Shoot from the HIP: Hessian Interatomic Potentials without derivatives

Andreas Burger[*]
University of Toronto
Vector Institute

Luca Thiede
University of Toronto
Vector Institute

Nikolaj Rønne
Technical University of Denmark

Nandita Vijaykumar
University of Toronto

Tejs Vegge
Technical University of Denmark

Arghya Bhowmik
Technical University of Denmark

Alán Aspuru-Guzik
Acceleration Consortium
University of Toronto
Vector Institute
NVIDIA

Abstract

Fundamental tasks in computational chemistry, from transition state search to vibrational analysis, rely on molecular Hessians, which are the second derivatives of the potential energy. Yet, Hessians are computationally expensive to calculate and scale poorly with system size, with both quantum mechanical methods and neural networks. In this work, we demonstrate that Hessians can be predicted directly from a deep learning model, without relying on automatic differentiation or finite differences. We observe that one can construct SE(3)-equivariant, symmetric Hessians from irreducible representations (irrep) features up to degree l=2 computed during message passing in graph neural networks. This makes HIP Hessians one to two orders of magnitude faster, more accurate, more memory efficient, easier to train, and enables more favorable scaling with system size. We validate our predictions across a wide range of downstream tasks, demonstrating consistently superior performance for transition state search, accelerated geometry optimization, zero-point energy corrections, and vibrational analysis benchmarks. We open-source the HIP codebase and model weights to enable further development of the direct prediction of Hessians.

1 Introduction

Over the past decades, molecular simulation has become a cornerstone for many tasks in material discovery and molecular design. Beyond energy and forces, chemists, chemical engineers, materials scientists, and physicists frequently rely on Hessians to enable many critical workflows [1, 2]. For example, *second-order geometry optimization* accelerates the determination of reliable equilibrium structures. *Transition-state searches* are needed to find reaction pathways and estimate barrier heights and transition rates. This determines whether a reaction is viable and how to tune selectivity, with high industrial relevance for the chemical and pharmaceutical industry. *Vibrational analyses* connect theory to experiment through infrared and Raman spectra and provide zero-point energies, which are essential for ranking isomers and estimating reaction free energies.

[*]andreas.burger@mail.utoronto.ca

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: SIMBIOCHEM.

Despite their broad utility, Hessian calculations remain a significant computational bottleneck. For a molecule with N atoms, the Hessian is a $3N \times 3N$ matrix, where each entry requires a mixed second derivative of the electronic energy with respect to two nuclear coordinates

$$\mathbf{H}_{IJ}^{\alpha,\beta} = \frac{\partial^2 E}{\partial \mathbf{R}_I^\alpha \, \partial \mathbf{R}_J^\beta} \tag{1}$$

with $I, J = 1, \ldots, N$ indexing atoms, and $\alpha, \beta \in \{x, y, z\}$ indexing the cartesian x, y and z components of the atom positions $\mathbf{R}$.

Traditional approaches rely either on analytic second derivatives or on numerical differentiation of gradients, both of which have severe limitations [3, 4, 5]. Analytic second derivatives are available only for a limited set of methods, such as Hartree-Fock, DFT, and MP2; whereas, for higher-level correlated wavefunction methods such as CCSD or CASPT2, they are either not yet implemented in most computational packages or computationally prohibitive. Numerical approaches sidestep analytic derivations but can become infeasible even for medium-sized molecules, requiring $O((3N)^2)$ gradient evaluations [6].

Most computational chemistry calculations are carried out with Kohn-Sham density functional theory [7] (KS-DFT), due to its balance of computational cost and accuracy [8]. However, the $O(N^4)$ scaling of hybrid KS-DFT with system size remains expensive, which has spurred the development of machine-learning interatomic potentials (MLIPs). MLIPs promise to deliver near-DFT accuracy at a fraction of the cost by incorporating symmetries, thereby drastically reducing the need for extensive training data [9, 10, 11, 12, 13, 14]. Recently, some MLIP models have also been adapted to compute Hessians via automatic differentiation (AD). These AD Hessians incur significant computational costs of $O(N^2)$, which presents practical challenges during training and inference [15, 16, 17, 18, 19].

In this work we introduce **HIP**: **H**essian **I**nteratomic **P**otentials. HIPs predict the Hessian directly, to eliminate the need for finite differences, coupled-perturbed equation solvers, or automatic differentiation. By building on SE(3)-equivariant neural networks, we demonstrate how HIP can predict Hessians at a fraction of the computational cost of traditional methods. A key observation is that **one can construct the Hessian from spherical harmonic features of the atom nodes and the messages between atoms**, while satisfying equivariance and symmetry constraints by design. This approach is not only faster but also more accurate, enabling a variety of critical tasks in molecular simulations.

Our contributions are the following:

1. We present Hessian Interatomic Potentials (HIP), a method to construct SE(3)-equivariant, symmetric Hessians *directly*, without finite differences, coupled-perturbed solvers, or automatic differentiation, using an equivariant neural network backbone.
2. We introduce a loss function that targets the part of the Hessian subspace particularly important for molecular optimization problems.
3. We show that, compared to AD, the predicted Hessians are 10-70x faster for single molecule evaluation and over 70x faster in batched molecular prediction, while also requiring 2-3x less peak memory on small molecules with 5-30 atoms.
4. We show that the predicted Hesians are more accurate on benchmark datasets, with 2x lower Hessian MAE, 3.5x lower eigenvalue MAE, and 1.5x higher eigenvector cosine similarity.
5. Finally, we rigorously validate the practical utility of our predicted Hessians in several downstream applications, such as (i) zero-point energy corrections, (ii) second-order geometry optimization, (iii) transition state search, and (iv) frequency analysis for extrema classification.

2 Background

2.1 MLIPs and equivariant neural networks

Early works on machine learning interatomic potentials used local atomic environment descriptors in combination with linear regression [20, 21], Gaussian processes [22], and simple neural networks [23]. SchNet [9] was the first work to introduce *rotation-invariant graph neural networks* to predict molecular properties like energies and forces. Next-generation invariant architectures include ViSNet [24] and QuinNet [25]. In parallel, *rotation-equivariant* architectures were developed, such as

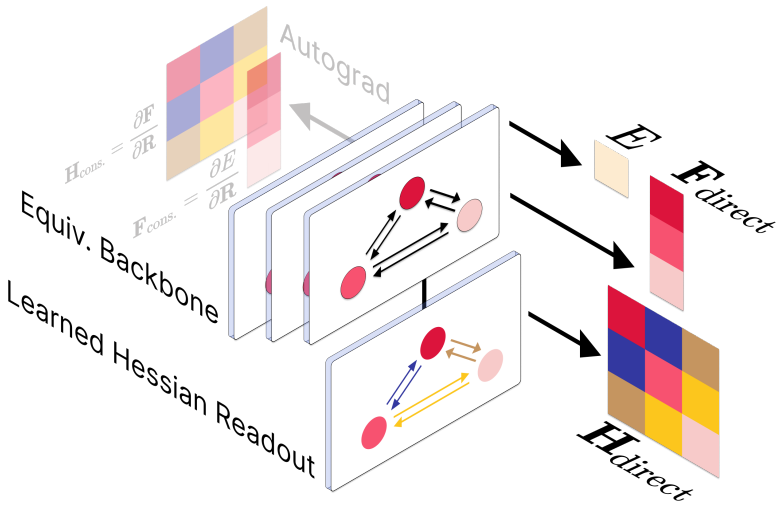

Figure 1: In contrast to previous methods that rely on AD to construct the Hessian matrix, Hessian Interatomic Potentials (HIP) use a learned Hessian readout that predict the Hessian *directly*.

Cormorant [26], DimeNet [27], GemNet [11], PaiNN [12], NequIP [14], SphereNet [28], MACE [13] and the Equiformer family [29, 30]. *Graph neural networks* represent a molecule as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in 3D Euclidean space $\mathbb{R}^3$, with nodes/atoms $I \in \mathcal{V}$ at positions $\mathbf{r}_I \in \mathbb{R}^3$ and edges $(I, J) \in \mathcal{E}$ defined by a cutoff radius. At layer t, each node carries a feature $\mathbf{h}_I^{(t)}$ that is a direct sum of irreducible $\mathrm{SO}(3)$ representations (irreps): $\mathbf{h}_I^{(t)} = \bigoplus_{l=0}^{l_{\max}} \mathbf{h}_I^{(t,l)}$, where $\mathbf{h}_I^{(t,l)} = \{\mathbf{h}_{I,l,m}^{(t)}\}_{m=-l}^{l}$ has $2l+1$ components. To update the node's features, the model sends messages between each connected node, which are then aggregated in a permutation invariant fashion (usually the sum, or attention-weighted-sum operation) and then added to the receiving node. In $\mathrm{SO}(3)$-equivariant GNNs, node features transform under a global rotation $\mathbf{Q}$ via Wigner $\mathbf{D}$-matrices:

$$\mathbf{h}_{i,l,m}^{(t)}\left(\mathbf{Q}\{\mathbf{r}_k\}_{k=1}^{N}\right) = \sum_{m'=-l}^{l} \mathbf{D}_{mm'}^{(l)}(\mathbf{Q})\, \mathbf{h}_{i,l,m'}^{(t)}\left(\{\mathbf{r}_k\}_{k=1}^{N}\right), \tag{2}$$

Translation equivariance is enforced by building messages from relative displacements $\mathbf{r}_{I,J} = \mathbf{r}_J - \mathbf{r}_I$. If we want O(3) instead of SO(3) equivariance, each order-l feature has an extra parity label that determines the transformation under a global coordinate inversion.
Two irreps $\mathbf{p}_{l_1}$ and $\mathbf{g}_{l_2}$ interact using the Clebsch-Gordan tensor product [10]

$$\mathbf{h}_{l_3,m_3} = \left(\mathbf{p}_{l_1,m_1} \otimes_{l_1,l_2}^{l_3} \mathbf{g}_{l_2,m_2}\right)_{m_3} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} \mathbf{C}_{(l_1,m_1),(l_2,m_2)}^{(l_3,m_3)} \mathbf{p}_{l_1,m_1} \mathbf{g}_{l_2,m_2} \tag{3}$$

A message from source atom J to target atom I is then constructed as

$$\mathbf{v}_{I,J,l_3} = \mathbf{v}_{l_3}(\mathbf{h}_I, \mathbf{h}_J, \mathbf{r}_{I,J}) = \sum_{l_1,l_2} w_{l_1,l_2,l_3}(||\mathbf{r}_{I,J}||) \left(\mathbf{f}_{l_1}(\mathbf{h}_I, \mathbf{h}_J) \otimes_{l_1,l_2}^{l_3} \mathbf{Y}_{l_2}\left(\frac{\mathbf{r}_{I,J}}{||\mathbf{r}_{I,J}||}\right)\right) \tag{4}$$

where $\mathbf{Y}_{l_2}$ is the spherical harmonic of degree l_2, $w_{l_1,l_2,l_3} : \mathbb{R} \to \mathbb{R}$ is a learned weighting function, and $\mathbf{f}_{l_1}$ a simple function that takes both node features in, for example concatenation in the case of EquiformerV2. A more efficient variation of equation 4 relying on the reduction of the $SO(3)$ tensor product to the $SO(2)$ tensor products was proposed by eSCN [31] and adopted by subsequent works, such as EquiformerV2. Finally, the messages are pooled to update the node features, which can be done, for example, via a sum or using graph attention [30].
After several layers of message passing, task-dependent readout heads map the node features to the desired targets while respecting symmetry. For example, energies E are $\mathrm{SO}(3)$-invariant per-graph scalars. The energy readout head therefore uses a global pooling and a reduction operation to output a l=0 feature. Forces $\mathbf{F}$ are $\mathrm{SO}(3)$ equivariant per-atom vectors ($l = 1$), and can be outputted either by a direct force readout head, or as the derivative of the energy $\mathbf{F} = -\nabla_{\mathbf{R}} E$. As we will see in section 3, Hessians are $\mathrm{SO}(3)$-equivariant per-graph matrices composed of per-atom-pair 3×3 equivariant sub-matrices.

2.2 Hessians from MLIPs

MLIPs have gained widespread popularity for their ability to accurately approximate molecular energies and forces. The forces can either be directly predicted [11, 30], or calculated as the derivative of the MLIP energy using automatic differentiation. Previous work has shown that the Hessian can be calculated using AD to some success [32, 15, 16, 17, 18, 19]. Although AD Hessians are much cheaper to calculate compared to DFT, they still have some downsides. First, accurately modeling the energy does not guarantee low errors on Hessians [15], but requires dedicated training [33, 19]. Second, compared to energies and forces, Hessians are expensive: Given a probing vector $\mathbf{v}$, AD only lets us calculate $\mathbf{Hv}$. To calculate the full $3N \times 3N$ Hessian, we need $3N$ Hessian vector products (HVPs) with the unit vectors $\mathbf{v} = \mathbf{e}_i, i = 0, ..., 3N-1$, each yielding one column of the Hessian. Since the energy calculation with an MLIP is usually $O(N)$, the total cost of the Hessian calculation is $O(N^2)$. While the HVPs can theoretically be parallelized, the $O(N^2)$ memory footprint requires even medium-sized systems to be computed sequentially.

Instead of relying on automatic differentiation, we propose to directly predict Hessians with an equivariant neural network. Unlike direct force prediction, Hessians only need to be symmetric rather than energy-conservative[34], and they offer a much greater speedup, as shown in section 4.

3 HIP Hessian Prediction

Any equivariant neural network [35] with features of at least $l = 2$ can be equipped with a Hessian readout head. In this work, we pick the EquiformerV2 architecture as the backbone with four transformer layers [30]. Each layer contains a layer norm, message passing with graph attention, another layer norm, and a feed-forward layer.

3.1 Hessian Symmetry Requirements

Hessians are symmetric, real-valued, Cartesian tensors. This means, under rotation of the coordinate system with rotation matrix $\mathbf{Q}$, each Hessian sub-block $\mathbf{H}_{I,J}$ transforms as

$$\mathbf{H}_{I,J} \xrightarrow{\mathbf{Q}} \mathbf{Q}\mathbf{H}_{I,J}\mathbf{Q}^\top \tag{5}$$

Any physically meaningful Hessian has to satisfy this rotation symmetry, as well as the symmetry of the upper and lower triangular $\mathbf{H} = \mathbf{H}^\top$. We elaborate on the symmetry of Hessians in A.3.

3.2 Hessian Prediction Head

Starting from T layers of message passing in the backbone, we use the atom features $\mathbf{h}^{(T)}_{I,m,c}$, and feed them into a Hessian readout module. Here, we first add another Hessian readout-specific EquiformerV2 layer to improve expressivity. We then construct the Hessian sub-blocks: We first send normal messages with equation 4, but instead of attention and summing, we treat the messages as atom-pair features:

$$\mathbf{h}_{I,J,l,m,c} = \mathbf{v}(\mathbf{h}^{(T)}_I, \mathbf{h}^{(T)}_J, \mathbf{r}_{I,J}) = \sum_{l_1,l_2} w_{l_1,l_2,l_3}(||\mathbf{r}_{I,J}||) \left(\mathbf{f}_{l_1}(\mathbf{h}_I, \mathbf{h}_J) \otimes^{l_3}_{l_1,l_2} \mathbf{Y}_{l_2} \left(\frac{\mathbf{r}_{I,J}}{||\mathbf{r}_{I,J}||} \right) \right) \tag{6}$$

As we are reusing the message passing machinery with an interaction cutoff radius, predicting Hessians scales initially $O(N)^2$ in memory and compute within the cutoff, and then reduces to $O(N)$ scaling for larger systems. This sparsity with distance assumption aligns with the sparsity of Hessians due to locality of electron interactions that is common in quantum chemistry and can be mathematically rigorously justified [36]. After message passing we have to transform the atom pair features $\mathbf{h}_{I,J,l,m,c}$ into the corresponding Hessian sub-blocks $\mathbf{H}_{I,J}$. First, we project the pair feature irreps down to a single $\{\tilde{\mathbf{h}}_{I,J,l,m}\}_{l\in\{0,1,2\}}$ irreps feature (`1x0e + 1x1e + 1x2e` in `e3nn` notation [37]) using a linear layer $\mathbf{W}_{l,c}$

$$\tilde{\mathbf{h}}_{I,J,l,m} = \mathbf{W}_{l,c}\mathbf{h}_{I,J,l,m,c} \tag{7}$$

We then expand $\tilde{\mathbf{h}}_{I,J,l,m}$ to an intermediate Hessian sub-block $\mathbf{H}'_{I,J}$ using the tensor product expansion [38]:

$$\mathbf{H}'_{I,J,m_1,m_2} = \sum_{l,m} \mathbf{C}^{l,m}_{l_1=1,m_1,l_2=1,m_2} \tilde{\mathbf{h}}_{I,J,l,m} \tag{8}$$

where $\mathbf{C}^{l,m}_{l_1,m_1,l_2,m_2}$ are the Clebsch-Gordon coefficients ensuring that the relation in equation 5 is enforced [38]. Physically, this is a simple change of basis from the coupled to the uncoupled angular momentum basis [39]. $\mathbf{H}'_{I,J,m_1,m_2}$ is a 3×3 block, since m_1 and m_2 run from $-l_1 = -l_2 = -1$ to $l_1 = l_2 = 1$. As $\mathbf{C}^{l,m}_{l_1=1,m_1,l_2=1,m_2} = 0$ for all $l > 2$, $\mathbf{h}_{I,J,l,m,c}$ only needs to contain irreps up to $l = 2$ to build the 3×3 Hessian sub-blocks. Intuitively, equation 8 is the inverse of the irreducible tensor decomposition, so it assembles a spherical tensor back from its irreducible components. Conversely, this also means we need all irreps up to $l = 2$ as only including irreps up to $l = 1$ would not be sufficient to express the 3×3 tensors that decompose into $l = 2$ irreps. Finally, we symmetrize the immediate Hessian to the final prediction with $\mathbf{H} = \mathbf{H}' + \mathbf{H}'^{\top}$.

3.3 Loss function design

To learn the Hessians, one can use standard loss functions like mean absolute error (MAE) or mean squared error (MSE) between predicted and actual Hessian elements:

$$\mathcal{L}_{\text{MAE / MSE}} = \sum_{i,j} |\mathbf{H}_{i,j} - \mathbf{H}^{\text{pred}}_{i,j}|_{\text{MAE / MSE}} \tag{9}$$

However, we are often mainly interested in the smallest eigenvalues and the corresponding eigenvectors [40, 41]. For this reason, we design a loss function to emphasize this subspace of the eigen-decomposition.
Let $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{3N}]$ be the matrix with columns made from eigenvectors of the ground truth Hessian $\mathbf{H}$, and $\mathbf{\Lambda}$ the corresponding matrix with eigenvalues on the diagonal and zeros everywhere else. Further denote $\mathbf{V}_{[:,:k]}$ and $\mathbf{V}_{[:,k:]}$ the matrices sliced to contain all columns up to/starting from the k^{th} one. We can then define a subspace loss by

$$\mathcal{L}_{\text{sub}} = \sum_{i,j} \left| \mathbf{V}^{\top}_{[:,:k]} \mathbf{H}^{\text{pred}} \mathbf{V}_{[:,:k]} - \mathbf{\Lambda}_{[:,:k]} \right|_{i,j} \tag{10}$$

If $\mathbf{H}^{\text{pred}} = \mathbf{H}$ we have $\mathbf{V}^{\top}\mathbf{H}^{\text{pred}}\mathbf{V} = \mathbf{\Lambda}$, and the loss has therefore the correct minimum. Using $\mathcal{L} = \mathcal{L}_{\text{MAE/MSE}} + \alpha\mathcal{L}_{\text{sub}}$ lets us emphasize the subspace corresponding to the k lowest eigenvectors and eigenvalues. Since the Hessian always has six redundant degrees of freedom with eigenvalues close to zero, we need to set $k > 6$. In practice, we use $k = 8$. As we will show in A.6, predicting Hessians with HIP using any loss function significantly outperforms AD Hessians. MAE combined with the subspace loss in equation 10 further improves results in transition state search and extrema classification with frequency analysis by a small margin, which is why we use them in the experiment of the main text. For details and ablations of the choice of loss function see A.6.

4 Experiments

We evaluate our proposed approach for direct prediction of molecular Hessians across multiple tasks that test both accuracy and practical utility. Specifically, we measure prediction accuracy and quantify computational speed-ups relative to AD and finite-difference Hessian calculation. We further assess downstream performance in geometry optimization, zero-point energy, and transition state (TS) searches. For TS searches, we additionally perform frequency analyses to evaluate the reliability of the predicted Hessians in distinguishing true transition states. See A.1 for details.

Dataset The HORM dataset contains DFT energies, forces, and Hessians computed at the ωB97X/6-31G* level of theory. The training dataset comprises 1,725,362 geometries sampled from the Transition1x dataset, with 50,844 additional geometries serving as the validation set. To isolate the Hessian from the forces and energy, we train only the Hessian prediction head using a fixed EquiformerV2 backbone from HORM [19]. This ensures that the energy and force predictions of AD-EquiformerV2 and HIP-EquiformerV2 are the same for all experiments. All baseline models originate from [19] and were also trained using the same ground-truth DFT Hessians. Therefore, all downstream task improvements are purely due to better Hessians.

Table 1: **Comparison of Hessian prediction errors on the HORM-Transition1x validation set.** Numbers denote the mean absolute error (MAE), except for the cosine similarity (CosSim) and average time per forward pass of a single molecule. Bold values highlight the best-performing model in each task. Baseline AD models are taken from [19].

Hessian	Model	Hessian ↓ eV/Å^2	Eigenvalues ↓ eV/Å^2	CosSim v_1 ↑ unitless	λ_1 ↓ eV/Å^2	Time ↓ ms
AD	AlphaNet	0.385	2.871	0.750	0.770	767.0
	LEFTNet	0.150	0.684	0.804	0.322	1110.7
	LEFTNet-df	0.197	0.669	0.532	0.742	341.3
	EquiformerV2	0.074	0.242	0.541	0.324	633.0
Predicted	HIP-EquiformerV2	**0.030**	**0.063**	**0.870**	**0.130**	**38.5**

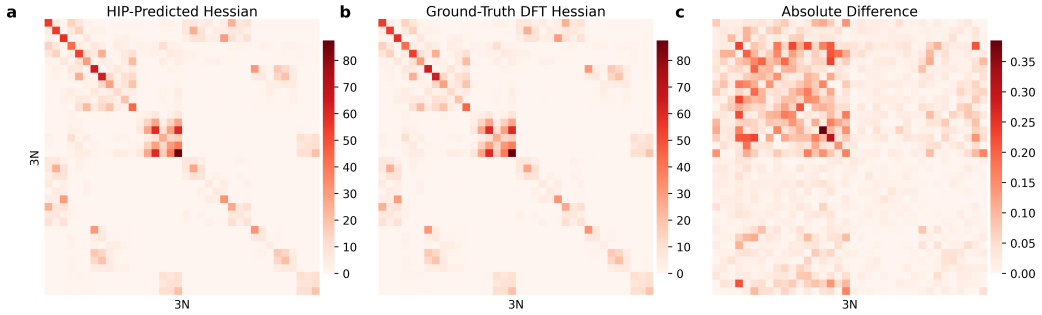

Figure 2: **HIP compared to ground-truth DFT Hessian.** The horizontal and vertical axis index the $(3N_{Atoms})^2$ entries of the Hessian. We depict the absolute values of test sample 12345 in eV/Å^2.

Accuracy To quantitatively assess the accuracy and speed of our directly predicted Hessians, we calculate the average elementwise mean absolute error, average eigenvalue error, and average time per prediction. Since, for many tasks such as ZPE and TS search, the smallest eigenvalue/eigenvector pair is particularly important, we calculate the cosine similarity and eigenvalue error for these tasks separately. The results are presented in Table 1. In every metric, our direct prediction Hessian approach outperforms the other models. The improvement is especially notable in prediction speed, where we are an order of magnitude faster than the models relying on automatic differentiation.
In Figures 2(a) and (b), we visually inspect the HIP Hessian compared to the ground-truth DFT Hessian, respectively. We observe good agreement with the absolute difference between the two shown in Figure 2(c).

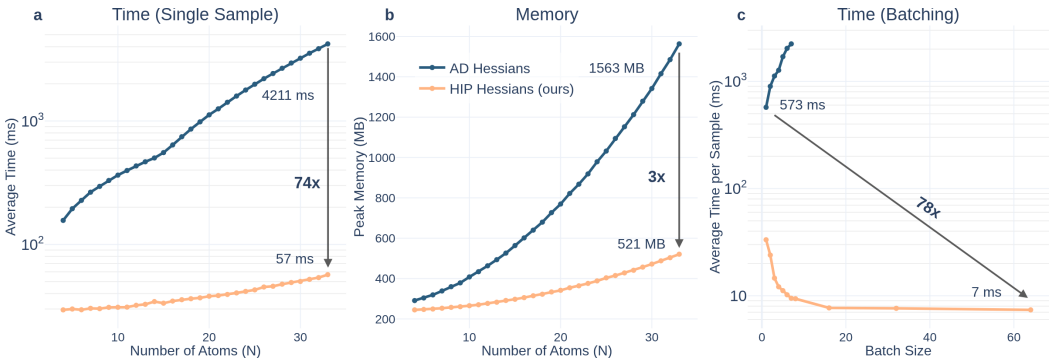

Figure 3: **Computational cost.** (a) Speed and (b) memory footprint for a single sample as a function of molecule size, and (c) scaling of Hessian predictions in parallel with increasing batch size. AD values are shown in blue, and HIP (this work) in orange.

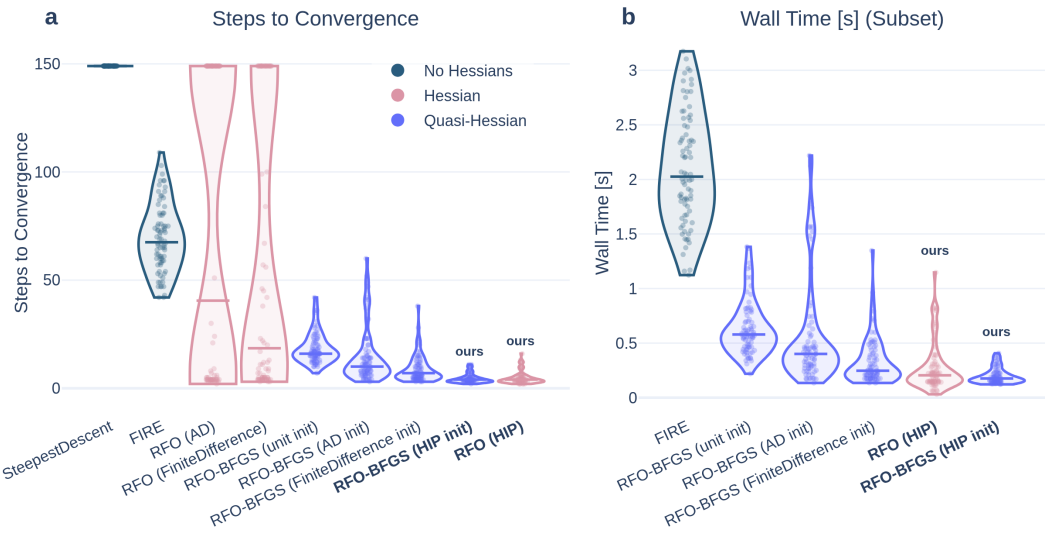

Figure 4: **Geometry optimization.** Methods ordered by their median (horizontal line) for (a) number of steps and (b) wall-clock time to convergence. For clarity, we omit to plot the wall-clock time of RFO with AD or finite difference Hessians, as they require one and two orders of magnitude more time than RFO with HIP predicted Hessians.

Speedup and memory In Figure 3(a-b), we further compare the prediction time and memory footprint as a function of molecule size. The direct prediction of Hessians is 10 - 74× faster than using AD for molecules in the HORM dataset. HIP predicted Hessians also benefit from a much more favorable scaling with the molecule size. Figure 3(c) compares the batched AD Hessian as implemented in [19] with our HIP-EquiformerV2. We observe at minimum a 78× speedup as a function of batch size compared to AD. We explain the significant performance degradation of batched AD Hessians in A.4.

Geometry optimization We now turn to downstream applications of the Hessian, first focusing on geometry optimization. We compare exact second-order approaches using Hessians (RFO) against first-order methods, and quasi-second-order RFO initialized with Hessians followed by BFGS updates. See A.7.1 for details. Figure 4(a) shows that RFO with HIP Hessians converges within the fewest steps in the median case. In contrast, both finite difference Hessians and AD Hessians frequently fail to converge. Among the Hessian-initialized BFGS methods (blue), the predicted Hessian also performs the best. Steepest descent with line search does not converge within the step budget, which demonstrates the difficulty of the task.
As seen in Figure 4(b), RFO and BFGS methods with HIP Hessians are also the fastest in terms of wall-clock time. The current implementation is bottlenecked by memory transfer from and to the CPU. We would expect an even more significant speedup if the optimization was performed on GPU and parallelized across samples.

Zero-point energy We evaluated an important thermochemical property of the relaxed equilibrium state, the zero-point energy (ZPE). We report both the ZPE of the reactant, as well as the relative ZPE between reactant and product ΔZPE. Experimental details are in A.7.3. Table 2 shows the mean absolute error with standard deviation (MAE (Std)) for each model compared to DFT. The HIP-predicted Hessians reach an order of magnitude lower MAE than the next-best model, and two orders of magnitude lower MAE than the same model using automatic differentiation Hessians. The results are also well below the chemical accuracy of 43 meV.

Transition state search Another important application of Hessians is transition state search. We use the recently introduced ReactBench benchmark [33], but *additionally* verify the predicted transition states with DFT. A step-by-step description of the workflow can be found in A.7.2. We present metrics of each step of the workflow in Figure 5. The growing string method (GSM) success rate is the same for both AD and HIP-EquiformerV2 models. GSM only uses energy and forces, which

are the same for both models. For all subsequent metrics, the HIP predicted Hessians outperform the AD ones. The RFO convergence and DFT-verified convergence metrics depend on the Hessian the most. For both, the HIP Hessians yield the most improvements over AD Hessians.

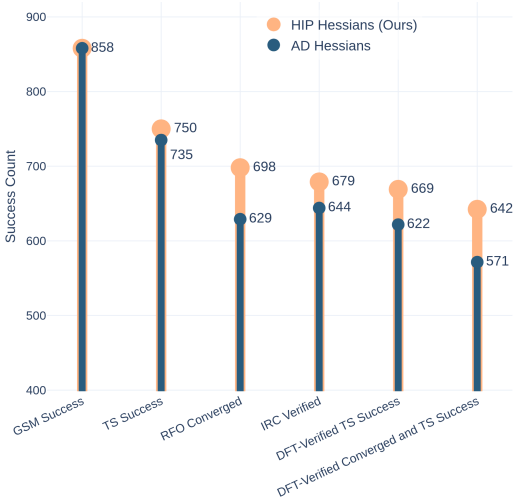

Figure 5: **Transition state search workflow success rate across diverse models.** "DFT-verified TS success" is defined when the DFT Hessian has one negative eigenvalue. "DFT-verified converged and TS success" additionally requires the DFT force RMS to be below 2×10^{-3} Ha/Bohr.

Hessian	Model	ZPE MAE (Std) [eV]	ΔZPE MAE (Std) [eV]
AD	AlphaNet	0.0565 (0.0283)	0.0625 (0.0770)
AD	LeftNet	1.2957 (0.3331)	0.1996 (0.2552)
AD	LeftNet-DF	0.0094 (0.0039)	0.0269 (0.0264)
AD	EquiformerV2	0.0600 (0.0719)	0.0539 (0.0706)
Predicted	HIP-EquiformerV2	**0.0004 (0.0003)**	**0.0016 (0.0018)**

Table 2: **Zero-point energy.** Mean average error and standard deviation of the signed error of the ZPE of the predicted/AD Hessian compared to DFT. We report both the ZPE at the reactant, as well as the relative ΔZPE between reactant and product.

Hessian	Model	Accuracy ↑
AD	AlphaNet	71%
AD	LeftNet-DF	78%
AD	LeftNet	82%
AD	EquiformerV2	75%
Predicted	HIP-EquiformerV2	**92%**

Table 3: **Classifying extrema using Hessian frequency analysis.** We report the accuracy by model in characterizing stationary points via the correct number of negative Hessian eigenvalues. Percentages were computed over 1000 samples, roughly half of which were true transition states and 37% were minima. Validated against DFT.

Frequency analysis for extrema classification We further investigate the ability of the Hessian methods to differentiate different extrema on the potential energy surface. We perform frequency analysis on 1000 geometries from the HORM-T1x validation set. Roughly half (44%) are 1st order transition states, 37% are minima, and the rest are higher order transition states according to their DFT Hessians. As shown in Table 2, the HIP-predicted Hessians achieve the best accuracy compared to all models using AD. In particular, HIP-EquiformerV2 has a significant advantage of 10 percentage points in accuracy over any other model, and +17% over the same model with AD (EquiformerV2).

Limitations Direct Hessian regression requires ground-truth data from DFT, which can be particularly expensive to obtain for larger systems. Therefore, the experiments in this work focus on small organic molecules. Future work should explore applying Hessian prediction to other chemical systems, such as materials, larger complexes, or even proteins. Furthermore, to isolate the Hessian from the force and energy predictions, we trained only the Hessian regression head using a fixed backbone. For optimal results, energy, forces, and Hessians should be trained end-to-end.

5 Conclusion

In this work, we have presented Hessian Interatomic Potentials (HIP), a novel method for directly predicting molecular Hessians using SE(3)-equivariant neural networks. Our approach eliminates the need for traditional computationally expensive methods, such as finite differences or automatic differentiation, offering a significant improvement in computational time, memory usage, and accuracy. Through extensive validation across a range of critical molecular tasks, we have shown that our predicted Hessians are highly effective for practical applications in computational chemistry. By making Hessians more accessible, we believe our method enables new possibilities for high-throughput screening, material discovery, and drug design. For future work, any base model using spherical harmonics and message passing can be made HIP.

Acknowledgments

We are thankful for funding by the Pioneer Center for Accelerating P2X Materials Discovery (CAPeX), DNRF grant number P3. A.A.-G. thanks Anders G. Frøseth for his generous support. A.B. and L.T. acknowledge the AIST support to the Matter lab for the project titled "SIP project - Quantum Computing". A.B. is thankful for support through the Google NSERC IRC award. L.T. thanks the NSERC for the support through the Discovery Grant. This research was undertaken thanks in part to funding provided to the University of Toronto's Acceleration Consortium from the Canada First Research Excellence Fund CFREF-2022-00042. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

[1] Chen Qu and Joel M. Bowman. An ab initio potential energy surface for the formic acid dimer: Zero-point energy, selected anharmonic fundamental energies, and ground-state tunneling splitting calculated in relaxed 1–4-mode subspaces. *Physical Chemistry Chemical Physics*, 18(36):24835–24840, September 2016. 1

[2] Jing Huang, Yanzi Zhou, and Daiqian Xie. Predicted infrared spectra in the HF stretching band of the H2–HF complex. *The Journal of Chemical Physics*, 149(9), September 2018. 1

[3] Poul Jørgensen and Jack Simons. Ab initio analytical molecular gradients and Hessians. *The Journal of Chemical Physics*, 79(1):334–357, July 1983. 1

[4] Nicholas C. Handy and Henry F. Schaefer. On the evaluation of analytic energy derivatives for correlated wave functions. *The Journal of Chemical Physics*, 81(11):5031–5033, December 1984. 1

[5] Andrew Komornicki and George Fitzgerald. Molecular gradients and hessians implemented in density functional theory. *The Journal of Chemical Physics*, 98(2):1398–1421, January 1993. 1

[6] Trygve Helgaker, Poul Jørgensen, and Jeppe Olsen. Multiconfigurational Self-Consistent Field Theory. In *Molecular Electronic-Structure Theory*, chapter 12, pages 598–647. John Wiley & Sons, Ltd, 2000. 1

[7] W. Kohn. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140(4A):A1133–A1138, 1965. 1

[8] Markus Bursch, Jan-Michael Mewes, Andreas Hansen, and Stefan Grimme. Best-Practice DFT Protocols for Basic Molecular Computational Chemistry. *Angewandte Chemie*, 134(42):e202205735, October 2022. 1

[9] 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, March 2018. 1, 2.1

[10] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds, May 2018. 1, 2.1

[11] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. GemNet: Universal Directional Graph Neural Networks for Molecules. In *Advances in Neural Information Processing Systems*, volume 34, pages 6790–6802. Curran Associates, Inc., 2021. 1, 2.1, 2.2

[12] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9377–9388. PMLR, July 2021. 1, 2.1

[13] Ilyes Batatia, David P. Kovacs, Gregor Simm, Christoph Ortner, and Gabor Csanyi. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. *Advances in Neural Information Processing Systems*, 35:11423–11436, December 2022. 1, 2.1

[14] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, May 2022. 1, 2.1

[15] Eric C.-Y. Yuan, Anup Kumar, Xingyi Guan, Eric D. Hermes, Andrew S. Rosen, Judit Zádor, Teresa Head-Gordon, and Samuel M. Blau. Analytical ab initio hessian from a deep learning potential for transition state optimization. *Nature Communications*, 15(1):8865, October 2024. 1, 2.2

[16] Nils Gönnheimer, Karsten Reuter, and Johannes T. Margraf. Beyond Numerical Hessians: Higher-Order Derivatives for Machine Learning Interatomic Potentials via Automatic Differentiation. *Journal of Chemical Theory and Computation*, April 2025. 1, 2.2

[17] Austin Rodriguez, Justin S. Smith, and Jose L. Mendoza-Cortes. Does Hessian Data Improve the Performance of Machine Learning Potentials? *Journal of Chemical Theory and Computation*, July 2025. 1, 2.2

[18] Nicholas J. Williams, Lara Kabalan, Ljiljana Stojanovic, Viktor Zólyomi, and Edward O. Pyzer-Knapp. Hessian QM9: A quantum chemistry database of molecular Hessians in implicit solvents. *Scientific Data*, 12(1):9, January 2025. 1, 2.2

[19] Taoyong Cui, Yunhong Han, Haojun Jia, Chenru Duan, and Qiyuan Zhao. HORM: A Large Scale Molecular Hessian Database for Optimizing Reactive Machine Learning Interatomic Potentials, May 2025. 1, 2.2, 4, 1, 4, A.5, A.7.2

[20] Aidan P Thompson, Laura P Swiler, Christian R Trott, Stephen M Foiles, and Garritt J Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015. 2.1

[21] Alexander V Shapeev. Moment tensor potentials: A class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016. 2.1

[22] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010. 2.1

[23] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007. 2.1

[24] Yusong Wang, Shaoning Li, Xinheng He, Mingyu Li, Zun Wang, Nanning Zheng, Bin Shao, Tie-Yan Liu, and Tong Wang. Visnet: an equivariant geometry-enhanced graph neural network with vector-scalar interactive message passing for molecules. *arXiv preprint arXiv:2210.16518*, 2022. 2.1

[25] Zun Wang, Guoqing Liu, Yichi Zhou, Tong Wang, and Bin Shao. Efficiently incorporating quintuple interactions into geometric deep learning force fields. *Advances in Neural Information Processing Systems*, 36, 2024. 2.1

[26] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019. 2.1

[27] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020. 2.1

[28] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *International Conference on Learning Representations*, 2022. 2.1

[29] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs, February 2023. 2.1

[30] Yi-Lun Liao, Brandon M Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. In *The Twelfth International Conference on Learning Representations*, 2024. 2.1, 2.1, 2.2, 3

[31] Saro Passaro and C. Lawrence Zitnick. Reducing SO(3) convolutions to SO(2) for efficient equivariant GNNs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 27420–27438. JMLR.org, July 2023. 2.1

[32] Xingyu Yang, Haitao Ma, Qing Lu, and Wensheng Bian. Efficient Method for Numerical Calculations of Molecular Vibrational Frequencies by Exploiting Sparseness of Hessian Matrix. *The Journal of Physical Chemistry A*, March 2024. 2.2

[33] Qiyuan Zhao, Yunhong Han, Duo Zhang, Jiaxu Wang, Peichen Zhong, Taoyong Cui, Bangchen Yin, Yirui Cao, Haojun Jia, and Chenru Duan. Harnessing Machine Learning to Enhance Transition State Search with Interatomic Potentials and Generative Models, May 2025. 2.2, 4, A.7.2, A.7.4

[34] Filippo Bigi, Marcel Langer, and Michele Ceriotti. The dark side of the forces: Assessing non-conservative force models for atomistic machine learning, July 2025. 2.2

[35] Alexandre Duval, Simon V. Mathis, Chaitanya K. Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen, Pietro Liò, Yoshua Bengio, and Michael Bronstein. A Hitchhiker's Guide to Geometric GNNs for 3D Atomic Systems, March 2024. 3

[36] Jörg Kussmann, Arne Luenser, Matthias Beer, and Christian Ochsenfeld. A reduced-scaling density matrix-based method for the computation of the vibrational hessian matrix at the self-consistent field level. *The Journal of Chemical Physics*, 142(9), 2015. 3.2

[37] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022. 3.2

[38] Oliver Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus-Robert Müller. Se (3)-equivariant prediction of molecular wavefunctions and electronic densities. *Advances in Neural Information Processing Systems*, 34:14434–14447, 2021. 3.2, 3.2

[39] Alan Robert Edmonds. *Angular momentum in quantum mechanics*, volume 4. Princeton university press, 1996. 3.2

[40] Emili Besalú and Josep Maria Bofill. On the automatic restricted-step rational-function-optimization method. *Theoretical Chemistry Accounts*, 100(5):265–274, December 1998. 3.3, A.7.1, A.7.2

[41] Charles J. Cerjan and William H. Miller. On finding transition states. *The Journal of Chemical Physics*, 75(6):2800–2806, September 1981. 3.3

[42] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. A.1

[43] Nikita Doikov, El Mahdi Chayti, and Martin Jaggi. Second-Order Optimization with Lazy Hessians. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8138–8161. PMLR, July 2023. A.1

[44] Jack Simons, Poul Joergensen, Hugh Taylor, and Judy Ozment. Walking on potential energy surfaces. *The Journal of Physical Chemistry*, 87(15):2745–2753, 1983. A.1, A.2.4, A.2.4

[45] Ajit Banerjee, Noah Adams, Jack Simons, and Ron Shepard. Search for stationary points on surfaces. *The Journal of Physical Chemistry*, 89(1):52–57, January 1985. A.1, A.2.4, A.2.4

[46] Eric D. Hermes, Khachik Sargsyan, Habib N. Najm, and Judit Zádor. Sella, an Open-Source Automation-Friendly Molecular Saddle Point Optimizer. *Journal of Chemical Theory and Computation*, 18(11):6974–6988, November 2022. A.1

[47] James D. Louck and Harold W. Galbraith. Eckart vectors, Eckart frames, and polyatomic molecules. *Reviews of Modern Physics*, 48(1):69–106, January 1976. A.1, A.7.4

[48] Peter Deglmann, Filipp Furche, and Reinhart Ahlrichs. An efficient implementation of second analytical derivatives for density functional methods. *Chemical Physics Letters*, 362(5):511–518, August 2002. A.2.1

[49] Erik Bitzek, Pekka Koskinen, Franz Gähler, Michael Moseler, and Peter Gumbsch. Structural Relaxation Made Simple. *Physical Review Letters*, 97(17):170201, 2006. A.2.4, A.7.1

[50] Graeme Henkelman and Hannes Jónsson. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *The Journal of Chemical Physics*, 111(15):7010–7022, 1999. A.2.5

[51] Graeme Henkelman, Blas P. Uberuaga, and Hannes Jónsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *The Journal of Chemical Physics*, 113(22):9901–9904, 2000. A.2.5

[52] Baron Peters, Andreas Heyden, Alexis T. Bell, and Arup Chakraborty. A growing string method for determining transition states: Comparison to the nudged elastic band and string methods. *The Journal of Chemical Physics*, 120(17):7877–7886, 2004. A.2.5, A.2.5

[53] Paul M. Zimmerman. Single-ended transition state finding with the growing string method. *Journal of Computational Chemistry*, 36(9):601–611, 2015. A.2.5, A.2.5, A.7.2

[54] N. van der Aa, H. ter Morsche, and R. Mattheij. Computation of eigenvalue and eigenvector derivatives for a general complex-valued eigensystem. *The Electronic Journal of Linear Algebra*, 16:300–314, January 2007. A.6

[55] Johannes Steinmetzer, Stephan Kupfer, and Stefanie Gräfe. Pysisyphus: Exploring potential energy surfaces in ground and excited states. *International Journal of Quantum Chemistry*, 121(3):e26390, 2021. A.7.1, A.7.2, A.7.4

[56] Henry C. Herbol, James Stevenson, and Paulette Clancy. Computational Implementation of Nudged Elastic Band, Rigid Rotation, and Corresponding Force Optimization. *Journal of Chemical Theory and Computation*, 13(7):3250–3259, July 2017. A.7.1

[57] Cody Aldaz, Tamas Stenczel, Joshua Kammeraad, and Paul M. Zimmerman. ZimmermanGroup/pyGSM. ZimmermanGroup, August 2025. A.7.2

[58] E. B. Wilson, J. C. Decius, P. C. Cross, and Benson R. Sundheim. Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra. *Journal of The Electrochemical Society*, 102(9):235Ca, September 1955. A.7.4

[59] Hrant P. Hratchian, Michael J. Frisch, and H. Bernhard Schlegel. Steepest descent reaction path integration using a first-order predictor–corrector method. *The Journal of Chemical Physics*, 133(22):224101, December 2010. A.7.4

A Appendix

A.1 Molecular optimization with Hessians

Rational function optimization (RFO) Geometry optimization locates minima $\arg\min_{\mathbf{R}} E(\mathbf{R})$ of the potential energy surface (PES) starting from non-equilibrium molecular geometries. It underpins most workflows from global searches to high-throughput screening. If Hessians are available, second-order methods are favored due to their fast convergence guarantees [42, 43]. RFO [44, 45] is a commonly used second-order optimizer (it makes use of Hessian information to accelerate convergence) for molecular geometries [46]. An attractive property of RFO is that it can be used both for minimizing $E(\mathbf{R})$ as well as for finding saddle points, and it is robust to indefinite Hessians [45]. As computing the Hessian is usually too expensive, a common practice is to maintain an approximate Hessian using the BFGS quasi-Newton scheme. We provide more details in A.2.4.

Zero-point energy (ZPE) Zero-point corrections account for the quantum mechanical vibrational energy that molecules have at absolute zero temperature. To calculate the ZPE, one relaxes a geometry to an extrema, computes the Hessian, and sums the frequencies $ZPE = \frac{\hbar}{2}\sum_i \sqrt{\tilde{\lambda}_i}$. Here, $\tilde{\lambda}_i$ are eigenvalues of the mass-weighted Hessian $\tilde{\mathbf{H}}_{AB} = \mathbf{H}_{AB}/m_A m_B$, and $\hbar$ is Planck's reduced constant. One is usually interested in the relative ZPE between reactant and product states: $\Delta ZPE = ZPE(\mathbf{R}_R) - ZPE(\mathbf{R}_P)$. The relative ZPE is relevant for reaction thermochemistry, as ΔZPE enters as a correction to the reaction free energy ΔG and therefore the equilibrium constant, which relates forward and backward rates via detailed balance.

Transition states Transition states (TS) are first-order saddle points on the PES. They represent the maximum barrier along the minimum energy path (MEP) between two minima. Identifying transition states is crucial for understanding reaction mechanisms and mapping reaction networks. A first-order saddle point is characterized by having exactly one negative eigenvalue of the Hessian. Over the years, various computational methods have been developed to describe MEPs and locate transition states. We review methods for transition state search in A.2.5.

Frequency analysis for extrema classification To carry out frequency analysis, one must first remove the five or six redundant degrees of freedom, corresponding to the invariance of the energy under rotation and translation. This is done by mass-weighing the Hessian and performing an Eckart projection [47], which we describe in A.7.4. Then, the projected matrix is decomposed into its eigenvalues, with all positive eigenvalues signaling a minimum, and exactly N negative eigenvalue signaling the presence of a order-N transition state.

A.2 Extended background

A.2.1 Hessians from DFT

The gold standard for computing nuclear Hessians is to derive them analytically. In Kohn-Sham DFT, this is achieved using the Coupled Perturbed Kohn–Sham (CPKS) method [48], which is employed to calculate the ground truth in the dataset and in the workflows of our experiments. If available, analytical Hessians provide the most efficient ab-initio way to obtain Hessians, but they still scale as $O(N^5)$. A major downside of the CPKS approach to Hessians is its highly complex implementation and need for exchange-correlation-functional-specific derivations, which makes it not always available. An alternative is to use a finite-difference scheme, requiring only energies or forces. Finite differences also scale $O(N^5)$ for hybrid DFT, although with more numerical noise and a much larger computational prefactor. We provide more details in A.2.3.

A.2.2 Analytical Hessians via CPKS

In Kohn-Sham DFT, Hessians are obtained with the Coupled Perturbation Kohn-Sham (CPKS) method. This is much more complex and computationally demanding than energy or force calculations. The difficulty arises from the need to calculate the response term of the electron density, which involves understanding how the self-consistent field (SCF) solution to the Kohn-Sham equations changes with a change in the nuclear coordinates. To see why this is necessary, write the DFT energy

in terms of the density matrix $\mathbf{P}(\mathbf{R}) = (\mathbf{C})^T(\mathbf{R})\mathbf{S}(\mathbf{R})\mathbf{C}(\mathbf{R})$ with molecular orbital coefficients $\mathbf{C}$ and overlap matrix $\mathbf{S}$:

$$E[\mathbf{P}] = \text{Tr}[h\mathbf{P}] + \frac{1}{2}\text{Tr}[\mathbf{J}[\mathbf{P}]] + \rho(\mathbf{P}) \tag{11}$$

To get the force, so the nuclear gradient, we first have to define the Lagrangian to enforce normalization of the wavefunction:

$$\mathcal{L}(C, \epsilon, \mathbf{R}) = E[\mathbf{R}] - \text{Tr}[\epsilon(\mathbf{C}^T\mathbf{S}\mathbf{C} - \mathbf{I})] \tag{12}$$

We then get the nuclear gradient by differentiating $\mathcal{L}$. The derivative w.r.t. nuclear component x is:

$$\frac{d\mathcal{L}}{dx} = \frac{\partial\mathcal{L}}{\partial x} + \underbrace{\frac{\partial\mathcal{L}}{\partial\mathbf{C}}}_{=0}\frac{\partial\mathbf{C}}{\partial x} - \text{Tr}\left[\epsilon\mathbf{C}^T\frac{\partial\mathbf{S}}{\partial x}\mathbf{C}\right] + 2\text{Tr}\left[(\mathbf{S}\mathbf{C}\epsilon)^T\frac{d\mathbf{C}}{dx}\right] \tag{13}$$

$$= \frac{\partial E}{\partial x} - 2\text{Tr}\left[(\mathbf{S}\mathbf{C}\epsilon)^T\frac{dC}{dx}\right] - \text{Tr}\left[\epsilon\mathbf{C}^T\frac{\partial\mathbf{S}}{\partial x}\mathbf{C}\right] + 2\text{Tr}\left[(\mathbf{S}\mathbf{C}\epsilon)^T\frac{d\mathbf{C}}{dx}\right] \tag{14}$$

$$= \frac{\partial E}{\partial x} - \underbrace{\text{Tr}\left[\epsilon\mathbf{C}^T\frac{\partial\mathbf{S}}{\partial x}\mathbf{C}\right]}_{\text{"Pulay terms"}} \tag{15}$$

Due to the stationarity $\frac{\partial\mathcal{L}}{\partial C} = 0$ we did not need to calculate the density response $\frac{\partial C}{\partial x}$, which is why calculating forces is cheap, on the same order as the cost of energies. However, once we want Hessians, we need to differentiate equation 12 twice:

$$\frac{d^2\mathcal{L}}{dxdy} = \frac{\partial^2\mathcal{L}}{\partial x\partial y} + \frac{\partial^2\mathcal{L}}{\partial\mathbf{C}\partial x}\frac{d\mathbf{C}}{dy} + \frac{\partial^2\mathcal{L}}{\partial\epsilon\partial x}\frac{d\epsilon}{dy} \tag{16}$$

$$= \frac{\partial^2\mathcal{L}}{\partial y\,\partial x} - \begin{bmatrix}\frac{\partial^2\mathcal{L}}{\partial\mathbf{C}\,\partial x} & \frac{\partial^2\mathcal{L}}{\partial\varepsilon\,\partial x}\end{bmatrix}\begin{bmatrix}\frac{\partial^2\mathcal{L}}{\partial\mathbf{C}^2} & \frac{\partial^2\mathcal{L}}{\partial\mathbf{C}\,\partial\varepsilon} \\ \frac{\partial^2\mathcal{L}}{\partial\varepsilon\,\partial\mathbf{C}} & \frac{\partial^2\mathcal{L}}{\partial\varepsilon^2}\end{bmatrix}^{-1}\begin{bmatrix}\frac{\partial^2\mathcal{L}}{\partial\mathbf{C}\,\partial y} \\ \frac{\partial^2\mathcal{L}}{\partial\varepsilon\,\partial y}\end{bmatrix}. \tag{17}$$

Solving the linear system of equations scales formally as $O(N^5)$ in computation and $O(N^4)$ in memory. Similar to energy calculations, density fitting, sparsification (integral screening), and other optimizations can reduce the cost for CPKS. Still, the scaling remains worse in both computation and memory compared to energy calculations.

A.2.3 Hessians via finite differences

A major downside of the CPKS approach to Hessians is its highly complex implementation and need for exchange-functional-specific derivations. Therefore, many new methods do not support Hessians, and most codes instead resort to finite difference calculations in these cases. Using central differences of the forces along the $3N$ Cartesian directions, the Hessian elements are constructed from

$$\mathbf{H}_{ij} \approx \frac{\mathbf{F}_i(\mathbf{R} + h\mathbf{e}_j) - \mathbf{F}_i(\mathbf{R} - h\mathbf{e}_j)}{2h} \tag{18}$$

This requires $2 \times 3N$ displaced *gradient* calculations. Each gradient calculation scales similarly to the energy calculation with $O(N^4)$, leading to a $O(N^5)$, the same as naive CPKS, although typically with more numerical noise and a much larger prefactor because we have to repeatedly converge an SCF, as opposed to solving a large linear system in analytical Hessians.

A.2.4 Geometry optimization

Geometry optimization locates minima $\arg\min_{\mathbf{R}} E(\mathbf{R})$ of the potential energy surface starting from non-equilibrium molecular geometries. It underpins most workflows from global searches to high-throughput screening. The efficiency of geometry optimization depends on four key factors: (i) the choice of coordinate system, (ii) the quality of the Hessian approximation, (iii) the update strategy, and (iv) the step size control. Cartesian coordinates are straightforward but strongly coupled, which limits step size and slows convergence for flexible systems. In contrast, internal coordinates are better aligned with chemically meaningful motions, separate stiff and soft modes, and allow constraints to be imposed naturally. These properties often lead to substantially fewer iterations. Some of the most widely used optimizers are the first-order FIRE [49] optimizer and the second-order rational function optimization (RFO) optimizer.

RFO Rational function optimization (RFO) [44, 45] is a commonly used second-order optimization technique in molecular geometry optimization. RFO starts with a [2/2]-Padé-expansion of the energy:

$$E(\mathbf{R_t} + \Delta\mathbf{x}) - E(\mathbf{R_t}) \approx \frac{\mathbf{g}^\top \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top \mathbf{H} \Delta\mathbf{x}}{1 + |\Delta\mathbf{x}|^2} \tag{19}$$

The extrema $\Delta x'$ of this surrogate are given by the solution of the generalized eigenvalue problem:

$$\begin{bmatrix} \mathbf{H} & \mathbf{g} \\ \mathbf{g}^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}' \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}' \\ 1 \end{bmatrix}. \tag{20}$$

An attractive property of RFO is that it can be used both for minimizing $E(\mathbf{R})$ as well as for finding saddle points (important for transition state search, see next section): If we pick the first eigenvector of equation 20, we get an update pointing to the minimum; if we select the second eigenvector, we get a transition state update. In contrast to Newton-Raphson optimization, RFO is also robust to indefinite Hessians. For detailed derivation, see [44, 45]. In this paper, for minimization, we are using restricted step RFO (RS-RFO), which adds a trust region to prevent unphysically large steps. For the transition point search, we are using restricted step partitioned RFO (RS-P-RFO), a slight modification that treats the subspace with negative and positive eigenvalues of the Hessian separately.

BFGS To apply RFO in practice, we need a Hessian at each step. As computing full Hessians is usually too expensive, a common practice is to maintain an approximate Hessian using the BFGS quasi-Newton scheme. BFGS updates approximate Hessians $\mathbf{B}_k$ of the true Hessian $\mathbf{H}(\mathbf{R}_k)$ using the update equation

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^\top \mathbf{B}_k}{\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}, \qquad \mathbf{y}_k^\top \mathbf{s}_k > 0, \tag{21}$$

where the curvature condition $\mathbf{y}_k^\top \mathbf{s}_k > 0$ (typically ensured by a Wolfe line search) guarantees that $\mathbf{B}_{k+1}$ remains positive-definite, which is desirable for minimization. In the RFO framework, $\mathbf{B}_k$ simply replaces the exact Hessian $\mathbf{H}$ in the augmented eigenvalue problem in equation 20.

A.2.5 Transition State Search

Computational methods for transition state search can broadly be categorized as *single-ended* and *double-ended*. In double-ended methods, we know the product and reactant states (the two minima on the energy surface) and try to find an interpolation on the MEP that goes through the TS. In contrast, in single-ended methods, we only know a starting point and try to climb up on the energy surface to find a nearby transition state. Often, double-ended methods are used to give good initial guesses, which are then refined by single-ended methods. In this study, we are using the Growing string method to find the initial states, which we then refine with RS-P-RFO (see above).
Different double-ended methods are distinguished by how they grow the interpolations between product and reactant states. The most common approaches are the nudge elastic band method [50, 51] and the growing string method (GSM) [52, 53].

Nudged elastic band method The nudged elastic band method aims to find the minimum energy path between given initial and final configurations. A set of images connecting these states is linked by classical spring forces, forming an elastic band. Each image experiences a total force comprising the spring force along the local tangent and the true force perpendicular to it. The images are simultaneously optimized to trace the MEP. To converge to the transition state, a climbing image is introduced, for which the force along the band is inverted, leaving only the perpendicular component of the true force. This drives the climbing image up the potential energy surface along the band while descending perpendicularly, reaching the transition state along the MEP.

Growing string method The growing string method (GSM) locates transition states by incrementally constructing a discrete path of structures on the potential energy surface. Starting from the reactant and product endpoints, two path fragments are iteratively grown together. At each iteration, a new node on the string is added along the local tangent direction and relaxed orthogonally to the path according to the force

$$\mathbf{F}_\perp = \mathbf{F} - (\hat{\mathbf{t}}^\top \mathbf{F})\hat{\mathbf{t}}, \tag{22}$$

where $\mathbf{F} = -\nabla E$ is the force and $\hat{\mathbf{t}}$ is the local tangent unit vector. Convergence is typically accelerated by approximate Hessians, constructed in delocalized internal coordinates and updated by quasi-Newton schemes, which enable eigenvector-guided optimization while avoiding full second-derivative evaluations [53]. Once the two fragments merge, the full string is reparameterized to maintain uniform node distance, and the highest energy node serves as a transition state estimate, which can be further refined using Hessian-based eigenvector following [52].

A.3 Hessian properties

The Hessian is a symmetric matrix $\mathbf{H} = \mathbf{H}^\top$ where each sub-block transforms under rotation as a cartesian tensor

$$\mathbf{H}_{I,J} \xrightarrow{\mathbf{Q}} \mathbf{Q}\mathbf{H}_{I,J}\mathbf{Q}^\top \tag{23}$$

or equivalently

$$\mathbf{H} \xrightarrow{\mathbf{Q}} (\mathbf{I}_N \otimes \mathbf{Q})\mathbf{H}(\mathbf{I}_N \otimes \mathbf{Q})^\top \tag{24}$$

The symmetry follows directly from Schwartz's theorem, which states that for every scalar function $E(\mathbf{R})$ that has continuous partial second derivatives in the neighborhood of a point $\mathbf{R_0}$, the partial second derivatives commute

$$\left(\frac{\partial^2 E}{\partial \mathbf{R}_i \partial \mathbf{R}_j}\right)_{\mathbf{R}=\mathbf{R_0}} = \left(\frac{\partial^2 E}{\partial \mathbf{R}_j \partial \mathbf{R}_i}\right)_{\mathbf{R}=\mathbf{R_0}} \tag{25}$$

The transformation law under rotation follows straightforwardly from the chain rule:
First, write the rotated coordinates as

$$\mathbf{R}' = (\mathbf{I}_N \otimes \mathbf{Q})\mathbf{R} \tag{26}$$

$$\frac{\partial \mathbf{R}'}{\partial \mathbf{R}} = (\mathbf{I}_N \otimes \mathbf{Q}) \tag{27}$$

Then, define the energy in the rotated frame

$$E'(\mathbf{R}') = E((\mathbf{I}_N \otimes \mathbf{Q})^\top \mathbf{R}') = E(\mathbf{R}) \tag{28}$$

Then the first order derivative is

$$\nabla_{\mathbf{R}'} E'(\mathbf{R}') = (\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}} E'(\mathbf{R}') \tag{29}$$

$$= (\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}} E((\mathbf{I}_N \otimes \mathbf{Q})^{-1}(\mathbf{I}_N \otimes \mathbf{Q})\mathbf{R}') \tag{30}$$

$$= (\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}} E(\mathbf{R}) \tag{31}$$

showing the equivariance of the forces. The second-order derivative is

$$\mathbf{H}' = \nabla^2_{\mathbf{R}'} E'(\mathbf{R}') = \nabla_{\mathbf{R}'}((\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}} E(\mathbf{R})) \tag{32}$$

$$= (\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}'}\nabla_{\mathbf{R}} E(\mathbf{R}) \tag{33}$$

$$= (\mathbf{I}_N \otimes \mathbf{Q})\nabla_{\mathbf{R}}\nabla_{\mathbf{R}} E(\mathbf{R})(\mathbf{I}_N \otimes \mathbf{Q})^\top \tag{34}$$

$$= (\mathbf{I}_N \otimes \mathbf{Q})\mathbf{H}(\mathbf{I}_N \otimes \mathbf{Q})^\top \tag{35}$$

which shows the equivariance of the Hessian matrix.

A.4 Batched AD Hessian performance

We observe in Figure 3 a fast degradation of the AD Hessian speed with batch size. To explain this, we have to understand exactly how AD treats Hessians. From the point of view of the AD engine, there is no difference between a batch of two molecules of system size N_A and N_B, and a larger system of size $N_A + N_B$. In both cases, the MLIP function takes in an array of dimension $3N_A + 3N_B$, and returns either a single scalar $E_A + E_B$ or even worse, an array $[E_A, E_B]$, so mathematically the energy function looks like $E(\mathbf{R}) : \mathbb{R}^{3N_A+3N_B} \to \mathbb{R}$ or $E(\mathbf{R}) : \mathbb{R}^{3N_A+3N_B} \to \mathbb{R}^2$. Consequently, a Hessian of this function is a block diagonal matrix of dimension $3(N_A + N_B) \times 3(N_A + N_B)$, or even $2 \times 3(N_A + N_B) \times 3(N_A + N_B)$. Therefore, the memory costs grow quadratically with the batch size. Since we have to implement the Hessian calculation with sequential Hessian-Vector products in order not to run out of memory, we can not even make use of parallelization in the batched processing of Hessians.

A.5 Training Details

We train our model HIP-EquiformerV2 for roughly six days on only a single H100 GPU. The hyperparameters are listed in 4. We inherit the model settings and large parts of the optimizer settings from the HORM codebase [19]. We expect much better results to be possible by training end-to-end, for longer, and tuning the hyperparameters.

Table 4: Hyperparameters used

Model	Type: EquiformerV2
	Layers: 4
	Sphere channels: 128
	Attention hidden channels: 64
	Attention heads: 4
	Attention alpha channels: 64
	Attention value channels: 64
	FFN hidden channels: 128
	Activation: SiLU
	Distance basis: 512 (Gaussian)
	Radius cutoff: 12 Angstrom
	Cutoff Hessian: 12 Angstrom
	Hessian layers: 3
	Spherical harmonics: Lmax=4, Mmax=2
	Grid resolution: 18, Sphere samples: 128
	Dropout: α=0, drop path=0
Loss	MAE weight: 1
	Eigen subspace k: 8
	Eigen loss weight α: 1.0
Optimizer	AdamW
	Betas: (0.9, 0.999)
	AMSGrad: True
	Weight decay: 0
	Batch size: 128 (train), 256 (val)
	Gradient clipping: 0.1 (norm)
Learning Rate	Learning rate: 0.0005
	Scheduler: StepLR
	Step size: 10
	Gamma: 0.85
Trainer	Epochs: 300 with limited batches (approx. 35 full epochs)
	Limit train batches per epoch: 1600

A.6 Loss function

We are trying to design a loss function that emphasizes the lowest lying eigenvalues/eigenvectors. To design such a loss function, one could naively compute a loss directly as $L = \sum_i^k |\mathbf{v}_i^{\text{predict}} - \mathbf{v}_1^{\text{true}}|$. This requires computing the eigenvalues and eigenvectors of the true and predicted Hessian via an eigenspectrum decomposition. Unfortunately, backpropagation through such an eigenspectrum decomposition is numerically unstable [54]. First, gradients computed using the eigenvectors tensor will only be finite when $\mathbf{H}$ has distinct eigenvalues. Furthermore, if the distance between any two eigenvalues is close to zero, the gradient will be highly sensitive, as it depends on the eigenvalues λ_i through the computation of $\min_{i \neq j} \frac{1}{\lambda_i - \lambda_j}$. Instead, we propose the following subspace loss.

In the following section, we compare the eigenspace loss that we introduced in equation 10 to the standard choice of mean square error (MSE) and mean absolute error (MAE).
In particular, we emphasize the subspace $k = 8$. The reason for this is that transition state search, ZPE, and frequency analysis for extrema classification all depend on the two lowest-lying eigenvectors and eigenvalues. We choose $k = 8$ instead of $k = 2$ to account for the usual 6 redundant degrees of freedom, which we do not remove during training for numerical stability.

Loss	Hessian ↓ eV/Å^2	Eigenvalues ↓ eV/Å^2	CosSim $\boldsymbol{v}_1$ ↑ unitless	λ_1 ↓ eV/Å^2	λ_2 ↓ eV/Å^2	λ_1^E ↓ eV/Å^2	λ_2^E ↓ eV/Å^2	CosSim $\boldsymbol{v}_1^E$ ↑ unitless	CosSim $\boldsymbol{v}_2^E$ ↑ unitless
MSE	0.033	0.072	0.728	0.168	0.088	0.037	0.013	0.959	0.928
MAE	**0.026**	**0.057**	0.825	**0.127**	0.052	**0.031**	0.012	0.976	0.952
MAE+Sub	0.030	0.063	**0.870**	0.130	**0.030**	0.031	**0.010**	**0.980**	**0.957**

Table 5: Accuracy of HIP-EquiformerV2 using different loss functions. Superscript E denotes after removing redundant degrees of freedom using Eckart-projection on the mass-weighted matrix.

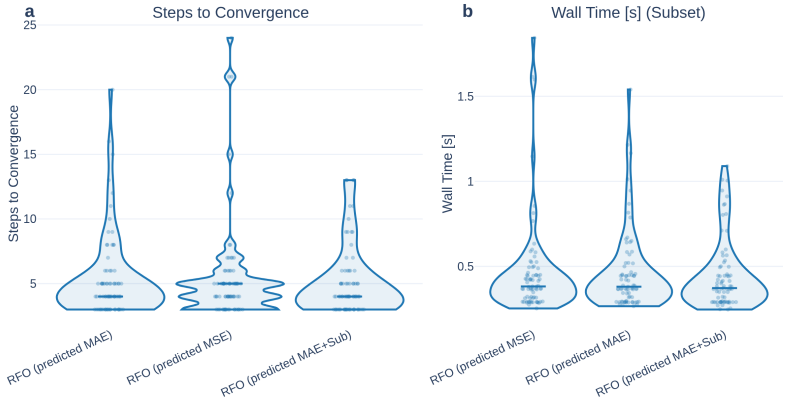

Figure 6: Geometry relaxation with HIP-EquiformerV2 trained using different loss functions

We combine the subspace loss with MAE, so the total loss becomes

$$\mathcal{L}_{\text{MAE+Sub}} = \mathcal{L}_{\text{MAE}} + \mathcal{L}_{\text{sub}}^{(k=8)} \tag{36}$$

$$= \sum_{i,j} |\mathbf{H}_{i,j} - \mathbf{H}_{i,j}^{\text{pred}}| + \sum_{i,j} \left| \mathbf{V}_{[:,:k]}^{\top} \mathbf{H}^{\text{pred}} \mathbf{V}_{[:,:8]} - \mathbf{\Lambda}_{[:,:8]} \right|_{i,j} \tag{37}$$

For simplicity, we weight both the MAE and subspace loss equally. Future work could benefit from carefully tuning the relative weights between the loss terms.

On downstream tasks, subspace loss improves over MAE and MSE in (i) transition state search, as shown in Table 7, and (ii) transition state identification frequency analysis, as shown in Table 8. We observe equal performance between MAE and the subspace loss for (i) geometry relaxation Table 6 and (ii) computing the zero-point energy Table 6.

Table 5 shows that the subspace loss improves the first eigenvector $\boldsymbol{v}_1$ cosine similarity, and the second eigenvalue λ_2 MAE. This is consistent with the better performance of the subspace loss on transition state-related tasks, since the transition state is characterized via the two smallest eigenvalues λ_1, λ_2, and is found by following the first eigenvector $\boldsymbol{v}_1$.

MSE generally underperforms compared to MAE and MAE with subspace loss. MSE has the theoretical advantage that it is rotation invariant, while the MAE is not. In practice, however, we find that the MSE leads to a high variance (spikes) in the training loss and gradient norm across batches. We speculate that MAE outperforms MSE due to training stability.
Hessian prediction with any loss function tested significantly outperforms prior AD Hessians.

Hessian	Model	ZPE MAE (Std) [eV]	ΔZPE MAE (Std) [eV]
MSE	HIP-EquiformerV2	0.0008 (0.0012)	0.0019 (0.0025)
MAE	HIP-EquiformerV2	0.0004 (0.0004)	**0.0013 (0.0019)**
MAE+Sub	HIP-EquiformerV2	**0.0004 (0.0003)**	0.0016 (0.0018)

Table 6: Zero-point energy by HIP-EquiformerV2 for different loss functions

Loss	Model	TS Success	RFO Converged	Both
MSE	HIP-EquiformerV2	718	669	659
MAE	HIP-EquiformerV2	740	693	683
MAE+Sub	HIP-EquiformerV2	**750**	**705**	**698**

Table 7: Transition state search with HIP-EquiformerV2 trained with different loss functions

Name	TPR ↑	FPR ↓	FNR ↓	TNR ↑	Precision ↑	Accuracy ↑	Accuracy All ↑
MSE	88%	11%	12%	89%	86%	89%	86%
MAE	93%	7%	7%	93%	91%	93%	91%
MAE+Sub	**93%**	**6%**	**7%**	**94%**	**92%**	**94%**	**92%**

Table 8: Identifying transition states (left) and any extrema (rightmost column) via frequency analysis using HIP-EquiformerV2 for different loss functions.

A.7 Experimental Details

A.7.1 Relaxations

We compare our RFO-based optimization [40] using predicted Hessians (RFO predicted), and our RFO with BFGS updates with initial predicted Hessian (RFO-BFGS predicted init) against first-order methods, quasi-second-order BFGS variants, and exact second-order approaches obtained via automatic differentiation or finite differences.
Our baselines include steepest descent with backtracking line search (SteepestDescent) and the first-order method FIRE [49]. For quasi-second-order methods, we consider RFO with BFGS updates initialized from either the identity (RFO-BFGS unit init), AD Hessian (RFO-BFGS AD init), or finite difference Hessian (RFO-BFGS FiniteDifference init). For full second-order methods, we include RFO with either automatic differentiation or finite difference Hessians. We run relaxations on 80 reactant geometries from the Transition1x validation set (which is different from HORM-T1x), that we noise with 0.5 Å RMS. We use the RS-RFO and redundant coordinate implementation in pysisyphus [55, 56]. We set the convergence criteria to "Gaussian default" (see A.7.4), with a budget of 150 steps.

A.7.2 Transtion State Search with ReactBench

We use the recently introduced ReactBench benchmark [33]. The workflow consists of generating an initial guess using the growing string method [57, 53], followed by local search using the restricted-step partitioned rational-function-optimization method (RS-P-RFO) [40]. Finally, convergence to the correct transition state is confirmed by frequency analysis, and following the intrinsic reaction coordinate (IRC) [55].
Following previous work [33, 19], we report success metrics of each step:

1. GSM successfully converged below a force RMS of $5e^{-5}$ Hartree/Bohr within 100 iterations ("GSM Success")
2. After RS-P-RFO, the frequency analysis determines geometry as a true transition state ("TS Success")
3. IRC converged to the same criteria and yields the original initial reactant and product ("IRC Verified").

We then treat the samples successfully passing (a)-(c) as transition state proposals, and verify for a random subset of 100, if the DFT Hessians have one negative eigenvalue and the DFT force RMS is below $2e^{-3}$ Ha/Bohr. We additionally report if RS-P-RFO converged to "Gaussian default" (A.7.4) within 50 steps.

A.7.3 Zero-Point energy

Starting from 80 reactant and product geometries from the Transition1x validation set, we relax the geometries using DFT and RS-RFO (BFGS, identity initialization), up to "Gaussian tight" convergence (see A.7.4). We use the same level of theory for the relaxation as for the training of the models. Then we compute the ZPE from the Eckart-projected, mass-weighted Hessian as predicted by the different models and compare to DFT. We report all energies in eV per molecule.

Setting	Max Force	RMS Force	Max Step	RMS Step	Used for
Gaussian loose	1.7e-3	1.0e-2	6.7e-3		
Gaussian default	4.5e-4	3.0e-4	1.8e-3	1.2e-3	Relaxation, TS search 4
Gaussian tight	1.5e-5	1.0e-5	6.0e-5	4.0e-5	ZPE 4
Gaussian very tight	1.0e-6		6.0e-6	4.0e-6	

Table 9: Convergence metrics used in our experiments

A.7.4 Convergence Criteria

Throughout our experiments, we adopt the following convergence criteria, set forth by the Gaussian software and widely used in different codebases. All criteria need to be met for a geometry to be considered converged.

Mass-weighting and Eckart Projection

We briefly describe the process of removing the redundant degrees of freedom of the Hessian [58, 47].

We start from a system of N atoms with Cartesian coordinates $\mathbf{q} = (x_1, y_1, z_1, \ldots, x_N, y_N, z_N)^\top \in \mathbb{R}^{3N}$, atomic positions $\mathbf{R}_i = (x_i, y_i, z_i)^\top$, and masses m_i.

Centering at the center of mass The first step is to use COM-centered coordinates $\mathbf{r}_i = \mathbf{R}_i - \mathbf{R}_{\text{com}}$, which are computed from the total mass $M = \sum_{i=1}^{N} m_i$, and the center of mass (COM)

$$\mathbf{R}_{\text{com}} = \frac{1}{M} \sum_{i=1}^{N} m_i \, \mathbf{R}_i. \tag{38}$$

Mass-weighing Given a Cartesian Hessian $\mathbf{H} \in \mathbb{R}^{3N \times 3N}$, its mass-weighted form is

$$\widetilde{\mathbf{H}} = \mathbf{M}^{-1/2} \, \mathbf{H} \, \mathbf{M}^{-1/2}. \tag{39}$$

where the diagonal mass matrix is

$$\mathbf{M} = \operatorname{diag}\big(\underbrace{m_1, m_1, m_1}_{\text{atom } 1}, \ldots, \underbrace{m_N, m_N, m_N}_{\text{atom } N} \big) \in \mathbb{R}^{3N \times 3N}. \tag{40}$$

Inertia tensor and principal axes To build the Eckart vectors, we require the eigendecomposition of the inertia tensor. The inertia tensor about the COM is

$$\mathbf{I} = \sum_{i=1}^{N} m_i \Big[\big(\mathbf{r}_i \cdot \mathbf{r}_i \big) \, \mathbf{1}_3 - \mathbf{r}_i \, \mathbf{r}_i^\top \Big] = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix}, \tag{41}$$

with components

$$\begin{aligned} I_{xx} &= \sum_i m_i (y_i^2 + z_i^2), \quad I_{yy} = \sum_i m_i (x_i^2 + z_i^2), \quad I_{zz} = \sum_i m_i (x_i^2 + y_i^2), \\ I_{xy} &= -\sum_i m_i x_i y_i, \quad I_{xz} = -\sum_i m_i x_i z_i, \quad I_{yz} = -\sum_i m_i y_i z_i. \end{aligned} \tag{42}$$

The eigendecomposition of the inertia tensor is then $\mathbf{I}\,\mathbf{E} = \mathbf{E}\,\mathbf{\Lambda}$, where the columns $\{\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_2, \hat{\mathbf{i}}_3\}$ of $\mathbf{E}$ are orthonormal principal axes.

Translational and rotational subspace (Eckart vectors) We can now define the three mass-weighted translational unit vectors $\{\mathbf{t}_\alpha\}_{\alpha=x,y,z} \in \mathbb{R}^{3N}$ (using unit vectors e.g., $\hat{\mathbf{e}}_x = (1,0,0)^\top$)

$$\mathbf{t}_\alpha^{(i)} = \sqrt{m_i} \, \hat{\mathbf{e}}_\alpha, \quad \alpha \in \{x, y, z\}, \quad i = 1, \ldots, N, \tag{43}$$

as well as the three mass-weighted rotational vectors $\{\mathbf{r}_k\}_{k=1,2,3} \in \mathbb{R}^{3N}$ using the COM-centered coordinates and the principal axes from the inertia tensor:

$$\mathbf{r}_k^{(i)} = \sqrt{m_i} \, \big(\hat{\mathbf{i}}_k \times \mathbf{r}_i \big), \quad k = 1, 2, 3. \tag{44}$$

For linear molecules, one rotational vector has (near-)zero norm and is discarded; in practice, we drop any $\mathbf{r}_k$ whose norm is below a small threshold. We collect the translation and rotation vectors as rows of a matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}_x^\top \\ \mathbf{t}_y^\top \\ \mathbf{t}_z^\top \\ \mathbf{r}_1^\top \\ \mathbf{r}_2^\top \\ \mathbf{r}_3^\top \end{pmatrix} \in \mathbb{R}^{d \times 3N}, \tag{45}$$

with $d = 6$ for non-linear molecules and $d = 5$ for linear molecules. We then orthonormalize these rows (e.g., QR factorization of $\mathbf{T}^\top$) to obtain an orthonormal set $\{\mathbf{u}_a\}_{a=1}^{d}$ spanning the rigid-body subspace in mass-weighted coordinates.

Eckart projector To build the projector, two equivalent forms are convenient in practice.
1) Explicit projector

$$\mathbf{P} = \mathbf{I}_{3N} - \sum_{a=1}^{d} \mathbf{u}_a \, \mathbf{u}_a^\top. \tag{46}$$

2) Nullspace basis via the SVD $\mathbf{T}^\top = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top$, which we use as default due to improved numerics. The columns of $\mathbf{U}$ associated with zero singular values span the vibrational subspace. Stack them row-wise to form $\mathbf{P}_{\text{basis}} \in \mathbb{R}^{(3N-d)\times 3N}$.
Finally, we apply the Eckart projection

$$\widetilde{\mathbf{H}}_{\text{Eckart}} = \mathbf{P}_{\text{basis}} \, \widetilde{\mathbf{H}} \, \mathbf{P}_{\text{basis}}^\top. \tag{47}$$

followed by re-enforcing the projected Hessian to be symmetric:

$$\widetilde{\mathbf{H}}_{\text{Eckart}} \leftarrow \tfrac{1}{2}\big(\widetilde{\mathbf{H}}_{\text{Eckart}} + \widetilde{\mathbf{H}}_{\text{Eckart}}^\top\big). \tag{48}$$

Intrinsic Reaction Coordinate with Euler Predictor-Corrector

The intrinsic reaction coordinate (IRC) traces the steepest-descent path on the potential energy surface (PES) from a transition state (TS) toward reactant and product minima. We use the IRC provided in the ReactBench benchmark [33], which itself uses the pysisyphus implementation [55]. We integrate in mass-weighted Cartesian coordinates to obtain a physically meaningful path, solving:

$$\frac{d\mathbf{x}}{ds} \;=\; -\,\frac{\mathbf{g}(\mathbf{x})}{\|\mathbf{g}(\mathbf{x})\|}\,,$$

where $\mathbf{x}$ are mass-weighted coordinates and $\mathbf{g}$ is the mass-weighted gradient. The pysisyphus implementation follows an Euler predictor-corrector (EulerPC) scheme [59]. Starting from a TS, we diagonalize the mass-weighted Hessian, identify the imaginary mode (transition vector), and displace slightly along it to initialize the path. We perform small Euler integrations along $-\mathbf{g}/\|\mathbf{g}\|$, updating the gradient via a local Taylor expansion using the current Hessian. The Hessian is updated with BFGS updates from $\Delta\mathbf{x}, \Delta\mathbf{g}$. We refine the predicted point using distance-weighted interpolation (DWI) of previously visited points (coordinates, energies, gradients, Hessians). The corrector integrates the IRC equation on this surrogate PES using a modified Bulirsch-Stoer procedure with Richardson extrapolation.