internal layers in regression on molecular properties using the data set QM9. Since other networks like SchNet and Comorant can be cast within our `e3nn` framework, this experiment provides intuition about architecture design for a wide variety of paradigms. Internal rotationally equivariant layers quantitatively improved performance by 23% on average while introducing new layers only helped by a mean of 4%. We provided physical intuition about what is gained by using rotating features using a simple example of a dipole built from two constituents. This example was chosen because the dipole moment was most impacted by the introduction of rotating features compared with increasing depth. However, it remains challenging to identify specific targets which would benefit the most from rotational features, in general. Our recommendation is to use rotationally equivariant internal layers when performing regression on (magnitudes of) geometric tensors where the angular contribution to constituent addition plays an important role. Our results imply that these contributions play a role in other properties as well, but of lower order.

Acknowledgements We acknowledge financial support from the European Commission (ERC CoG 772230 "ScaleCell"), the Berlin Mathematics center MATH+ (AA1-6, EF1-2) and the federal ministry of education and research BMBF (BIFOLD). Tess E. Smidt and Mario Geiger were supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory and Benjamin Kurt Miller was supported by CAMERA both under U.S. Department of Energy Contract No. DE-AC02-05CH11231. We are grateful for in-depth discussions with Moritz Hoffmann, Kostiantyn Lapchevskyi, Josh Rackers, Jonas Köhler, Jan Hermann, and Simon Batzner. Special thanks to Kostiantyn Lapchevskyi for having a diligent eye.

Code Our exact implementation can be found at https://github.com/bkmi/equivariant-benchmark.

References

[1] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688 – 702.e13, 2020.

[2] Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.*, 10:8016–8024, 2019.

[3] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018.

[4] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet – a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[5] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

[6] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pages 10381–10392, 2018.

[7] 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. *arXiv preprint arXiv:1802.08219*, 2018.

[8] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

[9] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[10] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[12] Mario Geiger, Tess Smidt, Benjamin K. Miller, Wouter Boomsma, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, and Jes Frellsen. github.com/e3nn/e3nn, May 2020.

[13] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, and K.-R. Müller. Schnetpack: A deep learning toolbox for atomistic systems. *Journal of Chemical Theory and Computation*, 15(1):448–455, 2019.

[14] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, 108:058301, Jan 2012.

[15] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104:136403, Apr 2010.

[16] Albert P Bartók, Sandip De, Carl Poelking, Noam Bernstein, James R Kermode, Gábor Csányi, and Michele Ceriotti. Machine learning unifies the modeling of materials and molecules. *Science advances*, 3(12):e1701816, 2017.

[17] Jiang Wang, Stefan Chmiela, Klaus-Robert Müller, Frank Noé, and Cecilia Clementi. Ensemble learning of coarse-grained molecular dynamics force fields with a kernel approach. *The Journal of Chemical Physics*, 152(19):194106, 2020.

[18] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

[19] Robin Winter, Floriane Montanari, Frank Noé, and Djork-Arné Clevert. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chem. Sci.*, 10:1692–1701, 2019.

[20] Brooke E. Husic, Nicholas E. Charron, Dominik Lemm, Jiang Wang, Adrià Pérez, Andreas Krämer, Yaoyi Chen, Simon Olsson, Gianni de Fabritiis, Frank Noé, and Cecilia Clementi. Coarse graining molecular dynamics with graph neural networks. *arXiv preprint arXiv:2007.11412*, 2020.

[21] Raphael J. L. Townshend, Brent Townshend, Stephan Eismann, and Ron O. Dror. Geometric prediction: Moving beyond scalars, 2020.

[22] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

[23] Johannes Hoja, Leonardo Medrano Sandonas, Brian G Ernst, Alvaro Vazquez-Mayagoitia, Robert A DiStasio Jr, and Alexandre Tkatchenko. Qm7-x: A comprehensive dataset of quantum-mechanical properties spanning the chemical space of small organic molecules. *arXiv preprint arXiv:2006.15139*, 2020.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

4 Supplementary Material

4.1 Normalization Coefficient

A feature of `e3nn` is the convolution defined in Equation 3. One heretofore undefined coefficient in the convolution is n. Recall that the normalization coefficient is selected such that component-wise second moment unity input features and component-wise normally distributed radial components produce output features which are component-wise unity in their second moments.

To better discuss the normalization properties, it makes sense to divide Equation 3 into a so-called "Kernel,"

$$K_{ui\,vj}^{l_{out}l_{in}} = \sum_{l_f\ k} C_{ijk}^{l_{out}l_{in}l_f}\ Y_k^{l_f}\ R_{uv}^{l_{out}l_{in}l_f}\ n^{l_{out}l_{in}}, \tag{5}$$

and a "Kernel-Feature Convolution,"

$$\tilde{F}_{ui\,A}^{l_{out}} = \sum_{B\ l_{in}\ vj} K_{ui\,vj\,AB}^{l_{out}l_{in}}\ F_{vj\,B}^{l_{in}}. \tag{6}$$

As you can see, calculating the Kernel followed by the Kernel-Feature Convolution yields $\tilde{F}$ which are the intermediate features before the application of the Gated Block, i.e. Convolution$(\cdot)$ = Kernel-Feature Convolution $\circ$ Kernel$(\cdot)$.

Now that the Kernel is defined, we can discuss the normalization in simpler language. Using the $\langle H \rangle$ notation for the mean of H, we write four useful, true statements:

$$\text{(by independence)}\ \ \text{var}\left[\sum_{l_{in}\,vj} K_{ui\,vj}^{l_{out}l_{in}} F_{vj}^{l_{in}}\right] = \sum_{l_{in}\,vj} \text{var}\left[K_{ui\,vj}^{l_{out}l_{in}} F_{vj}^{l_{in}}\right], \tag{7}$$

$$\text{(by independence)}\ \ \text{var}\left[K_{ui\,vj}^{l_{out}l_{in}} F_{vj}^{l_{in}}\right] = \langle (KF)^2 \rangle - \langle KF \rangle^2 = \langle K^2 \rangle \langle F^2 \rangle - \langle K \rangle^2 \langle F \rangle^2, \tag{8}$$

$$(\text{since } \langle R \rangle = 0)\ \ \langle K_{ui\,vj}^{l_{out}l_{in}} \rangle = \sum_{l_f\ k} \langle C_{ijk}^{l_{out}l_{in}l_f}\ Y_k^{l_f}\ R_{uv}^{l_{out}l_{in}l_f}\ n^{l_{out}l_{in}} \rangle = 0, \tag{9}$$

$$\begin{aligned}
\langle (K_{ui\,vj}^{l_{out}l_{in}})^2 \rangle &= \sum_{l_f\ k} \sum_{l'_f\ k'} \langle C_{ijk}^{l_{out}l_{in}l_f}\ C_{ijk'}^{l_{out}l_{in}l'_f}\ Y_k^{l_f}\ Y_{k'}^{l'_f}\ R_{uv}^{l_{out}l_{in}l_f}\ R_{uv}^{l_{out}l_{in}l'_f}\ (n^{l_{out}l_{in}})^2 \rangle \\
(\text{since } \langle RR \rangle = \delta) &= \sum_{l_f} \sum_{kk'} C_{ijk}^{l_{out}l_{in}l_f}\ C_{ijk'}^{l_{out}l_{in}l_f}\ Y_k^{l_f}\ Y_{k'}^{l_f}\ (n^{l_{out}l_{in}})^2 \\
&= (n^{l_{out}l_{in}})^2 \sum_{l_f} \left(\sum_k C_{ijk}^{l_{out}l_{in}l_f}\ Y_k^{l_f} \right)^2.
\end{aligned} \tag{10}$$

Now we can combine Equations 9 and 10 with Equation 8 to write,

$$\text{var}\left[K_{ui\,vj}^{l_{out}l_{in}} F_{vj}^{l_{in}}\right] = (n^{l_{out}l_{in}})^2 \langle (F_{vj}^{l_{in}})^2 \rangle \sum_{l_f} \left(\sum_k C_{ijk}^{l_{out}l_{in}l_f}\ Y_k^{l_f} \right)^2. \tag{11}$$

This implies that Equation 7 can be written

$$
\begin{aligned}
(7) &= \sum_{l_{in}} (n^{l_{out}l_{in}})^2 \sum_{vj} \tau_{l_{in}}^2 \sum_{l_f} \left(\sum_k C_{ijk}^{l_{out}l_{in}l_f} \, Y_k^{l_f} \right)^2 \\
&= \sum_{l_{in}} \left(\sum_v 1 \right) (n^{l_{out}l_{in}})^2 \tau_{l_{in}}^2 \sum_{l_f} \sum_j \left(\sum_k C_{ijk}^{l_{out}l_{in}l_f} \, Y_k^{l_f} \right)^2 \\
&= \sum_{l_{in}} \left(\sum_v 1 \right) (n^{l_{out}l_{in}})^2 \tau_{l_{in}}^2 \sum_{l_f} (4\pi(2l_{out}+1))^{-1} \\
&= (4\pi(2l_{out}+1))^{-1} \sum_{l_{in}} \left(\sum_v 1 \right) (n^{l_{out}l_{in}})^2 \, \tau_{l_{in}}^2 \left(\sum_{l_f} 1 \right),
\end{aligned}
\tag{12}
$$

where $\langle (F_{vj}^{l_{in}})^2 \rangle \coloneqq \tau_{l_{in}}^2$. Note that we want $(7) = \tau_{l_{in}}^2$, and we assume that $\tau_{l_{in}}^2 = 1$. This enforces that the second moment is unity. Therefore,

$$
\begin{aligned}
4\pi(2l_{out}+1) &= \sum_{l_{in}} \left(\sum_v 1 \right) (n^{l_{out}l_{in}})^2 \left(\sum_{l_f} 1 \right) \\
(n^{l_{out}l_{in}})^2 &= \frac{4\pi(2l_{out}+1)}{\left(\sum_v 1\right) \left(\sum_{l_f} 1 \right) \left(\sum_{l_{in}} 1 \right)}.
\end{aligned}
\tag{13}
$$

4.2 Shift & Scale Function

Neural networks operate best when their outputs are normally distributed. For this reason, we perform a decomposition of the target value such that the regression network's output fits this criteria. The implementation of this part of the network was handled by the SchNetPack package [13]. First, the decomposition utilizes the reference values in the QM9 data set so the network starts with a good guess and predicts a perturbation from that guess. The network's prediction is decomposed into a reference bias, an atom-wise sum from the Table 2, and a scaled contribution from each atom.

Element	ZPVE Hartree	U (0 K) Hartree	U (298.15 K) Hartree	H (298.15 K) Hartree	G (298.15 K) Hartree	Heat Capacity Cal/(Mol Kelvin)
H	0.000	-0.500	-0.499	-0.498	-0.511	2.981
C	0.000	-37.847	-37.845	-37.844	-37.861	2.981
N	0.000	-54.584	-54.582	-54.582	-54.599	2.981
O	0.000	-75.065	-75.063	-75.062	-75.080	2.981
F	0.000	-99.719	-99.717	-99.716	-99.734	2.981

Table 2: Table is adapted from the "atom ref" table in the QM9 paper [10].

For any target in Table 2 and atom in QM9, we can create a map from element Z and prediction column C to the corresponding reference value $ref(Z, C)$. For example, $ref(\mathrm{H}, \mathrm{U}_0) = -0.5$ Hartree. Given a training set of M molecules indexed by $m \in \{1, 2, ..., M\}$ each with A_m atoms indexed by $a_m \in \{1, 2, ..., A_m\}$ with a corresponding element Z_{a_m}, we write the reference bias

$$
p_m = \sum_{a_m=1}^{A_m} ref(Z_{a_m}, C). \tag{14}
$$

To further our decomposition consider the target regression value for a certain molecule t_m. From the ground truth, we can write the atom-wise deviation from the reference value,

$$
\tilde{t}_m = \frac{t_m - p_m}{A_m}. \tag{15}
$$

By gathering statistics from the training data on this $\tilde{t}_m$, we will achieve our goal of normalizing the output of the regression network. Let $\bar{\tilde{t}}$ and $\sigma_{\tilde{t}}$ be the mean and standard deviation of $\tilde{t}_m$ over molecules respectively. Given the atom-wise output of a regression network $\mathcal{R}_{a_m}$, we predict the ground truth target $\hat{t}_m$ by

$$\begin{aligned}\hat{t}_m &= p_m + \sum_{a_m=1}^{A_m} \left(\bar{\tilde{t}} + \sigma_{\tilde{t}} \mathcal{R}_{a_m}\right) \\ &= p_m + A_m \bar{\tilde{t}} + \sigma_{\tilde{t}} \sum_{a_m}^{A_m} \mathcal{R}_{a_m} \\ &= p_m + \frac{A_m}{M} \sum_{n=1}^{M} \tilde{t}_n + \sigma_{\tilde{t}} \sum_{a_m}^{A_m} \mathcal{R}_{a_m}.\end{aligned} \tag{16}$$

We presented several equivalent formulations in order to provide clarity.

4.3 Hyperparameter Search Technique

The technique applied in the paper was to do an ablation study of the rotating features in an L1Net but first we had to determine which hyperparameters defined the L1Net. In order to find a network architecture which was well suited for every QM9 target, the hyperparameter search utilized multi-target training using a featurization-output design. By searching in the multi-target regime, as opposed to doing 12 individual searches utilizing the same architecture, we traded the accuracy of single-target training for a factor of 12 decrease in hyperparameter search time. This allowed for significantly more architectures to be tested.

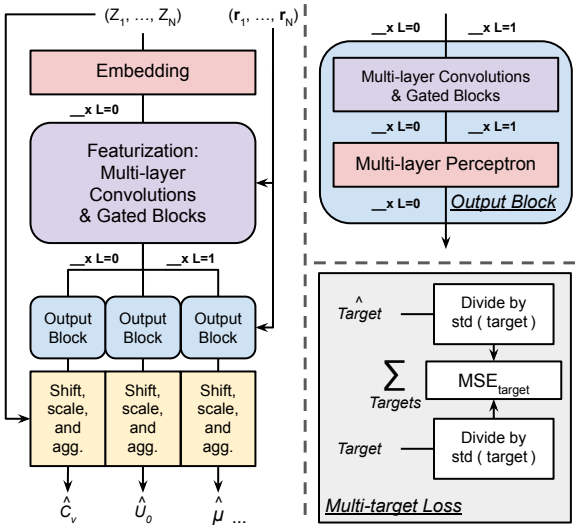

Figure 3: Illustration of the hyperparameter search with the set of possible architectures on the left, the set of possible output blocks on the top right, and the multi-target, normalized loss function shown in the bottom right.

The featurization section in L1Net (see Figure 1) is represented by the atom-wise embedding and two convolution & gated block layers. In L1Net the so-called output block represents the remaining convolution & gated block layer. Each final atom-wise layer and shift, scale, and aggregate is unique to the target being predicted. Given that examples, we focus next on how we achieved the multi-target training aspect using multiple output blocks.

Each output block receives a copy of the same learned featurization; however, the output blocks do not share gradients or other information. This allows for each output block to transform the

learned input features in parallel, each predicting a single target; thereby, the whole network makes predictions on multiple targets. The network design is depicted in Figure 3.

The featurization section is an embedding followed by layers of Convolutions and Gated Blocks. A combination of order zero and order one spherical harmonic features are copied and passed to each output block. Each output block then passes those features through Convolutions and Gated Blocks and calculates an array of scalar features. Since they are scalar features, we can pass them through one or more atom-wise layers with a rectified linear unit activation without breaking total network rotation invariance. The last layer of that multi-layer perceptron predicts a single scalar using an atom-wise layer with the identity activation function, which is then passed to the shift and scale operation as seen in Section 4.2.

The loss calculation is different from other learning problems because we attempt to normalize the losses across targets. Since all targets are equally important, we normalize their variance to one based off of statistics from our training data. This formulation depends on the assumption that each output block predicts mean zero at initialization. Recall Equation 16.

Therefore, using the notation from Section 4.2, we write the loss using the the total molecule-wise offset $s_m = p_m + A_m \bar{\tilde{t}}$. The molecule-wise loss for target t_m looks like,

$$\mathcal{L}_m(t_m, \hat{t}_m) = (\frac{t_m - s_m}{\sigma_{\tilde{t}}} - \frac{\hat{t}_m - s_m}{\sigma_{\tilde{t}}})^2 = \frac{1}{\sigma_{\tilde{t}}^2}(t_m - \hat{t}_m - 2s_m)^2. \tag{17}$$

For a batch of molecules M and pairs of targets with corresponding predictions $\{(t_m, \hat{t}_m) : m \in M\}$, the total loss is calculated by

$$\frac{1}{M} \sum_{(t_m, \hat{t}_m)} \sum_m \mathcal{L}_m(t_m, \hat{t}_m). \tag{18}$$

Although it is possible to train a model against all targets at the same time using this methodology, it is often much more difficult to achieve simultaneously good performance across targets. Therefore this model was only used for hyperparameter search, not for reported performance results.

Hyperparameter	Minimum	Maximum
Batch Size	8	25
Learning Rate	10^{-6}	3×10^{-1}
Size of Embedding	80	144
Featurization Components (FC)	80	144
Featurization Representation	(FC) randomly divided	between Y_m^0 and Y_m^1
Featurization Conv & GBs	2	5
Residual Network	True	True
Radial Basis	ϕ_C, ϕ_G, ϕ_B	ϕ_C, ϕ_G, ϕ_B
Number of Radial Bases	25	100
Radial Maximum	1.2 Å	30.0 Å
Radial MLP Layers	1	3
Radial MLP Neurons	80	144
Output Components (OC)	64	128
Output Representation	(OC) randomly divided	between Y_m^0 and Y_m^1
Output Conv & GBs	1	2
Output MLP Layers	1	3
Output MLP Neurons	80	144

Table 3: The ranges of hyperparameters for the random hyperparameter search are written in this table. Cosine ϕ_C, Gaussian ϕ_G, and Bessel ϕ_B are defined in Equation 2, SchNet [4], or DimeNet [22] respectively.

The hyperparameter search involved sampling forty different sets of hyperparameters from the ranges in Table 3 and doing multi-target training for ten epochs with each set of hyperparameters. The test

set performance for each of the forty models was compared. We took L1Net to be the winner since it produced the minimum loss averaged over normalized losses on all targets.

4.4 L1Net, L0Net, etc. Learning Plots

We compared the performance of L1Net to several different L0Net-style architectures. The most important question in this paper was: "Can an L0Net make-up for the L1Net performance by increasing depth?" However, given our architecture design, "increasing depth" could mean one of several things. L0Net Deep increased added another Convolution & Gated Block layer, L0Net Outdeep added another atom-wise layer after the convolutions, and L0Net Both Deep did both of those things. Their performance on validation data is plotted in Figure 4. We found that the L0Net Deep performed the best when compared with the other L0Net-style architectures.

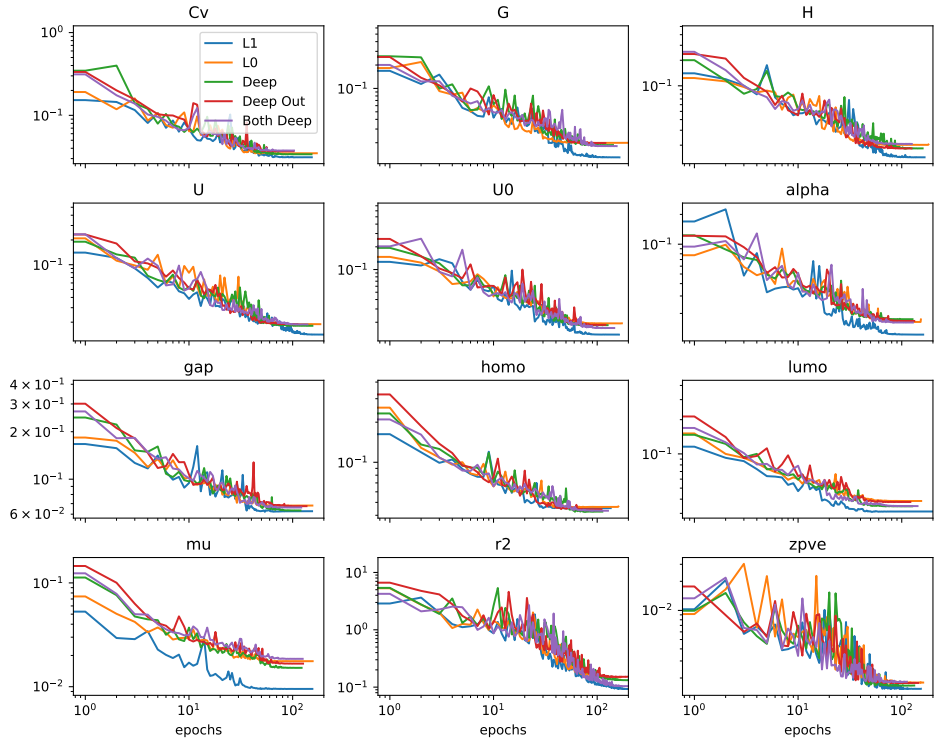

Figure 4: Plotted above is the logarithm of the mean absolute error on the validation set versus the logarithm of training epochs for every regression target. The plots contain the training curves for the L1Net, L0Net, L0Net Deep, L0Net Outdeep, and L0Net Both Deep architectures. Just like in the main article, the adam optimizer was employed with standard parameters and an initial learning rate of 6.53×10^{-3}. The learning rate was decayed given a loss plateau of five epochs to a minimum of 10^{-7}. The maximum number of training epochs was set at 200 with early stopping patience of 50.