STRIDERNET: A Graph Reinforcement Learning Approach to Optimize Atomic Structures on Rough Energy Landscapes

Anonymous Authors[1]

Abstract

Optimization of atomic structures presents a challenging problem, due to their highly rough and non-convex energy landscape, with wide applications in the fields of drug design, materials discovery, and mechanics. Here, we present a graph reinforcement learning approach, STRIDERNET, that learns a policy to displace the atoms towards low energy configurations. We evaluate the performance of STRIDERNET on three complex atomic systems, namely, binary Lennard-Jones particles, calcium silicate hydrates gel, and disordered silicon. We show that STRIDERNET outperforms all classical optimization algorithms and enables the discovery of a lower energy minimum. In addition, STRIDERNET exhibits a higher rate of reaching minima with energies, as confirmed by the average over multiple realizations. Finally, we show that STRIDERNET exhibits inductivity to unseen system sizes that are an order of magnitude different from the training system. All the codes and datasets are available at https://anonymous.4open.science/r/StriderNET-F64D.

1. Introduction and Related Work

Optimization of functions exhibiting non-convex landscapes is a ubiquitous problem in several fields, such as the design of mechanical structures (Mistakidis & Stavroulakis, 2013), robotics and motion planning (Alonso-Mora et al., 2018; Schwager et al., 2011), materials (Le & Winkler, 2016), and biological systems (Yang et al., 2019), such as proteins. Specifically, materials discovery relies on finding stable structures of atomic systems, such as new battery materials, novel drugs, or ultralight super-hard materials, through efficient optimization (Xiang et al., 1995). These materials predicted through optimization are then verified and validated through experiments and tests for industrial applications. However, even for a given material having a few hundred atoms, a large number of possible structures can be obtained by allowing various configurational arrangements of the atoms. For instance, Fig. 1 shows the structure of a 100-atom Lennard-Jones system (detailed later), where the potential energy and positions of the atoms before and after optimization are shown. Extrapolation of previous work (Tsai & Jordan, 1993) on simple atomic clusters suggests that a system containing 147 atoms can have as many as $10^{60} - 10^{259}$ minima. These possible configurations of the atomic network, represented by local minima in the energy landscape separated by high energy barriers, make the optimization problem extremely challenging (Wales et al., 2003).

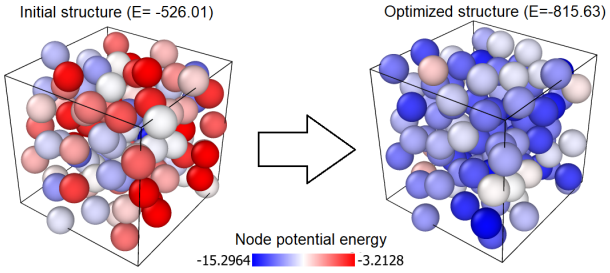

Figure 1. Optimization of 100 atoms LJ system (Colorbar shows node potential energy).

Several classical approaches have been proposed for optimization of atomic structures. These include *fast inertial relaxation engine (FIRE)* (Bitzek et al., 2006), gradient-based approaches (Stillinger & LaViolette, 1986; Leach, 2001), perturbation-based approaches (Wales & Doye, 1997), Bayesian approaches (Gonzalez et al., 2015), evolutionary algorithms (Daven et al., 1996), and learned optimizers (Merchant et al., 2021). However, most of these approaches present several drawbacks, namely, *(i)* a significant number of iterations, *(ii)* carefully hand-crafted update rules that are sensitive to parameters, *(iii)* inability to scale to larger system sizes, *(iv)* representation of atomic structures, and, most importantly, *(v)* the inability to overcome high-energy barriers (Wales et al., 2003). Another approach toward finding minima is to combine classical molecular simulation (MS) with machine learning. These approaches focus on accelerated MS using machine-learned force fields or coarse graining (Noé et al., 2020; Park et al., 2021; Li et al., 2022).

An alternative approach is to allow the system *learn* policies that discover better minimum energy structures through *reinforcement learning* (RL) (Christiansen et al., 2020; Simm et al., 2020; Rumelhart et al., 1986; Meldgaard et al., 2020). Most studies using RL for materials have focussed on small atomic clusters or simple molecules having a limited num-

ber of atoms. For extending the work to realistic structures, the first challenge is to develop a scalable representation of atomic structures. To this extent, graph neural networks (GNNs) is an excellent choice—thanks to their ability to capture the local topology, while being inductive to unseen system sizes. GNNs have been used extensively for modeling atomic and physical structures (Batzner et al., 2022; Bhattoo et al., 2023; Thangamuthu et al., 2022; Bhattoo et al., 2022; Battaglia et al., 2018; Bishnoi et al., 2022).

Here, we propose a framework combining GNNs and RL, namely STRIDERNET[1], that allows optimization of atomic structures exhibiting a rough energy landscape. Specifically, we show that combining a graph representation of atomic structures with a *policy-gradient* approach outperforms the standard optimization algorithms. The main contributions of the present work are as follows.

- **STRIDERNET:** A graph reinforcement learning framework (Section 3) that outperforms state-of-the-art optimizers on atomic structures (Section 4.2).
- **Graph matters:** The neighborhood information of atomic structure as captured by the graph architecture enables efficient optimization (Section 4.4). More importantly, a graph-based optimization framework for atomistic configurations has hitherto been unexplored, and this work initiates a new direction.
- **Model adaptation:** Adaptation of the model to a specific atomic structure allows the discovery of low energy states (Section 4.5).
- **Inductivity:** The graph architecture allows the adaptation of a trained model to *unseen* system sizes in an *inductive* fashion (Section 4.6).

2. Preliminaries and Problem Formulation

The configuration $\Omega_c(\mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x}_N)$ of an atomic system is given by the positions of all the atoms in the system $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_N)$ and their types ω_i. Each $\mathbf{x_i}$ represents the position of the i^{th} atom in a d-dimensional space, where d is typically 2 or 3. The potential energy U of an N-atom structure is a function of Ω_c. Specifically, the energy of a system can be written as the summation of one-body $U(r_i)$, two-body $U(r_i, r_j)$, three-body $U(r_i, r_j, r_k)$, up to N-body interaction terms as:

$$U = \sum_{i=1}^{N} U(r_i) + \sum_{\substack{i,j=1;\\ i\neq j}}^{N} U(r_i, r_j) + \sum_{\substack{i,j,k=1;\\ i\neq j\neq k}}^{N} U(r_i, r_j, r_k) + \cdots \tag{1}$$

However, the exact computation of this energy is highly challenging and involves expensive quantum mechanical computations (Cohen et al., 2012). Alternatively, empirical potential functions (Torrens, 2012) can approximately capture this interaction while maintaining the minima associated with these structures. These potentials are developed relying only on two-, three- or four-body interactions and ignoring higher-order terms for computational efficiency. In this work, we rely on well-validated empirical potentials to compute the energy of the different atomic structures. Accordingly, the atomic structure optimization can now be posed as a problem of identifying the configuration of N-atoms in terms of their position vectors, such that the system's total energy is minimum.

The major challenge in such optimization is the rough landscape featuring an enormous number of stable structures (local minima) and a large number of degrees of freedom associated with an atomic structure (Nd for an N-atom structure in d dimensional space; typically $d = 2$ or 3). While characterizing the number of minima in the energy landscape of actual material is challenging, several studies have focused on simple model systems. One of the classical systems extensively characterized includes the Lennard-Jones (LJ) system, which can be used to model noble gases (Tsai & Jordan, 1993; Wales & Doye, 1997; Malek & Mousseau, 2000; Doye et al., 1999). The energy of a system of N-atoms interacting through the LJ potential is given by:

$$U = \lambda \sum_{i=1}^{N-1} \sum_{\substack{j=2;\\ j>i}}^{N} \left[\left(\frac{\beta}{|x_{ij}|} \right)^{12} - \left(\frac{\beta}{|x_{ij}|} \right)^{6} \right] \tag{2}$$

where $|x_{ij}| = |\mathbf{x_i} - \mathbf{x_j}|$ is the distance is between the atoms i and j, and λ and β are constants depending on the atom types. By extrapolating the studies on small LJ structures, the scaling of minima with the number of atoms N can be obtained as $e^{(k_1+k_2 N)}$ or $e^{(k_1+k_2 N+k_3 N^2)}$, where k_1, k_2 and k_3 are constants obtained by fitting (Wales & Doye, 1997). Thus, it becomes incredibly challenging for a system with thousands of atoms to get the global minima or even local minima with extremely low energy compared to the global minima.

Traditional approaches for optimizing atomic structures exploit the gradient of the energy U with the positions to find stable structures near the starting configuration leading to local minima. Some of these approaches include steepest descent (Stillinger & LaViolette, 1986), conjugate gradient, and Newton-Raphson (Leach, 2001). Alternatively, FIRE relies on a momentum-based approach and has been shown to outperform purely gradient-based methods (Bitzek et al., 2006). These approaches aim to find the most stable atomic structure, starting from an arbitrary configuration. Thus, once trapped in a local minimum, these approaches cannot escape the minima to move toward more stable structures. Further, these approaches do not learn any new heuristics based on the trajectory they followed. Thus, there is no possibility of "adapting" these algorithms to obtain more stable structures closer to the global minimum. To address

[1] In our approach, RL trains the policy network to progressively take small *strides* towards optimizing the graph representation of the atomic structure.

these challenges, we propose a framework that exploits the atomic structure and energy relationship to discover stable configurations.

Problem: (Discovering stable structures) *Let $\Omega_c(\mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_N})$ be a configuration of an N-atom system with energy U^{Ω_c} sampled from the energy landscape $\mathbb{U}^{Nd}$ of the system. Starting from Ω_c, our goal is to obtain the configuration Ω_{min} exhibiting the minimum energy $U^{\Omega_{min}}$ by displacing the atoms. To this end, we aim to learn a policy π that displaces the atom so that the system moves toward lower energy configurations while allowing it to overcome local energy barriers.*

In addition to the ability to find low-energy configurations, we also desire π to satisfy the following properties:

- **Permutation Invariance:** Policy π is permutation invariant if $\pi(\Omega_c(\mathbf{x_1}, \ldots, \mathbf{x_N})) = \pi(P(\Omega_c(\mathbf{x_1}, \ldots, \mathbf{x_N})))$, where $P(\cdot)$ is a permutation over the constituent atoms. An atomistic configuration is a set of positions. Sets are permutation invariant by definition. Hence, if the policy is not permutation invariant, it will generate multiple representations for the same set (configuration) depending on the index ordering of atoms. This hampers generalizability to unseen configurations.
- **Inductivity:** Policy π is inductive if the number of parameters in the model is independent of N, i.e., the number of atoms in the system. If the policy is not inductive, it will be restricted to inference *only* on atoms of size N, which limits generalizability to configurations of unseen sizes.

3. STRIDERNET: Proposed Methodology

Fig. 2 describes the architecture of STRIDERNET. To achieve the above-outlined objectives of permutation invariance and inductivity, we represent an atomistic configuration as a graph (more details in Section 3.1). Subsequently, we develop a message-passing GNN to embed graphs into a feature space. The message-passing architecture of the GNN ensures both permutation invariance and inductivity. The graph, in turn, predicts the displacements of each of the atoms based on which the rewards are computed. Finally, the policy π is learned by maximizing the discounted rewards. Note that we learn the parameters of π using a set of training graphs exhibiting diverse energies that are sampled from the energy landscape $\mathbb{E}^{Nd}$ of an atomic system with N-atoms in d dimensions. Thus, the initial structure, although arbitrary and possibly unstable, is realistic and physically feasible. Then given a new structure, we adapt the parameters of our learned policy network π to the new structure while optimizing the new graph structure. All notations used in the present work are given in Tab. 3 in App. A. Before we define the parametrization of our policy, we first discuss how our atomic system is transformed into a graph.

3.1. Transforming atomic system to graph

The total energy U of an atomic system is closely related to the local neighborhood of an atom. In order to leverage this neighborhood information, we transform the atomic structure into a graph, where the nodes and edges of the graph represent the atoms and the chemical bonds between the atoms, respectively. Thus, an atomic system is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the nodes $v \in \mathcal{V}$ denotes the atoms and $e_{vu} \in \mathcal{E}$ represents edges corresponding to the interactions between atoms v and u. Note that the edges can be dynamic in nature; new edges may form, or existing ones may break depending on the configuration Ω_i. Thus, the edges are defined for each Ω_c as a function of the distance between two nodes as $\mathcal{E} = \{e_{uv} = (u, v) \mid d(u, v) \leq \delta\}$ where $d(u, v)$ is a distance function over node positions and δ is a distance threshold. This threshold can be selected based on the first neighbor cutoff of the atomic structures as obtained from the pair-distribution function or based on the cutoff of the empirical potential. The cutoff thus defines the neighborhood of a node v given by $\mathcal{N}_v = \{u | (u, v) \in \mathcal{E}\}$.

3.2. Learning policy π as Markov decision process

Given an atomic structure represented as a graph $\mathcal{G}$ with the potential energy $U_{\mathcal{G}}$, our goal is to update the positions of the nodes $v \in \mathcal{V}$ for t steps, such that the graph structure obtained after these updates $\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$, has a lower potential energy $U_{\mathcal{G}^t}$. We model this task of iteratively updating the node positions as a *Markov decision process* defined by the tuple $(S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, S is the *state space*, $\mathcal{A}$ is the set of all possible *actions*, $\mathcal{P} : S \times S \times \mathcal{A} \to [0, 1]$ denotes the *state transition probability function*, $\mathcal{R} : S \times \mathcal{A} \to \mathbb{R}$ denotes the *reward function* and $\gamma \in (0, 1)$ the *discounting factor*. We next details each of these MDP components.

State: We denote the state of a graph $\mathcal{G}$ at step t as a matrix $S_{\mathcal{G}^t}$, where the i^{th} row in the matrix corresponds to the input node representation for the i^{th} node. Intuitively, the state should contain information that would help our model make a decision regarding the magnitude and direction of each node's displacement. In this context, we note that the overall potential energy of the system is a function of the potential energy of individual atoms[2], which in turn depends upon the local neighborhood around an atom. To capture these intricacies, we construct our state space using a set of semantic and topological node features.

- **Node type:** Each node $v \in \mathcal{V}$ is characterized by its type ω_v. The type ω_v is a discrete variable and is useful in distinguishing particles of different characteristics within a system (Ex. two different types of atoms). We use *one-hot encoding* to represent the node type.
- **Node potential energy:** Potential energy, being a scalar and extensive quantity, is additive in nature; that is, the

[2]We use the terms atoms and nodes interchangeably.

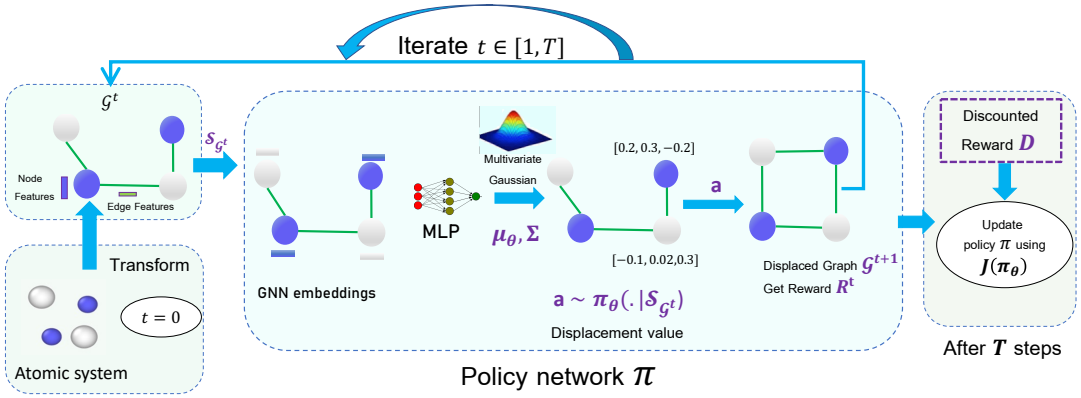

Figure 2. STRIDERNET architecture: The atomic structure is transformed into a graph, which is passed to a policy network that predicts node displacement, and reward is computed. Finally, the policy parameters are updated based on the discounted reward.

potential energy of a system $U_\mathcal{G}$ is the sum of the potential energy of individual atoms. Consequently, the potential energy of a node can be a useful feature to identify the nodes that need to be displaced to reduce the overall energy. We denote the potential energy of node v after t steps as U_v^t (Ex. see Fig. 1 for the distribution of potential energy per atom in an LJ system).

- **Neighborhood potential energy of a node:** As detailed earlier, the potential energy of an atom depends on its neighborhood (see Eq. 2). Thus, the energy of the neighborhood represents whether the atom is located in a relatively stable or unstable region. To this extent, we use the mean and the sum of the potential energy of atoms in the locality of the central atom as a node feature. We denote the sum of the potential energy of a node v's neighborhood at step t as $\text{SUM}(U^t_{\mathcal{N}_v})$, and the mean as $\text{MEAN}(U^t_{\mathcal{N}_v})$.

Additionally, in order to capture the interactions of atoms, we use edge features. Specifically, we use the L^1 distance between two nodes u and v to characterize each edge e_{uv}. Finally, the empirical potentials modeling atomic structures present an *equilibrium bond length* $|x^{equi}_{vu}|$ between two atoms; the distance at which these two atoms exhibit a minimum energy configuration. Note that $|x^{equi}_{vu}|$ for an atomic system can be directly obtained from the potential parameters (Ex. $2^{1/6}\beta$ for LJ; see Eq. 2). To represent this, we include an additional feature $|x^{equi}_{vu}| - |x_{vu}|$, where $|x_{vu}|$ is the bond length of the edge e_{vu} connecting two atoms v and u. This feature quantifies how much stretched/compressed the edge is from its equilibrium configuration. Finally, the initial features of a node v at step t are:

$$\mathbf{s}_v^t = \omega_v \parallel U_v^t \parallel \text{SUM}(U^t_{\mathcal{N}_v}) \parallel \text{MEAN}(U^t_{\mathcal{N}_v}) \quad (3)$$

where, $\mathbf{s}_v^t \in \mathbb{R}^{d_s}$ and $\parallel$ denotes the *concatenation* operation. Further, for an edge e_{vu} with terminal nodes v and u, its initial representation at step t is:

$$\mathbf{s}_e^t = x_v - x_u \parallel y_v - y_u \parallel z_v - z_u \parallel (|x^{equi}_{vu}| - |x_{vu}|) \quad (4)$$

Using the above-designed node features, the state of a graph $\mathcal{G}$ at step t is denoted by a matrix $S_{\mathcal{G}^t} \in \mathbb{R}^{|\mathcal{V}| \times d_s}$ where each row $S_{\mathcal{G}^t}[i] = \mathbf{s}_i^t$.

Action: We displace all the nodes of the graph differently at each step, hence the action space is continuous in our case and is represented as $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $d = 3$.

Reward: Our objective is to reduce the overall potential energy of the system. One option is to define the reward R^t at step $t \geq 0$ as the reduction in potential energy of the system at step t, i.e., $U_{\mathcal{G}^t} - U_{\mathcal{G}^{t+1}}$. However, this definition of reward focuses on short-term improvements instead of long-term. In rough energy landscapes, the path to the global minima may involve crossing over several low-energy barriers. Hence, we use *discounted rewards* D^t to increase the probability of actions that lead to higher rewards in the long term. The discounted rewards are computed as the sum of the rewards over a *trajectory* of actions with varying degrees of importance (short-term and long-term). Mathematically,

$$D^t = R^t + \gamma R^{t+1} + \gamma^2 R^{t+2} + \ldots = \sum_{k=0}^{T-t} \gamma^k R^{t+k} \quad (5)$$

where T is the length of the trajectory and $\gamma \in (0, 1]$ is a *discounting factor* (hyper-parameter) describing how much we favor immediate rewards over the long-term future rewards.

State transition: At each step t, all the nodes in the graph $\mathcal{G}^t$ are displaced based on the translation determined by the policy function π. The graph state thus transits from $S_{\mathcal{G}^t}$ to $S_{\mathcal{G}^{t+1}}$. Since it is hard to model the transition dynamics $p(S_{\mathcal{G}^{t+1}} | S_{\mathcal{G}^t})$ (Hu et al., 2020), we learn the policy in a *model-free* approach. Sec. 3.3 discusses the details.

3.3. Neural method for policy representation

The atoms in a system interact with other atoms in their neighborhood. In order to capture these interactions and infuse topological information, we parameterize our policy by a GNN. At each step t, we first generate the representation of nodes using our proposed GNN. These embeddings are next passed to an MLP to generate a $|\mathcal{V}| \times d$-dimensional vector that represents the mean displacement for each node in each direction. The entire network is then trained end-to-end. We now discuss each of these components in detail.

Graph neural network: Let $\mathbf{h}_v^0 = \mathbf{s}_v^t$ denote the initial node representation of node v and $\mathbf{h}_{vu}^0$ denote the initial

edge representation of edge e_{vu}. We perform L layers of message passing to generate representations of nodes and edges. To generate the embedding for node v at layer $l+1$ we perform the following transformation:

$$\mathbf{h}_v^{l+1} = \sigma\left(\textsc{Mlp}\left(\mathbf{h}_v^l \parallel \sum_{u\in\mathcal{N}_v} \mathbf{W}_{\mathcal{V}}^l(\mathbf{h}_u^l \parallel \mathbf{h}_{vu}^l)\right)\right) \quad (6)$$

where $\mathbf{h}_v^{(l)}$ is the node embedding in layer l and $\mathbf{h}_{vu}^{(l)}$ is the embedding of the edge between node v and u and $u \in \mathcal{N}_v$. $\mathbf{W}_{\mathcal{V}}^l$ is a trainable weight matrix and σ is an activation function. The edge embedding is computed as follows:

$$\mathbf{h}_{vu}^{l+1} = \sigma\left(\textsc{Mlp}\left(\mathbf{h}_{vu}^l \parallel \mathbf{W}_{\mathcal{E}}^l(\mathbf{h}_v^l \parallel \mathbf{h}_u^l)\right)\right) \quad (7)$$

where $\mathbf{h}_{vu}^{(l)}$ is edge embedding in layer l for edge e_{vu}. $\mathbf{W}_{\mathcal{E}}^l$ is a trainable parameter.

Following L layers of message passing, the final node representation of node v in the L^{th} layer is denoted by $\mathbf{h}_v^L \in \mathbb{R}^{d_h}$. Intuitively $\mathbf{h}_v^L$ characterizes v using a combination of its own features and features aggregated from its neighborhood. Note that the equations presented here correspond to the specific GNN implementation used in STRIDERNET. Indeed, we evaluate the effect of graph architecture by replacing our GNN with other architectures such as graph attention network (GAT) (Veličković et al., 2017), full graph network (FGN) (Battaglia et al., 2018) later in Sec. 4.4.

As discussed, at each step t, the nodes in $\mathcal{G}^t$ are displaced based upon the action determined by policy function π. Since our actions are continuous values, we must define the probability distribution over real-valued vectors. To this end, we employ *multivariate Gaussian distribution*[3] $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for modeling the probability distribution over nodes. Here, $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$. Gaussian distribution is commonly used for continuous control in reinforcement learning (Duan et al., 2016; Mnih et al., 2016) since it is easy to sample from and its gradients can also be easily computed (Duan et al., 2016; Rumelhart et al., 1986).

For an action $\mathbf{a}_i \in \mathbb{R}^d$ on node i, we define the policy $\pi_\theta(\mathbf{a}_i|S_{\mathcal{G}^t})$ constructed from the distribution parameters $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ as follows:

$$\pi_\theta(\mathbf{a}_i|S_{\mathcal{G}^t}) = \left(\frac{1}{2\pi}\right)^{d/2} |\boldsymbol{\Sigma}|^{-1/2}\times \exp\left[-\frac{1}{2}(\mathbf{a}_i - \boldsymbol{\mu}_i)'\boldsymbol{\Sigma}^{-1}(\mathbf{a}_i - \boldsymbol{\mu}_i)\right] \quad (8)$$

In the above equation, we parameterize mean $\boldsymbol{\mu}_i$ for node i as:

$$\boldsymbol{\mu}_i = \mu_\theta(\mathbf{h}_i^L)$$

Recall $\mathbf{h}_i^L$ is the embedding of node i generated by GNN in Eq. 6 and is a function of the state of the graph $\mathcal{G}^t$. We do not parameterize $\boldsymbol{\Sigma}$ and instead use a fixed value, i.e., $\boldsymbol{\Sigma} = \alpha \times \mathbf{I}$ where α is a hyper-parameter and $\mathbf{I} \in \mathbb{R}^{d\times d}$ is identity matrix. This is done in order to simplify the learning process (Turner et al., 2022). Nonetheless, our design can be extended to output $\boldsymbol{\Sigma}$ as well.

For a trajectory of length T, we sample actions for all nodes of the graph at each step t using policy π. Consequently, for $\mathcal{G}^t$, we obtain an action vector $\mathbf{a}^t \in \mathbb{R}^{|\mathcal{V}|\times d}$.

3.4. Policy loss computation with baseline

Our goal is to learn parameters such that actions that lead to an overall reduction in energy are favored more over others. Towards this, we use *REINFORCE gradient estimator* with baseline (Williams, 1992) to optimize the parameters of our policy network. Specifically, we wish to maximize the reward obtained for the trajectory of length T with discounted rewards D^t. To this end, we define a reward function $J(\pi_\theta)$ as:

$$J(\pi_\theta) = \mathbb{E}\big[\sum_{t=0}^{T}\left(D^t\right)\big] \quad (9)$$

We, then, optimize $J(\pi_\theta)$ with a *baseline* b as:

$$\nabla J(\pi_\theta) = \left[\sum_{t=0}^{T}\left(D^t - b(\mathcal{S}_{\mathcal{G}^t})\right)\nabla_\theta log\pi_\theta(\mathbf{a}^t/\mathcal{S}_{\mathcal{G}^t})\right] \quad (10)$$

The role of a baseline $b(\mathcal{S}_{\mathcal{G}^t})$ is to estimate the difficulty of a state $S_{\mathcal{G}^t}$ (that is, how difficult it is to perform the task on $S_{\mathcal{G}^t}$ for the baseline) and better contextualize the rewards obtained by the actions generated by π (Kool et al., 2018). Empirically, it often reduces variance and speeds up learning. In our case, we use FIRE (Bitzek et al., 2006) as the baseline since empirical performance obtained by FIRE was found to be better than other optimization techniques for rough landscapes (see Sec. D).

3.5. Training and adaptation

Training phase: For a given set of training graphs, we optimize the parameters of the policy network π_θ for T steps using Eq. 10.

Adaptation Phase: Once we obtain the trained model π_θ, we adapt it to a target graph $\mathcal{G}_{target}$, which was unseen during training. Toward this, we optimize the parameters π_θ as well as the target graph $\mathcal{G}_{target}$ using Eq. 10. The central idea is to keep optimizing the graph structure for an extremely long trajectory (much larger than the training trajectory). However, training policy gradient with large values of T can be difficult due to long-horizon problem (Wang et al., 2020). To overcome this challenge, we sample a lower energy configuration (graph) obtained from the last three steps of the optimization trajectory (of length T) of the target graph $\mathcal{G}_{target}$. This sampled graph (configuration) now becomes the target graph, and we optimize this graph structure and the policy parameters. This process continues for a large number of steps($\gg T$).

[3] Since we deal with d dimensional action space, we use multivariate Gaussian.

It enables the policy to adapt to a low-energy environment, completely unseen during the training, and successively get more stable configurations after each iteration without suffering from the long-horizon problem.

4. Experiments

In this section, we evaluate the performance of STRIDERNET to optimize atomic structures and compare it with classical optimizers. We also analyze the effect of modifying the reward function, including additional features, and graph architectures. Further, we show how the graph architecture enables generalization to unseen system sizes.

4.1. Experimental setup

• **Simulation environment:** All the training and forward simulations are carried out in the JAX environment (Schoenholz & Cubuk, 2020). The graph architecture is implemented using the jraph package (Godwin* et al., 2020). All the codes and datasets are available at https://anonymous.4open.science/r/StriderNET-F64D. The software packages and the hardware details can be found in Appendix Sec. E.

• **Atomic systems and datasets:** To evaluate the performance of STRIDERNET, we consider three systems that are characterized by rough energy landscape, namely, (i) binary LJ mixture, (ii) Stillinger-Weber (SW) silicon, and (iii) calcium-silicate-hydrate (C-S-H) gel. The systems are discussed briefly below. The detailed equations of energy functions for these systems can be found in App. B.

Binary LJ: We select a well-known binary mixture of two atom types with the atoms A and B in the ratio 80 and 20, respectively (Kob & Andersen, 1995). The interactions in this system are pair-wise LJ (Eq. 2). However, this system is a good glass former and hence exhibits a large number of stable local minima. Further, the presence of two types of atoms makes optimization challenging for this system.

SW Silicon (SW Si): The empirical potential of SW Si is more complex, owing to the three-body angular term, thereby making the energy landscape more challenging to optimize (Stillinger & Weber, 1985). Similar to the LJ system, SW Si also exhibits a large number of stable amorphous (disordered) states, although exhibiting a stable ordered crystalline state as well.

Calcium silicate hydrate (C-S-H): C-S-H is a coarse-grained model colloidal gel with interactions similar to LJ (Masoero et al., 2012), but of a higher degree polynomial. This structure is rarely found in an ordered state and, thus, similar to other systems, exhibits a rough landscape.

Dataset generation: The atomic structures corresponding to each of the systems are generated through molecular dynamics or Monte Carlo simulations at high temperatures. This ensures that the initial disordered structures are realistic and sampled from the high-energy regions of the landscape. For each system, 100 atomic structures are selected randomly from the simulation. The detailed data generation procedure is given in App. B.

• **Baselines:** We compare the performance of STRIDERNET with the following three classical optimizers, namely, (i) gradient descent (Stillinger & LaViolette, 1986), (ii) Adam (Kingma & Ba, 2014), and (iii) FIRE (Bitzek et al., 2006). It is worth noting that while gradient descent and FIRE are widely used for atomic structures, Adam is rarely used. Nevertheless, due to the wide use of Adam for other optimization tasks, we include it in the present work. The hyper-parameters of the baseline have been chosen for each system to reach the lowest energy possible.

• **Evaluation metric:** Since the goal of the present work is to find the most stable structure starting from a random initial structure, we use the potential energy of the structure as the metric to evaluate the performance of the algorithms. A more stable structure corresponds to lower energies, with the global minima exhibiting the lowest energy structure. Note that the energy for each of the systems considered is computed using the respective empirical potential. Additionally, to evaluate the performance of the model during the training phase, we compute the change in energy during a given trajectory of length T on the validation graphs. Specifically, at different training epochs, we calculate the average reduction in energy of the system in 20 optimization steps (5 steps longer than the training trajectory), $< E_{20} - E_0 >$, where E_{20} is the energy at the 20^{th} step and E_0 is the energy of the initial configuration from the validation set.

• **Model architecture and training setup:** All the hyperparameters of the model are given in Tabs. 5 and 6 in App. D. For the GNN, the node and edge embeddings are chosen to be of size 48 with a single message passing layer. All MLPs, except the initial node embedding generation MLP and the final displacement prediction MLP, have two hidden layers, each having 48 hidden layer units. The initial node embedding generation MLP has an additional batch-normalization layer, while the final MLP has four hidden layers. Leaky-ReLU is used for all the MLPs as the activation function.

For each system, a dataset of 100 initial states of the environment sampled from the simulation randomly split into $75:25$ training and validation sets, respectively, are used to train the model. During training, at each epoch, a trajectory length of $T = 15$ is used to compute the reward function $J(\pi_\theta)$, and the batch-average loss is used to compute the policy gradient. Validation is performed for the trained model on a trajectory of $T = 20$ steps by selecting graphs randomly from the validation set. Note that validation is performed every 20 epoch. For the adaptation of the trained model to obtain minimum energy, 10 new *target structures(graphs)*, that were not part of the training or validation sets and randomly sampled from the simulation, were used as starting structures. Adaptations of these graphs were carried out for 1000 epochs, with each epoch having a trajectory length of 15 steps. Further, for each structure, the adaptation of STRIDERNET was performed

on 10 random seeds, and the model that gave the minimum energy structure was selected. For each system, the mean of the minimum energy obtained on the 10 structures and the lowest minima among the 10 structures are reported.

For the baselines, the minimization was carried out for 1000 steps in the case of LJ and SW Si, and for 2000 steps in the case of C-S-H. In all the cases, the steps were long enough to ensure that the energy of the structures obtained by baselines was saturated. Similar to STRIDERNET, the minimization was performed on the same 10 configurations, and both the mean minimum energy and lowest minimum energy obtained are reported.

4.2. STRIDERNET: Comparison with baselines

First, we analyze the performance of STRIDERNET on the three systems, namely, LJ, C-S-H, and SW Silicon, to optimize the structures. Figs. 5, 4 in Appendix show the validation and reward curves, respectively, for these models during the training. Table 1 shows the minimum and mean energies obtained by STRIDERNET compared to the baselines for the three systems on 10 initial structures. We note that STRIDERNET achieves better minima than the baselines for LJ, C-S-H, and SW Silicon systems, both in terms of the minimum energy achieved and the mean over 10 structures. We also note that both FIRE and Adam consistently outperform gradient descent. Interestingly, Adam outperforms FIRE on SW Silicon. For the C-S-H system, Adam and FIRE exhibit comparable performance, while for the LJ system, FIRE outperforms Adam. Nevertheless, we observe that STRIDERNET exhibits notably better performance than all the other classical optimization algorithms in obtaining a stable low-energy structure. We also observe that as the structure gets optimized, the distribution of energy and stress gets narrower (see App. K). The superior performance of STRIDERNET could be attributed to several components, such as discounted rewards and graph topology. While discounted reward allows it to overcome local barriers, graph-based modeling enables richer characterization of atomistic configurations through topology. The role of different features in STRIDERNET are discussed in App. F.

4.3. Effect of baseline and additional components

Now, we analyze the role of several components in STRIDERNET such as the use of FIRE as baseline in eq. 10 and additional features towards its performance. STRIDERNET uses FIRE as baseline during training and adaptation. To analyze the effect of baseline, the first variation, termed RL, discards the FIRE baseline and is trained with $b(\mathcal{S}_{\mathcal{G}^t})=0$. The second variation, termed RL+FIRE, equivalent to the STRIDERNET, uses FIRE as a baseline during the training. The third variation, termed RL+Radial, employs vanilla RL with the radial symmetry functions (Behler, 2011) as an additional node input feature for the GNNs, which has been shown to provide excellent neighborhood representation for atomic structures. The final variation, termed RL+Radial+FIRE, uses both FIRE as the baseline and the radial functions as additional input features for the nodes in the GNNs for better neighborhood representation.

Fig. 3(a) in the appendix shows the validation curve of the trained models with the above-mentioned variations. We observe that the best performance is achieved by RL+FIRE and RL+Radial. Note that including radial features (RL+Radial) makes the computation more expensive for this model (Behler, 2011). We also observe that RL performs similarly to RL+FIRE, although for larger epochs. However, the forward trajectory of the RL without baseline occasionally exhibits instability, whereas the RL+FIRE exhibits highly stable inference. We observe that RL+Radial+FIRE shows poorer performance than RL+FIRE and RL+Radial. Altogether, we observe that the STRIDERNET, represented by RL+FIRE, represents the optimal model in terms of computational efficiency and inference.

4.4. GNN architectures: MLP, GAT, FGN, STRIDERNET

We evaluate the role of the GNNs architecture on the performance of STRIDERNET. To this extent, we compare three models with different graph architectures, namely, GAT, FGN, and STRIDERNET, which has our own architecture (see Sec. 3.3). In order to evaluate the role of GNNs, we also trained a model with a fully-connected feed-forward multilayer perceptron (MLP). Details of the hyperparameters for all the models are in App. L. In Fig. 3(b) we observe that the proposed GNN architecture in STRIDERNET provides superior performance, although GAT also leads to similar performance for larger epochs. We note that the FGN architecture is unable to achieve comparable performance. Interestingly, the MLP-based model fails to train and shows no reduction in energy, even at large epochs. This suggests that the topology and neighborhood information, as captured by the GNN through message passing plays a crucial role in the performance of STRIDERNET.

4.5. Model adaptation

Now, we analyze the evolution of the energy of a structure during adaptation. Fig. 3(c) shows the performance of STRIDERNET along with the baselines on 10 structures. It should be noted that for STRIDERNET, the adaptation of the trained model involves back-propagation; hence, the evolution of energy is plotted with the number of epochs in this case. In the case of both LJ and C-S-H systems, we observe that STRIDERNET consistently exhibits lower energy than other models. In the case of SW Si, we observe that STRIDERNET, although initially exhibiting higher energy, eventually outperforms other models. Thus, we observe that the model adaptation on an unseen target graph structure allows STRIDERNET to outperform classical optimization algorithms.

Atomic system	Metric	Gradient Descent	FIRE	Adam	STRIDERNET
LJ (ε units)	Min	-799.53	-813.66	-808.62	**-815.63**
	Mean	-795.38	-806.29	-801.96	**-811.99**
C-S-H (kcal/mol)	Min	-1583539.3	-1637194.1	-1622905.9	**-1671916.8**
	Mean	-1548798.6	-1588792.4	-1596680.4	**-1648965.9**
SW Silicon (eV)	Min	-249.22	-256.98	-258.86	**-259.94**
	Mean	-247.56	-256.37	-256.93	**-257.35**

Table 1. Comparison of STRIDERNET with classical optimization algorithms for LJ, C-S-H, and SW Silicon systems. For each system, the minimum and mean energies are evaluated on 10 random initial structures.

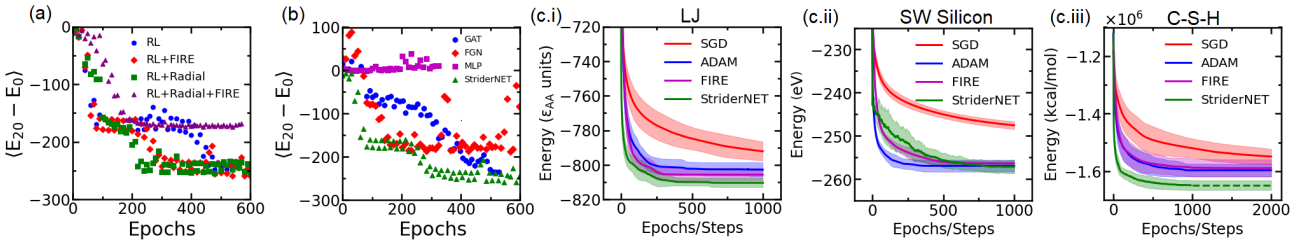

Figure 3. (a) Validation curve during training for different models. (b) Comparison of different graph architectures for RL algorithm, namely, GAT, FGN, STRIDERNET, and MLP. (c) Evolution of energy during adaptation of STRIDERNET for: (i) LJ, (ii) SW Silicon, and (iii) C-S-H system, respectively. The curve represents the mean over 10 structures, and the shaded regions represent the standard deviation. Note that the STRIDERNET for C-S-H is run only for 1000 epochs and the dotted line represents the value at the 1000^{th} step.

Number of atoms	Metric	Gradient descent	Adam	FIRE	STRIDERNET
25	Min.	-6.94	-7.00	-6.99	**-7.08**
	Mean	-6.79	-6.91	-6.81	**-6.97**
50	Min.	-7.67	-7.70	-7.67	**-7.77**
	Mean	-7.57	-7.62	-7.63	**-7.71**
100	Min.	-8.00	-8.09	-8.14	**-8.16**
	Mean	-7.92	-8.03	-8.06	**-8.12**
250	Min.	-8.02	**-8.15**	**-8.15**	**-8.15**
	Mean	-7.98	-8.10	-8.11	**-8.13**
500	Min.	-8.02	-8.14	-8.14	**-8.16**
	Mean	-7.99	-8.12	-8.12	**-8.14**
1000	Min.	-8.00	-8.13	**-8.14**	-8.13
	Mean	**-8.12**	**-8.12**	**-8.12**	**-8.12**

Table 2. Minimum energy obtained by adaptation of STRIDERNET trained on a 100-atom LJ system to varying system sizes. For comparison among multiple sizes, total energy normalized by the number of atoms in the system is shown.

4.6. Inductivity to varying system sizes

Finally, we evaluate the ability of STRIDERNET trained on a given graph size to adapt to unseen graph sizes. To this extent, we consider the STRIDERNET trained for the LJ system having 100 atoms and adapt it to different system sizes with $N = 25, 50, 250, 500, 1000$. Table 2 shows the performance of STRIDERNET on all the system sizes. Interestingly, for all structures from 25 to 500 atoms, we observe that STRIDERNET gives the best performance in terms of both the overall minimum and the mean of the minimum energies of 10 structures. For the 1000 atom system, we observe that STRIDERNET gives the same performance as Adam and FIRE for mean energy, while FIRE outperforms Adam and STRIDERNET in terms of the minimum energy achieved. However, it is worth noting that STRIDERNET gives comparable performance for the mean energy even for 1000 atom structures; that is one order larger than the trained graph. We also note that STRIDERNET performs better when trained on 250-atom system in comparison to when trained on 100-atom system, confirming its applicability to more complex and larger systems. Details are present in Appendix Sec. H.

5. Conclusion

In this work, we present STRIDERNET, a graph reinforcement learning approach that enables the optimization of atomic structures on a rough landscape. We evaluate the model on three systems, namely, LJ, C-S-H, and SW Silicon, and show that STRIDERNET outperforms the classical optimization algorithms such as gradient descent, FIRE, and Adam. We also show that the model exhibits inductivity to completely unseen system sizes; STRIDERNET trained on 100 atom yields superior performance for a 500 atom system. Altogether, STRIDERNET presents a promising framework to optimize atomic structures.

Limitations and future work: Although promising, STRIDERNET is limited to a relatively small number of atoms. Scaling it to a larger number of atoms presents a major computational challenge. Further, although STRIDERNET outperformed classical local optimizers, the energy reached by STRIDERNET is not the global minimum. Thus, there is further scope for improvement that enables one to discover the global minimum in these structures. The GNN employed in STRIDERNETis not SE(3) equivariant. While it is known that SE(3) equivariant GNNs have better expressibility, they are computationally expensive. It would be interesting to explore the application of such architectures on the performance STRIDERNET. In many cases, energy might not be the sole criterion for optimizing atomic structure. For such problems, the reward function in STRIDERNET can be modified to be multi-objective. Finally, an iteration of STRIDERNET is computationally more expensive compared to non-neural baselines, the acceleration of which also is an open challenge.

References

Alonso-Mora, J., Beardsley, P., and Siegwart, R. Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics*, 34(2):404–420, 2018.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of chemical physics*, 134(7):074106, 2011.

Bhattoo, R., Ranu, S., and Krishnan, N. A. Learning articulated rigid body dynamics with lagrangian graph neural network. In *Advances in Neural Information Processing Systems*, 2022.

Bhattoo, R., Ranu, S., and Krishnan, N. A. Learning the dynamics of particle-based systems with lagrangian graph neural networks. *Machine Learning: Science and Technology*, 2023.

Bishnoi, S., Bhattoo, R., Ranu, S., and Krishnan, N. Enhancing the inductive biases of graph neural ode for modeling dynamical systems. *arXiv preprint arXiv:2209.10740*, 2022.

Bitzek, E., Koskinen, P., Gähler, F., Moseler, M., and Gumbsch, P. Structural relaxation made simple. *Physical review letters*, 97(17):170201, 2006.

Christiansen, M.-P. V., Mortensen, H. L., Meldgaard, S. A., and Hammer, B. Gaussian representation for image recognition and reinforcement learning of atomistic structure. *The Journal of Chemical Physics*, 153(4):044107, 2020.

Cohen, A. J., Mori-Sánchez, P., and Yang, W. Challenges for density functional theory. *Chemical reviews*, 112(1): 289–320, 2012.

Daven, D., Tit, N., Morris, J., and Ho, K. Structural optimization of lennard-jones clusters by a genetic algorithm. *Chemical physics letters*, 256(1-2):195–200, 1996.

Doye, J. P., Miller, M. A., and Wales, D. J. The double-funnel energy landscape of the 38-atom lennard-jones cluster. *The Journal of Chemical Physics*, 110(14):6896–6906, 1999.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Godwin*, J., Keck*, T., Battaglia, P., Bapst, V., Kipf, T., Li, Y., Stachenfeld, K., Veličković, P., and Sanchez-Gonzalez, A. Jraph: A library for graph neural networks in jax., 2020. URL http://github.com/deepmind/jraph.

Gonzalez, J., Longworth, J., James, D. C., and Lawrence, N. D. Bayesian optimization for synthetic gene design. *arXiv preprint arXiv:1505.01627*, 2015.

Hu, S., Xiong, Z., Qu, M., Yuan, X., Côté, M.-A., Liu, Z., and Tang, J. Graph policy network for transferable active learning on graphs. *Advances in Neural Information Processing Systems*, 33:10174–10185, 2020.

Ioannidou, K., Kanduč, M., Li, L., Frenkel, D., Dobnikar, J., and Del Gado, E. The crucial effect of early-stage gelation on the mechanical properties of cement hydrates. *Nature communications*, 7(1):12106, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kob, W. and Andersen, H. C. Testing mode-coupling theory for a supercooled binary lennard-jones mixture i: The van hove correlation function. *Physical Review E*, 51(5): 4626, 1995.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Le, T. C. and Winkler, D. A. Discovery and optimization of materials using evolutionary approaches. *Chemical reviews*, 116(10):6107–6132, 2016.

Leach, A. R. *Molecular modelling: principles and applications*. Pearson education, 2001.

Li, Z., Meidani, K., Yadav, P., and Barati Farimani, A. Graph neural networks accelerated molecular dynamics. *The Journal of Chemical Physics*, 156(14):144103, 2022.

Liu, H., Dong, S., Tang, L., Krishnan, N. A., Masoero, E., Sant, G., and Bauchy, M. Long-term creep deformations in colloidal calcium–silicate–hydrate gels by accelerated aging simulations. *Journal of colloid and interface science*, 542:339–346, 2019a.

Liu, H., Dong, S., Tang, L., Krishnan, N. A., Sant, G., and Bauchy, M. Effects of polydispersity and disorder on the mechanical properties of hydrated silicate gels. *Journal of the Mechanics and Physics of Solids*, 122:555–565, 2019b.

Liu, H., Xiao, S., Tang, L., Bao, E., Li, E., Yang, C., Zhao, Z., Sant, G., Smedskjaer, M. M., Guo, L., et al. Predicting the early-stage creep dynamics of gels from their static structure by machine learning. *Acta Materialia*, 210: 116817, 2021.

Malek, R. and Mousseau, N. Dynamics of lennard-jones clusters: A characterization of the activation-relaxation technique. *Physical Review E*, 62(6):7723, 2000.

Manzano, H., Masoero, E., Lopez-Arbeloa, I., and Jennings, H. M. Shear deformations in calcium silicate hydrates. *Soft Matter*, 9(30):7333–7341, 2013.

Masoero, E., Del Gado, E., Pellenq, R.-M., Ulm, F.-J., and Yip, S. Nanostructure and nanomechanics of cement: polydisperse colloidal packing. *Physical review letters*, 109(15):155503, 2012.

Meldgaard, S. A., Mortensen, H. L., Jørgensen, M. S., and Hammer, B. Structure prediction of surface reconstructions by deep reinforcement learning. *Journal of Physics: Condensed Matter*, 32(40):404005, 2020.

Merchant, A., Metz, L., Schoenholz, S. S., and Cubuk, E. D. Learn2hop: Learned optimization on rough landscapes. In *International Conference on Machine Learning*, pp. 7643–7653. PMLR, 2021.

Mistakidis, E. S. and Stavroulakis, G. E. *Nonconvex optimization in mechanics: algorithms, heuristics and engineering applications by the FEM*, volume 21. Springer Science & Business Media, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Noé, F., Tkatchenko, A., Müller, K.-R., and Clementi, C. Machine learning for molecular simulation. *Annual review of physical chemistry*, 71:361–390, 2020.

Park, C. W., Kornbluth, M., Vandermause, J., Wolverton, C., Kozinsky, B., and Mailoa, J. P. Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture. *npj Computational Materials*, 7(1):73, 2021.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Schoenholz, S. and Cubuk, E. D. Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33, 2020.

Schwager, M., Rus, D., and Slotine, J.-J. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *The International Journal of Robotics Research*, 30(3):371–383, 2011.

Simm, G., Pinsler, R., and Hernández-Lobato, J. M. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pp. 8959–8969. PMLR, 2020.

Singh, S., Ediger, M. D., and De Pablo, J. J. Ultrastable glasses from in silico vapour deposition. *Nature materials*, 12(2):139–144, 2013.

Stillinger, F. H. and LaViolette, R. A. Local order in quenched states of simple atomic substances. *Physical Review B*, 34(8):5136, 1986.

Stillinger, F. H. and Weber, T. A. Computer simulation of local order in condensed phases of silicon. *Phys. Rev. B*, 31:5262–5271, Apr 1985. doi: 10.1103/PhysRevB.31.5262. URL https://link.aps.org/doi/10.1103/PhysRevB.31.5262.

Thangamuthu, A., Kumar, G., Bishnoi, S., Bhattoo, R., Krishnan, N. A., and Ranu, S. Unravelling the performance of physics-informed graph neural networks for dynamical systems. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., in 't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., and Plimpton, S. J. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022. doi: 10.1016/j.cpc.2021.108171.

Torrens, I. *Interatomic potentials*. Elsevier, 2012.

Tsai, C. and Jordan, K. Use of an eigenmode method to locate the stationary points on the potential energy surfaces of selected argon and water clusters. *The Journal of Physical Chemistry*, 97(43):11227–11237, 1993.

Turner, M., Koch, T., Serrano, F., and Winkler, M. Adaptive cut selection in mixed-integer linear programming. *arXiv preprint arXiv:2202.10962*, 2022.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *ICLR*, 2017.

Wales, D. et al. *Energy landscapes: Applications to clusters, biomolecules and glasses*. Cambridge University Press, 2003.

Wales, D. J. and Doye, J. P. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

Wang, R., Du, S. S., Yang, L. F., and Kakade, S. M. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*, 2020.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Xiang, X.-D., Sun, X., Briceno, G., Lou, Y., Wang, K.-A., Chang, H., Wallace-Freedman, W. G., Chen, S.-W., and Schultz, P. G. A combinatorial approach to materials discovery. *Science*, 268(5218):1738–1740, 1995.

Yang, K. K., Wu, Z., and Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019.

ICML 2023 Paper Checklist Guidelines

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? Yes.
   (b) Did you describe the limitations of your work? Yes; See Sec.5.
   (c) Did you discuss any potential negative societal impacts of your work? N/A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? N/A
   (b) Did you include complete proofs of all theoretical results? N/A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes; See Sec. 4, code link: https://anonymous.4open.science/r/StriderNET-F64D
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes. See Table 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes; shaded region in Fig. 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? See Sec. 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? Yes; See Sec. 4
   (b) Did you mention the license of the assets? N/A
   (c) Did you include any new assets either in the supplemental material or as a URL? N/A
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? N/A
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? N/A
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? N/A
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? N/A
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? N/A

A. Notations

All the notations used in this work are outlined in Tab. 3.

Symbol	Meaning
$\mathcal{G}^t$	Graph at step t
$\mathcal{V}$	Node set
$\mathcal{E}^t$	Edge set at step t
$\mathcal{S}_{\mathcal{G}^t}$	State of Graph at step t
$\mathcal{N}_v$	Neighboring nodes of node v
U_v	Potential energy of node v
U_v^t	Potential energy of node v at step t for graph $\mathcal{G}^t$
$U_{\mathcal{G}^t}$	Potential energy of graph $\mathcal{G}$ at step t
e	Edge $e \in \mathcal{E}$
d	Number of Dimensions in the system
$\mathbf{s}_v^t$	Initial feature representation of node v at step t
T	Length of trajectory
π	Policy function
$\mathbf{a}$	Action vector for all nodes of a graph. $\mathbf{a} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d}$
μ_i	Predicted mean displacement for the i^{th} node. $\mu_i \in \mathbb{R}^d$
$\mathbf{\Sigma}$	Covariance Matrix. $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$

Table 3. Notations used in the paper

B. System Details

B.1. Binary Lennard-Jones (LJ)

The system has two types of particles with composition $A_{80}B_{20}$ consisting of total N(=25,50,100,250,500) particles in a cubic ensemble with periodic boundaries. The interaction between the particles is governed by

$$V_{\mathrm{LJ}}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{11}$$

where r refers to the distance between two particles, σ is the distance at which inter-particle potential energy is minimum and ε refers to the depth of the potential well. Here, we use the LJ parameters $\varepsilon_{AA} = 1.0$, $\varepsilon_{AB} = 1.5$, $\varepsilon_{BB} = 0.5$, $\sigma_{AA} = 1.0$, $\sigma_{AB} = 0.8$ and $\sigma_{BB} = 0.88$. The mass for all particles is set to 1.0. All the quantities are expressed in reduced units with respect to σ_{AA}, ε_{AA}, and Boltzmann constant k_B. We set the interaction cutoff $r_c = 2.5\sigma$ (Singh et al., 2013) and the time step $dt = 0.003$ for simulations.

We perform all the molecular dynamic simulations at constant volume and temperature. For preparing the initial high energy structures, the ensemble is taken to a high temperature $T = 2.0$ where it equilibrates in the liquid state. Once it equilibrates, 100 random configurations are sampled.

B.2. Stillinger Weber (SW) Silicon

The system consists of N=64 particles in a cubic ensemble with periodic boundaries interacting via the Stillinger Weber(SW) potential, as given by the following equation.

$$\begin{aligned} E &= \sum_i \sum_{j>i} \phi_2\left(r_{ij}\right) + \sum_i \sum_{j\neq i} \sum_{k>j} \phi_3\left(r_{ij}, r_{ik}, \theta_{ijk}\right) \\ \phi_2\left(r_{ij}\right) &= A_{ij}\epsilon_{ij}\left[B_{ij}\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{q_{ij}}\right] \exp\left(\frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right) \\ \phi_3\left(r_{ij}, r_{ik}, \theta_{ijk}\right) &= \lambda_{ijk}\epsilon_{ijk}\left[\cos\theta_{ijk} - \cos\theta_{0ijk}\right]^2 \exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right) \exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik} - a_{ik}\sigma_{ik}}\right) \end{aligned} \tag{12}$$

where ϕ_2 is the two body term and ϕ_3 is the three-body angle term. The following are the standard parameters(Stillinger & LaViolette, 1986) used in the equation:

Parameter	ε	σ	A	B	p	q	a	λ	γ	$cos\theta_0$
Value	2.1683 eV	2.0951 Å	7.0495	0.6022	4	0	1.80	21.0	1.20	-1/3

Table 4. Parameters for Stillinger weber potential

We equilibrate the system at a high temperature of T=3500 K in an isochoric-isothermal (NVT) ensemble to obtain the initial high-energy configurations.

B.3. Calcium silicate hydrate (C-S-H) gel

Calcium silicate hydrate(C-S-H) is the binding phase in concrete. C-S-H is known to govern various properties of concrete, including strength and creep. The coarse-grained colloidal gel model of C-S-H used in this work was proposed by Masoero et al.(Masoero et al., 2012). The model has been studied extensively and found to be capable of simulating the realistic mesoscale structure of C-S-H as well as long-term creep behavior(Liu et al., 2019a; 2021).

The C-S-H particles interact with each other via a generalized Lennard-Jones interaction potential as given by the following equation:

$$U_{ij}(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{2\alpha} - \left(\frac{\sigma}{r_{ij}}\right)^{\alpha}\right] \tag{13}$$

Where U_{ij} is the interaction potential energy between any pair particles 'i' and 'j', r_{ij} is the distance between the particles, and σ is the grain diameter which is taken to be 5 nm in the model. α is a parameter that controls the potential well's narrowness. α is chosen to be 14 such that the tensile strain at failure is close to that obtained in previous simulations of bulk C–S–H. ε is the potential well's energy depth. The energy depth is given by $\varepsilon = A_0\sigma^3$, where $A_0 = kE$ and E is the young's modulus of bulk C–S–H grain, which is around 63.6 GPa (Manzano et al., 2013) and k=0.0023324.

B.3.1. PREPARATION OF C-S-H BY GCMC SIMULATIONS AND OBTAINING HIGH ENERGY STATES

During the hydration process, the chemical reaction between the cement and electrolytes in water occurs via a dissolution-precipitation reaction. The grand canonical Monte Carlo (GCMC) simulations mimic the precipitation process during the hydration of cement. The C-S-H particles are iteratively inserted in an empty cubic box ensemble with periodic boundary conditions. In each step of the simulation, 'X' attempts of grain exchanges(i.e., insertions and deletions) are performed, which is followed by 'M' attempts of randomly displacing the grains to achieve a more stable configuration. The following equation gives the Monte Carlo acceptance probability according to the Metropolis algorithm:

$$P_{acceptance} = min\left\{1, exp\left[-\left(\Delta U - \frac{\mu\lambda}{k_B T}\right)\right]\right\} \tag{14}$$

where ΔU is the change in energy after the Monte Carlo trial move, μ is the chemical potential which represents the free energy gained by the formation of C-S-H hydrates, λ is the variation in the number of C-S-H particles, k_B is Boltzmann constant. T is the temperature of an infinite reservoir source. The chemical potential of the reservoir is kept as $2k_BT$ as per the previous studies(Ioannidou et al., 2016; Liu et al., 2019b). The GCMC steps are performed until the no. of inserted C-S-H grains reaches saturation. The simulations are performed at a temperature of T=300 K. The final saturated configurations so obtained are relaxed in the isothermal-isobaric (NPT) ensemble at 300 K and zero pressure for 50 ns to release ant macroscopic tensile stress induced during GCMC simulation. Finally, energy minimization is performed to reach the inherent state of the configuration.

Next, the obtained structure is taken to a high temperature of T=1000K in an isothermal-isochoric (NVT) ensemble and allowed to equilibrate. Once it equilibrates, 100 random configurations are sampled. The GCMC simulation was performed in Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) (Thompson et al., 2022) software.

C. Reward and validation curves of StriderNet

Figure 4 shows the reward at the end of each of the validation trajectories for STRIDERNET trained on LJ, SW Si, and C-S-H systems. Positive values of the rewards suggest that the model has outperformed FIRE on the validation graphs.

Figure 5 shows the difference between the energy at the beginning and the end of the trajectory on the validation set. We observe that the curve saturates for both LJ and C-S-H systems. However, SW Si exhibits a further downward trend after 800 epochs. It is worth noting that the SW Si has a tendency for crystallization and exhibits a global minimum crystalline structure. Thus, it would be worth exploring further on continuing the training of the SW Si systems towards exploration of a lower minimum.

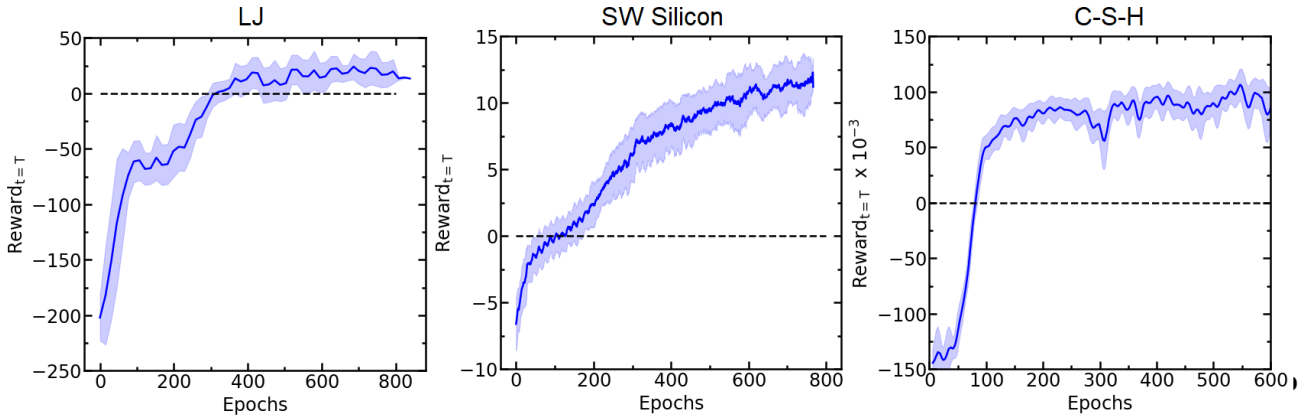

Figure 4. Reward at the end of trajectory during the training of STRIDERNET for LJ, SW Si, C-S-H systems.

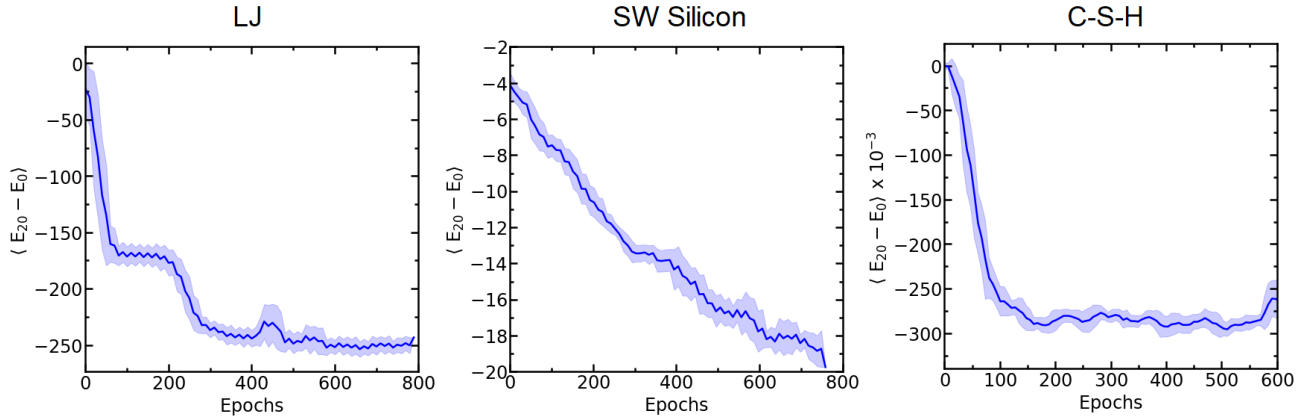

Figure 5. Validation curves: Average reduction in energy in 20 steps of optimization during the training of STRIDERNET for LJ, SW Si, and C-S-H.

D. Hyperparameters of STRIDERNET and baselines

Hyperparameters of STRIDERNET are included in Tab. 5. Further, the hyperparameters associated with the baselines, namely, FIRE, Adam, and gradient descent are included in Tab. 6. To reduce computational overhead, we run baseline only on the initial state and use that value across all steps in the trajectory during the training of STRIDERNET.

E. System details

Software packages: numpy-1.24.1, jax-0.4.1, jax-md-0.2.24, jaxlib-0.4.1, jraph-0.0.6.dev0, flax-0.6.3, optax-0.1.4
Hardware: Processor: 2x E5-2680 v3 @2.5GHz/12-Core "Haswell" CPU RAM: 62 GB"

Hyper-parameters	
PARAMETER	VALUE
Edge embedding size	48
Node embedding size	48
Initial node embedding MLP f_a layers	3
Initial edge embedding MLP f_b layers	2
Edge update MLP layers	2
Node update MLP layers	2
Node displacement MLP layers	4
Message passing steps(L)	1
Batch-norm layer decay rate for the exponential moving average	0.9
Trajectory length(T)	15
Gradient accumulation steps	2
Graphs training batch size	4
Edge to node aggregaton function	Mean
Activation functions(all MLPs)	Leaky ReLU
Multivariate Gaussian constant factor(α)	10^{-5}
Rewards discount factor (γ)	0.9
Training optimizer	Adam
Training optimizer learning rate	0.005
Node displacement MLP neighborhood aggregation	Mean
Predicted displacement scaling factor	2.0 (LJ), 2.0 (SW Si),5.0 (C-S-H)
Gradient clipping	0.1

Table 5. Hyper-parameters of STRIDERNET

Baseline	Parameter	LJ	SW SIlicon	CSH
Gradient descent	Learning rate	5×10^{-4}	10^{-3}	5×10^{-4}
Adam	Learning rate	0.05	0.1	1.0
	β_1	0.9		
	β_2	0.999		
	ε	10^{-8}		
	$\bar{\varepsilon}$	0.0		
FIRE	dt_{start}	0.01	0.5	5×10^{-3}
	dt_{max}	0.4		
	N_{min}	5		
	f_α	0.99		
	f_{dec}	0.5		
	f_{inc}	1.1		
	α_{start}	0.1		

Table 6. Baselines hyperparameters

F. Feature Ablation

To understand the contribution of each of the node and edge features toward the performance of STRIDERNET, we performed the ablation studies by removing the node and edge features one by one in STRIDERNET. The validation score of each of the models is studied in Fig. 6. First, we observe that the edge features play a crucial role as the model without edge features exhibits poor performance. We also note that STRIDERNET with all the features exhibits the best performance among all the models. Thus, although the edge features play a major role in the model performance, the node features enhance the performance when used in conjunction with the edge features.

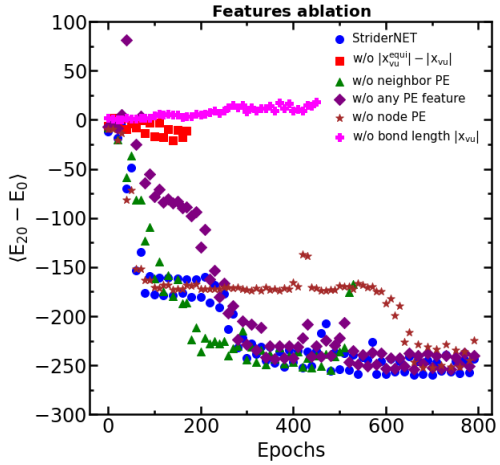

Figure 6. **Feature ablation:** Validation curves of STRIDERNET on binary LJ system with 100 atoms using different set of input features.

G. Training time of STRIDERNET with respect to system size

In Table 7 we study the scalability of STRIDERNET against different system sizes.

System Size	10	25	50	100	250	500	1000
Training time(100 epochs) in Hours	0.92	1.25	1.75	2.11	2.50	3.83	8.89

Table 7. Training time of STRIDERNET for binary LJ system with different sizes.

H. Training on larger systems

We trained STRIDERNET on a larger LJ-system of 250 atoms. For the model trained on larger system, adaptation was further performed on 10 random configurations with (250, 500, and 1000) atoms. The results of these models are compared with the ones obtained from the original STRIDERNET trained on 100 atoms (see Tables below). Interestingly, we observe that the model trained on 250 atoms outperform STRIDERNET trained on 100 atoms. This suggests that the model performance improves when trained on larger number of atoms. This result suggests that STRIDERNET can indeed be used for more larger complex systems with potentially better performance.

System Size	**Metric**	For model trained on $N = 100$	For model trained on $N = 250$
250	Min	-2037.06	**-2038.45**
	Mean	-2033.25	**-2035.74**
500	Min	-4080.65	**-4086.12**
	Mean	-4069.59	**-4071.64**
1000	Min	-8126.29	**-8139.19**
	Mean	-8121.44	**-8122.94**

Table 8. Performance of STRIDERNET when trained on 100 and 250 atom systems for the LJ-system. The table shows minimum energy obtained for different system sizes.

I. Comparison of running time of all optimizers

In Table 10 we study the running time of all optimizers. We observe that the baseline optimizers are faster than the StriderNET. The major cost comes from the requirement of creation of graph based representation at each step. The other methods doesn't require the graph representation. As mentioned earlier our graph update function is currently not JIT compiled, whereas the baseline optimizers are JIT compiled. Once we have an efficient JIT compiled implementation of

System Size	Metric	Training on N = 100	Training on N = 250	Gradient Descent	Adam	FIRE
250	Min	-8.148	**-8.154**	-8.025	-8.152	-8.153
	Mean	-8.133	**-8.143**	-7.978	-8.103	-8.109
500	Min	-8.161	**-8.172**	-8.016	-8.144	-8.136
	Mean	-8.139	**-8.142**	-7.990	-8.118	-8.120
1000	Min	-8.126	**-8.139**	-8.003	-8.135	-8.139
	Mean	-8.121	**-8.123**	-7.984	-8.119	-8.124

Table 9. Normalized Performance(w.r.t number of atoms) of STRIDERNET when trained on 100 and 250 atom systems for the LJ-system. The table shows minimum energy obtained(normalized) for different system sizes.

graph update function, our model shall become comparable to other optimizers. Also running on GPU will significantly improve the running time.

Method	Runtime(100 steps) (in seconds)
STRIDERNET Adaptation	1170(for 100 epochs)
STRIDERNET forward pass	54.3
Gradient Descent	0.72
Adam	0.76
FIRE	1.02

Table 10. Comparison of running time of all optimizers on LJ binary system with 100 nodes.

J. Impact of learning rate on non-neural optimizers

we have investigated the effect for all the baselines (dt_start, in case of FIRE algorithm) by performing additional experiments with different learning rates. Table 11 shows the minimum energies obtained at different learning rates in 2000 steps. Here unstable means that algorithm increases the energy instead of decreasing. We have chosen the learning rate that gives minimum energy.

Learning rate	Gradient Descent	Adam	FIRE
1e-1	Unstable	Unstable	Unstable
5e-2	Unstable	**-808.6**	Unstable
1e-2	Unstable	-807.4	**-813.7**
5e-3	Unstable	-806.8	-811.4
1e-3	Unstable	-806.8	-811.3
5e-4	**-811.3**	-805.4	-803.4
1e-4	-804.1	-806.6	-787.4
5e-4	-804.1	-806.6	-779.7
1e-5	-802.5	-813.5	-732.3

Table 11. The performance of baselines with different learning rates for 100 atom LJ-system. The best result for each of the models is shown in boldface.

K. Variation of energy and stress distribution during optimization

To understand the behavior of STRIDERNET during model adaptation, we study the variation in the distribution of energy and stress over the particles. Figure 7 shows the energy distributions. We observe that as the structure gets optimized, the energy distribution shift towards lower energies and sharpens. Figure 8 shows the variation of atomic stresses $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$. We observe that as the structure gets optimized, the stresses tend to sharpen toward zero stress.

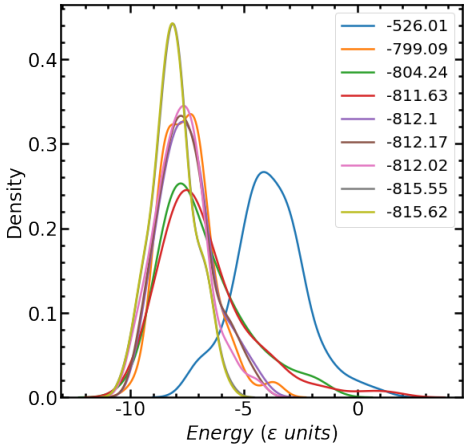

Figure 7. **Energy distribution:** Variation of energy distribution during model adaptation by STRIDERNET on binary LJ system with 100 atoms

L. Details of GNN architectures

The details of all the graph architectures are included below.

L.1. Graph attention network (GAT)

The GAT architecture uses the same node and edge features as StriderNET architecture(section 3.2).It consists of softmax attention on the edges. Other hyper parameters are tabulated in Table 12

Hyper-parameters	
PARAMETER	VALUE
Edge embedding size	48
Node embedding size	48
Initial node embedding MLP f_a layers	2
Initial edge embedding MLP f_b layers	2
Attention query MLP layers	2
Attention logits MLP layers	2
Node displacement MLP layers	4
No. of GAT Layers(L)	3
Activation functions(all MLPs)	Leaky ReLU
Node displacement MLP neighborhood aggregation	Mean

Table 12. Hyper-parameters of GAT

L.2. Full graph network (FGN)

The full graph network(FGN) architecture uses the same node and edge features as StriderNET architecture(section 3.2). Other hyper parameters are tabulated in Table 13

L.3. Multi-layer perception (MLP)

The multi-layer perception (MLP) uses only node features as the input feature vector. The node features are same as StriderNET(section 3.2). Other hyper parameters are tabulated in Table 14.

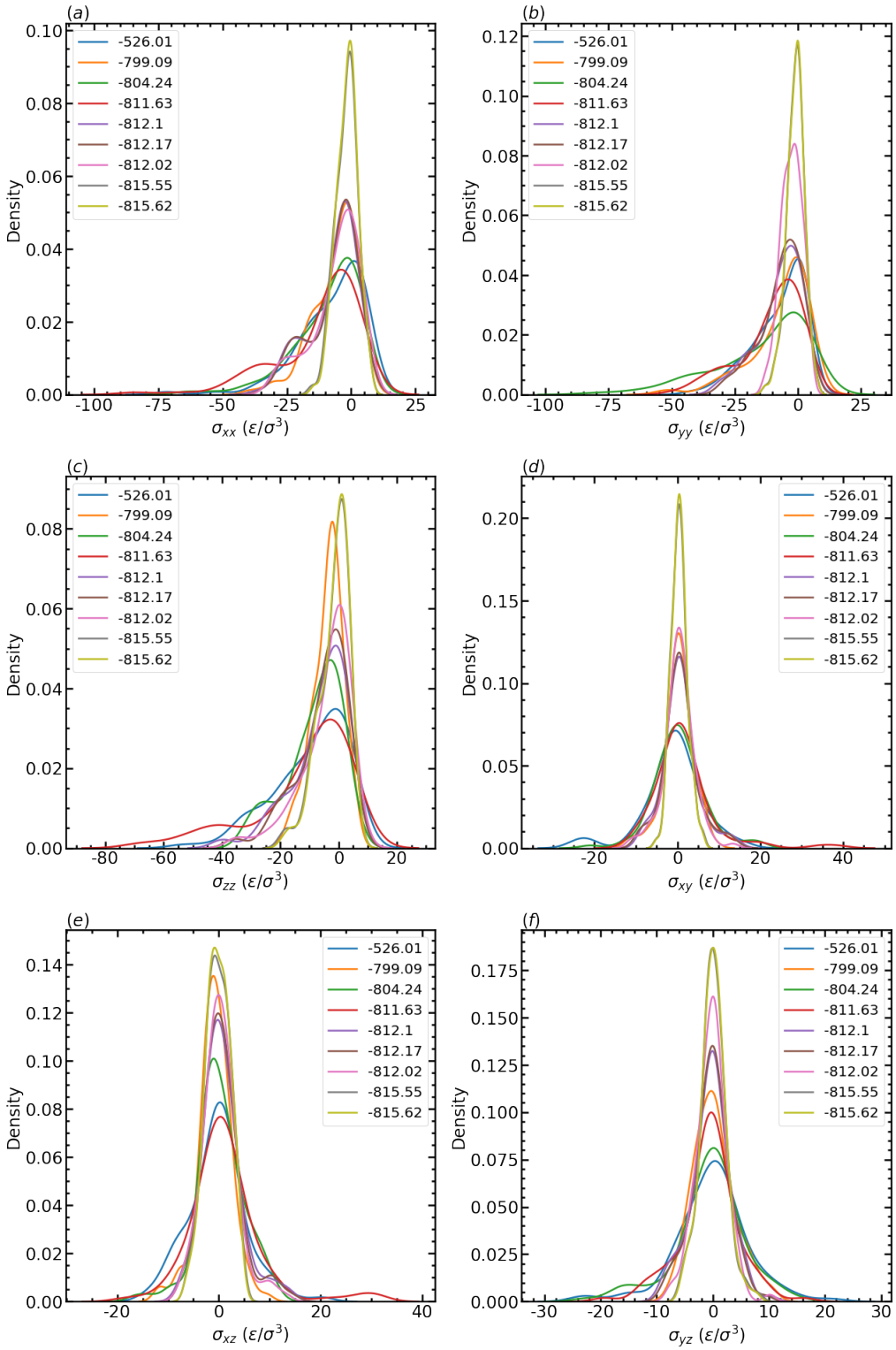

Figure 8. **Stress distribution:** Variation of stress distribution during model adaptation by STRIDERNET on binary LJ system with 100 atoms.$(a)\sigma_{xx}, (b)\sigma_{yy}, (c)\sigma_{zz}, (d)\sigma_{xy}, (e)\sigma_{xz}, (f)\sigma_{yz}$

Hyper-parameters	
PARAMETER	VALUE
Edge embedding size	32
Node embedding size	32
Initial node embedding MLP f_a layers	2
Initial edge embedding MLP f_b layers	2
Edge update MLP layers	2
Node update MLP layers	2
Node displacement MLP layers	4
No. of FGN Layers(L)	2
Activation functions(all MLPs)	Leaky ReLU
Node displacement MLP neighborhood aggregation	Mean

Table 13. Hyper-parameters of FGN

Hyper-parameters	
PARAMETER	VALUE
Hidden layer size	128
No. of hidden layers	6
Activation functions	Leaky ReLU

Table 14. Hyper-parameters of MLP

M. Comparison with baselines at fixed runtime

To ensure fairness of comparison with baselines in terms of running time, we allow the baselines to run for the same time as the StriderNET model adaptation. Figure 9 shows the comparison; StriderNET outperforms the baseline methods.

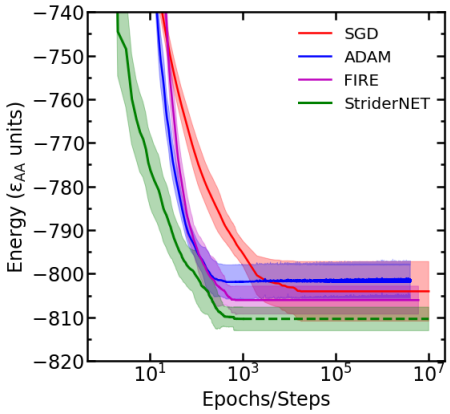

Figure 9. Comparison of STRIDERNET model adaptation on 100 atom Binary LJ system against baselines. All the models are for the same duration (runtime) as STRIDERNET, that is, 3 hours. The dotted line represents the value of STRIDERNET at 1000 epochs.

N. Effect of learning rate on training

We study the effect of learning rate on training the STRIDERNET model. Figure 10 shows the validation curves at different learning rates. At learning rates faster than 10^{-3}, the training is unstable.

O. Effect of Multivariate Gaussian factor α on the training of STRIDERNET

The Multivariate Gaussian constant factor (α)(see Sec. 3.3) controls how far the sampled displacements are from the mean displacement predicted by StriderNET. If α is kept large, the predicted and sampled displacements will have large deviation

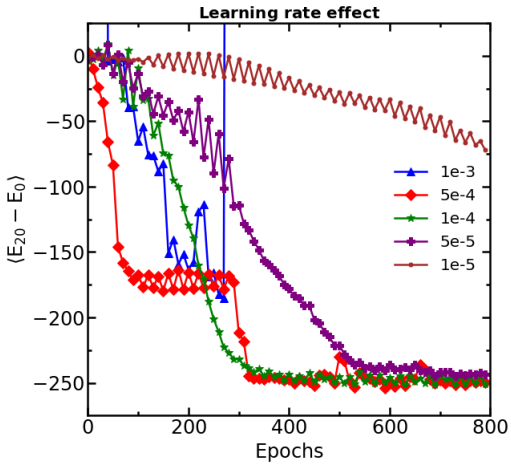

Figure 10. Validation curve of training STRIDERNET at different learning rates for 100 atom Binary LJ system.

and this will reflect in deviation in the corresponding change in energies (ΔE). If the deviation in ΔE is too large, the reward thus obtained will not correspond to the action the model has taken. Thus, it is a tradeoff between exploration and exploitation. We have empirically chosen this value such that the deviation in ΔE due to sampled displacement is no more than $5 - 10\%$.

To demonstrate the effect of α in Multivariate Gaussian we calculate the ΔE first using predicted displacements directly and then using the sampled displacements from the Multivariate Gaussian with given α. The percentage deviation in both the values of ΔE is reported in Table 15.

Effect of α	
α	$\%$ deviation in ΔE
10^{-1}	2590716.57
10^{-2}	2377.76
10^{-3}	408.70
10^{-4}	45.76
10^{-5}	4.64
10^{-6}	0.47

Table 15. Percentage deviation in ΔE at different values of α of Multivariate Gaussian distribution for 100 atom Binary LJ system.