Relevance of Rotationally Equivariant Convolutions for Predicting Molecular Properties

Benjamin Kurt Miller\textsuperscript{1,3}, Mario Geiger\textsuperscript{2,4}, Tess E. Smitd\textsuperscript{3,4} and Frank Noé\textsuperscript{1,5}
\textsuperscript{1}Freie Universität Berlin; \textsuperscript{2}EPFL; Lawrence Berkeley National Laboratory, \textsuperscript{3}Center for Advanced Mathematics for Energy Research Applications (CAMERA), \textsuperscript{4}Computational Research Division; \textsuperscript{5}Rice University
b.k.miller@uva.nl, mario.geiger@epfl.ch, tsmidt@lbl.gov, 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 using the QM9 data set. We find that for fixed network depth, adding angular features improves the accuracy on most targets. For most, but not all, molecular properties, distance-only e3nn (L0Nets) can compensate by increasing convolutional layer depth. Our angular-feature e3nn (L1Net) architecture beats previous state-of-the-art results on the global electronic properties dipole moment, isotropic polarizability, and electronic spatial extent.

1 Introduction

The discovery of novel molecules has been accelerated by advances in computational quantum chemistry and machine learning assisted exploration of chemical space \cite{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 \cite{4,5}. Meanwhile, there has also been development on neural networks which are equivariant to group action \cite{6}, some with molecules in mind \cite{7,8}. Equivariant neural networks can be seen as a super-set of invariant ones because a group necessarily contains the identity operator. The question considered in this paper can loosely be stated as: When doing regression on 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 \cite{9,10} by predicting quantum chemical properties of small molecules. While the molecules can rotate and translate, effecting 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 differ by whether their internal features are rotationally invariant (convolutions depend on distances) or equivariant (convolutions depend on distances and angles). The networks are implemented in the PyTorch \cite{11} library e3nn \cite{12} using the $SE(3)$ equivariant point modules. QM9 data handling and training routines were borrowed from SchNetPack \cite{13}.

Given atomic positions $r \in \mathbb{R}^{3\times N}$ and atomic features $F^h$, layer $h$ of an e3nn produces atomic features $F^{h+1}$ by $L^h(r, F^h) = F^{h+1}$. $F^h$ is a collection of $u^h_0$ scalars $F^h_{\ell=0}$ and $u^h_1$ vectors $F^h_{\ell=1}$ flattened into a column by $F^h = \text{vec}(F^h_{\ell=0} \oplus F^h_{\ell=1})$. The total multiplicity of features at layer $h$ is $u^h = u^h_0 + u^h_1$. The rotation matrix $R$ acts on $F^h$ in block matrix notation by

Preprint. Under review.
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 \( N \) 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 \( \{(r_1, Z_1), ..., (r_N, Z_N)\} \mapsto F(r_1, ..., r_N, Z_1, ..., Z_N) \) 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.

### 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 \( N \) 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 \( \{(r_1, Z_1), ..., (r_N, Z_N)\} \mapsto F(r_1, ..., r_N, Z_1, ..., Z_N) \) 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.

#### AtomicWISE

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

#### Radial Basis Function (rbf)

The radial basis \( \phi : \mathbb{R} \to \mathbb{R}_B \), expands \( d = \| \mathbf{r}_b - \mathbf{r}_u \| \) by

\[
\phi(d) = \begin{cases} 
\cos^2\left( \frac{d}{2 \mu_B - \mu_B} \right), & -1 \leq \frac{d - \mu_B}{\mu_{B+1} - \mu_B} \leq 1 \\
0, & \text{otherwise},
\end{cases}
\]

where \( 0 \AA \leq \mu_B \leq 11.1 \AA \) is a sequence of \( B = 84 \) 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 \( \ell_f \). 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 \( C \), spherical harmonics \( Y^{\ell_f} (\frac{\mathbf{r}_u - \mathbf{r}_B}{\| \mathbf{r}_u - \mathbf{r}_B \|}) \), 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,

\[
\tilde{F}_{u,A}^{\ell_f} = \sum_{B \ell_{out}, v,j,k} C_{ijk}^{\ell_{out}, \ell_{in}, \ell_f} Y^{\ell_f}_{k AB} R_{uv AB}^{\ell_{out}} n_{v,j,k}^{\ell_{in}} F_{v,j,B}^{\ell_{in}}.
\]

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, \( \tilde{F}_{i}^{\ell_{max}} = \tilde{F}_{i}^{\ell_{max}} \), 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 \([12, 23]\). In said framing, features are seen as spherical harmonics of degree \( \ell \).

### 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.
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 $\ell_{in}, \ell_{out}$. 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. $\mathcal{L}(F^{\ell=0}) = \text{Softplus}(F^{\ell=0})$, while vector, $\ell = 1$, features are multiplied by a scalar passed through an activation function. Specifically, $\mathcal{L}(F^{\ell=1}) = \text{Sigmoid}(F^{\ell=0}) \cdot F^{\ell=1}$. This introduces nonlinearity while maintaining equivariance. The previous layer produces extra learned scalar features, of multiplicity $u_1$ with index offset $O$, 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, $P = \sum_{a=0}^{N} F_{a}^{h_{max}}$. 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, $\hat{t}$. We employ the MSE loss between $t$ and $\hat{t}$.

![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 $T \times L = D^*$ implies that this connection contains a multiplicity $T$ of features with degree $D$. The target output $U_0$ is shown as an example.](image)

**2.1 Experiment**

Using QM9, 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 $F^{\ell=1}_{u=1,...,29}$ features are dropped and the multiplicity of $F^{\ell=0}$ features is increased by $3 \times 29 = 87$.

The adam optimizer [22] was employed with standard parameters and an initial learning rate of $6.53 \times 10^{-3}$. 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.

As seen in the results from Table I, increasing the depth of the L0Net did not improve the accuracy of the $\mu$ prediction. This empirical evidence implies that networks restricted to non-rotating, scalar
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. Internal rotationally equivariant layers helped achieve state-of-the-art results on three properties. We found that rotationally invariant network depth could qualitatively compensate for the gains offered by rotationally equivariant layers on some predictions, but, notably, not for the dipole moment $\mu$. We argued that predicting the magnitude of a rotating vector features cannot compensate the loss of angular information in the internal representation by increasing depth on targets like the dipole moment. We argue that predicting the magnitude of a rotating vector quantity, like $\mu$, requires functional dependence on the angles between constituents in order to make unbiased predictions. Consider the case of an estimator $F$ which predicts the magnitude squared total dipole moment of two constituent dipoles $p^2 = ||p_1 + p_2||^2 = p_1^2 + 2p_1p_2\cos\theta_{12} + p_2^2$. $F$ is restricted from functional dependence on $\theta_{12}$, thus $E[F] = F$. If we assume the best-case scenario, $E[\cos^2\theta_{12}] = 0$, and the likely scenario, $E[\cos^2\theta_{12}] > 0$, then the expected squared error is

$$
\min_F E[(F - p^2)^2] = \min_F F^2 - 2F(p_1^2 + p_2^2) + (p_1^2 + p_2^2)^2 + 4p_1^2p_2^2E[\cos^2\theta_{12}]
$$

$$
= 4p_1^2p_2^2E[\cos^2\theta_{12}] > 0;
$$

implying $F$ is a biased estimator. This is exactly the case in L0Net and L0Net Deep, while L1Net, applied to this problem, would be allowed functional dependence on $\theta_{12}$ and estimate without bias.

On other parameters, notably $\langle R^2 \rangle$, increasing the depth of rotationally invariant layers improved prediction accuracy, nearly to the accuracy of L1Net. For these parameters L1Net offered layer efficiency; however, accuracy was nearly achievable using deeper networks with non-rotating features.

An important counterexample was the isotropic polarizability, $\alpha$, which was not affected by increasing L0Net’s depth. Although polarizability is also a geometric tensor, like dipole moment, the argumentation regarding predicting the magnitude does not hold because the isotropic part of the polarizability tensor is merely the scalar contribution, not a magnitude. See supplementary material.

## 3 Conclusion

![Figure 2: The magnitude of the total dipole moment depends on the orientation of the constituents, which L0Net convolutions do not consider.](image)

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. Internal rotationally equivariant layers helped achieve state-of-the-art results on three properties. We found that rotationally invariant network depth could qualitatively compensate for the gains offered by rotationally equivariant layers on some predictions, but, notably, not for the dipole moment $\mu$. We argued that the scheme of summing constituents leads to a biased estimator when predicting the magnitude of vector quantities with L0Nets; however, L1Nets solve this problem. 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. We hypothesize that these contributions might play a role in other properties as well, but of lower order. That investigation is left for further work.

| Target          | schnetpack | Cormorant | DimeNet | L1Net | L0Net | L0Net Deep |
|-----------------|------------|-----------|---------|-------|-------|------------|
| $\mu$ (D)       | 0.021      | 0.038     | 0.0286  | 0.009 | 0.018 | 0.019      |
| $\alpha$ ($a_0^2$) | 0.124      | 0.085     | 0.0469  | 0.013 | 0.017 | 0.017      |
| $\epsilon_{HOMO}$ (meV) | 47        | 34        | 27.8    | 46.015 | 47.069 | 45.294     |
| $\epsilon_{LUMO}$ (meV) | 39        | 38        | 19.7    | 34.646 | 39.947 | 37.217     |
| $\epsilon_{gap}$ (meV) | 74        | 61        | 34.8    | 67.543 | 70.344 | 67.873     |
| $\langle R^2 \rangle$ ($a_0^2$) | 0.158      | 0.961     | 0.331   | 0.099 | 0.162 | 0.107      |
| $zpve$ (meV)    | 1.616      | 2.027     | 1.29    | 1.561 | 1.804 | 1.800      |
| $U_0$ (meV)     | 12         | 22        | 8.02    | 13.464 | 19.943 | 18.487     |
| U (meV)         | 12         | 21        | 7.89    | 13.834 | 19.889 | 19.533     |
| H (meV)         | 12         | 21        | 8.11    | 14.358 | 21.001 | 20.744     |
| G (meV)         | 13         | 20        | 8.11    | 13.989 | 20.057 | 18.744     |
| $C_v$ ($\frac{\text{cal}}{\text{molK}}$) | 0.034      | 0.026     | 0.0249  | 0.031 | 0.035 | 0.037      |

Table 1: This table quantifies the mean average error of relevant models on the QM9 regression targets. On the targets, $\mu$, $\alpha$, and $\langle R^2 \rangle$, L1Net achieves state-of-the-art performance. The L1 and L0Nets are compared to their closest relatives, SchNet and Cormorant. To the best of the author’s knowledge, DimeNet is the leading architecture on other QM9 regression targets.
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.

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., 9:513–550, 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, Michal 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. Schnet-pack: 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] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Benjamin Kurt Miller. Se(3) equivariant neural networks for regression on molecular properties: The qm9 benchmark, 2020.

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

[25] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks, 2018.

[26] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don’t decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[27] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.
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 \( \eta \). 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,”

\[
K_{ui vj}^{\text{out} \text{lin}} = \sum_{l_f k} C_{ij k}^{l_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} Y_{f k}^{l_f} R_{uv}^{l_f \text{lin} \text{lin}} n_{l_f \text{out} \text{lin}}, \tag{5}
\]

and a “Kernel-Feature Convolution,”

\[
\tilde{F}_{ui A}^{\text{out} \text{lin}} = \sum_{B l_i v_j} K_{ui v_j}^{\text{out} \text{lin}} F_{v_j}^{l_i \text{lin}}. \tag{6}
\]

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(\( \tilde{\cdot} \)) = 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:

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

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

\[
\text{(since } \langle R \rangle = 0 ) \quad \langle K_{ui vj}^{\text{out} \text{lin}} \rangle = \sum_{l_f k} \langle C_{ij k}^{l_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} Y_{f k}^{l_f} R_{uv}^{l_f \text{lin} \text{lin}} n_{l_f \text{out} \text{lin}} \rangle = 0, \tag{9}
\]

\[
\langle (K_{ui vj}^{\text{out} \text{lin}})^2 \rangle = \sum_{l_f k} \sum_{l'_f k'} \langle C_{ij k}^{l_f \text{out} \text{lin}} C_{ij k'}^{l'_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} f_{l'_f}^{l'_f \text{in} \text{lin}} Y_{f k}^{l_f} Y_{f k'}^{l'_f} R_{uv}^{l_f \text{lin} \text{lin}} R_{uv}^{l'_f \text{lin} \text{lin}} (n_{l_f \text{out} \text{lin}})^2 \rangle
\]

\[
\text{(since } \langle RR \rangle = \delta ) = \sum_{l_f} \sum_{kk'} \sum_{l_f} C_{ij k}^{l_f \text{out} \text{lin}} C_{ij k'}^{l_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} Y_{f k}^{l_f} Y_{f k'}^{l_f} (n_{l_f \text{out} \text{lin}})^2 \tag{10}
\]

\[
= (n_{l_f \text{out} \text{lin}})^2 \sum_{l_f} \left( \sum_{k} C_{ij k}^{l_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} Y_{f k}^{l_f} \right)^2.
\]

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

\[
\text{var} \left[ K_{ui vj}^{\text{out} \text{lin}} F_{vj}^{l_i \text{lin}} \right] = (n_{l_f \text{out} \text{lin}})^2 (\langle F_{vj}^{l_i \text{lin}} \rangle^2 \sum_{l_f} \left( \sum_{k} C_{ij k}^{l_f \text{out} \text{lin}} f_{l_f}^{l_f \text{in} \text{lin}} Y_{f k}^{l_f} \right)^2. \tag{11}
\]

This implies that Equation \[7\] can be written
\[ (7) = \sum_{l} (n_{l_{out}})^2 \sum_{v} \tau_{l_{in}}^2 \sum_{l_{f}} \left( \sum_{k} C_{ij}^l Y_{k}^{l_{f}} \right)^2 \]

\[ = \sum_{l_{in}} \left( \sum_{v} 1 \right) (n_{l_{out}})^2 \tau_{l_{in}}^2 \sum_{l_{f}} \left( \sum_{k} C_{ij}^l Y_{k}^{l_{f}} \right)^2 \]

\[ = \sum_{l_{in}} \left( \sum_{v} 1 \right) (n_{l_{out}})^2 \tau_{l_{in}}^2 \sum_{l_{f}} (4\pi(2l_{out} + 1))^{-1} \]

\[ = (4\pi(2l_{out} + 1))^{-1} \sum_{l_{in}} \left( \sum_{v} 1 \right) (n_{l_{out}})^2 \tau_{l_{in}}^2 \left( \sum_{l_{f}} 1 \right), \]

where \( \left( \langle F_{v,j}^2 \rangle \right) = \tau_{l_{in}}^2 \). 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,

\[ 4\pi(2l_{out} + 1) = \sum_{l_{in}} \left( \sum_{v} 1 \right) (n_{l_{out}})^2 \left( \sum_{l_{f}} 1 \right) \]

\[ (n_{l_{out}})^2 = \frac{4\pi(2l_{out} + 1)}{(\sum_{v} 1) (\sum_{l_{f}} 1) (\sum_{l_{in}} 1)}. \]

### 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 SchNetPack [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 | U (0 K) | U (298.15 K) | H (298.15 K) | G (298.15 K) | Heat Capacity |
|---------|------|--------|--------------|-------------|-------------|--------------|
|         | Hartree | Hartree | Hartree | Hartree | Hartree | 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, \( ref(H, U_0) = -0.5 \) 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

\[ p_m = \sum_{a_m=1}^{A_m} ref(Z_{a_m}, C). \]  

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,

\[ \tilde{t}_m = \frac{t_m - p_m}{A_m}. \]
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 $\bar{t}$ 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 $R_{a_m}$, we predict the ground truth target $\hat{t}_m$ by

$$\hat{t}_m = p_m + \sum_{a_m=1}^{A_m} \left( \bar{t} + \sigma_\tilde{t} R_{a_m} \right)$$

$$= p_m + A_m \bar{t} + \sigma_\tilde{t} \sum_{a_m}^{A_m} R_{a_m}$$

$$= p_m + \frac{A_m}{M} \sum_{n=1}^{M} \tilde{t}_n + \sigma_\tilde{t} \sum_{a_m}^{A_m} R_{a_m}.$$  \hspace{1cm} (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.

![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.](image)

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 \( s_m = p_m + A_m \tilde{t} \). The molecule-wise loss for target \( t_m \) looks like,

\[
L_m(t_m, \hat{t}_m) = \left( \frac{t_m - s_m}{\sigma_t} - \frac{\hat{t}_m - s_m}{\sigma_t} \right)^2 = \frac{1}{\sigma_t^2} (t_m - \hat{t}_m - 2s_m)^2. \tag{17}
\]

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

\[
\frac{1}{M} \sum_{(t_m, \hat{t}_m) \in M} \sum_m L_m(t_m, \hat{t}_m). \tag{18}
\]

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                        | \(10^{-6}\) | \(3 \times 10^{-1}\) |
| Size of Embedding                    | 80      | 144     |
| Featurization Components (FC)        | 80      | 144     |
| Featurization Representation (FC) randomly divided | between \(Y_0^m\) and \(Y_1^m\) |
| Featurization Conv & GBs             | 2       | 5       |
| Residual Network                     | True    | True    |
| Radial Basis                         | \(\phi_C, \phi_G, \phi_B\) | \(\phi_C, \phi_G, \phi_B\) |
| Number of Radial Bases               | 25      | 100     |
| Radial Maximum                       | 1.2 Å   | 30.0 Å  |
| Radial MLP Neurons                   | 80      | 144     |
| Output Components (OC)               | 64      | 128     |
| Output Representation (OC) randomly divided | between \(Y_0^m\) and \(Y_1^m\) |
| 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 \(\phi_C\), Gaussian \(\phi_G\), and Bessel \(\phi_B\) are defined in B.K. Miller’s thesis [23] or in Equation 2, SchNet [4], or DimeNet [24] 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 $6.53 \times 10^{-3}$. 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.

4.5 Improved $\alpha$ Results with Non-rotating Features

The isotropic polarizability of a molecule is the constant which couples an applied electric field $E$ to the induced dipole $p$ by the following equation,

$$ p = \alpha E. $$

The isotropic polarizability is merely an approximation of a higher-order tensor that couples an applied electric field to its induced dipole $p$ using this equation,
\[ p_i = \sum_j \alpha_{ij} E_j. \] 

(20)

The approximation in Equation 19 assumes that polarizability is a linear effect, i.e. applied electric fields induce a dipole which is proportional to the electric field. In reality this is not the whole story and there are anti-isotropic effects; however, that information is not contained in the QM9 data set so we cannot attempt to model it.

Since the network performed so well predicting the isotropic polarizability \( \alpha \), it is worth figuring out why. The difference in generalization error between L1Net and L0Net on this quantity was not as big as with \( \mu \), yet L0Net Deep did not improve from L0Net’s performance at all. It’s very possible that L1Net included necessary rotations like with \( \mu \); however, we presume that these effects are small and offer another explanation.

e3nn has been trained on very small batch sizes compared to SchNet which was trained on batch sizes of 100 molecules per batch. Smaller batch sizes have been shown to improve generalization [25, 26, 27] in some cases. We can validate that batch size played a role by training the a SchNet with smaller batch sizes. This is easily done by modifying the tutorial example in SchNetPack [13]. In Figure 5, one can see the training progress of a basic SchNet model against the isotropic polarizability target. Although the test set was not evaluated, the mean average validation error reduced by nearly an order of magnitude from the initially released SchNet results. The validation mean average error after training turned out to be 0.0130. Since SchNet is an L0Net-style architecture, this result shows that there is a set of hyperparameters for an L0Net which reduce the generalization error to levels qualitatively similar to the L1Net. Thus it is likely that another batch size, or another hyperparameter played a bigger role than the ablation of rotating internal features.

Figure 5: Mean average error in isotropic polarizability is plotted against epochs. The batch size was set to 64 molecules per batch. The training was accomplished using schnetpack with the default settings as found in qm9_tutorial.py. Using a small batch size significantly reduced the validation mean average error from schnetpack’s reported 0.124 test error down to 0.0130 validation error.