# Geometric Considerations for Normalization Layers in Equivariant Neural Networks

**Max Aalto**[1], **Ekdeep Singh Lubana**[1,2,3], **Hidenori Tanaka**[1,2]
[1]Physics and Informatics Laboratories, NTT Research, Inc., CA, USA
[2]Center for Brain Science, Harvard University, MA, USA
[3]EECS Department, University of Michigan, Ann Arbor, MI, USA

## Abstract

In recent years, neural networks that incorporate physical symmetry into their architecture have become indispensable tools for overcoming the scarcity of molecular and material data. However, despite its importance in deep learning, the design and selection of the normalization layer has often been treated as a side issue. In this study, we shed light on normalization and discuss how the naive application of batch normalization (BatchNorm) to materials science may face unique geometric challenges. By conducting case studies of the existing literature, we observe that the normalization layer can not only break desired symmetries in the architecture, but may also introduce undesired symmetries that remove task-relevant information from the representation. To address these issues, we review alternative normalization layers under a consistent framework and lay out the conditions for making a proper choice, *geometric-match*. Overall, our survey provides a useful summary and guidelines of normalization layers for practitioners and highlights open issues for further research and development of deep learning architectures for materials design.

## 1 Introduction

Although deep neural networks have been demonstrated to be highly capable and useful tools, they are often very difficult to optimize, costing as many as millions of dollars and several weeks to train. The technique of *batch normalization* has proven to be a critically important method for stabilizing optimization techniques, allowing for the efficient training of very deep neural networks [1] [2].

BatchNorm was motivated by the need to train deeper spatial convolutional neural networks, which have since become a relatively mature tool. More recently, practitioners have developed novel architectures, such as graph and group-equivariant networks, which are effective at processing molecular and materials data for a variety of prediction tasks. However, in the course of attempting to solve these tasks, it has been empirically shown that naive applications of BatchNorm-based techniques are often insufficient for the training of these models [3] [4] [5] [6].

Moreover, when a practitioner develops a novel normalization technique to enable the training of their network, it is typically the case that this technique is designed specifically for the corresponding task and without consideration of similar problems in other fields of study. This has led to many similarities and redundancies in the literature, where two or more groups have developed similar or even identical techniques, presumably without any knowledge of each other.

To facilitate the development of novel neural networks for geometric and graph-based tasks, it is imperative to create a broader understanding of normalization techniques that will allow for less complicated and expensive optimization of such networks. Additionally, providing a more unified characterization of existing normalization techniques will reduce the burden of the practitioner in

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

finding and selecting an appropriate technique for their task, as well as prevent redundant techniques from being rediscovered for the same class of problems.

In this context, our work aims to:

1. Summarize existing normalization techniques, with a particular focus on those used in the training of recent state-of-the-art group equivariant neural networks
2. Identify similarities and redundancies in the mechanics of such techniques
3. Explain the existence of these similarities from the perspective of satisfying geometric constraints
4. Demonstrate that it is a necessary (but not sufficient) condition that normalization techniques must obey the same geometric constraints as the task

## 2 When is BatchNorm Insufficient?

To demonstrate the insufficiency of BatchNorm as a normalization technique for graph and group-equivariant networks, we first examine four specific cases in which BatchNorm is insufficient for the training of a particular network. Subsequently, we create a generalization of the conditions in each example, which captures a necessary (but not sufficient) condition for a normalization technique to succeed on these architectures.

**Case 1. BatchNorm introduces undesired molecular correlations.** The authors of GemNet, a geometric message passing neural network intended for molecular simulations, state that in their network, applying "batch normalization introduces correlations between separate molecules and atoms", and further demonstrate that applying traditional normalization methods from spatial convolutional network literature reduces network performance [3]. In other words, when applied to GemNet, BatchNorm would result in the network learning correlations between the behaviors of separate molecules and atoms, even though in reality their behaviors are disjoint.

**Case 2. BatchNorm breaks rotational equivariance.** In "3D-Rotation-Equivariant Quaternion Neural Networks", an architecture intended for the processing of geometric data, the authors create a revised version of BatchNorm which is explicitly intended to be rotationally equivariant, respecting the symmetries of their task [4].

**Case 3. BatchNorm is not compatible with polar coordinates.** In "Rotation equivariant vector field networks", also an architecture intended for processing geometric data, the authors state that, "BN should only normalize magnitudes of the vectors to unit standard deviation. It would not make sense to normalize the angles, since their values are already bounded and changing their distribution would alter important information about relative and global orientations." Instead, they use an altered form of BatchNorm that satisfies these conditions [5].

**Case 4. Mean-subtraction can remove task-relevant information.** The authors of SetNorm, a work that develops an architecture intended for making predictions on set data, note that normalization layers for architectures that operate on sets should (a) maintain permutation equivariance and (b) not deteriorate performance due to removal of information that is useful for prediction [6]. They further note that batch norm is unsuited to their task as it can violate this second condition, and additionally may introduce dependence between inputs. This is another example of a violation of the constraints supplied by the task.

In each of these cases, there exists a geometric constraint specified by the task, and applying the vanilla version of BatchNorm violates that constraint, leading to a performance decrease. To formalize this phenomenon, we must first introduce a few definitions:

**Definition 1.** *(Equivariance and Invariance)*

*If $G$ is a group that acts on $\mathbb{R}^N$, a function $\psi : \mathbb{R}^N \to \mathbb{R}^N$ is **G-equivariant** if $\psi(Ax) = A\psi(x)$ for any $A \in G$ and any $x \in \mathbb{R}^N$. If instead, $\psi(Ax) = \psi(x)$, then the function is is **G-invariant**.*

**Definition 2.** *(Task)*

*A **task** is a function $\psi^\star(X) = y^\star$ that takes as input a representation $X$ of the input data, and returns a target object $y^\star$. The target object can be a classification target, i.e. $y^\star \in \{0, ..., k\}$, or a regression target $y^\star \in \mathbb{R}^{d_y}$. The objective of deep learning is to find a function that well-approximates this task.*

**Definition 3.** *(Geometric Match and Mismatch)*

*If a neural network is trained to approximate a task, and that task is equivariant or invariant with respect to a group $G$, then the network is* ***geometrically matched*** *to the task if:*

1. *All of its layers are also equivariant or invariant with respect to that group.*
2. *None of its layers introduce invariances which are not respected by the task.*

*If either condition is violated, then we say that the network is* ***geometrically mismatched*** *to the task.*

Geometric matching is a fundamentally important principle in deep learning. For example, the utility of spatial convolutional neural networks in image classification is enabled by the fact that such networks are invariant to the translation of objects within an image, which is geometrically matched to the task of image classification. Thus, instead of having to learn this symmetry, the network is fundamentally constrained to obey it.

## 3 Normalization Beyond Language and Vision Models

When confronted with a situation in which BatchNorm (or similar variants borrowed from language or vision literature, such as InstanceNorm and LayerNorm) do not work, practitioners developing applications typically create an alternative normalization method for that specific task. Below, in Table 1, we summarize a representative set of normalization layers developed for applications to graph and geometric data.

The notation is as follows. $\mathcal{A}$ represents the 'activations', or the outputs from the activation function following a convolutional layer, which have shape $b \times x \times v$, where $b$ is the batch size, $x$ is the number of nodes in a graph or pixels in an 'image', and $v$ is the number of features per node, or channels per pixel. $\mu$ and $\sigma$ represent the mean and standard deviation of the activations, respectively, and the subscript describes which subset of the tensor dimensions these statistics are computed over. $g$ represents a subset of the feature dimension $v$ (a 'group'). $|X|$ represents the size of a graph in the dataset, and $\mathcal{N}$ represents the average number of nearest neighbors, where the average is computed across the tensor dimensions represented in the subscript. RMS represents the root mean square, MLP is a multilayer perceptron, and $m$ is the 'hidden message' computed in the internal layers of a graph neural network. $s$ is a scaling hyperparameter, and $\alpha, \beta, \gamma$, and $\Gamma$ are learnable parameters. For EvNorm and Vector Neuron, the normalization layer is broken into two components - one that acts on the geometric component of the data, and one that acts on the invariant component of the data.

| **Language and Vision** | | **Graph and Geometric** | |
|---|---|---|---|
| BN [1] | $\frac{\mathcal{A}-\mu_{\{b,x\}}}{\sigma_{\{b,x\}}}$ | Quaternion [4] | $\frac{\mathcal{A}}{\text{RMS}_{\{x,v\}}}$ |
| LayerNorm [7] | $\frac{\mathcal{A}-\mu_{\{x,v\}}}{\sigma_{\{x,v\}}}$ | Vector Perceptron / EGAN [8] [9] | $\frac{\mathcal{A}}{\text{RMS}_{\{x,v\}}}$ |
| InstanceNorm [10] | $\frac{\mathcal{A}-\mu_{\{x\}}}{\sigma_{\{x\}}}$ | GraphNorm [11] | $\frac{\mathcal{A}-\alpha\odot\mu_{\{x\}}}{\sigma_{\{x\}}}$ |
| GroupNorm [12] | $\frac{\mathcal{A}-\mu_{\{x,v/g\}}}{\sigma_{\{x,v/g\}}}$ | PairNorm [13] | $s\cdot\left(\frac{\mathcal{A}-\mu_{\{x,v\}}}{\sigma_{\{x,v\}}}\right)$ |
| FilterResponseNorm [14] | $\frac{\mathcal{A}}{\text{RMS}_{\{x\}}}$ | MeanSubtractionNorm [15] | $\mathcal{A}-\mu_{\{b,v\}}$ |
| VarianceNorm [2] | $\frac{\mathcal{A}}{\sigma_{\{b,x\}}}$ | MessageNorm [16] | $\text{MLP}\left(\mathcal{A}+s\cdot\|\mathcal{A}_{\{x\}}\|_2\frac{m_{\{x\}}}{\|m_{\{x\}}\|_2}\right)$ |
| WeightNorm [17] | $g\frac{\mathcal{W}}{\|\|\mathcal{W}\|\|_2}$ | DiffGroupNorm [18] | $\mathcal{A}+s\sum_g \text{BN}\left(\text{softmax}_g\left(\mathcal{A}\Gamma\right)\odot\mathcal{A}\right)$ |
| SpectralNorm [19] | $\max_{h:h\neq 0}\frac{\|Wh\|_2}{\|h\|_2}\cdot W$ | SetNorm [6] | $\frac{\mathcal{A}-\mu_{\{b,v\}}}{\sigma_{\{b,v\}}}$ |
| Scaled Weight [20] | $g\frac{\mathcal{W}-\mu_{\{x,v\}}}{\sigma_{\{x,v\}}}$ | EvNorm [21] | $\left(\frac{\|\mathcal{A}_{\text{Positional}}\|_2-\mu_{\{x,v\}}}{\sigma_{\{x,v\}}},\ \frac{\mathcal{A}_{\text{Invariant}}}{\|\mathcal{A}_{\text{Invariant}}\|_2+\frac{1}{\beta}}\right)$ |
| | | Vector Neuron [22] | $\left(\frac{\mathcal{A}_{\text{Positional}}}{\|\mathcal{A}_{\text{Positional}}\|_2},\text{BN}(\mathcal{A}_{\text{Invariant}})\right)$ |
| | | GraphSizeNorm [23] | $\frac{\mathcal{A}}{\|X\|}$ |
| | | SchNet [24] | $\frac{\mathcal{A}}{\mathcal{N}_{\{v\}}}$ |
| | | Allegro / MACE [25] [26] | $\frac{\mathcal{A}}{\mathcal{N}_{\{b,x,v\}}}$ |

Table 1: Normalization layers used in graph and group equivariant neural networks

These techniques can first be divided into two categories - those that follow the approach of the original BatchNorm, and subtract a mean and divide by a standard deviation computed over some subset of the tensor dimensions, and those that perform a rescaling of the activations based on other data-specific quantities.

There are several trends we can observe from the above examples. Many variations on the original BatchNorm were in fact created to solve a geometric mismatch with a specific task; for instance, InstanceNorm was created for the task of image style transfer, and is invariant to the contrast of content images, which is a symmetry respected by the task, while BatchNorm is not. Similarly, SetNorm computes statistics over different tensor dimensions than BatchNorm, because the latter may map two samples whose activations differ in only a scale and bias to the same output, introducing an invariance that the task does not respect - a geometric mismatch that can hurt predictive ability.

### 3.1 Normalization for Geometric Data

In the particular case of geometric data, such as point clouds or molecules represented by their atomic coordinates, there are several works which introduce normalization techniques designed to be rotationally equivariant, so that two data points representing the same input at different rotations will be mapped to the same output modulo rotation. Among these are the techniques developed in "Rotation equivariant vector field networks", "3D-Rotation-Equivariant Quaternion Neural Networks", and "Learning From Protein Structure with Geometric Vector Perceptrons" [5] [4] [9]. Computing either variances or root-mean-squares over the feature dimension and scaling the activations by these quantities is a rotationally equivariant operation which still provides the benefits of normalization. The methods introduced in 'Vector Perceptrons' and 'Quaternion Neural Networks' are identical in form, although each method was developed independently and apparently without mutual knowledge of the others' work, and despite any differences in the task. Similarly, the method introduced in 'Vector Field Networks' is practically identical to VarianceNorm [2], modulo differences in the datatype, as VarianceNorm was developed for image data.

There have additionally been multiple geometric networks that utilize 'edge-based' information for normalization (referring to the implicit or explicit graph structure in the data), which again share commonalities. For instance, the normalization techniques used in 'SchNet' [24] and 'Allegro' [25] both scale activations by a quantity derived from the connectivity of each molecular or material graph; in the former, the quantity is computed for each atom in a molecule or material, whereas in the latter, the quantity is computed across all atoms in all molecules in the entire dataset. Both networks involve the individual node (atom) level prediction of quantities which are explicitly invariant to the number of neighbors of each node, and this normalization technique aims to preserve this invariance property.

## 4 Empirical Analysis

To demonstrate the importance of normalization techniques on the performance of equivariant neural networks, we present a series of graphs that depict training and testing metrics for three popular models used in chemistry and materials science: Allegro [25], GemNet [3], and SchNet [24]. For each network, we train a variant with and without the normalization techniques described by the authors in their respective papers.

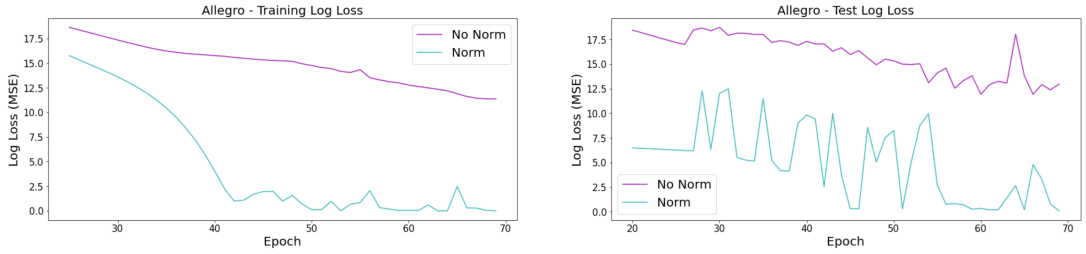


Figure 1: **Allegro training and test losses blow up without normalization:** The train and test losses when training Allegro on the 'Uracil' dataset [27] immediately explode when training without the normalization methods developed by the authors; they begin at a MSE of $10^{14}$ and although they decrease, over 100 epochs they remain 5 orders of magnitude greater than when training with normalization.

To test the effects of normalization on Allegro, we utilize the official PyTorch / e3nn implementation available on Github [28], and train on the 'Uracil' data included within the MD17 dataset [27], using the hyperparameters specified in the tutorial written by the authors. For SchNet, we utilize an open-source implementation [29] based on the architecture described in [24], with a batch size of 8, 3 hidden layers, which each have 3 interaction layers of sizes 128, 128, and 4 respectively. We train SchNet on the 'CHAMPS Scalar Coupling' dataset [30], using Adam with an initial learning rate of 0.001 and exponential weight decay with coefficient 0.99. For GemNet, we utilize the official PyTorch implementation available on Github [31], and train it on the 'COLL' dataset [32] included in the repository, using the default hyperparameters defined by the authors. In each case, we train one network with the normalization methods implemented in the respective repository, and one without, using identical hyperparameter settings.

Figure 1, above, depicts the log of the train and test losses for the Allegro network trained with and without normalization. For the network that has no normalization, we observe 'gradient explosion' [33], and losses that are initially 11 orders of magnitude greater than its normalized counterpart. Though the unnormalized loss decreases, it is unable to achieve a value comparable to the normalized architecture. Figure 2, below, depicts the train and test losses for the SchNet and GemNet networks, again trained with and without normalization. Although the effects are not as dramatic, we still observe that applying normalization techniques yields an order of magnitude decrease in the loss.

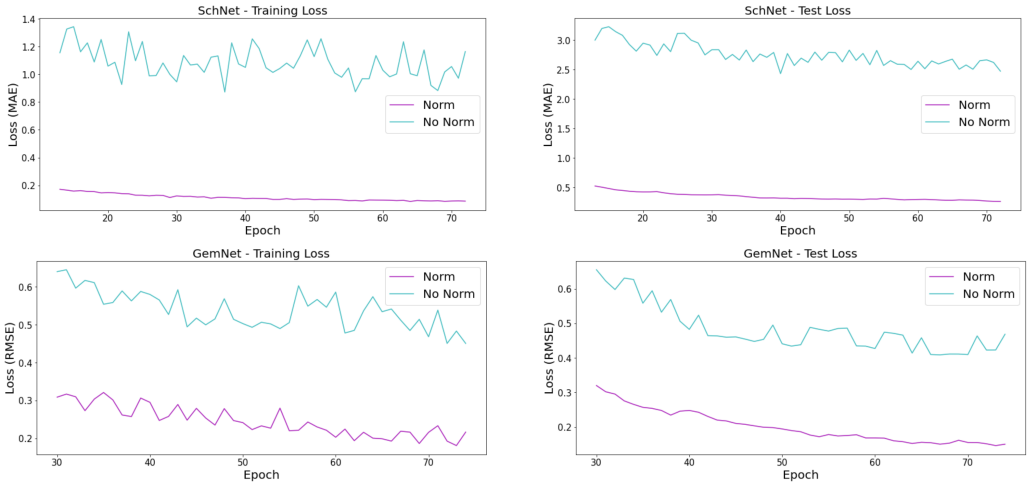


Figure 2: **Normalization leads to improved performances in SchNet and GemNet:** Both the GemNet and SchNet architectures achieve an order of magnitude better performance when using normalization methods.

## 5 Conclusion

Normalization techniques have been proven to be a powerful tool for enabling the training of larger networks, and yet a large proportion of existing deep learning applications to chemistry problems completely forego using them [34] [35] [36] [37] [38]. As the datasets involved in molecular and materials prediction tasks grow, so will the networks, and thus the difficulty involved in successfully training an effective neural network to solve these tasks will increase correspondingly. Although several techniques have been developed for existing geometric datasets, no general principles of normalization exists for these new classes of architectures. Here, we initiated the discussion by providing the above survey, noting the commonalities between these newly developed normalization techniques that make them effective for their specific tasks, and using the guideline of geometric matching as a necessary condition for maintaining stability during optimization.

As both the interest and the size of the datasets in molecular and materials deep learning grows, we end by posing two important open challenges that will be crucial to its practical implementations: what are the most general conditions under which traditional normalization techniques fail, and what is the most general formulation of a normalization layer which may be applied to any graph-based or geometric dataset underlying a molecular or materials prediction task?

## References


[1] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. Int. Conf. on Machine Learning (ICML)*, Jul 2015.

[2] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules, 2021. URL `https://arxiv.org/abs/2106.08903`.

[4] Binbin Zhang, Wen Shen, Shikun Huang, Zhihua Wei, and Quanshi Zhang. 3d-rotation-equivariant quaternion neural networks. *CoRR*, abs/1911.09040, 2019. URL `http://arxiv.org/abs/1911.09040`.

[5] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. *CoRR*, abs/1612.09346, 2016. URL `http://arxiv.org/abs/1612.09346`.

[6] Lily H Zhang, Veronica Tozzo, John M Higgins, and Rajesh Ranganath. Set norm and equivariant skip connections: Putting the deep in deep sets. 2021.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv*, abs/1607.08022, 2016.

[8] Tuan Le, Frank Noé, and Djork-Arné Clevert. Equivariant graph attention networks for molecular property prediction, 2022. URL `https://arxiv.org/abs/2202.09891`.

[9] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons, 2020. URL `https://arxiv.org/abs/2009.01411`.

[10] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv*, abs/1607.08022, 2016.

[11] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, pages 1204–1215. PMLR, 2021.

[12] Yuxin Wu and Kaiming He. Group Normalization. In *Proc. European Conf. on Computer Vision (ECCV)*, 2018.

[13] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

[14] Saurabh Singh and Shankar Krishnan. Filter Response Normalization Layer: Eliminating Batch Dependence in the Training of Deep Neural Networks. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020.

[16] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[17] Tim Salimans and Diederik P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2016.

[18] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 33:4917–4928, 2020.

[19] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[20] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.

[21] Zhuoran Qiao, Anders S. Christensen, Frederick R. Manby, Matthew Welborn, Anima Anandkumar, and Thomas F. Miller III. Unite: Unitary n-body tensor equivariant network with applications to quantum chemistry. *CoRR*, abs/2105.14655, 2021. URL https://arxiv.org/abs/2105.14655.

[22] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. *CoRR*, abs/2104.12229, 2021. URL https://arxiv.org/abs/2104.12229.

[23] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[24] 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, jun 2018. doi: 10.1063/1.5019779. URL https://doi.org/10.1063%2F1.5019779.

[25] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics, 2022. URL https://arxiv.org/abs/2204.05249.

[26] Ilyes Batatia, Dávid Péter Kovács, Gregor N. C. Simm, Christoph Ortner, and Gábor Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields, 2022. URL https://arxiv.org/abs/2206.07697.

[27] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017. doi: 10.1126/sciadv.1603015. URL https://www.science.org/doi/abs/10.1126/sciadv.1603015.

[28] Albert Musaelian and Simon Batzner. e3nn implementation of allegro, 2022. URL https://github.com/mir-group/allegro.

[29] toshi k. Schnet starter kit, 2020. URL https://www.kaggle.com/code/toshik/schnet-starter-kit/notebook.

[30] Jia Fang, Linyuan Hu, Jianfeng Dong, Haowei Li, Hui Wang, Huafen Zhao, Yao Zhang, and Min Liu. Predicting scalar coupling constants by graph angle-attention neural network. *Scientific Reports*, 11(1):18686, Sep 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-97146-1. URL https://doi.org/10.1038/s41598-021-97146-1.

[31] Johannes Gasteiger and Nicholas Gao. Pytorch implementation of gemnet, 2021. URL https://github.com/TUM-DAML/gemnet_pytorch.

[32] Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules, 2020. URL https://arxiv.org/abs/2011.14115.

[33] George Philipp, Dawn Song, and Jaime G. Carbonell. The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions, 2017. URL https://arxiv.org/abs/1712.05577.

[34] So Takamoto, Satoshi Izumi, and Ju Li. TeaNet: Universal neural network interatomic potential inspired by iterative electronic relaxations. *Computational Materials Science*, 207:111280, may 2022. doi: 10.1016/j.commatsci.2022.111280. URL https://doi.org/10.1016%2Fj.commatsci.2022.111280.

[35] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31 (9):3564–3572, apr 2019. doi: 10.1021/acs.chemmater.9b01294. URL https://doi.org/10.1021%2Facs.chemmater.9b01294.

[36] Yeji Kim, Yoonho Jeong, Jihoo Kim, Eok Kyun Lee, Won June Kim, and Insung S. Choi. Molnet: A chemically intuitive graph neural network for prediction of molecular properties, 2022. URL https://arxiv.org/abs/2203.09456.

[37] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. iab initio/i solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3), sep 2020. doi: 10.1103/physrevresearch.2.033429. URL https://doi.org/10.1103%2Fphysrevresearch.2.033429.

[38] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra, 2021. URL https://arxiv.org/abs/2102.03150.