Curvature-informed multi-task learning for graph networks

Alexander New [1] Michael J. Pekala [1] Nam Q. Le [1] Janna Domenico [1] Christine D. Piatko [1]
Christopher D. Stiles [1]

Abstract

Properties of interest for crystals and molecules, such as band gap, elasticity, and solubility, are generally related to each other: they are governed by the same underlying laws of physics. However, when state-of-the-art graph neural networks attempt to predict multiple properties simultaneously (the multi-task learning (MTL) setting), they frequently underperform a suite of single property predictors. This suggests graph networks may not be fully leveraging these underlying similarities. Here we investigate a potential explanation for this phenomenon – the curvature of each property's loss surface significantly varies, leading to inefficient learning. This difference in curvature can be assessed by looking at spectral properties of the Hessians of each property's loss function, which is done in a matrix-free manner via randomized numerical linear algebra. We evaluate our hypothesis on two benchmark datasets (Materials Project (MP) and QM8) and consider how these findings can inform the training of novel multi-task learning models.

1. Introduction

Graph networks (GNs) (Battaglia et al., 2018) are considered state-of-the-art machine learning (ML) methods for many scientific problems, including property prediction of both inorganic crystalline materials (Xie & Grossman, 2018; Park & Wolverton, 2020) (hereafter "crystals", e.g., Figure 1) and small organic molecules (Gilmer et al., 2017; Gasteiger et al., 2020) (hereafter "molecules", e.g., Figure 2); the same GN architectures have shown success in both domains (e.g., (Chen et al., 2019), who suggest the "crystal" vs. "molecule" terminology and use "material" to refer to both). A common use case might involve training a model from materials with known properties and then screening a separate dataset lacking those properties to obtain potential candidates to further investigate. Typically, property-prediction tasks are formulated as single-target regression problems, where the goal is to predict a scalar property like band gap, formation stability, elasticity, or solubility. However, practical materials design requires optimizing multiple properties of interest. For example, when designing a cell phone screen, a material with both high optical transparency and hardness would be desired.

Predicting a single property can be posed as a single task for an ML model; using a single ML model to predict multiple properties is thus a type of multi-task learning (MTL). The most common family of MTL approaches is hard parameter sharing, in which a single representation space is shared across multiple tasks of interest, and task-specific networks map from that space to output predictions (Caruana, 1997).

However, the latest research in novel MTL techniques largely has been focused on classification problems in the natural language processing (NLP) and computer vision (CV) domains – for example, the MultiMNIST problem (Sener & Koltun, 2018), where a model must identify two separate digits in the same picture. With some recent exceptions – e.g., (Yang et al., 2021; Tan et al., 2021; Kong et al., 2021) – there has been little work exploring the application of MTL to multi-property GN regression problems.

Some existing papers have used hard parameter sharing for material property prediction, in which a single model predicts a set of properties. However, results suggest that this approach has degraded performance compared to a set of single-property models – see, e.g., (Gasteiger et al., 2020), which predicts twelve quantum mechanical properties of molecules and finds that a single multi-output model performs worse than a set of single-output models.

Works have explored reasons for this: (Yu et al., 2020) identifies a "tragic triad" of reasons that MTL models might perform worse compared to single-task ones, one of which is that multi-task loss functions can have high curvature, as characterized by the Hessian of the loss. However, (Yu et al., 2020) rely on first-order approximations to curvature and do not directly incorporate second-order information.

[1]Research and Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory. Correspondence to: Alexander New <alex.new@jhuapl.edu>.

2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML), 2022. Copyright 2022 by JHU/APL.

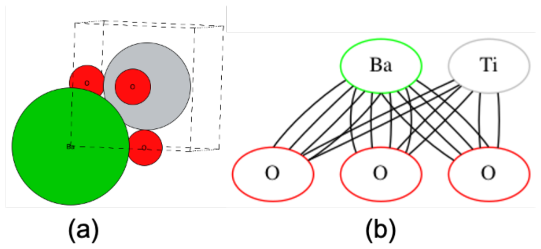


Figure 1. **Example of converting a periodic crystal lattice structure (Ba Ti O_3) to a multigraph**
(a): Crystal lattice structure, created using `ase`[2].
(b): The corresponding multigraph representation of the crystal that is ingested by a graph network.

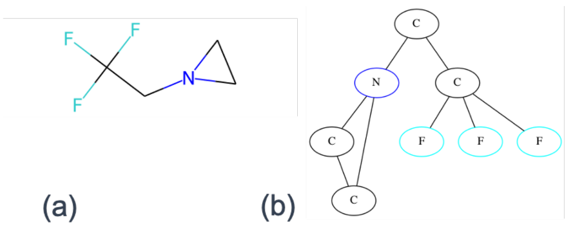


Figure 2. **Example of converting a 2D molecule structure (C_4 H_6 F_3 N) to a graph**
(a) Molecular structure as represented by `rdkit`
(b) The corresponding graph

Concurrently, several works have applied techniques from randomized numerical linear algebra to directly probe the Hessians of large ML models (Sagun et al., 2017; Alain et al., 2019; Yao et al., 2019; Ghorbani et al., 2019; Papyan, 2020). These rely (1) on the fact that the product of a model's Hessian and another vector (a Hessian-vector product) may be efficiently obtained, without explicitly calculating the full Hessian, and (2) that given a matrix-free Hessian-vector product function, several key spectral properties of the Hessian, including its eigenvalues and trace, may be estimated. However, these papers focus only on standard image classification problems, and do not consider the MTL setting, GNs, or regression problems.

In this paper, we formulate the application of curvature-informed techniques to MTL models (Section 2.1), describe the application of graph networks to the problem of multi-property prediction in the domains of materials science and chemistry (Section 2.2 and Section 2.3), and analyze training dynamics of multi-property prediction models using methods based on assessing local curvature of loss function surfaces (Section 3). Our results suggest how novel multi-property prediction models might inherently account for differences in curvature to enhance learning efficiency.

2. Approach

Let $\mathcal{G}$ be a space of directed multigraphs corresponding to crystal or molecule structures. A directed graph $g = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V} = \{v\}$ and edges $\mathcal{E} = \{(v, v', k)\}$. Each node v has a node embedding $x_v \in \mathbb{R}^V$, and each edge (v, v', k) has an edge embedding $e_{v,v',k} \in \mathbb{R}^E$. A pair of nodes v and v' may have multiple edges connecting them, and these multi-edges are indexed by $k = 1, \dots, K_{v,v'}$. Furthermore, a graph g has T properties $y = (y_1, \dots, y_T) \in \mathcal{Y}$, here assumed to all be scalars.

We require a model that can predict all targets y for a given g. This entails minimizing a loss with respect to a set of neural network weights $\{\theta_{sh}, \theta_1, \dots, \theta_T\}$, where θ_t is used only in predicting property t, and θ_{sh} is shared across predictions. The form of this problem and its neural network is discussed in Section 2.2. First we consider the loss's curvature.

2.1. Curvature assessment techniques

We view the shared parameters θ_{sh} of a graph network as a flattened vector in $\mathbb{R}^P$; let $L_t : \mathbb{R}^P \to \mathbb{R}$ be a property-specific loss function of θ_{sh}, where the property-specific parameters $\theta_1, \dots, \theta_T$ are held constant. Typically, machine learning models consider only first-order derivative information of L_t when training neural networks. However, second-order information is also useful in optimization problems – consider how a quasi-Newton method like limited-memory L-BFGS (Liu & Nocedal, 1989) is more efficient than a first order method like gradient descent. This suggests that additional insight into the training problems of ML models can be found in using second-order information.

The curvature of a function L_t is characterized by its Hessian $H_t = \nabla^2_{\theta_{sh}} L_t \in \mathbb{R}^{P \times P}$ for a weight vector $\theta_{sh} \in \mathbb{R}^P$. For typical deep neural networks, $P \approx 10^6$, and calculating H directly is computationally infeasible due to storage constraints (H_t will have 10^{12} entries). However, the product of H_t with a vector v is computable in approximately the same time-complexity as calculating the gradient of L_t with respect to θ (Pearlmutter, 1994).[3] This enables calculation of properties of H_t that can be obtained from observing its action on a chosen vector v (Yao et al., 2019; Ghorbani et al., 2019). In particular, the trace of H_t, and an approximate probability density function of its eigenvalues can be efficiently estimated.

The trace of H_t is both the sum of its eigenvalues and L_t's Laplacian (i.e., $\operatorname{tr} H_t = \sum_p \partial_{\theta_{sh,p}\theta_{sh,p}} L_t(\theta)$). Thus, the trace of H_t can be viewed as a measurement for the complexity of its curvature (Yao et al., 2019). The sign of the trace also indicates the sign of L_t's local curvature. The

[3]In `jax`, this is implemented by composing a forward-mode Jacobian-vector product `jvp` function with a reverse-mode gradient function `grad`.

trace of H_t may be estimated with the expectation

$$\operatorname{tr} H_t = \mathbb{E}_{v\sim\mathbb{P}_R}[v^T H_t(\theta)v],$$

where $\mathbb{P}_R$ is the distribution of random variables in $\mathbb{R}^P$ with iid Rademacher components (Avron & Toledo, 2011).

Having access to all of H_t's eigenvalues gives us a full picture of L_t's curvature. These eigenvalues are often represented as a function ψ called the spectral density or density of states that is given by $\psi(t) = \frac{1}{P}\sum_p \delta(t-\lambda_p)$, where $\lambda_1 \leq \cdots \leq \lambda_P$ are the eigenvalues of H_t, and δ is the Dirac distribution (Lin et al., 2016). However, the weight vector θ_{sh} is too high-dimensional for all of H_t's eigenvalues to be directly calculated in practice. Methods like power iteration can be used to estimate the large-magnitude eigenvalues of a Hessian (Yao et al., 2019), but they scale poorly as its dimensionality increases.

A solution is to first use the Lanczos algorithm to estimate $P' \ll P$ eigenvalues of H_t and then smooth these estimated values into a continuous approximation to the discrete density ψ by convolving the values with a Gaussian kernel (Golub & Welsch, 1969; Ghorbani et al., 2019).[4] This yields a continuous function $\hat{\psi}(t)$ that approximates ψ. See Appendix E for an example of estimating $\hat{\psi}$ for a simplified loss function.

Interpreting estimated spectral densities remains a challenging task: (Ghorbani et al., 2019) and (Yao et al., 2019) claim that a spectral density should be "smooth" (i.e., concentrated within a dense region) and consider the effect of different network architectures on density smoothness. (Alain et al., 2019) argues that having a large concentration of negative eigenvalues can lead to inefficient training, because most optimizers fail to leverage this local negative curvature. (Papyan, 2020) relate the outlier structure of a Hessian's eigenvalues to the number of classes in a dataset.

Compared to CV problems, graph-structured datasets have a number of interesting attributes. For example, a graph input has a variable number of node and edge features, which complicates the GN learning problem. Furthermore, in contrast to the use of very deep networks in CV problems, increasing depth in graph networks has often not been found to be effective (see, e.g., discussion in (Godwin et al., 2022)). Hessian-based information has the potential to inform some of these comparisons.

Note that the estimations of both the trace and the spectral densities are based on random quantities. For trace, multiple vectors v must be sampled and the quantity $v^T H_t(\theta)v$ computed, and then convergence in mean can be assessed. In practice, we find that approximately 500 samples are sufficient. For spectral density, each Lanczos iterate is initialized as a random Gaussian vector. We adapt an implementation of the Lanczos algorithm and smoothing process developed by (Ghorbani et al., 2019), in which we reorthogonalize the Lanczos iterates during each step of estimation to promote numerical stability. We follow (Yao et al., 2019) and use $P' = 100$, which is more than the $P' = 90$ that (Ghorbani et al., 2019) validate by comparing to an exact calculation of every eigenvalue of a smaller network.

[4]The Lanczos algorithm accurately estimates the Hessian's most positive and most negative eigenvalues, and the convolution approximates the distribution of eigenvalues between these extrema.

2.2. Graph networks for property prediction

Let $\hat{y} : \mathcal{G} \to \mathcal{Y}$ be a neural network parameterized by θ. We follow (Sener & Koltun, 2018) in splitting θ into a set of shared weights θ_{sh}, and a set of property-specific weights $\theta_1, \ldots, \theta_T$. Then the predicted t^{th} property is

$$\hat{y}_t(g; \theta_{sh}, \theta_t) = f_t(z(g; \theta_{sh}); \theta_t),$$

where $z : \mathcal{G} \to \mathbb{R}^D$ is a task-independent GN (Battaglia et al., 2018), and each $f_t : \mathbb{R}^D \to \mathbb{R}$ is a task-specific feedforward network.

We briefly outline the functionality of our graph network z, which maps an input graph g to a graph-level representation vector u^M. First, the multigraph g is given a global feature $u^0 \in \mathbb{R}^U$ that is initialized as a vector of ones, and, similarly to (Chen et al., 2019), the node features x_v and edge features $e_{v,v',k}$ are linearly projected: $x_v^0 = W_V\, x_v$ and $e^0_{v,v',k} = W_E\, e_{v,v',k}$, where W_V and W_E are matrices learned during training. Then information is propagated across the multigraph during $m = 1, \ldots, M$ message-passing steps.

The m^{th} message-passing step proceeds as follows: First, the edge features are updated:

$$e^m_{v,v',k} = \phi^E(\operatorname{cat}(e^{m-1}_{v,v',k}, x^{m-1}_v, x^{m-1}_{v'}, u^{m-1})) + e^{m-1}_{v,v',k},$$

where ϕ^E is a feed-forward neural network, and cat is the concatenation operator. Then node updates are collected. For every node v, first edge updates are calculated:

$$\begin{aligned} h^m_{v,out} &= \sum_{(v',k):(v,v',k)\in\mathcal{E}} e^m_{v,v',k} \\ h^m_{v,in} &= \sum_{(v',k):(v',v,k)\in\mathcal{E}} e^m_{v',v,k}, \end{aligned}$$

where the first summation is over the tuples (v', k) such that (v, v', k) is an edge for g, and the second summation is defined similarly. With these updates, the new node representation for v is given by

$$x^m_v = \phi^V(\operatorname{cat}(h^m_{v,out}, h^m_{v,in}, x^{m-1}_v, u^{m-1})) + x^{m-1}_v,$$

where ϕ^V is a feed-forward neural network. Finally, the global feature is updated:

$$u^m = \phi^U \left(\text{cat} \left(\sum_{v,v',k} e^m_{v,v',k}, \sum_v x^m_v, u^{m-1} \right) \right) + u^{m-1},$$

where ϕ^U is a feed-forward neural network, and each feature vector $x^m_v, e^m_{v,v',k}, u^m$ is processed with a layer normalization (Ba et al., 2016) layer. After M steps, the final global state u^M is fed into each property-specific network f_t.

Our graph networks are implemented in the `jraph`[5] framework that is built on `jax`[6]. We use `flax`[7] to implement component neural networks ϕ^V, ϕ^E, and ϕ^U. Because we seek to evaluate second-order derivative information, we ensure that our neural networks ϕ^V, ϕ^E, ϕ^U are smooth functions with respect to their weights by using tanh activation functions. Further details on specific hyperparameters used to instantiate models are given in Appendix A.

We impose on each property a loss function L_t (here assumed to be mean squared loss for all t), and we collect a training dataset of multigraph data points $\mathcal{D} = \{(g^n, y^n)\}_{n=1}^N$. Then we solve the minimization problem

$$\min_\theta L(\theta) = \sum_t L_t(\theta_{sh}, \theta_t),$$

where

$$L_t(\theta_{sh}, \theta_t) = \sum_n |\hat{y}_n(g^n; \theta_{sh}, \theta_t) - y^n_t|^2$$

are task-specific losses and $\theta = \{\theta_{sh}, \theta_1, \ldots, \theta_T\}$.

We solve this minimization with stochastic gradient descent using the `optax`[8] implementation of AdamW (Loshchilov & Hutter, 2019) to choose stepsizes; we handle the potential difference in scales across properties by normalizing all targets prior to training and set $\mu_t = 1$ for all t. Further details on the specifics of model training are given in Appendix B.

2.3. Data sources

We evaluate our graph network curvature assessment techniques on datasets from two scientific domains: materials science (crystals) and chemistry (molecules). For crystals, our data featurization follows (Park & Wolverton, 2020); for molecules, it follows (Gilmer et al., 2017). To reduce the complexity of our considered problem space, we choose to use simple featurizations (one-hot encoding of atom-types as node featurizations), but our methods are compatible with other featurizations (e.g., hand-crafted node descriptors like those provided by `rdkit`). Further details about the data used are available in Appendix C.

For materials science, we collect data from Materials Project (MP) (Jain et al., 2013), which contains results of density functional theory (DFT) calculations for different crystals. crystal records contain compositional and structural information, as well as some related properties. Each crystal's structure information is captured in a Crystallographic Information File (CIF), which we convert into a multigraph using the `VoronoiNN` function from `pymatgen`[9], following the approach of (Park & Wolverton, 2020). As crystals are periodic structures, this process yields multiple edges between many given pairs of nodes (Xie & Grossman, 2018; Park & Wolverton, 2020; Sanyal et al., 2018) (e.g., Figure 1). As targets, we use several assessments of a crystal's elastic properties: the Voigt-Reuss-Hill (Hill, 1952) calculation for shear (G_{VRH}) and bulk (K_{VRH}) modulus, both measured in units of GPa, as well as the isotropic Poisson ratio μ, a dimensionless quantity.

These properties are inherently coupled – μ can be calculated as a function of G_{VRH} and K_{VRH}. Thus, an ML model that predicts all three serves as a test for how well it can learn underlying physical relationships. A one-hot encoding of atom-type is used to obtain initial node features x_v, and four bond-related properties calculated by `pymatgen` are used as edge features, resulting in a dataset of 10,500 crystal structures with known elastic properties.

For chemistry, we use the QM8 (Ramakrishnan et al., 2015) dataset of organic molecules. We use the `rdkit` package[10] to convert SMILES strings (Weininger, 1988) into molecular structure graphs (see, e.g., Figure 2). As targets, we use the first and second transition energies, E_1 and E_2, and the first and second oscillator strengths f_1 and f_2. QM8 contains several versions of transition energies and oscillator strengths that have been calculated via different levels of DFT. We use the predictions of the approximate coupled-cluster (CC2) (Hättig & Weigend, 2000) method as our targets, as these are treated as the "ground truth" in (Ramakrishnan et al., 2015). A one-hot encoding of atom-type is used to obtain initial node features x_v, and a one-hot encoding of bond-type is used to obtain initial edge features $e_{v,v',k}$.

Note that, in Section 2.2, we describe our input data points g as directed multigraphs. However, for our actual data points, the edges are not inherently directed – bonds are symmetric. Thus, we duplicate each edge feature (i.e., $e_{v,v',k} = e_{v',v,k}$ for all edges) during data preprocessing.

[5] https://github.com/deepmind/jraph
[6] https://github.com/google/jax
[7] https://github.com/google/flax
[8] https://github.com/deepmind/optax
[9] https://pymatgen.org
[10] https://github.com/rdkit/rdkit

3. Results

We demonstrate the application of our curvature-assessment techniques (Section 2.1) on trained GNs (Section 2.2) in two application domains: materials science and chemistry (Section 2.3). Although here we focus on GNs (Battaglia et al., 2018), our assessment techniques are applicable to the broad family of graph neural networks used in property prediction tasks.

Figure 3 shows training and test set errors for each dataset. These results do not necessarily align with domain intuition, suggesting that the models do not leverage scientific knowledge in their learned representations. For example, for MP, the test set error for Poisson ratio is higher than that of G_{VRH} or K_{VRH}, despite the fact that Poisson ratio can be calculated as a function of G_{VRH} and K_{VRH}. This is interesting and suggests that the GNs are not fully leveraging known scientific knowledge. For QM8, the oscillator strengths f_1 and f_2 have the lowest and highest, respectively, test set errors.

Next we summarize the curvature of the loss functions using the trace of each task's loss's Hessian. In Figure 4, we show that, despite the general decreasing training error, the curvature of each property's loss surface, as measured by the trace of its Hessian, is highly variable. In particular, both datasets show high heterogeneity across properties and across training epochs in their estimated traces. Our observations here do not align with prior work that has analyzed Hessian traces for CV problems (Yao et al., 2019). In those works, traces increased monotonically during training and were consistently positive. Here, our estimated traces often flip between being positive and negative.

The Hessian trace is a high-level summary of the curvature of a loss surface; for a more granular examination, we estimate the spectral densities of each property's Hessian. In Figure 6a in D, we show that the spectral densities are similarly variable and often feature the presence of outliers that briefly occur during training. These outliers vary across properties. For the QM8 dataset, we zoom in on the range of eigenvalues where most density is concentrated in Figure 5. Similar to (Yao et al., 2019; Ghorbani et al., 2019), we see that most density is concentrated near 0. This suggests that there is a great deal of redundancy in the latent spaces learned by these models, and their true dimensionality is likely significantly less than the full P parameters of the θ_{sh} weight vector. Unlike these works, the spectral densities are more symmetric around 0. This matches Figure 4, since the Hessian trace (sum of all its eigenvalues) varies between being very positive and very negative. The exact spectral densities vary across properties, even at the end of training.

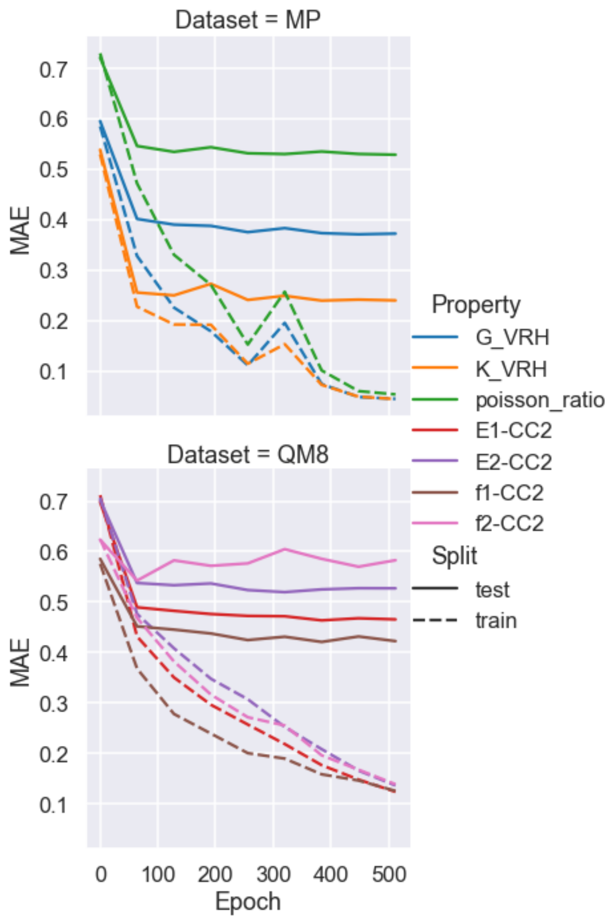
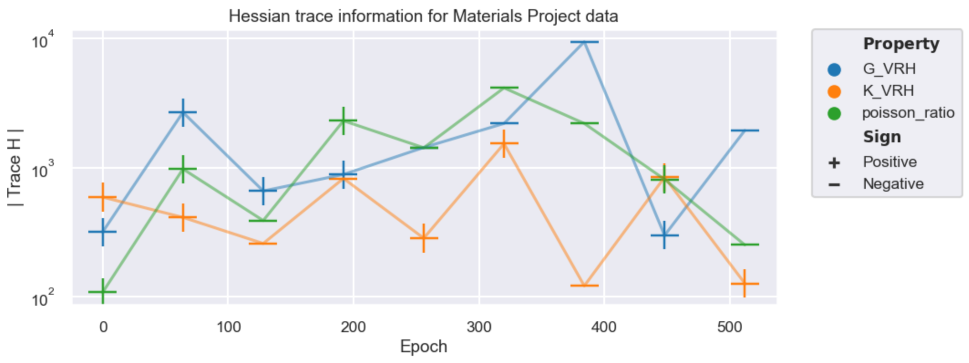
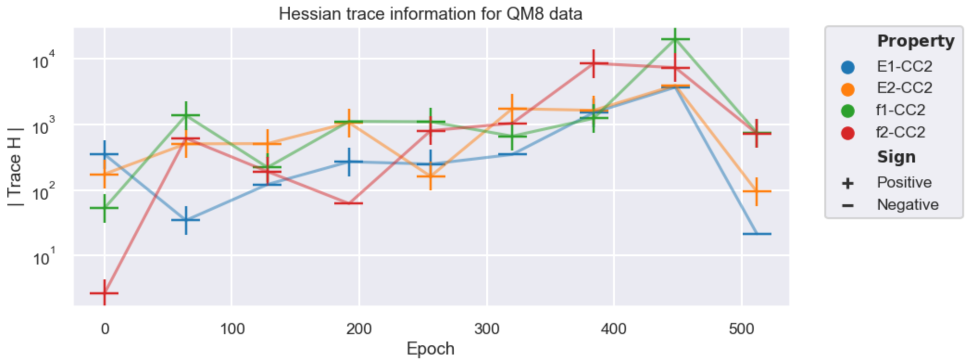


Figure 3. Standardized MAE values for each property across train and test splits, for the MP and QM8 prediction tasks. Despite the high variability in curvature across training (Figure 4), training error reduces relatively smoothly, with occasional regressions in performance for MP. Although the training error for each property converges to rough equality, significant inter-property variety is observed for test set error.



(a) MP



(b) QM8

Figure 4. We track property-specific Hessian traces $\operatorname{tr} H_t$ while training GNs for the MP and QM8 prediction problems. The loss surface of the MTL problem for GNs appears to be complicated and varying across properties. Trace signs (indicating the sign of the curvature) and trace magnitudes vary across training and across properties, the latter by orders of magnitude. In prior work that looked at traces of Hessians (Yao et al., 2019) in CV tasks, traces were found to be less variable.

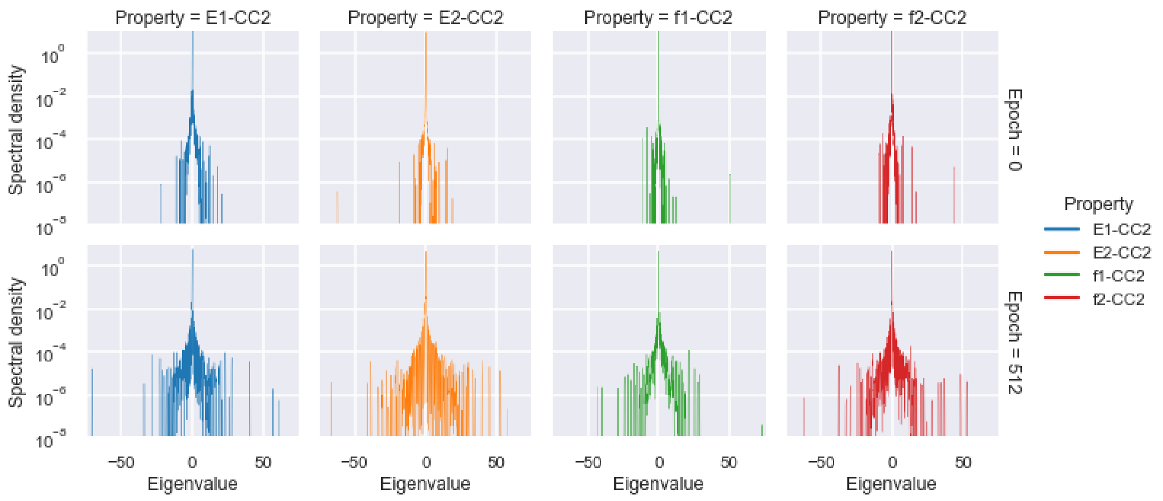


Figure 5. Snapshots of the spectral densities of property-specific Hessian losses for the QM8 dataset. All properties start with comparable densities closely concentrated near 0; but, as training finishes, the densities spread out as the loss surface increases in complexity. Each property has a different spread of eigenvalues, suggesting that the loss surfaces do vary in curvature by property. Note that this plot trims the range of the x-axis to remove outlier eigenvalues. The full spectra are displayed in Figure 6 in Appendix D. For these plots, the x-axis gives the range of eigenvalues for the loss function, and the y-axis gives the density $\hat{\psi}(t)$ of eigenvalues concentrated at that point t (as described in 2.1).

4. Discussion

In this work, we have posited that the performance of multi-task GNs for property prediction may be stymied by underlying variation in property-specific loss-surface curvatures. Some existing work considers this question (Yu et al., 2020) in a simplified theoretical setting, but, to the best of our knowledge, no one has empirically investigated the Hessian properties of multi-task GNs. In two domains – chemistry and materials science – our results suggest that loss surface curvature varies across each modeled property.

In order to assess curvature without calculating the full Hessian, we build on recent work that uses matrix-free methods to estimate Hessian properties (Alain et al., 2019; Yao et al., 2019; Ghorbani et al., 2019; Papyan, 2020). We extend their results by considering a novel type of learning problem – multi-output scalar regression – and a novel class of neural network – graph networks. Our results echo some previous results – the majority of a Hessian's spectral density is concentrated near 0 – and diverges in other respects – Hessian properties appear far noisier and more variable for GNs than for CV tasks. We leave further investigation of this question to future work. Potential explanations include the difference in loss functions for regression vs. classification tasks, a higher level of noise in the datasets we examined, and some special characteristic of GN vs. other neural network architectures.

We have here focused on a specific but representative subset of GN prediction problems, but considerable potential variation exists. For example, many recent GN architectures incorporate a notion of equivariance (Satorras et al., 2021; Gasteiger et al., 2020) into their feature-extraction models, and this might impact their curvature in different ways. In addition, we have not evaluated how the choice of optimizer (e.g., stochastic gradient descent vs. Adam (Kingma & Ba, 2014) vs. AdamW) impacts the curvature properties of a learned loss surface.

Our curvature assessment enable several future research directions. First, our analysis here was primarily empirical. The phenomena we identify here (a diversity of curvatures across multi-task loss functions) and the previously-observed degradation of performance for multi-task models (Gasteiger et al., 2020) could be connected by a theoretical justification. Similarly, we find that curvature properties of GNs appear to be much noisier and more variable than the properties of other network architectures, and we currently lack a theoretical justification of why. Intermediate steps might entail investigating the curvature properties of MTL methods that, in other domains, do out-perform single-task models (e.g., (Sener & Koltun, 2018)). Existing work in analyzing curvature for computational geometry (e.g. (Goldman, 2005)) might provide techniques to build upon.

Acknowledgements

This work was supported by internal research and development funding from the Research and Exploratory Development Mission Area of the Johns Hopkins University Applied Physics Laboratory. Thanks to Christopher Ratto for assistance in editing and improving the work.

References

Alain, G., Roux, N. L., and Manzagol, P.-A. Negative eigenvalues of the hessian in deep neural networks, 2019. URL https://arxiv.org/abs/1902.02366.

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 2011. doi: 10.1145/1944345.1944349.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., et al. Relational inductive biases, deep learning, and graph networks, 2018. URL https://arxiv.org/abs/1806.01261.

Caruana, R. Multitask learning. *Machine Learning*, 1997. doi: 10.1023/A:1007379606734.

Chen, C., Ye, W., Zuo, Y., Zheng, C., and Ong, S. P. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 2019. doi: 10.1021/acs.chemmater.9b01294.

Gasteiger, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2232–2241, 09–15 Jun 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proc. of the 34th Int. Conf. on Machine Learning - Volume 70*, 2017.

Godwin, J., Schaarschmidt, M., Gaunt, A. L., Sanchez-Gonzalez, A., Rubanova, Y., et al. Simple GNN regularisation for 3d molecular property prediction and beyond. In *Int. Conf. on Learning Representations*, 2022.

Goldman, R. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 2005. doi: 10.1016/j.cagd.2005.06.005. Geometric Modelling and Differential Geometry.

Golub, G. H. and Welsch, J. H. Calculation of gauss quadrature rules. *Mathematics of Computation*, pp. 221–s10, 1969.

Hill, R. The elastic behaviour of a crystalline aggregate. *Proc. of the Phys. Soc. Section A*, 65(5):349–354, may 1952. doi: 10.1088/0370-1298/65/5/307.

Hättig, C. and Weigend, F. Cc2 excitation energy calculations on large molecules using the resolution of the identity approximation. *J. Chem. Phys.*, 2000. doi: 10.1063/1.1290013.

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., et al. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 2013. doi: 10.1063/1.4812323.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Kong, S., Guevarra, D., Gomes, C. P., and Gregoire, J. M. Materials representation and transfer learning for multi-property prediction. *Appl. Phys. Rev.*, 2021. doi: 10.1063/5.0047066.

Lin, L., Saad, Y., and Yang, C. Approximating spectral densities of large matrices. *SIAM Review*, 2016. doi: 10.1137/130934283.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 1989. doi: 10.1007/BF01589116.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *Int. Conf. on Learning Representations*, 2019.

Papyan, V. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

Park, C. W. and Wolverton, C. Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery. *Phys. Rev. Materials*, 2020. doi: 10.1103/PhysRevMaterials.4.063801.

Pearlmutter, B. A. Fast Exact Multiplication by the Hessian. *Neural Computation*, 1994. doi: 10.1162/neco.1994.6.1.147.

Ramakrishnan, R., Hartmann, M., Tapavicza, E., and von Lilienfeld, O. A. Electronic spectra from tddft and machine learning in chemical space. *The Journal of Chemical Physics*, 2015. doi: 10.1063/1.4928757.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks, 2017. URL https://arxiv.org/abs/1706.04454.

Sanyal, S., Balachandran, J., Yadati, N., Kumar, A., Rajagopalan, P., Sanyal, S., and Talukdar, P. Mt-cgcnn: Integrating crystal graph convolutional neural network with multitask learning for material property prediction, 2018. URL https://arxiv.org/abs/1811.05660.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9323–9332, 18–24 Jul 2021.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Proc. 32nd Int. Conf. on Neural Information Processing Systems*, NIPS'18, 2018.

Tan, Z., Li, Y., Shi, W., and Yang, S. A multitask approach to learn molecular properties. *J. Chem. Inf. Mod.*, 61 (8):3824–3834, 2021. doi: 10.1021/acs.jcim.1c00646. PMID: 34289687.

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.*, 1988. doi: 10.1021/ci00057a005.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., et al. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018. doi: 10.1039/C7SC02664A.

Xie, T. and Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 2018. doi: 10.1103/PhysRevLett.120.145301.

Yang, A., Su, Y., Wang, Z., Jin, S., Ren, J., et al. A multi-task deep learning neural network for predicting flammability-related properties from molecular structures. *Green Chem.*, 2021. doi: 10.1039/D1GC00331C.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. Pyhessian: Neural networks through the lens of the hessian, 2019. URL https://arxiv.org/abs/1912.07145.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. In *Adv. Neural Inf. Proc Sys.*, 2020.

A. Model specifics

We use $M = 5$ message-passing steps. Node, edge, and global features are projected into a 64-dimensional feature space. ϕ^V and ϕ^E have two layers with 256 units and a skip connection, followed by an output layer of 64 units. ϕ^U has two layers with 192 units and a skip connection, followed by an output layer of 64 units. All networks use `tanh` activations.

B. Training specifics

Models are trained for 512 epochs using the AdamW (Loshchilov & Hutter, 2019) optimizer and the default hyperparameters used in the `optax` implementation. The initial rate is set as 10^{-3}, with an exponential decay rate of 0.997 applied every epoch after the first 256 epochs.

C. Data

We summarize our datasets in Table 1.

We scraped Materials Project for crystal records present in it as of October 2020 using the `MPRester` class from the `pymatgen` package. The Supplemental Information contains a table of MP IDs used in this study. The raw entries of G_{VRH}, K_{VRH}, and Poisson ratio μ contained several anomalously small and large values, so we removed entries with values less than the 5th percentile or greater than the 95th percentile of obtained elastic properties. This left us with a total dataset of 10,500 crystals. Summary statistics for the targets used are given in Table 2. Furthermore, G_{VRH} and K_{VRH} were log-transformed to create a more normal distribution. Roughly 70% of the dataset (7,400 crystals) was used as training data, and all targets were standardized prior to training.

As initial node features, we used a one-hot encoding of atom type. For initial edge features, we used four features calculated by `pymatgen`: `area`, `face_dist`, `solid_angle`, and `volume`. Following similar work in applying graph neural networks to crystals (Xie & Grossman, 2018), we discretize these four features based on deciles.

We use the version of QM8 hosted by MoleculeNet (Wu et al., 2018), except that we drop molecules with negative oscillator energies. Summary statistics for the targets are given in Table 3. As initial node features, we use one-hot encodings of atom type. As initial edge features, we use one-hot encodings of bond type. Roughly 70% of the dataset (15,300 molecules) was used as training data, and all targets were standardized prior to training.

Table 1. Summary statistics of datasets

Dataset	# Data Points	# Targets	# Node Features	# Edge Features
MP	10,500	3	112	40
QM8	21,725	4	9	4

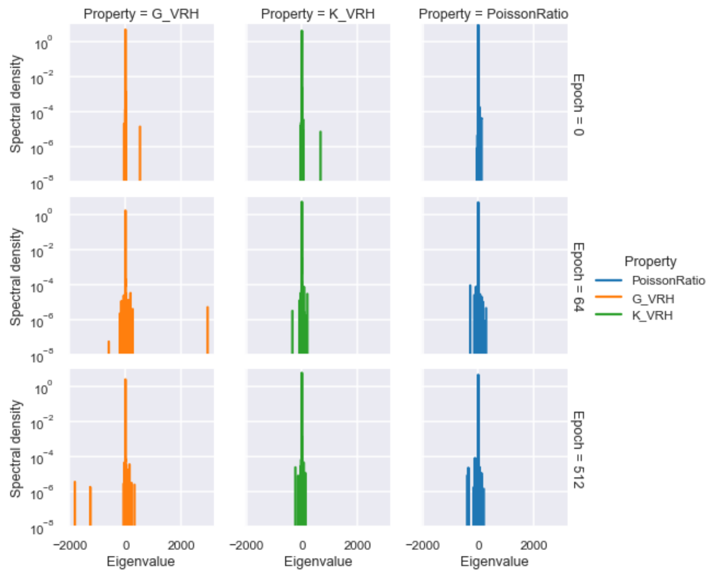
Table 2. Summary statistics of MP targets

	log G_{VRH} (log(GPa))	log K_{VRH} (log(GPa))	μ
mean	3.55068348	4.3697952	0.30674381
std	0.73963662	0.67564539	0.06501633
min	0.0	2.63905733	0.18
max	4.82028157	5.45103845	0.5

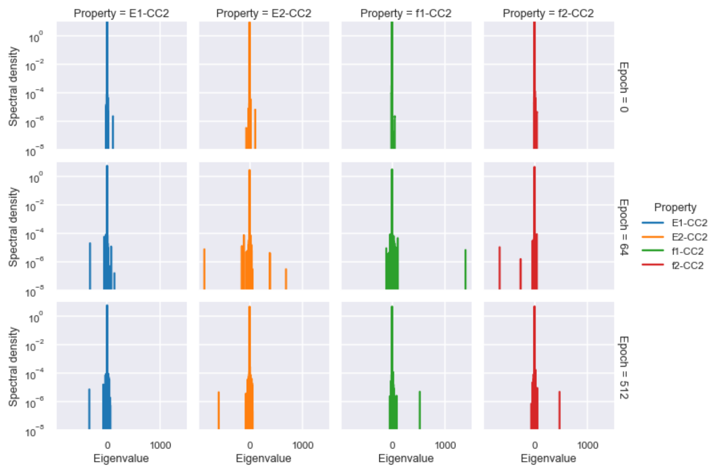
Table 3. Summary statistics of QM8 targets

	E_1	E_2	f_1	f_2
mean	0.22010506	0.24894142	0.02289806	0.04326408
std	0.04382473	0.0345149	0.05336499	0.07312536
min	0.06956711	0.10063996	0.0	0.0
max	0.51383669	0.51384601	0.61611219	0.56561272

D. Supplemental figures



(a) MP



(b) QM8

Figure 6. We visualize snapshots of property-specific estimated spectral densities during selected training points. For both MP and QM8 models, most of the spectral density is concentrated near 0, especially after random initialization. Occasionally, very high-magnitude eigenvalues, both positive and negative, spike up for a short period. Our results here elaborate on Figure 4 – the high and varying curvature of property-specific losses is driven by singularly large eigenvalues that vary across properties. For these plots, the x-axis gives the range of eigenvalues for the loss function, and the y-axis gives the density $\hat{\psi}(t)$ of eigenvalues concentrated at that point t (as described in 2.1).

E. Example spectrum estimation

We consider a loss function defined by $L(x) = \frac{1}{2}x^T Ax$, where $A = \frac{1}{2}(B + B^T)$, for a matrix B with entries sampled iid from a standard normal distribution. In this case, the Hessian is given by A and so its estimated eigenvalues can be compared to its true eigenvalues. In Figure 7, we plot the results of a sample calculation, for a $1,000 \times 1,000$ matrix. We show that the estimated density has reasonable similarity to the true distribution, and the estimated trace also matches the true trace.

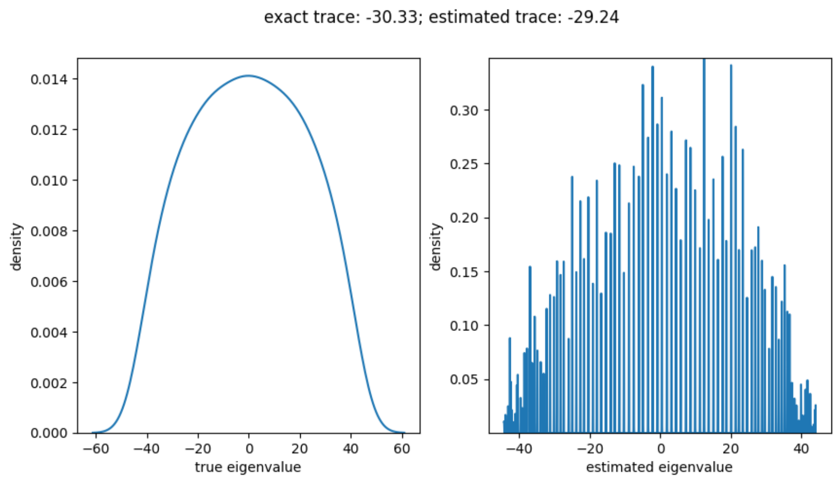


Figure 7. An example showing the use of the Lanczos algorithm to estimate the spectral density of a Hessian. The left figure is the distribution of the true eigenvalues, and the right is the estimated spectral density. The estimated values accurately characterize the maximal and minimal eigenvalues, and the interior shows qualitative agreement with the known distribution.