DEQUIFY YOUR FORCE FIELD: MORE EFFICIENT SIMULATIONS USING DEEP EQUILIBRIUM MODELS

Andreas Burger*, Luca Thiede*, Alán Aspuru-Guzik & Nandita Vijaykumar
University of Toronto, Canada
Vector Institute, Toronto, Canada
{andreas.burger,luca.thiede}@mail.utoronto.ca

ABSTRACT

Machine learning force fields show great promise in enabling more accurate molecular dynamics simulations compared to manually derived ones. Much of the progress in recent years was driven by exploiting prior knowledge about physical systems, in particular symmetries under rotation, translation and reflections. In this paper, we argue that there is another piece of important prior information that thus far hasn't been explored: Simulating a molecular system is necessarily continuous, and successive states are therefore extremely similar. Our contribution is to show that we can exploit this information by recasting an state-of-the-art equivariant base model as a deep equilibrium model. This allows us to recycling intermediate neural network features from previous time steps, enabling us to improve both accuracy and speed by $10\% - 20\%$ on the MD17, MD22, and OC20 200k datasets, compared to the non-DEQ base model. The training is also much more memory efficient, allowing us to train more expressive models on larger systems.

1 INTRODUCTION

With increasingly more available compute, molecular dynamics (MD) simulations emerged as an integral tool for studying the behaviour of molecules to develop a mechanistic understanding of a large class of processes in drug discovery and molecular biology Lin & MacKerell (2019); Hollingsworth & Dror (2018); Sinha et al. (2022); Durrant & McCammon (2011). The backbone of an MD simulation is a force field, which predicts the forces acting on each of the atoms in a molecule, given the current atom positions. These forces are then used to integrate the equations of motion numerically by multiplying the forces with a small time step dt to obtain velocities, which in turn is used to update the atom's positions. Traditionally, force fields were designed by hand to capture known physical effects such as covalent bonds, electrostatics, and van der Waals forces Weiner & Kollman (1981); Pearlman et al. (1995). These hand-crafted force fields are compact and fast but lack the expressivity to capture more complex quantum mechanical many-body interactions. Alternatively, force fields can be calculated from highly accurate but costly quantum mechanical calculations, so-called ab-initio molecular dynamics (AIMD). Therefore, a new approach has gained traction over recent years: Training an expressive machine learning model on data from expensive ab-initio methods. This results in models at near-quantum chemical accuracy at only a fraction of the cost.
Some early works on machine learning force fields used local atom environment descriptors in combination with linear regression Thompson et al. (2015); Shapeev (2016), Gaussian processes Bartók et al. (2010), and feed-forward neural networks Behler & Parrinello (2007). The pioneering work SchNet Schütt et al. (2017) used a rotation invariant graph neural network to predict energies, forces and other properties. This was later improved by the use of equivariant neural networks that model angular dependencies more directly, such as Cormorant Anderson et al. (2019), DimeNet Gasteiger et al. (2020), PaiNN Schütt et al. (2021), GemNet Gasteiger et al. (2021), SphereNet Liu et al. (2022), and NequIP Batzner et al. (2022). Recent models have further improved the expressivity and scalability of equivariant models. Equiformer introduces an attention mechanism Liao & Smidt (2023), Allegro focuses on edge features with non-growing receptive fields Musaelian et al. (2023), MACE introduces an efficient mechanism to calculate many-body interactions with high-order tensor polynomials Batatia et al. (2022), eSCN improves the scaling of Clebsch-Gordan products involved in equivariant convolutions Passaro & Zitnick (2023), and ViSNet Wang et al. (2022) and QuinNet Wang

et al. (2024) derive ways to incorporate four and five body terms much more efficiently. CHGNet Deng et al. (2023) incorporates magmoms for extra physical supervision. VisNet-LSRM Li et al. (2023) and 4G-HDNNP Ko et al. (2021) focus on modeling long-range and non-local effects.

Much of the architectural designs were guided by the incorporation of prior information about the systems, in particular, invariance to permutation of atom IDs, translation, rotation, and inversion symmetries of energy and forces, as well as size extensivity and smoothness under the movement of atoms. Our main observation is that in physically meaningful simulation, there is additional prior information that is not yet incorporated in any model: The evolution of atom coordinates in time needs to be continuous. In practical simulations, this is enforced by adaptively picking integration time steps small compared to the fastest moving part of the system. This way even processes that are seemingly abrupt on a macroscopic scale, such as shock simulations, are smooth on the atomic scale. Violation of this principle would lead to nonphysical energy dissipation.

We aim to use this extra information about our simulation by adapting the Deep Equilibrium Model (DEQ) framework Bai et al. (2019) to equivariant architectures. DEQs replace the typical deep stack of layers with a lightweight shallow model and a fixed-point solver, see section 3.1. This allows us to reuse latent features across simulation time steps by warm starting the fixed point solution from previous time steps, see figure 1. This lets us build effectively deep models at the cost of shallow ones. Additionally, the formulation allows for much more memory-efficient training, permitting us to train expressive models on large systems that would otherwise not fit in memory.

Formulating energies and forces as fixed points is natural since the ab initio ground truth data used to train ML force fields are fixed points of self-consistent field (SCF) methods themself. This raises the hopes that our DEQ formulation induces an inductive bias leading to better generalization, and in section 4.1 we present some preliminary evidence for that. Tricks like restarting from previous fixed points are also used in ab-initio molecular dynamics simulations, where the density is initialized from previous time steps, or even more advanced extrapolation Kolafa (2004) and Car-Parrinello schemes Car & Parrinello (1985). Therefore, our formulation represents an interesting link between AIMD and ML MD.

We implement our method by transforming the EquiformerV2 architecture Liao & Smidt (2023); Liao et al. (2024), which has the highest accuracy on the Open Catalyst Project leaderboard [1], into a DEQ. In principle however, the methodology is compatible with other similar force field architectures. Our results show that, compared to the original EquiformerV2, DEQuiformer achieves: (1) significantly improved accuracy for the OC20 200k dataset, (2) 10-20% faster inference and equally or higher accuracy on the MD17/MD22 datasets and in MD relaxations, (3) all at reduced training memory cost and (4) with up to 5x fewer model parameters.

We summarize our contributions as follows:

1. We identify the temporal continuity of molecular simulations as additional prior information that has not yet been used in any ML-based architecture.
2. We design the first DEQ-based equivariant neural network and apply it to ML force fields, which lets us exploit this temporal continuity by reusing fixed points across time steps
3. We demonstrate that our model can improve upon speed, accuracy, training memory, and parameter efficiency compared to non-DEQ counterparts on common benchmarks as well as real MD simulations

2 BACKGROUND

State-of-the-art ML force fields like EquiformerV2 belong to the class of equivariant graph neural networks (GNN) Batatia et al. (2022); Musaelian et al. (2023); Liao et al. (2024); Batzner et al. (2022). The central shared feature is the stacking of equivariant message passing layers, typically between five Batzner et al. (2022) and twenty Liao et al. (2024). 3D rotational and translational equivariance is achieved by building on irreducible representations and spherical harmonics, improving data efficiency Batzner et al. (2022). Such symmetry exploiting networks have emerged as the SOTA for molecular data Passaro & Zitnick (2023); Musaelian et al. (2023); Batatia et al. (2022); Liao et al. (2024); Thomas et al. (2018). A Graph Neural Network (GNN) takes in a graph $\mathcal{G}$ and maps it to a target space in a permutation equivariant way. If the graph is embedded in 3d space as molecules are,

[1] https://opencatalystproject.org/leaderboard.html

we use $O(3)-$Equivariant graph neural networks, which are equivariant to translations, rotations and optionally inversions. In these networks, node features h_t of node t, are concatenations of irreducible representations (irreps) $h_t^l \in \mathbb{R}^{2l+1}$, organized by their degree l (we omit an additional channel dimension for simplicity). Irreps transform under rotation R as

$$h^l(R \cdot (r_1, ..., r_N)) = D_l(R) \cdot h^l(r_1, ..., r_N) \tag{1}$$

where $r_1, ..., r_N$ are the coordinates of the atoms, and $D_l(R) \in \mathbb{R}^{(2l+1)\times(2l+1)}$ is the Wigner-D matrix. Intuitively, higher-degree features rotate faster with rotation of the input features. $l = 0$ features are rotation invariant scalars, and $l = 1$ are ordinary vectors. A vector $r \in \mathbb{R}^3$ can be mapped to an l graded feature using the spherical harmonics $Y_l({}^r\!/_{||r||}) \in \mathbb{R}^{2l+1}$.
Two irreps f^{l_1} and g^{l_2} of different degrees interact using the Clebsch-Gordan tensor product Thomas et al. (2018)

$$h^{l_3}_{m_3} = \left(f^{l_1}_{m_1} \otimes^{l_3}_{l_1,l_2} g^{l_2}_{m_2}\right)_{m_3} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C^{(l_3,m_3)}_{(l_1,m_1),(l_2,m_2)} f^{l_1}_{m_1} g^{l_2}_{m_2} \tag{2}$$

where we index the elements within the $2l + 1$ dimensional tensor by m, and $C^{(l_3,m_3)}_{(l_1,m_1),(l_2,m_2)}$ are the Clebsch-Gordan coefficients. Every combination of l_1, l_2, l_3 is called a path, and every path is weighted individually by $w_{l_1,l_2,l_3}(\cdot)$. The weight itself is predicted by a neural network, conditioned on rotation invariant features like the distance $||r_{ts}||$.
An equivariant GNN builts on top of equation 2 to define an equivariant message passing scheme: Given a target node h_t and a source node h_s with a relative coordinate vector r_{ts}, an equivariant GNN sends a message from the source to the target using

$$v^{l_3}_{ts} = v^{l_3}(h_t, h_s, r_{ts}) = \sum_{l_1,l_2} w_{l_1,l_2,l_3}(||r_{ts}||) \left(f^{l_1}(h_t, h_s) \otimes^{l_3}_{l_1,l_2} Y^{l_2}({}^{r_{ts}}\!/_{||r_{ts}||})\right) \tag{3}$$

where $f(h_t, h_s)$ is a function of both target and source node features; in EquiformerV2, it is simply the concatenation operation. Instead of using equation 3 directly, EquiformerV2 relies on eSCN convolutions, which calculates basically the same expression but in a more efficient way; please refer to Passaro & Zitnick (2023) and Liao et al. (2024) for details.

2.1 EQUIFORMERV2

EquiformerV2 Liao et al. (2024) is a graph transformer, where each message passing layer is an equivariant transformer block. To initialize the node features, the embedding block $\mathcal{U}$ first encodes the input molecule, based the atom numbers $\mathbf{z}$ and positions $\mathbf{r}$.

$$\mathbf{h}_i^{(0)} = \mathcal{U}(x_i) = \mathcal{U}\left(\mathbf{z}_i, \{\mathbf{r}_{ij}\}_{j\in\mathcal{N}(i)}\right) \tag{4}$$

The L transformer layers then perform repeated attention-weighted message passing to update the node features based on nodes in the neighborhood.

$$\mathbf{h}_i^{(l+1)} = f_\theta^{(l)}\left(\mathbf{h}_i^{(l)}, \{\mathbf{h}_j^{(l)}, \mathbf{r}_{ij}\}_{j\in\mathcal{N}(i)}\right) \tag{5}$$

After several transformer blocks update the node features, they are passed to two separate output heads for the final force and energy predictions. The total energy of the molecule is just the sum of the energies of the individual nodes.

$$E = \sum_i \mathcal{D}^{\text{scalar}}\left(\mathbf{h}_i^{(L)}\right),\ \mathbf{F}_i = \mathcal{D}^{\text{vector}}\left(\mathbf{h}_i^{(L)}, \mathbf{z}_i, \mathbf{r}_{ij}\right)_{j\in\mathcal{N}(i)} \tag{6}$$

We provide more details on equivariant GNNs and EquiformerV2 in section A.1.1.

3 DEQUIFORMER

Our goal is to incorporate temporal correlation as an inductive bias, to effectively reuse computation between timesteps of the simulation. Inspired by SCF methods where the density can be initialized from previous calulations, we do so by reusing the features of previous timesteps. It is unclear

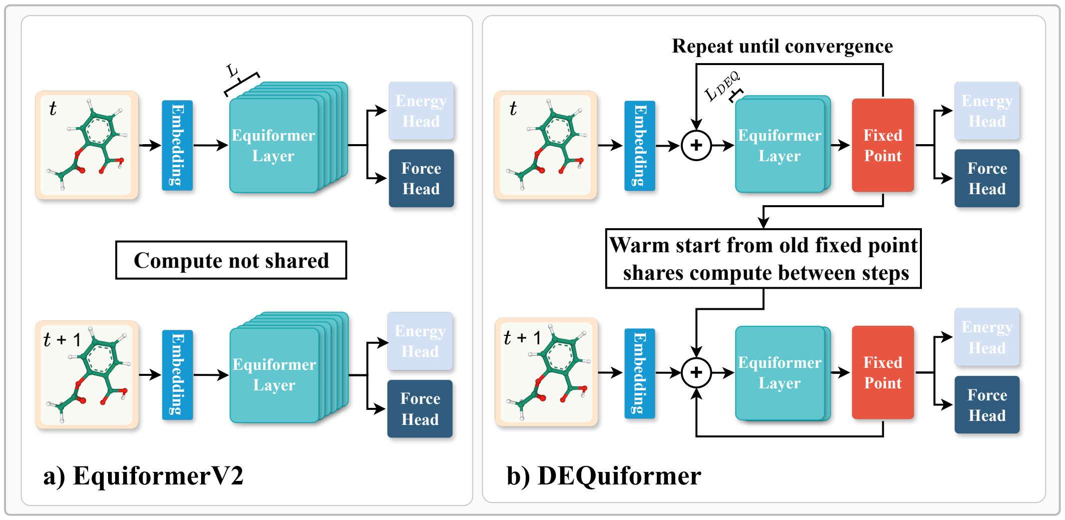


Figure 1: Comparison of the EquiformerV2 and DEQuiformer architectures. While the Equiformer model considers every input state independently, the DEQuiformer exploits the temporal continuity between input states to share compute. This works because neighbouring time steps in an MD simulation are highly similar by design. Therefore, we drastically reduce the required compute by reusing the fixed-point from the previous step.

however how one could use latents from previous MD steps with an architecture like EquiformerV2. The issue is that the model would have to learn to take into account previous latents during training. Fundamentally, there is no way for EquiformerV2 to adapt to the (dis-)similarity from the previous timestep. This is because explicit models define a fixed computational graph to map the input to an output. Instead, we use an implicit architecture like DEQ, that iterates until self-consistency is reached. In this sense, DEQs (and other implicit models) have the ability to adaptively increase compute depending on how difficult the problem is. Concretely, we ”DEQuify” EquiformerV2 by replacing the L Equiformer layers with a fixed-point solver over $L_{DEQ} \ll L$ Equiformer layers, as shown figure 1. In the following we discuss the techniques to make this work in practice.

3.1 Deep Equilibrium Networks

Implicit layer Deep Equilibrium models Bai et al. (2018; 2019) drastically reduce the model size by replacing the deep stack of layers with just one or two layers and an iterative fixed-point solver. In Bai et al. (2019), the authors showed that the network converges to a fixed-point in the limit of infinite depth $L \to \infty$. Therefore, we say that these models have ”continuous layers” or ”infinite depth” To formalize this, consider a function f_θ, usually a small neural network. Given some input x, repeated passes through f_θ updates the features $\mathbf{h}^s$

$$\mathbf{h}^{i+1} = f_\theta(\mathbf{h}^s, x) \tag{7}$$

until the features converge to a fixed-point $\mathbf{h}^* = f_\theta(\mathbf{h}^*, x)$. The ”fixed-point” or ”equilibrium point” is then considered the output of the fixed-point layer. This replaces the intermediate features $\mathbf{h}^{(l)}$ after l layers with a fixed-point estimate of the features $\mathbf{h}^s$. EquiformerV2 predicts the forces and energy via separate output heads, by acting on the node features after L layers $\mathbf{h}^{(L)}$. The node features are instead replaced by the fixed-point estimate of the node features $\mathbf{h}^*$ from the root solver, which we pass as input to the output heads.

Input injection via embedding block The neural network layer f_θ has to take in the input x, in addition to the current features $\mathbf{h}^s$, at every solver iteration, which is called the input injection. Equiformer initializes the node features via an embedding block $\tilde{x} = \mathcal{U}(x)$. Using the embedding to initialize the initial fixed-point estimate $\mathbf{h}^0$ however would stop gradients to flow to the encoder, since the gradient calculation is independent of the solver trajectory. Instead, we use the embedding block's output as the input injection, by adding the embedding to the fixed-point estimate $\mathbf{h}^s$ at every solver step before passing it through the layer g_θ. The node features are instead initialized as all zeros $\mathbf{h}^0 = 0$ Bai et al. (2019). To prevent the norm of the features to grow with depth, we rescale the vector 2-norm $\|\cdot\|$ to be the same as before the addition.

$$f_\theta(\mathbf{h}^s, x) = g_\theta\left((\mathbf{h}^s + \tilde{x})\frac{\|\tilde{x}\|}{\|\mathbf{h}^s + \tilde{x}\|}\right) \tag{8}$$

This setup reduces the model size from many layers to just a few. However, naively passing $\mathbf{h}$ through the NN layer f_θ until equilibrium is slow in practice. Instead, we search for the fixed-point directly by using a root-solving algorithm, which computes more sophisticated updates of $\mathbf{h}$ to reduce the number of passes until the fixed-point is reached. We found Broyden's method Broyden (1965) to be instable during training, so we use Anderson acceleration Anderson (1965).

Fast inference via fixed-point reuse The SCF-like structure of DEQs allows us to incorporate the inductive bias that consecutive time steps in a molecular dynamics simulation are highly similar. We do so by initializing the fixed-point estimate during inference not from all zeros, but the fixed-point of the previous time step: $\mathbf{h}^0_{t+1} = \mathbf{h}^*_t$ Bai et al. (2022). With this fixed-point reuse the number of solver steps can be significantly reduced to gain a speedup at inference time.

Memory efficient gradient Backpropagating through this solver trajectory would incur a prohibitive memory cost. Fortunately, a unique feature of DEQs is that the gradient can be computed by the Implicit Function Theorem (IFT) Bai et al. (2019):

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \mathbf{h}^*}\left(1 - \frac{\partial f_\theta}{\partial \mathbf{h}^*}\right)^{-1}\frac{\partial f_\theta\left(\mathbf{h}^*, x\right)}{\partial \theta} \tag{9}$$

Using IFT, the forward passes is performed without tracking gradients, i.e. without storing the layer activations. Thus, the memory cost during training, becomes independent of the DEQ's "depth", which starkly contrasts explicit models, where the memory complexity grows linearly with each layer. With the Implicit Function Theorem (IFT) the gradient is computed by solving a second fixed-point system, for which we again use a root solver Bai et al. (2019).

$$\mathbf{g}^* = \mathbf{g}^*\frac{\partial f}{\partial \mathbf{h}^*} + \frac{dL}{d\mathbf{h}^*} \tag{10}$$

Computing the gradient via IFT reduces the memory requirements during training, at the cost of extra time. Recent DEQ works Cao et al. (2024); Bai et al. (2022); Geng et al. (2023) circumvent solving equation 10 by the so-called 1-step (phantom) gradient approximation Fung et al. (2022).

$$\frac{\partial L}{\partial \theta} \approx \frac{\partial L}{\partial \mathbf{h}^*}\frac{\partial f_\theta\left(\mathbf{h}^*, x\right)}{\partial \theta} \tag{11}$$

We found however, that while the 1-step gradient leads to 2-3x faster training compared to solving the fixed-point system in equation 10, it resulted in a significant reduction in accuracy EquiformerV2. We thus remove the 1-step gradient and use the IFT instead.

Recurrent dropout Dropout is a widely used regularization that tends to hurt DEQ performance. This is because dropout samples a new mask for each pass through the implicit layer, which hinders finding a fixed-point Bai et al. (2019). EquiformerV2 uses two types of dropout, alpha dropout (acting on nodes) and path dropout, also known as stochastic depth (acting on edges). For DEQuiformer we instead use recurrent dropout, which applies the same mask at each step of the fixed-point solver, but a different mask for each sample Bai et al. (2019); Gal & Ghahramani (2016). We found that recurrent path dropout and no alpha dropout work best in DEQuiformer.

Training stability with fixed-point correction loss Without further regularization, DEQs may become unstable over the course of training, noticeable by increasing number of root solver steps Bai et al. (2022; 2021); Geng & Kolter (2023). An effective remedy is the *sparse fixed-point correction* regularization loss Bai et al. (2022). Given a fixed-point solver trajectory $\mathbf{h}^0, \cdots, \mathbf{h}^s, \cdots, \mathbf{h}^*$ we pick some fixed-point estimates $\mathbf{h}^s, s \in \mathcal{I}$ and add their gradient as if they were the final fixed-point estimate. We follow Bai et al. (2022) and uniformly pick three $\mathbf{h}^s$ along the solver trajectory.

Accuracy-compute tradeoff in the root solver During training, we require low fixed-point errors to ensure that gradients can be calculated with the IFT. However, we can trade off performance and time during inference by relaxing the error threshold for the root solver Bai et al. (2022). With the right threshold, this significantly speeds up inference while only marginally affecting performance. For simplicity, we adhere to the settings of Bai et al. (2022). During training we stop after the absolute fixed-point error falls below a relative threshold $|f_\theta(\mathbf{h}^s) - \mathbf{h}^s|/\|\mathbf{h}^s\| < \epsilon_{train} = 10^{-4}$. During inference, we compute the first fixed-point at the same tight tolerance $\epsilon_{test} = \epsilon_{train}$, but then relax the threshold for the following time steps to $\epsilon_{test}^{FPreuse} = 10^{-1}$. Relaxing the tolerance further reduces the number of forward steps and thus inference time, without sacrificing accuracy.

4 Experiments

OC20 The Open Catalyst Project (OC20) is one of the largest quantum chemistry datasets, containing 1.3 million molecular relaxations from 260 million DFT calculations. It is focused on catalyst simulation, where each system consisting of a surface and an adsorbate that is relaxed onto the surface. We train on the structure to energy and forces (S2EF) 200k split to evaluate the accuracy of our approach. We then run relaxation simulations, starting from configurations in the dataset to validate that our fixed-point reuse scheme speeds up the simulation and does not induce additional errors. The energies and forces are in units of eV and eV/Å. Time is measured as the forward pass on an AMD MI100.

MD17 MD17 contains one trajectory of a molecular dynamics simulations each for eight small molecules with 9 to 21 atoms. For each molecule, there are between 100,000 to 1,000,000 data points. Different to OC20, the data points in MD17 are from consecutive MD timesteps, which we will use to evaluate fixed-point reuse. Following Equiformer, a random subset of 950 data points is used for training, 50 for validation, and testing on all remaining samples. To aggregate the results over all molecules in MD17, we use minmax normalization per molecule and average over all molecules. We describe the procedure in section A.2. The energies and forces are in units of kcal/mol and kcal/mol/Å.

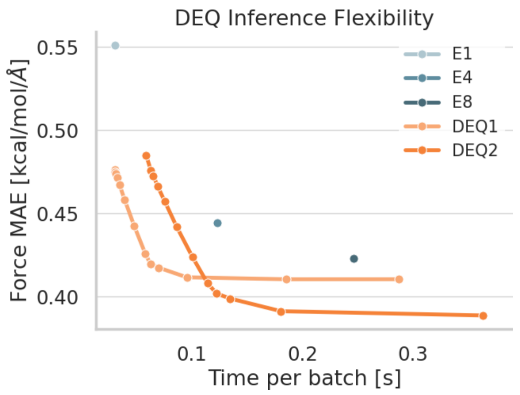
MD22 The MD22 dataset extends MD17 by seven larger molecules with 42 to 370 atoms Chmiela et al. (2023). The size of the training set variies for each molecule. The number of samples is defined such that their sGDML model reaches a root mean squared test error for the forces of 1 kcal/mol/Å. Chmiela et al. (2023).

Training Following previous work, we train separate models for each molecule on MD17/22 and one big model on OC20. We use the default model settings in the EquiformerV2 repository for OC20 as well as MD17/22. The training hyperparameters were similarly taken from EquiformerV2 for OC20, but slightly changed for MD17/22 to account for the smaller dataset sizes. We did not optimize the hyperparameters for DEQuiformer. Please refer to section A.2 for details.

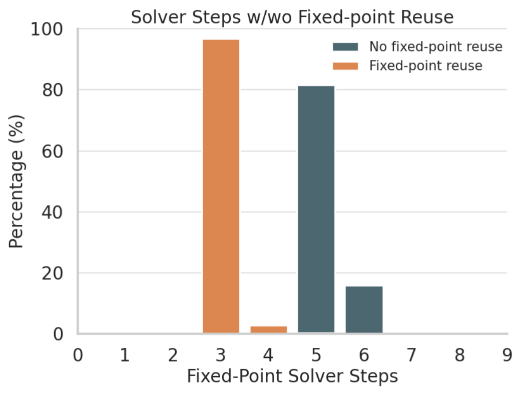
4.1 Results

Immproved accuracy on OC20 We first demonstrate how DEQs improve peak accuracy by having expressivity comparable to very deep models. To examine how the models scale with the number of layers, in figure 3a, we plot the force error over the number of layers, using a maximum of 14 layers for EquiformerV2, the maximum our GPU memory could support, and up to two layers for our DEQuiformer. DEQuiformer reaches significantly better accuracy than EquiformerV2 while using far fewer parameters. Interestingly EquiformerV2 seems not to benefit much from an increase in depth after a certain point, such that a one- or two-layer DEQuiformer is outperforming even an 14-layer EquiformerV2. This gives some preliminary evidence that the formulation of force fields as DEQs might induce a useful inductive bias, due to the connection to SCF procedures. We also train selected models for up to three times as long (see section A.3). The results are in table 1b.

Speed-accuracy tradeoff at inference time While we get good accuracy with our models, a vanilla DEQ would be slow. To achieve a speedup we exploit the temporal continuity by reusing fixed-points across simulation time steps. We examine the impact of fixed-point reuse in figure 2b.



(a) Compute-accuracy-tradeoff at inference time. DEQuiformer is remarkably robust to its fixed-point error up to a threshold of about 10^{-1}, where the error starts to rapidly increase. As expected, higher fixed-point tolerances lead to faster inference speed.



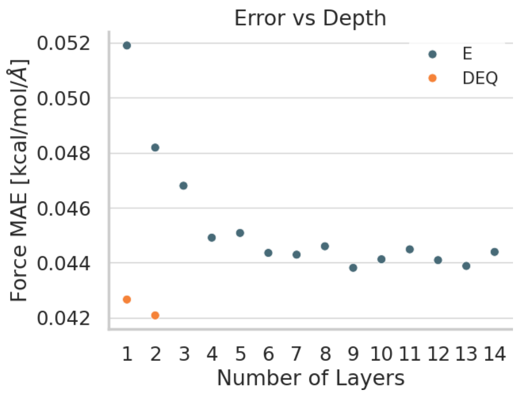
(b) Reusing the fixed-point significantly reduces the number of solver steps in DEQuiformer to enable a speedup. We plot two distributions, with and without fixed-point reuse. Percentage denotes the number of samples that required a given number of solver steps.

Figure 2: Reusing fixed-points and relaxing the solver threshold lead to better inference speed.

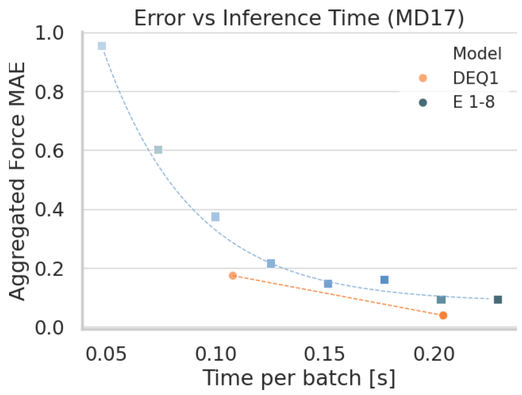
We plot the number of solver steps needed to find the fixed-point in the test set on Aspirin with and without reusing the fixed-point from the previous time step. Since MD17 contains one continuous MD trajectory per molecule, we can test on the whole trajectory and initialize the fixed-point of each sample with the fixed-point of the previous sample. While DEQuiformer takes about 5-6 steps per sample to find the fixed-point when starting from zero initialization $\mathbf{h}^0 = 0$; this gets reduced to 3 steps if we warm-start the solver, supporting our claim that we ”share compute” between successive time steps by leveraging the temporal continuity.
A unique feature of DEQs is that we can trade off accuracy for extra speed post-training by loosening the fixed-point error threshold. The looser this threshold, the faster the model, since the fixed-point solver terminates earlier. We are examining how sensitive the force error is with respect to this fixed-point error tolerance. We calculate the validation error and time per batch for different solver tolerances on a logarithmic scale, with the Aspirin molecule as an example. The results are plotted in figure 2a. As expected, looser thresholds lead to faster inference time but higher force errors. Remarkably, the model's predictions seem robust until a threshold of about 10^{-1}, after which the force error shoots up. Thus, we choose a threshold of 10^{-1} as the sweetspot to further speedup inference.

Accuracy and speed on sequential MD data We also validate the accuracy and speed of DEQuiformer on the molecular dynamics data across MD17, using both fixed-point reuse and relaxed error threshold. For each molecule, we again test the inference speed on the full dataset, reusing the fixed-point from the previous sample to initialize the solver. In figure 3b DEQuiformer improves upon the Pareto front of various EquiformerV2 layers. A full breakdown of the force test errors and inference time can be found in table 2 and table 4. DEQuiformer achieves consistently faster inference speeds at the same or better accuracy than EquiformerV2. Comparing DEQuiformer to EquiformerV2, we measure an average inference time improvement of 19 %, at 15 % better accuracy. In total DEQuiformer is the best model in 5/8 molecules. We also report the accuracy for the larger molecules in the MD22 dataset. DEQuiformer reaches state of the art accuracy in 3/7 cases and outperforms EquiformerV2 on average. The double-walled nanotube is much larger than the other systems, causing the 8-layer EquiformerV2 to run out of memory on our compute setup while our DEQuiformer easily fits in memory.

Markov property of DEQs We know from physics that that the forces should only depend on the current state, known as the Markov property. By initializing our features from a previous time step, one might fear that this property is lost. To test if reusing fixed points breaks the Markov property, we compare the predicted forces with and without fixed-point reuse over Aspirin and OC20 relaxation trajectories. At each timestep, we calculate the relative difference in the forces as described in appendix A.3. This results in an average deviation of less than 1%. Since the deviation is much



(a) **Accuracy on OC20 200k** DEQuiformer outperforms EquiformerV2 despite using much fewer parameters. Even with 14 layers (maxing out our memory), DEQuiformer still performs much better, indicating that DEQs are more data efficient for force fields.



(b) **Speed and accuracy on MD17** DEQuiformer is faster than Equiformer during inference at the same or better accuracy. Colors indicate number of layers from one to eight.

Figure 3: Results on MD17/22 and OC20 200k: DEQuiformer is faster and more accurate than EquiformerV2 while also using much fewer parameters.

OC20 Relaxation	FP reuse	$\epsilon_{test}^{FPreuse}$	Time [s]	# Solver steps
EquiformerV2 (14 layers)			12.92 ± 0.26	-
DEQ (1 layer)	✗	✗	32.98 ± 0.41	29.37 ± 7.03
DEQ (1 layer)	✓	✗	20.37 ± 0.43	18.05 ± 2.19
DEQ (1 layer)	✓	✓	$\mathbf{12.38 \pm 0.33}$	$\mathbf{11.03 \pm 2.91}$

(a) **Speed in relaxation simulation.** We ablate the impact of fixed point reuse and the lower error threshold $\epsilon_{test}^{FPreuse}$ (section 3.1) on DEQuiformer and compare it to an EquiformerV2. Both techniques make a significant impact and DEQuiformer is faster than EquiformerV2 when both are combined.

OC20 200k [eV] [eV/Å]	Force	Energy	# Weights
CGCNN	0.075	1.111	3.6M
SchNet	0.060	0.975	7.4M
MACE	0.051	0.565	6.2M
PaiNN	0.053	0.482	13M
PaiNN Direct	0.047	0.457	14M
DimeNet++	0.049	0.497	3.5M
GemNet-dT	0.041	0.443	32M
LEIGNN	0.044	0.415	17M
EquiformerV2 (8 layers)	0.038	**0.392**	3M
DEQuiformer (2 layers)	**0.035**	0.498	**1M**

(b) **Accuracy on OC20.** The force and energy errors of various models on OC20 200k. DEQuiformer is more accurate than EquiformerV2 on forces and energy. Baselines from Yang et al. (2024).

Table 1: DEQuiformer is (a) faster in relaxation simulations (b) more accurate on OC20.

smaller than the average prediction error, we conclude that fixed-point reuse approximately preserved the Markov property. This is congruent with results from ab initio molecular dynamics methods, where warm starting SCF iterations from density matrices of previous time steps do not affect the simulation, and are are therefore common practice.

Speedup in relaxations Finally, we validate our findings in a real world test case. To test the speedup of DEQuiformer in realistic simulation, we run relaxations based on configurations from OC20. Each sample includes a slab model for the surface and an adsorbate on it as an initial guess. Starting from 100 samples of the OC20 200k data set we run 100 relaxation steps each to get the lowest energy geometry. We compare a one-layer DEQuiformer to the biggest model we can afford, the 14-layer EquiformerV2. We also ablate reusing previous fixed-points and relaxing the solver threshold and summarize the results in 1a. We see that the speedup is only possible when using both techniques, reducing the number of layer evaluations significantly, from roughly 29 to 11. We find that DEQuiformer is faster than EquiformerV2 in practical scenarios while being much smaller, and more accurate on the test set.

MD17 Force MAE [kcal/mol/Å]	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic acid	Toluene	Uracil
DimeNet	0.499	0.187	0.230	0.383	0.215	0.374	0.216	0.300
PaiNN	0.371	230.000	0.230	0.319	0.083	**0.209**	0.102	**0.140**
SchNet	1.350	0.310	0.390	0.660	0.580	0.850	0.570	0.560
SphereNet	0.430	0.178	0.208	0.340	0.340	0.360	0.155	0.267
sGDML	0.680	**0.060**	0.330	0.410	0.110	0.280	0.140	0.240
EquiformerV2 (8 layers)	0.359	0.161	0.175	0.230	**0.063**	0.243	0.100	0.219
DEQuiformer (2 layers)	**0.298**	0.166	**0.162**	**0.216**	**0.063**	0.218	**0.086**	0.203

MD22 Force MAE [kcal/mol/Å]	AT-AT	AT-AT-CG-CG	Ac-Ala3-NHMe	DHA	Stachyose	Buckyball	Nanotube
Allegro	0.095	0.128	0.107	0.073	0.097	-	-
Frank	0.099	0.115	0.088	0.065	0.088	-	-
GN-OC-L	0.137	0.130	0.145	0.091	0.089	0.189	**0.222**
GN-OC-S	0.124	0.134	0.117	0.066	**0.051**	0.239	0.258
GemNetOC	0.124	0.130	0.117	0.066	0.089	-	-
Kovacs	0.088	0.106	0.089	0.053	0.063	-	-
MACE	0.099	0.115	0.088	0.065	0.088	**0.085**	0.277
PaiNN	0.238	0.370	0.230	0.136	0.233	-	-
SO3krates	0.095	0.128	0.107	0.073	0.097	-	-
Shoghi	0.216	0.332	0.244	0.242	0.435	-	-
TorchMD-NET	0.204	0.326	0.188	0.121	0.192	-	-
sGDML	0.691	0.703	0.797	0.747	0.674	-	-
EquiformerV2 (8 layers)	0.060	0.039	0.051	**0.043**	0.092	0.088	OOM
DEQuiformer (2 layers)	**0.059**	**0.038**	**0.050**	0.044	0.088	0.092	0.233

Table 2: **Accuracy on MD17 and MD22.** Force errors by molecular system in the MD17/22 dataset. Our DEQuiformer improves upon EquiformerV2's accuracy on most systems. The other baseline numbers are from Schütt et al. (2021); Gasteiger et al. (2020); Liu et al. (2022) (MD17) and Li et al. (2024); Shoghi et al. (2023) (MD22).

5 Conclusion

In this work, we explored the integration of Deep Equilibrium models and machine learning force fields to enhance the efficiency of molecular dynamics (MD) simulations. We "DEQuify" the state-of-the-art model EquiformerV2 by replacing its deep stack of layers with a more compact fixed-point layer. This approach allows us to leverage the temporal similarity between successive MD simulation states by reusing fixed-points, as well as the ability to trade off accuracy and speed. On the MD17 and MD22 datasets, our DEQuiformer model achieves substantial improvements in parameter efficiency and inference speed at similar or better accuracy compared to the original EquiformerV2. On the much larger OC20 200k dataset, DEQuiformer reaches significantly higher accuracy compared to the base model. This suggests a promising new research direction for machine learning force fields, focusing on exploiting the temporal nature of MD simulations to enhance computational efficiency. Since DEQs are in principal orthogonal to the base model, we expect that any improvements in the base architectures or DEQs in the future should complement each other. Future work could furhter expand on the idea of initializing features, akin to classical SCF methods like Pulay mixing, Anderson mixing, or density extrapolation methods.

Acknowledgments

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. Computations were performed on the Acceleration Consortium AMD Tacozoid cluster. 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

Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019. 1

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, October 1965. ISSN 0004-5411. doi: 10.1145/321296.321305. URL `https://doi.org/10.1145/321296.321305`. 3.1, A.3

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018. 3.1

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1, 3.1, 3.1, 3.1, 3.1, 3.1, A.3

Shaojie Bai, Vladlen Koltun, and Zico Kolter. Stabilizing Equilibrium Models by Jacobian Regularization. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 554–565. PMLR, July 2021. 3.1

Shaojie Bai, Zhengyang Geng, Yash Savani, and J. Zico Kolter. Deep Equilibrium Optical Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 620–630, 2022. 3.1, 3.1, 3.1, 3.1, A.1.2, A.3

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. 1

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

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. ISSN 2041-1723. doi: 10.1038/s41467-022-29939-5. 1, 2

Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007. 1

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. A.1.1

C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19(92):577–593, 1965. ISSN 0025-5718. doi: 10.2307/2003941. 3.1

Jiezhang Cao, Yue Shi, Kai Zhang, Yulun Zhang, Radu Timofte, and Luc Van Gool. Deep equilibrium diffusion restoration with parallel sampling. In *CVPR*, 2024. 3.1, A.1.2

Richard Car and Mark Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical review letters*, 55(22):2471, 1985. 1

Stefan Chmiela, Valentin Vassilev-Galindo, Oliver T. Unke, Adil Kabylda, Huziel E. Sauceda, Alexandre Tkatchenko, and Klaus-Robert Müller. Accurate global machine learning force fields for molecules with hundreds of atoms. *Science Advances*, 9(2):eadf0873, 2023. doi: 10.1126/sciadv.adf0873. URL `https://www.science.org/doi/abs/10.1126/sciadv.adf0873`. 4

Bowen Deng, Peichen Zhong, KyuJung Jun, Janosh Riebesell, Kevin Han, Christopher J. Bartel, and Gerbrand Ceder. Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 5(9):1031–1041, Sep 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00716-3. URL `https://doi.org/10.1038/s42256-023-00716-3`. 1

Jacob D Durrant and J Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC biology*, 9:1–9, 2011. 1

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6648–6656, 2022. 3.1, A.1.2

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/076a0c97d09cf1a0ec3e19c7f2529f2b-Paper.pdf`. 3.1

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020. 1, 2

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

Zhengyang Geng and J. Zico Kolter. TorchDEQ: A Library for Deep Equilibrium Models, October 2023. 3.1, A.2, A.3

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021. A.1.2

Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3.1, A.1.2

Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018. 1

Tsz Wai Ko, Jonas A. Finkler, Stefan Goedecker, and Jörg Behler. A fourth-generation high-dimensional neural network potential with accurate electrostatics including non-local charge transfer. *Nature Communications*, 12(1):398, Jan 2021. ISSN 2041-1723. doi: 10.1038/s41467-020-20427-2. URL https://doi.org/10.1038/s41467-020-20427-2. 1

Jiří Kolafa. Time-reversible always stable predictor–corrector method for molecular dynamics of polarizable molecules. *Journal of computational chemistry*, 25(3):335–342, 2004. 1

Yunyang Li, Yusong Wang, Lin Huang, Han Yang, Xinran Wei, Jia Zhang, Tong Wang, Zun Wang, Bin Shao, and Tie-Yan Liu. Long-short-range message-passing: A physics-informed framework to capture non-local interaction for scalable molecular dynamics simulation. *arXiv preprint arXiv:2304.13542*, 2023. 1

Yunyang Li, Yusong Wang, Lin Huang, Han Yang, Xinran Wei, Jia Zhang, Tong Wang, Zun Wang, Bin Shao, and Tie-Yan Liu. Long-short-range message-passing: A physics-informed framework to capture non-local interaction for scalable molecular dynamics simulation, 2024. URL https://arxiv.org/abs/2304.13542. 2

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs, February 2023. 1, A.1.1, A.1.1, A.2, A.2

Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. EquiformerV2: Improved Equivariant Transformer for Scaling to Higher-Degree Representations, March 2024. 1, 2, 2, 2.1, A.1.1, A.2, A.3

Fang-Yu Lin and Alexander D MacKerell. Force fields for small molecules. *Biomolecular simulations: Methods and protocols*, pp. 21–54, 2019. 1

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. URL https://openreview.net/forum?id=givsRXsOt9r. 1, 2

Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, February 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-36329-y. 1, 2

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*, pp. 27420–27438. JMLR.org, July 2023. 1, 2, 2

David A Pearlman, David A Case, James W Caldwell, Wilson S Ross, Thomas E Cheatham III, Steve DeBolt, David Ferguson, George Seibel, and Peter Kollman. Amber, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Computer Physics Communications*, 91(1-3):1–41, 1995. 1

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017. 1

Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pp. 9377–9388. PMLR, 2021. 1, 2

Alexander V Shapeev. Moment tensor potentials: A class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016. 1

Nima Shoghi, Adeesh Kolluru, John R Kitchin, Zachary W Ulissi, C Lawrence Zitnick, and Brandon M Wood. From molecules to materials: Pre-training large generalizable models for atomic property prediction. *arXiv preprint arXiv:2310.16802*, 2023. 2

Siddharth Sinha, Benjamin Tam, and San Ming Wang. Applications of molecular dynamics simulation in protein study. *Membranes*, 12(9):844, 2022. 1

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. *arXiv preprint arXiv:1802.08219*, 2018. 2, 2

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. 1

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. 1

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. 1

Paul K Weiner and Peter A Kollman. Amber: Assisted model building with energy refinement. a general program for modeling molecules and their interactions. *Journal of Computational Chemistry*, 2(3):287–303, 1981. 1

Ziduo Yang, Xian Wang, Yifan Li, Qiujie Lv, Calvin Yu-Chian Chen, and Lei Shen. Lightweight equivariant model for efficient machine learning interatomic potentials, 2024. URL https://arxiv.org/abs/2311.02869. 1b

A Appendix

A.1 Additional Background

A.1.1 EquiformerV2 architecture

We provide some further detail on the EquiformerV2 architecture and its three main components: the embedding, the transformer layers, and the output heads. A complete description with additional details on layer norm, multi-head-attention and non-linearities can be found in the original papers Liao & Smidt (2023); Liao et al. (2024).

Embedding An input sample consists of the positions and types of all the atoms in the molecule. The embedding block maps each atom i to a higher dimensional node embedding $\mathbf{h}^{()}i$, consisting of atom and edge-degree embeddings. The edge-degree embeddings transform a constant one vector with an message passing SO(2) layer, multiplied with edge distance embeddings, and aggregated by summing. Edge distance embeddings are the relative distances between the nodes, encoded by a learnable radial function on top of a Gaussian radial basis. This sum is rescaled by a scalar α and added to an linear embedding of the one-hot atom number $\mathbf{z}$:

$$u_t = \alpha \sum_{s \in \mathcal{N}(i)} v(1, 1, r_{is}) \tag{12}$$

$$\mathbf{h}^{()}i = \mathcal{U}(\mathcal{G})_i = \text{linear}(\text{one-hot}(\mathbf{z}_i)) + u_i \tag{13}$$

$\mathcal{N}(t)$ means the neighbourhood of atom i, defined by the set of atoms that are within a user-specified cutoff radius from the atom i.

EquiformerBlock We write *EquiformerBlock*$(\mathcal{G})$ to refer to a stack of L Equiformer layers. Each layer consists of equivariant graph attention, layer norm and feed-forward networks. The equivariant graph attention updates the node features h using equivariant messages (equation 3). However, instead of just summing up the messages directly to update a target node, Equiformer weights each message with an attention weight to get the final message which is then summed over all source nodes:

$$m_{ts} = a_{ts} \cdot v_{ts} \tag{14}$$

$$h'_t = h_t + \text{linear}\left(\sum_{s \in \mathcal{N}(t)} m_{ts} \right) \tag{15}$$

The attention weights are calculated using MLP attention Liao & Smidt (2023); Brody et al. (2021) operating only on the rotation invariant $L = 0$ features:

$$z_{ts} = k^\top \text{LeakyReLU}(f(h_t^0, h_s^0)) \tag{16}$$

$$a_{ts} = \frac{\exp(z_{ts})}{\sum_{k \in \mathcal{N}(t)} \exp(z_{tk})} \tag{17}$$

with a learnable weight vector k.

Output Heads The output heads take all the node features and process them depending on the type of target. For the energy, the $l = 0$ features of each node are transformed by an MLP and summed together for the final prediction. For the forces, an additional layer of equivariant graph attention is used, and the $l = 1$ features of each atom are directly treated as the prediction for the force.

A.1.2 Inexact gradients in DEQ

The computational bottleneck in equation 9 is to compute the inverse. Previous work has therefore explored approximating it via its Neumann series, sometimes called the phantom gradient Fung et al. (2022); Geng et al. (2021). Often, keeping only the first term (the identity) is good enough, which leads to the so-called 1-step gradient

$$\frac{\partial L}{\partial \theta} \approx \frac{\partial L}{\partial \mathbf{h}^*} \frac{\partial f_\theta(\mathbf{h}^*, x)}{\partial \theta} \tag{18}$$

The 1-step gradient can be implemented by simply passing the fixed-point through the implicit layer one additional time, this time with tracked gradients using autograd. Many recent works have used the 1-step gradient with great success Cao et al. (2024); Bai et al. (2022); Geng et al. (2023). We found however that while the 1-step gradient leads to 2-3x faster training compared to solving the fixed-point system in equation 10, it resulted in a significant reduction in accuracy, which is why we do not use the 1-step gradient in this paper.

A.2 Method

Aggregated metric over MD17/MD22 We use minmax normalization to rescale the errors of the different models on each molecule to $[0, 1]$, where the models are DEQuiformer and EquiformerV2 with various number of layers $M \in \{DEQ1, DEQ2, E1, E4, E8\}$. To get summary statistics per model, we then take the mean (Avg) over all normalized molecules.

$$\text{NormMAE}_M^{mol} = \frac{\text{MAE}_M^{mol} - min_M \left(\text{MAE}_M^{mol}\right)}{max_M \left(\text{MAE}_M^{mol}\right) - min_M \left(\text{MAE}_M^{mol}\right)} \tag{19}$$

$$\text{Avg}_M = \frac{1}{N_{mol}} \sum_{mol} \text{NormMAE}_M^{mol} \tag{20}$$

Hyperparameters for MD17/MD22 To facilitate a fair and straightforward comparison, we follow the hyperparameters set out by EquiformerV1 Liao & Smidt (2023) and EquiformerV2 Liao et al. (2024). Since EquiformerV2 did not evaluate on MD17/MD22, we refer to the EquiformerV1 Liao & Smidt (2023) codebase for training settings, which also provided the training loop for MD17/MD22 of our implementation. To be economical with our GPU resources in terms of training and inference speed, we used a smaller maximum feature degree of $l = 3$ (from previously $l = 6$), which was also used in EquiformerV1. We observed that benefits from higher l are neglectable on small datasets like MD17, as Liao et al. (2024) also noted for the similarly sized QM9 dataset. We kept all parameters of the optimizer identical to EquiformerV1.

Hyperparameters for OC20 S2EF 200k EquiformerV2 provides hyperparameters for OC20 2M, which we take as a proxy for the OC20 200k split we train on. The only changes made to the EquiformerV2 model are (1) a reduction in the number of layers down from 12 and (2) limiting the maximum spherical harmonics degree to $l = 3$, since the 200k split is ten times smaller than the 2M split, and because it significantly increases the computational cost. EquiformerV2 made minor changes to the optimizer parameters compared to V1. A full breakdown of hyperparameters is in table 3.

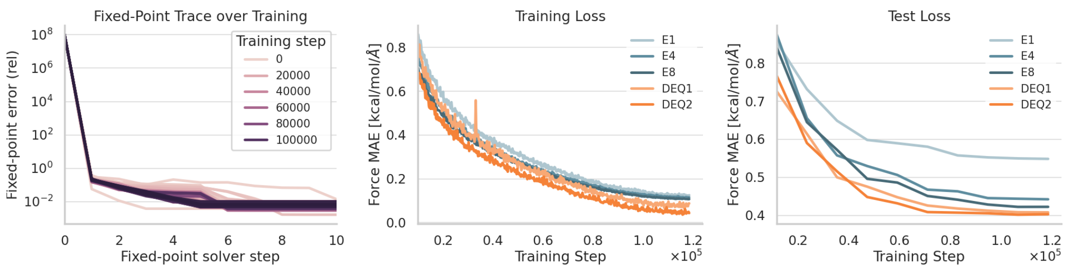
Implementation We use the model of EquiformerV2 from commit fa32143, which depends on open catalyst commit 5a7738f. The open catalyst repository (now called FairChem) has since undergone significant changes. For MD17/MD22 we modify the training loop from the code of EquiformerV1 Liao & Smidt (2023) from commit b7e7a0d. The DEQ solver is adapted from the TorchDEQ library Geng & Kolter (2023).
For MD17/MD22 each model is trained on a single AMD MI100 GPU with 32GB GPU RAM for 1000 epochs, which takes 12 to 72 hours. For OC20 200k training takes about 40 to 120 hours for six epochs.

A.3 Additional results

Training dynamics Our first question is whether or not DEQuiformer converges to a fixed-point. Since no prior work has combined DEQs with a rotation equivariant architecture, this is not at all obvious. To answer this question, we look at the relative fixed-point error on the Aspirin molecule as a function of the fixed-point solver steps at different steps in training; see figure 4a. We see that the fixed-point error decreases with the number of solver steps, as expected. The fixed-point iteration is stable over the training, even slightly improving, resulting in slightly faster convergence later in training.
Additionally, we look at the loss curves of aspirin as an example, figure 4b and figure 4c. We see that the training stability of DEQuiformer is similar to that of EquiformerV2. The same holds for

Hyperparameters	MD17/MD22	OC20 S2EF 200k
Optimizer	AdamW	
Learning rate scheduling	Cosine with linear warmup	
Warmup epochs	10	0.1
Initial learning rate	1×10^{-6}	4×10^{-5}
Maximum learning rate	5×10^{-4}	2×10^{-4}
Minimum learning rate	1×10^{-6}	2×10^{-6}
Number of epochs	1000	6
Batch size	4	
Force loss metric	L2 MAE	L2 MAE
Energy loss metric	L2 MAE	L1 MAE
Force loss weight λ_F	80	100
Energy loss weight λ_E	1	2
Weight decay	5×10^{-3}	10^{-3}
Gradient norm clipping	1000	100
Dropout rate (alpha dropout)	0.1 EquiformerV2 / 0 DEQ	
Stochastic depth (path dropout)	0.05	
Cutoff radius (Å)	5.0	12
Maximum number of neighbors	500	20
Number of radial bases	128	
Maximum degree l_{max}	3	
Maximum order M_{max}	2	
Grid resolution of point samples R	14	
Hidden dimension in feed forward networks d_{ffn}	128	
Dimension of hidden scalar features in radial functions d_{edge}	128	
Embedding dimension (spherical channels) d_{embed}	128	
$f_{ij}^{(L)}$ dimension d_{attn_hidden}	64	
Number of attention heads h	8	
$f_{ij}^{(0)}$ dimension d_{attn_alpha}	64	
Value dimension d_{attn_value}	16	
DEQ root solver	Anderson	
Maximum number of forward steps (stopping criterion)	40	
Absolute error tolerance (stopping criterion) training ϵ_{train}	10^{-4}	
Absolute error tolerance (stopping criterion) inference ϵ_{test}	10^{-1}	
Fixed-point correction loss terms	3	

Table 3: Hyperparameters for EquiformerV2 and DEQuiformer. Training hyperparameters for MD17/MD22 are taken from the EquiformerV1 codebase. Model parameters are reduced to roughly a quarter to match the smaller MD17/MD22 benchmark. For OC20 training and model settings are taken from the EquiformerV2 repository.



(a) DEQuiformer converges stably to a fixed-point over training.

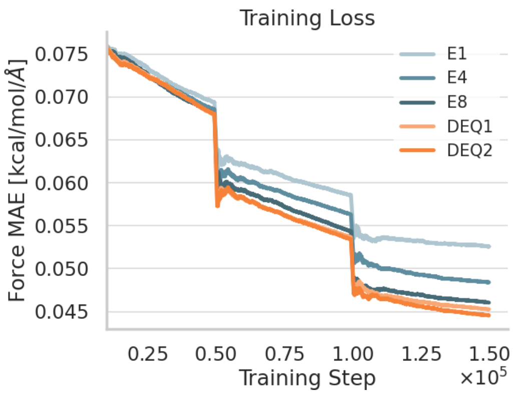
(b) DEQuiformer trains faster, achieving lower train error.

(c) Lower train error translates to lower test error in DEQuiformer.

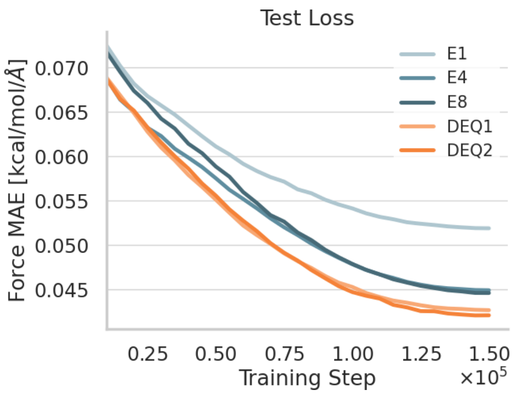
Figure 4: DEQuiformer enjoys stable training dynamics, reaching lower train and test error.

OC20 200k; see figure 5a in the supplementary material. We can also see that our DEQuiformer has both better training and better test errors, indicating that the improvements are not just due to reduced overfitting from a lower parameter count.

Training run on OC20 In figure 4 of the main text we depicted that DEQuiformer achieves lower train and test error than EquiformerV2 throughout training on Aspirin. For completion we also plot the training run for OC20 200k in figure 5. Note that the choppy behavior of the training curve is due to resets of averaging statistics after each epoch.



(a) DEQuiformer trains faster, achieving lower train error. We plot the error averaged over the current epoch. The step-like jumps are due to resetting the average at the start of a new epoch.



(b) Lower train error translates to lower test error.

Figure 5: DEQuiformer enjoys stable training dynamics, reaching lower train and test error than EquiformerV2 on OC20 200k.

MD17 Inference Time [s]	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic acid	Toluene	Uracil
EquiformerV2 (8 layers)	0.218	0.230	0.245	0.224	**0.200**	0.218	0.243	0.220
DEQuiformer (2 layers)	**0.215**	0.094	**0.218**	**0.249**	0.199	0.210	**0.201**	0.094
EquiformerV2 (4 layers)	0.131	0.146	**0.127**	0.132	**0.113**	0.125	**0.129**	0.121
DEQuiformer (1 layer)	**0.074**	**0.104**	0.097	**0.094**	0.073	**0.098**	0.077	**0.084**

Table 4: **Inference time on MD17.** DEQuiformer is faster at comparable accuracy. We highlight the lowest time per batch comparing EquiformerV2 (4 layers) to DEQuiformer (1 layer), and EquiformerV2 (8 layers) to DEQuiformer (2 layers), since they respectively closest in the speed-accuracy pareto front.

MD-17 timings in detail A detailed table with runtimes per molecule in MD17 is given in table 4

Fixed-point reuse approximately preserves Markovianity An important property of molecular dynamics is that the forces only depend on the current state, known as the Markovian property. To test if reusing fixed-points breaks Markov property, we compare the predicted forces $\mathbf{F}$ with and without fixed-point reuse. At each timestep we calculate the relative difference in the forces as

$$\Delta F_{rel}^{atom}(atom\ i) = \frac{|\mathbf{F}_i^{fpr} - \mathbf{F}_i|_x}{\frac{1}{2}\left(|\mathbf{F}_i^{fpr}|_x + |\mathbf{F}_i|_x\right)} \tag{21}$$

$$\Delta F_{rel}^{sample}(sample\ j) = \frac{1}{N}\sum_{i \in atoms}^{N}\left(\Delta F_{rel}^{atom}(i)\right) \tag{22}$$

$$\Delta F_{rel} = \frac{1}{M}\sum_{j \in test}^{M}\left(\Delta F_{rel}^{sample}(j)\right) \tag{23}$$

where $|\cdot|_x$ denotes the l2-norm over the three spatial components of a force vector on one atom i. We run and average over $M = 1$k consecutive samples of Aspirin from the MD17 dataset. The relative force difference ΔF_{rel} is depicted in figure 6. We see a deviation in the predicted forces between starting from zero initialization and from the previous fixed-point of, on average, 0.4%. The deviation remains constant over time. We repeat the experiment for the 100 times 100 relaxation steps reported in section 4.1, and measured a deviation of 0.8%. The deviation is much smaller than the average prediction error, so we conclude that fixed-point reuse approximately preserves the Markov property.

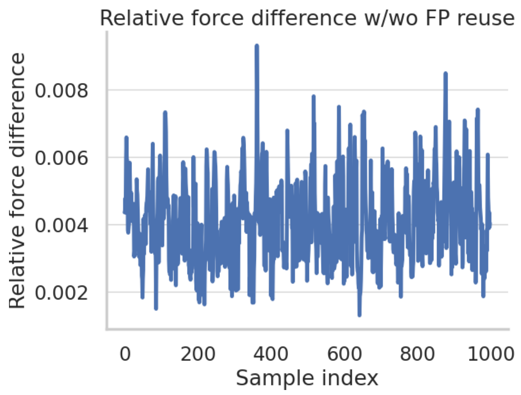
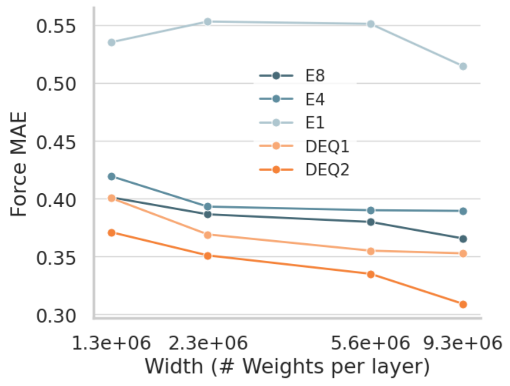
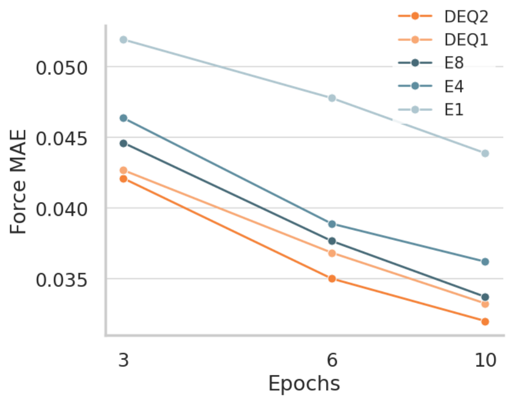


Figure 6: **Markov property.** Initializing from the previous fixed-point, compared to initializing from zero, leads to very small deviation in forces ΔF_{rel} below one percent. This means, initialization from the past fixed point has almost no effect on the accuracy of the prediction.



(a) Error scaling with model width on Aspirin (MD17), trained at 1k epochs. On the x-axis is the number of parameters per layer. This means that the 8-layer Equiformer has approximately 8 times the parameter compared to the 1-layer DEQuiformer.



(b) Error scaling with more epochs on OC20 200k. All models are getting better with more epochs, but DEQuiformer remains the leader in accuracy over Equiformer.

Figure 7: Error scaling with more epochs and model size.

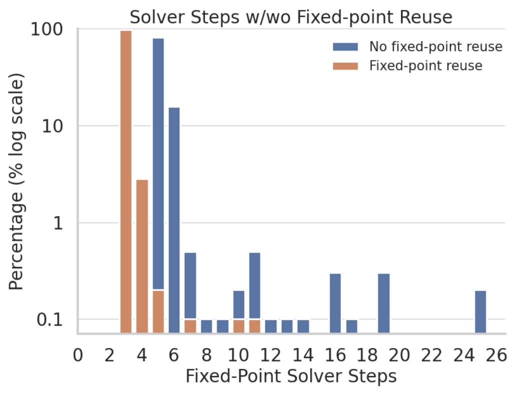
Scaling compute The paper directly compares DEQuiformer against EquiformerV2, albeit with limited compute compared to the EquiformerV2 paper Liao et al. (2024), which trained up to 135M parameters on a larger datapslit (200k vs >100M) for >1500 GPU days.
To demonstrate that our results are robust, we scale up selected runs. In figure 7a we train Aspirin for the same number of epochs as in the paper (1000) at increasing model sizes. The smallest datapoint (left) is the same model size that we used in the main text for MD17/MD22, and the largest (right) the same as previously used for OC20. Note that at the same width DEQuiformer has much fewer total parameters, e.g. DEQ1$\sim$4.8M compared to E8$\sim$21M for the right-most width. The accuracy gap between DEQuiformer to EquiformerV2 remains when scaling the model size.
We report the scaling with an increase in training epochs on OC20 200k in figure 7b. We did not scale up the model size, as EquiformerV2 would run out of memory. Instead we depict the same model size as in paper for OC20, with the left-most data point also trained on the same number of epochs. Again DEQuiformer increase in accuracy is robust when scaling up.

Pseudocode To clarify our algorithm, we provide pseudocode for DEQuiformer in algorithm 2 as well as for the original DEQ Bai et al. (2019) in algorithm 1.
The original DEQ paper (algorithm 1) is based on a transformer acting on a sequence of language tokens. $x_{1:T}$ denote the input sequence and $y_{1:T}$ the output sequence of tokens. f_θ is a (weight tied)

transformer layer.
DEQuiformer (algorithm 2) acts on an cloud of atom positions and atom types. We drop the token indices $\cdot_{1:T}$ and omit the atom indices for readability. The `BackwardDEQ` procedure remains the same. The the predicted and ground truth labels y each consist of forces and the energy instead of sequences. We made a couple of changes to the original DEQ. The original DEQ paper Bai et al. (2019) used a linear initialization of the input injection, whereas we use EquiformerV2's encoder. We also added a decoder (EquiformerV2's force and energy prediction heads). The solver is similar, but we use Anderson acceleration instead of Broyden's method, where β is the mixing parameter, c_j are coefficients determined by minimizing the residuals, and m is the number of previous iterations used in the mix Anderson (1965); Geng & Kolter (2023). We also add a normalization after each input injection. The original DEQ initialized fixed-points as zeros, whereas we took inspiration from Bai et al. (2022) and initialized with the previous fixed-point during inference. From [Bai et al. (2022) we also take the fixed-point correction loss and the relaxed solver tolerance ϵ. The main change we made to EquiformerV2 was to remove alpha dropout as it hurt performance and replace path dropout with a recurrent path dropout (not shown in the algorithm).

Algorithm 1 Deep Equilibrium Model (DEQ), Bai 2019

```
1:  procedure DEQ($\hat{x}_{1:T}, \theta, \epsilon$)
2:      Define $g_\theta(z_{1:T}; \hat{x}_{1:T}) = f_\theta(z_{1:T} + \hat{x}_{1:T}) - z_{1:T}$
3:      Initialize $z_{1:T}^{(0)} \leftarrow 0$
4:      $i \leftarrow 0$
5:      while $\|g_\theta(z_{1:T}^{(i)}; \hat{x}_{1:T})\| > \epsilon$ do                        ▷ fixed-point solver
6:          $z_{1:T}^{(i+1)} \leftarrow z_{1:T}^{(i)} - \alpha B g_\theta(z_{1:T}^{(i)}; \hat{x}_{1:T})$          ▷ Broyden's method
7:          $i \leftarrow i + 1$
8:      end while
9:      $z_{1:T}^* \leftarrow z_{1:T}^{(i)}$
10:     return $z_{1:T}^*$
11: end procedure
12:
13: procedure BackwardDEQ($z^*, y_{pred}, y_{gt}, \theta, \epsilon$)
14:     Compute $\frac{\partial \mathcal{L}}{\partial z^*}$ using the loss function $\mathcal{L}(y_{pred}, y_{gt})$
15:     Solve the linear system (IFT, second fixed-point solver):
            $$\left(J_{g_\theta}^\top\Big|_{z^*}\right) x + \left(\frac{\partial \mathcal{L}}{\partial z^*}\right)^\top = 0$$
16:     Compute the gradient:
            $$\frac{\partial \mathcal{L}}{\partial \theta} = -\left(\frac{\partial \mathcal{L}}{\partial z^*}\right)\left(J_{g_\theta}^{-1}\Big|_{z^*}\right)\frac{\partial f_\theta}{\partial \theta}$$
17:     return $\frac{\partial \mathcal{L}}{\partial \theta}$
18: end procedure
19:
20: procedure UseDEQ($x_{1:T}, y_{1:T}, \theta, \epsilon, \alpha$)
21:     while not done do
22:         $\hat{x}_{1:T} \leftarrow W^T x_{1:T}$                                          ▷ input injection
23:         $z_{1:T}^* \leftarrow$ DEQ($\hat{x}_{1:T}, \theta, \epsilon$)
24:         $y_{pred} \leftarrow z_{1:T}^*$                                                 ▷ no decoder
25:         if inference then
26:             $\frac{\partial \mathcal{L}}{\partial \theta} \leftarrow$ BackwardDEQ($z_{1:T}^*, y_{pred}, y_{1:T}, \theta, \epsilon$)
27:             Update $\theta \leftarrow$ optimizer($\theta, \frac{\partial \mathcal{L}}{\partial \theta}$)
28:         end if
29:     end while
30:     return $\theta$
31: end procedure
```



(a) Reusing the fixed-point significantly reduces the number of solver steps in DEQuiformer to enable a speedup. We plot two distributions, with and without fixed-point reuse. Percentage denotes relative number of samples in the test set that required a given number of solver steps.

Figure 8: Examining DEQuiformers fixed-point behaviour.

Algorithm 2 DEQuiformer

```
procedure DEQ($\hat{x}, \theta, \epsilon, z^*_{t-1}$)
    Define $g_\theta(z; \hat{x}) = f_\theta \left( (z + \hat{x}) \frac{||\hat{x}||}{||z+\hat{x}||} - z \right)$        ▷ added normalization
    Initialize $z^{(0)} \leftarrow 0$        ▷ if training
    if inference then
        Initialize $z^{(0)} \leftarrow z^*_{t-1}$        ▷ fixed-point reuse
    end if
    $i \leftarrow 0$
    $\{z^{(i)}\} \leftarrow \{\}$        ▷ intermediate fixed-points for correction loss
    while $\|g_\theta(z^{(i)}; \hat{x})\| > \epsilon$ do
        $z^{(i+1)} \leftarrow (1-\beta) g(z^{(i)}; \hat{x}) + \beta \sum_{j=0}^{m} c_j z^{(i-j)}$        ▷ Anderson acceleration
        if training then
            $\{z^{(i)}\}$ append $z^{(i+1)}$        ▷ if $i$ in $\mathcal{I}$, save intermediate fixed-point
        end if
        $i \leftarrow i + 1$
    end while
    $z^* \leftarrow z^{(i)}$
    return $z^*, \{z^{(i)}\}$
end procedure

procedure UseDEQ($x, (\mathbf{F}_{gt}, E_{gt}), \theta, \epsilon, \alpha$)
    $z^*_{t-1} \leftarrow 0$        ▷ if inference, save previous fixed-point
    while not done do
        $\hat{x} \leftarrow \text{Enc}(x)$        ▷ input injection via Equiformer encoder
        $z^*, \{z^{(i)}\} \leftarrow \text{DEQ}(\hat{x}, \theta, \epsilon, z^*_{t-1})$
        $z^*_{t-1} \leftarrow z^*$        ▷ save for fixed-point reuse
        $\mathbf{F} \leftarrow \text{Dec}_F(z)$        ▷ Eqiformer decoder
        $E \leftarrow \text{Dec}_E(z)$        ▷ Eqiformer decoder
        if training then
            $\frac{\partial \mathcal{L}}{\partial \theta} \leftarrow \text{BackwardDEQ}(z^*, (\mathbf{F}, E), (\mathbf{F}_{gt}, E_{gt}), \theta, \epsilon)$
            for $z^{(i)}$ in $\{z^{(i)}\}$ do        ▷ sparse fixed-point correction loss
                $\frac{\partial \mathcal{L}}{\partial \theta} += \text{BackwardDEQ}(z^{(i)}, (\mathbf{F}, E), (\mathbf{F}_{gt}, E_{gt}), \theta, \epsilon)$
            end for
            Update $\theta \leftarrow \text{optimizer}(\theta, \frac{\partial \mathcal{L}}{\partial \theta})$
        end if
    end while
    return $\theta$
end procedure
```

Distribution over solver steps figure 2b In figure 2b in the main text the distribution over solver steps for without fixed-pint reuse (blue) seemingly does not add up to 100%. We plot the same data again on a log scale on the y-axis in figure 8a. There is a long tail of up to 25 solver steps for no fixed point reuse, that individually contribute less than 1%, too small to show up in the plot of the main text.