ADAPT: Lightweight, Long-Range Machine Learning Force Fields Without Graphs

Evan Dramko*[1], Yihuang Xiong†[3], Yizhi Zhu[2,3], Geoffroy Hautier[2,3], Thomas Reps[4], Christopher Jermaine[1], and Anastasios Kyrillidis[1]

[1] *Department of Computer Science, Rice University, Houston, TX, USA*
[2] *Department of Nanoengineering, Rice University, Houston, TX, USA*
[3] *Thayer School of Engineering, Dartmouth College, Hanover, NH, USA*
[4] *Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI, USA*

Abstract

Silicon crystals with defects are a primary area of research for the creation of advanced computer chips. Progress in developing these crystals is limited by the time and computational cost required to determine their atomic structures—something accomplished with Density Functional Theory (DFT). Machine-learning force fields (MLFFs) have been introduced as efficient alternatives to first-principles calculations, with most existing works using graph-based neural networks. Such approaches consider atomic interactions in local neighborhoods; in cases like crystalline defects, however, long-range interactions between the bulk crystal and the defect centers can have a substantial effect on structure, motivating architectures that account for interactions across all length scales.

We introduce the *Accelerated Deep Atomic Potential Transformer* or ADAPT, a MLFF capable of accurately modeling the relaxation trajectories of silicon defects. Key to the success of ADAPT is the use of a coordinates-in-space representation of structures rather than representing them with graphs. Individual atoms are treated as "tokens" and a Transformer encoder is employed to model their interactions. We consider a dataset of point defects in silicon, and achieve roughly a 33% reduction in both force and energy prediction error as compared to a current state-of-the-art model, while using only a few percent of the compute.

1 Introduction

First-principles computations offer a powerful way to compute and predict materials and molecular structure and energetics. However, these physics-based approaches have a substantial computational cost. Machine learning force fields (MLFFs)—also referred to as machine learning interatomic potentials (MLIPs)—present a computationally efficient alternative. MLFFs often exhibit runtimes orders of magnitude lower than Density Functional Theory (DFT), making them increasingly considered in materials-discovery pipelines. MLFFs leverage large datasets to build a function approximating the original DFT calculations.

State-of-the-art MLFFs are often graph-based and equivariant neural networks (GNNs) [1, 2], excelling on bulk datasets and many chemistry tasks [3–8]. Additionally, GNNs often excel when

*Corresponding author: ed55@rice.edu
†Present address: School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu, China.

training data is scarce; exactly the situation with expensive DFT trajectories. Recent efforts in further GNN MLFF development have involved the introduction of specialized attention mechanisms [6, 9] and higher-order information in message passing [3].

GNNs have been considered to compute point-defect properties, which are usually simulated on a large periodic supercell with an isolated defect center. The first approaches focused on fitting GNNs to defect-formation energies data [10, 11], but more recent work has used MLFFs to compute forces and accelerate first-principles atomic relaxation [12]. However, challenges in directly applying GNNs to point defects have been raised; for instance, one work [13] suggested modifying GNNs to focus on the local defect region to combat oversmoothing [14]. We also note that defect computations typically involve large supercells of hundred to thousands of atoms, and are computationally demanding for the message-passing algorithms used in GNNs.

Consideration of only local interactions is inherent to graph architectures; however, non-local interactions play a vital role in the structural formation of defects. Recent work [15] showed success on a GNN "one-hop" initial-to-relaxed approach for 2D defects. Such an approach though would require prohibitive amounts of data [16–18] for use in complicated 3D defect trajectories, like those in our dataset. Inspired by the success of Transformers [19] in natural language [20], computer vision [21], and computational biology [22], we explore an alternative to directly handle such relationships: a coordinate-based Transformer with attention computed over all possible atom interactions, trained to predict per-atom forces from raw Cartesian coordinates and atomic features. This new approach is referred to as Accelerated Deep Atomic Potential Transformer (ADAPT), and is trained on a DFT database of defects in silicon, primarily consisting of complex defects. We show that ADAPT achieves state-of-the-art performance (both in energy and forces), outperforming pretrained universal MLFFs, such as MACE [3] and MatterSim [5], as well as MACE retrained on the same data set. Further, ADAPT demonstrates a training cost two orders of magnitude lower than message-passing architectures.

2 Results

In contrast to MACE [3] and related model architectures, ADAPT employs distinct networks for predicting atomic forces and structure energies. As mentioned before, both proposed architectures eschew graphs and inductive biases entirely, instead focusing on precise representations of geometries.

ADAPT adopts the now standard tokenization paradigm [23] from deep learning of breaking inputs into sequences of *tokens*. Here, each token corresponds to a single atom, so a structure with n atoms is represented by n tokens. Every token is initially a 12-dimensional vector containing:

$$(x, y, z, \text{column}, \text{row}, \chi, r_{\text{cov}}, N_{\text{val}}, E_{\text{ion}_1}, E_{\text{EA}}, r_{\text{atom}}, V_{\text{mol}}),$$

where we define x, y, z as the coordinates of the atom, *column* is the atom's group, *row* is the atom's period, χ is the electronegativity, r_{cov} is the covalent radius, N_{val} is the number of valence electrons, E_{ion_1} is the first ionization energy of the atom, E_{EA} is the electron affinity, r_{atom} is the atomic radius, and V_{mol} is the molar volume. These specific descriptors are used because they were naturally present in the raw data.

2.1 Force-Prediction Methodology

We consider the model architecture used to predict per-atom force vectors. It can be viewed as a function mapping each token to a corresponding force vector.

Embedding. Rather than working in the native 12-dimensional space, we embed each token into a higher-dimensional space of size d_{model} (a user-set hyperparameter). A multi-layer perceptron (MLP) [24] learns the transformation:

$$\texttt{MLP}(\mathbf{x}) = \mathbf{W}_k \sigma\Big(\mathbf{W}_{k-1}\,\sigma\big(\dots\sigma(\mathbf{W}_0\mathbf{x}+\mathbf{b}_0)\dots\big)+\mathbf{b}_{k-1}\Big)+\mathbf{b}_k, \tag{1}$$

where $\mathbf{x}\in\mathbb{R}^{12}$ is the input token, σ is the element-wise ReLU operation,[1] and $\mathbf{W}_j\in\mathbb{R}^{d_{\text{out},j}\times d_{\text{in},j}}$ are learnable weight matrices. Here $d_{\text{in},0}=12$, and $\mathbf{W}_k\in\mathbb{R}^{d_{\text{model}}\times d_{\text{out},k-1}}$. The embedding MLP is applied independently to each token.

2.1.1 Transformer Encoder

The embedded sequence is processed by k encoder blocks. Each block has the same structure but distinct parameters. A block is defined by:

$$\mathbf{H}_1 = \texttt{LN}\big(\mathbf{X}_{\text{in}} + \texttt{Attn}(\mathbf{X}_{\text{in}},\mathbf{X}_{\text{in}},\mathbf{X}_{\text{in}})\big), \tag{2}$$

$$\mathbf{H}_2 = \texttt{FFN}\big(\texttt{LN}(\mathbf{H}_1)\big), \tag{3}$$

$$\mathbf{X}_{\text{out}} = \texttt{LN}(\mathbf{H}_2+\mathbf{H}_1). \tag{4}$$

The main components are:

(i) **Layer Normalization (`LN`).** Given an input $\mathbf{x}\in\mathbb{R}^H$, layer norm normalizes across feature channels:

$$\mu = \tfrac{1}{H}\sum_{i=1}^{H} x_i, \qquad \sigma^2 = \tfrac{1}{H}\sum_{i=1}^{H}(x_i-\mu)^2, \tag{5}$$

$$\hat{x}_i = \frac{x_i-\mu}{\sqrt{\sigma^2+\epsilon}}, \qquad y_i = \gamma_i\hat{x}_i+\beta_i, \quad i=1,\dots,H, \tag{6}$$

where $\boldsymbol{\gamma},\boldsymbol{\beta}\in\mathbb{R}^H$ are learnable parameters and ϵ is a small constant for stability.

(ii) **Multi-Head Attention (`Attn`).** Given $\mathbf{X}\in\mathbb{R}^{n\times d_{\text{model}}}$ (sequence length n), each head $i=1,\dots,h$ computes:

$$\text{head}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V}_i, \tag{7}$$

where

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_{\mathbf{Q}_i}, \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_{\mathbf{K}_i}, \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_{\mathbf{V}_i}, \tag{8}$$

with projection matrices $\mathbf{W}_{\mathbf{Q}_i},\mathbf{W}_{\mathbf{K}_i},\mathbf{W}_{\mathbf{V}_i}\in\mathbb{R}^{d_{\text{model}}\times d_k}$. The raw similarity matrix $\mathbf{Q}_i\mathbf{K}_i^{\mathsf{T}}\in\mathbb{R}^{n\times n}$ encodes pairwise token similarities. The row-wise softmax[2] maps each row into a probability distribution over tokens.

Outputs from all heads are concatenated and projected:

$$\texttt{Attn}(\mathbf{X},\mathbf{X},\mathbf{X}) = \text{Concat}(\text{head}_1,\dots,\text{head}_h)\mathbf{W}_O, \tag{9}$$

with $\mathbf{W}_O\in\mathbb{R}^{hd_k\times d_{\text{model}}}$.

[1] $\text{ReLU}(x)=\max(0,x)$

[2] Softmax: $\text{softmax}(\mathbf{z}_i)=\frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{n}e^{\mathbf{z}_j}}$

(*iii*) **Feed-Forward Network (FFN).** A position-wise MLP, applied identically to each token:

$$\texttt{FFN}(\mathbf{H}) = \mathbf{W}_2\ \mathrm{ReLU}\big(\mathbf{W}_1\mathbf{H}^\mathsf{T} + \mathbf{b}_1\big) + \mathbf{b}_2, \tag{10}$$

where

$$\mathbf{H} \in \mathbb{R}^{n\times d}, \quad \mathbf{W}_1 \in \mathbb{R}^{d_{\text{ff}}\times d}, \quad \mathbf{W}_2 \in \mathbb{R}^{d\times d_{\text{ff}}}, \quad \mathbf{b}_1 \in \mathbb{R}^{d_{\text{ff}}}, \quad \mathbf{b}_2 \in \mathbb{R}^{d}.$$

(*iv*) **Dropout.** Dropout randomly masks neuron activations (set to 0), resampled at each pass during training. It is applied to the outputs of attention and feed-forward layers. Following convention, we exclude it from the equations for the model definition since it is only used during training and not inference.

Force Projection. Finally, after the encoder blocks, forces are obtained by a linear projection:

$$\widehat{\mathbf{y}} = \mathbf{X}_{\text{enc}}\mathbf{W}_{\text{out}}, \quad \mathbf{W}_{\text{out}} \in \mathbb{R}^{d_{\text{model}}\times 3}, \tag{11}$$

producing per-token force vectors (f_x, f_y, f_z). The resulting tensor has shape $n \times 3$. Appendix B covers standard Transformer computations in further detail.

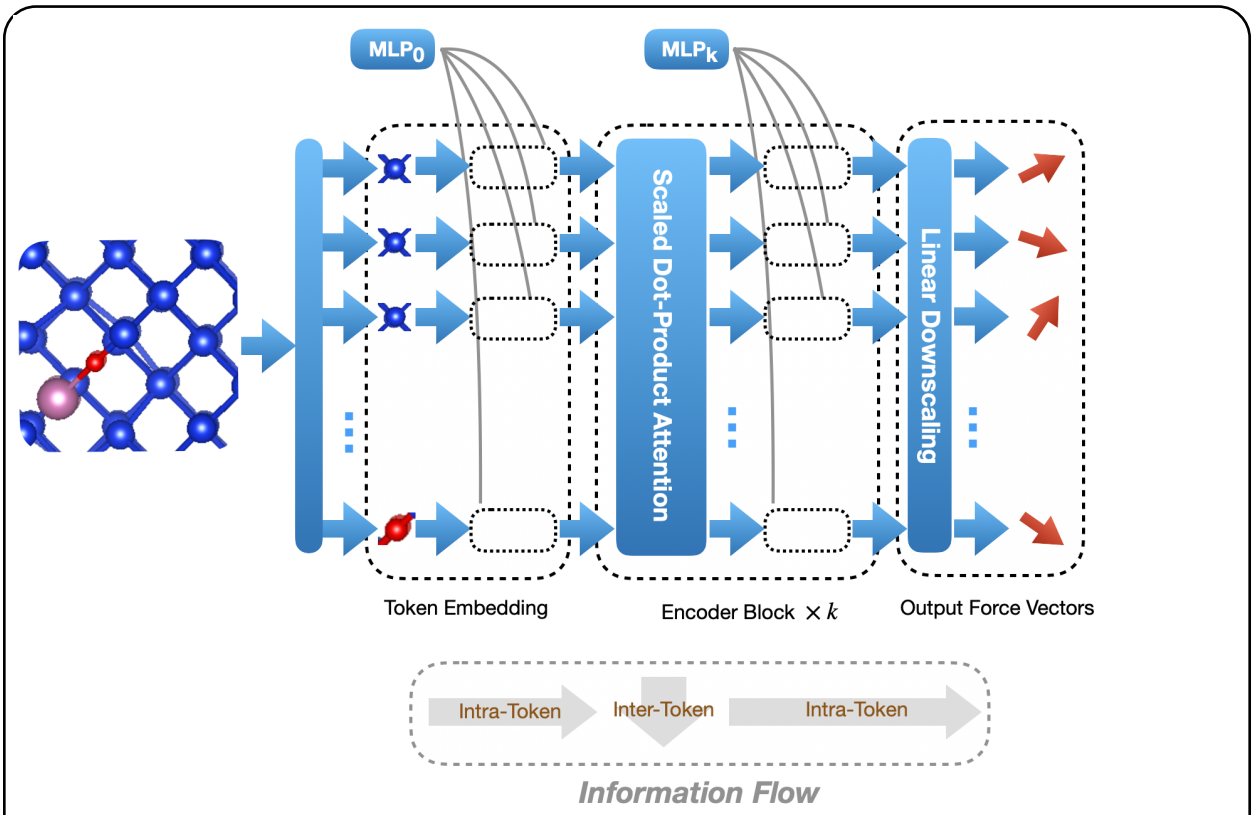


Figure 1: ADAPT architecture. Each of n atoms is embedded via an identically weighted MLP, passed through a stack of Attention-based encoder blocks, and linearly projected from $(n \times d_{\text{model}})$ to $(n \times 3)$ force vectors.

2.1.2 Handling Imbalance in Scaling

In crystalline defects, we see that there is a substantial disparity between the scale of forces in the local area of the defects, and in the bulk lattice. A similar imbalance occurs across atomic feature magnitudes, where certain descriptors (see Section 2.1) differ by several orders of magnitude. Such imbalance in the scale of features is known to cause issues in the training of NNs [25, 26]. This disparity motivates the use of a specialized loss function, as discussed below.

Loss Function. Training requires a differentiable objective that captures the mismatch between predicted and true atomic forces. A natural baseline is the mean-squared error (MSE). Plain MSE, however, does not bias towards any one atom implicitly, even though domain knowledge tells us that atoms nearest the defects dominate the crystal's mechanical response.

To emphasize these critical regions, we introduce a new loss function: "importance-weighted MSE." In particular, we create an importance mask $\mathbf{m} \in \mathbb{R}^n_+$, where each of the n atoms, a_i, receives weight:

$$m_i = \prod_{j \in \mathcal{D}} \Big(1 + \frac{\lambda_1}{\|\mathbf{r}_i - \mathbf{r}_j\|^2 + \lambda_2}\Big), \quad \mathcal{D} = \{\text{dopants}\}, \tag{12}$$

where $\mathcal{D}$ is the set of dopant locations[3], and $\mathbf{r}_i$ is the coordinate vector for atom i. This is similar to laws observed in nature, where the effect of many interactions decay as a power law of the distance between them.[4]It is possible that other weighting rules perform well; we present one that worked well for our training data. Hyperparameters λ_1, λ_2 are used to ensure numerical stability and to "temper" the scaling. The resulting loss becomes:

$$\mathcal{L}(\widehat{\mathbf{y}}, \mathbf{y}) = \sum_i m_i \sum_j (\widehat{y}_{i,j} - y_{i,j})^2$$

Where $\mathbf{y}, \widehat{\mathbf{y}}$ are the actual and predicted forces for each of the atoms (indexed i) and across each of the 3 components of the force vectors (indexed j).

where the force vectors predicted by the model is denoted $\widehat{\mathbf{y}}$, and we have actual force vectors $\mathbf{y}$. While this weighting produces comparable—but often slightly worse—$\mathcal{L}_2$ error as a plain MSE loss function, we find that it performs better when we consider practical use of the network. Section 2.3 details this difference.

2.2 Energy Prediction

We train a separate formation energy-predictor model to complement the MLFF. Three distinct architectures are considered for this task, with us ultimately selecting the third: (1) a decoder, (2) a multilayer perceptron (MLP) (see Section 2.1 for the architecture definition), and (3) an MLP+residual network. In each case, the model receives only the atomic structure and returns an estimated crystal energy. We test architecture (1) because it is the natural extension of the encoder framework to a single output case, architecture (2) has been widely used [27–29] for energy prediction in literature, and architecture (3) substantially outperforms both models.

[3]The formulation used herein does not consider vacancies, but could easily be modified to do so if necessary.

[4]An alternative weighting would be

$$\sum \ln \frac{1}{\|\mathbf{r}_i - \mathbf{r}_j\|^2 + \lambda_2}. \tag{13}$$

We experimented with Eq. (13), but found that for silicon defects Eq. (12) gave better results. It is possible that Eq. (13) would perform better in some applications.

2.2.1 Decoder

The natural extension of using an encoder to predict forces is to use a decoder to predict energy. While the encoder architecture produces a per-token output 2.1.1, the decoder architecture produces individual outputs, like a scalar crystal energy, using a similar Attention/Transformer based architecture. The decoder design we use starts with a stack of encoder layers like in the force-prediction model2.1.1, but instead of the final linear down-scaling, the stack is followed by a decoder head defined as:

$$\begin{aligned}
\mathbf{M} &= \texttt{encoder}(\mathbf{X}); \\
\mathbf{h}_0 &= \texttt{LN}(\mathbf{q} + \texttt{Attn}(\mathbf{q}, \mathbf{M}, \mathbf{M})); \\
\mathbf{h}_1 &= \texttt{LN}(\mathbf{h}_0 + \texttt{MLP}(\mathbf{h}_0)); \\
\widehat{\mathbf{y}} &= \mathbf{W}\mathbf{h}_1 + \mathbf{b},
\end{aligned}$$

where the notation follows that used in Section 2.1.1, and dropout is applied after `Attn` and `MLP`. Recall that $\mathbf{M} \in \mathbb{R}^{n \times d_{model}}$, and note that $\mathbf{W} \in \mathbb{R}^{\mathbb{1} \times \mathbb{n}}$. Although is a matrix of shape $\mathbf{q} \in \mathbb{R}^{(1 \times d_{model})}$ we denote it in lowercase vector form to make clear that it has only one non-trivial dimension. It is used as a "placeholder" token to allow the calculations to shrink the output to a constant size. We train both the encoder and decoder layers jointly.

2.2.2 MLP+Residual Architecture

The architecture of a residual network for raw input tokens $\mathbf{x}$ is:

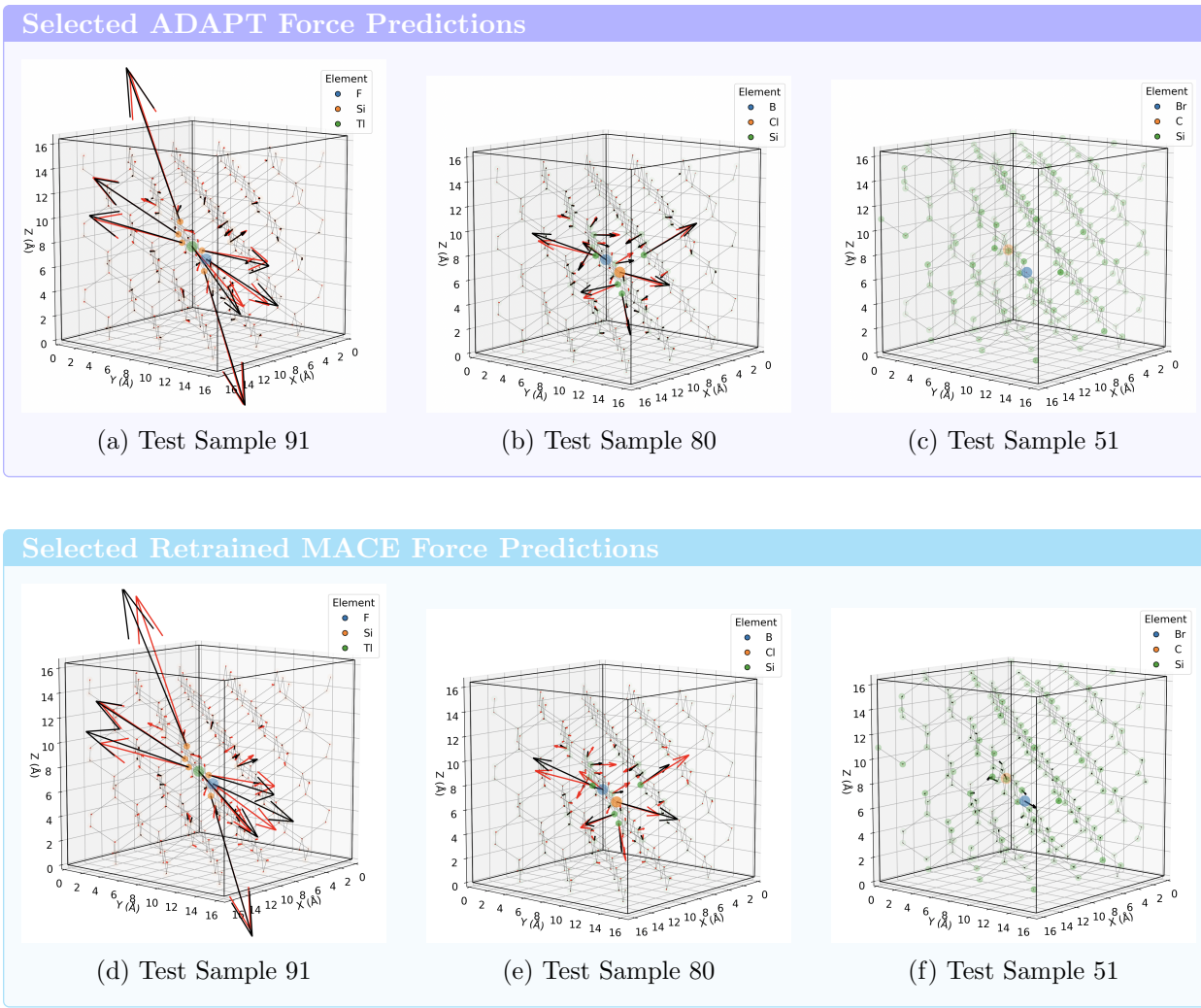
$$\begin{aligned}
\mathbf{t}_0 &= \sigma(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0) \\
\mathbf{h}_0 &= \texttt{LN}(\mathbf{P}_0\mathbf{t}_0 + \mathbf{t}_0) \\
\mathbf{t}_1 &= \sigma(\mathbf{W}_1\mathbf{h}_0 + \mathbf{b}_1) \\
\mathbf{h}_1 &= \texttt{LN}(\mathbf{P}_1\mathbf{t}_1 + \mathbf{t}_1) \\
&\ldots \\
\widehat{\mathbf{y}} &= \mathbf{W}_k\mathbf{h}_k + \mathbf{b}_k
\end{aligned}$$

where $\mathbf{W}_i, \mathbf{P}_i, \mathbf{b}_i$ are learnable weight matrices/vectors of any mathematically valid dimensions. Dropout 2.1.1 is applied after each ReLU activation function σ, and all other notation matches that used in Section 2.1.1. MLPs and MLP+residuals, unlike Transformers, require fixed-length inputs. Based on the structures present in our data, we pad[5] every structure to 220 atoms before feeding it to the network. The selection of 220 atoms stems from the regular Si lattice box in the dataset having $6^3 = 216$ atoms, and a need to allow for the inclusion of dopants. For larger systems, the energy-predictor model can be retrained or fine-tuned with a higher maximum length rather than truncating atoms.

It has been noted that the residual architecture bears striking resemblance to Euler integration [30, 31] making it a common choice [32–34] when considering modeling physical systems which are governed by differential equations.

2.2.3 Selection of Model

[5] "Padding" refers to the creation of dummy atoms where all values are 0.



(a) Test Sample 91 (b) Test Sample 80 (c) Test Sample 51

(d) Test Sample 91 (e) Test Sample 80 (f) Test Sample 51

Figure 2: Side-by-side comparison of outputs. Top row: ADAPT. Bottom row: MACE retrained on the data used to train ADAPT. Predicted forces are shown in black, actual forces are shown in red.

We evaluate between the three energy-prediction architectures by saving the weights for each model where it performed best on the validation set in an initial training of 200 epochs, then evaluating on the test set.

Table 1: Selection Performance

Information	$\mathcal{L}_2$ Error
Decoder	23.5508
MLP Only	50.3728
MLP + residual	11.1683

The results after 200 epochs of training are shown in Table 1. The best performing architecture was MLP+residual, justifying the choice of it as our recommended architecture for energy prediction. After settling on it as our architecture, we find that we can further refine MLP+residual with another 200 epochs of training until we reach convergence.

2.3 Numerical Results

The primary criterion for comparing MLFFs is accuracy in force and energy prediction, typically measured by $\mathcal{L}_2$ or MAE error. We benchmark ADAPT against two state-of-the-art models: MACE

[3] and MatterSim [5]. To ensure comparability, we retrain MACE from scratch on the same dataset used for ADAPT, and treat this as the principal baseline. We additionally report results from previously benchmarked MACE models. For MatterSim, which is positioned as a large-scale foundation model, retraining is computationally prohibitive; we therefore evaluate using its publicly released checkpoints. All models are tested on 100 structures whose trajectories were not included in training.

Recall that the primary motivation for MLFFs is to generate relaxation trajectories. Metrics such as $\mathcal{L}_2$ loss of predicted forces and energies are a proxy used to compare MLFFs, but they are not the main goal. In practice, the decisive measure of MLFF capability is its performance in the metastable structure-determination pipeline, diagrammed in Figure 3. To this end, we do not evaluate on full trajectories because $\mathcal{L}_2$ error can be misleading in the latter steps of crystalline-defect structure relaxation. When atomic forces are near zero, $\mathcal{L}_2$ often favors trivial or uninformative predictions. For example, the zero vector, $\vec{\mathbf{0}}$, can achieve lower error than nontrivial force predictions—even though it is not helpful in practice. This phenomenon occurs because most atoms in the bulk lattice undergo negligible displacement, allowing a model to minimize error by suppressing *all* motion across the lattice, at the cost of missing the subtle, yet critical, displacements that govern structural evolution.

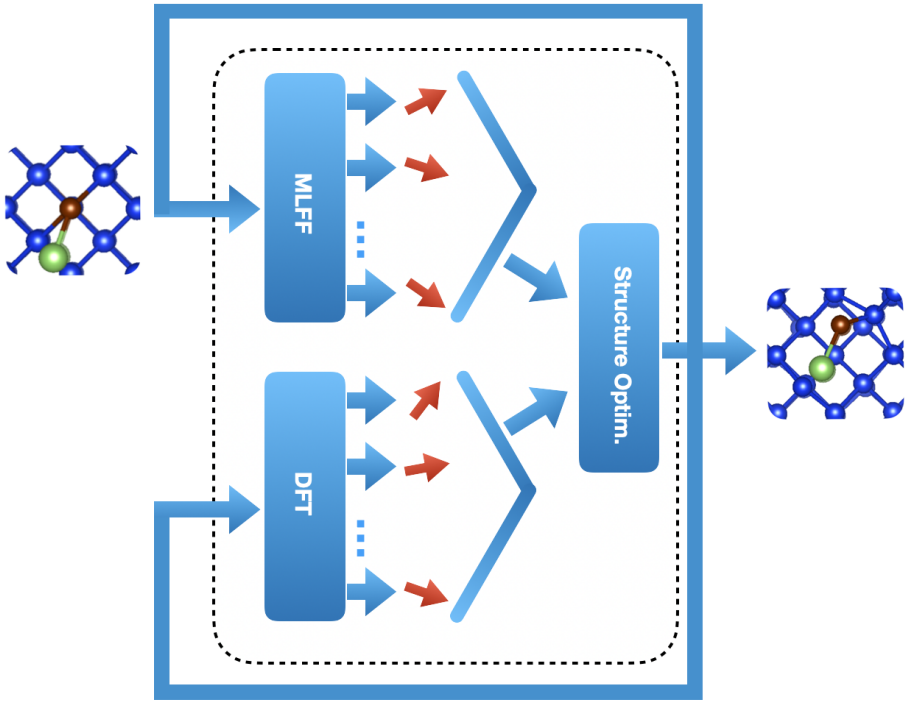


Figure 3: Predictor (Structural Relaxation) Loop

In practice, however, MLFFs and relaxation procedures are often tolerant to small perturbations in the bulk lattice. Predictions typically exhibit small stochastic deviations, yet these are often self-correcting over successive relaxation steps. The practical utility of MLFFs lies in their ability to capture the significant atomic-force vectors that drive structural rearrangements. By evaluating on candidate structures from the beginnings of trajectories rather than full trajectories, the standard $\mathcal{L}_2$ metric better reflects practical utility *for defects*. These initial configurations often contain larger force magnitudes, reducing the advantage of trivial predictions.

Results are summarized in Table 2: ADAPT achieves a 33% error reduction relative to retrained MACE, and far outperforms the strongest pretrained model. Scatter plots of force and energy errors are shown in Figure 4.

Force Predictions. Table 2 shows that the small ADAPT configuration ($d_{\text{model}} = 256$, $d_{\text{ff}} = 512$, 80 epochs) outperforms its larger counterpart ($d_{\text{model}} = 512$, $d_{\text{ff}} = 1024$, 750 epochs). The larger configuration exhibited overfitting, indicating that the smaller model already distilled nearly all available information from the data. Accordingly, no further model training on the same inputs is likely to achieve a meaningful performance gain[6].

Energy Predictions. We also show that the ADAPT energy-predictor model produces performance superior to both MACE and MatterSim. A table of results is given as Table 2, and scatter plots showing the results are given in Figure 5. We achieve near identical error to MatterSim 5M—the best of the existing energy predictors—after 200 epochs, and reach our final result—with a better than 30% reduction in MAE error over MatterSim 5M—after 400 epochs.

2.4 Computational Efficiency

Force Predictions. An advantage of the ADAPT architecture is its computational efficiency. Training Small ADAPT required approximately 2.24 minutes per epoch on a single NVIDIA A100, and converged after 80 epochs (totaling 3 compute hours). In comparison, retraining MACE required 8.5 minutes per epoch for 300 epochs on 16 NVIDIA A100s, amounting to 680 compute hours: more than 227× the amount of compute used to train ADAPT's force-prediction model. The compact design of ADAPT permits training on commodity hardware, including workstations and even consumer-grade laptops equipped with GPUs,[7] thereby significantly reducing hardware requirements for adoption. This accessibility is consistent with the overarching objective of the MLFF literature: to accelerate structural determination by reducing dependence on large-scale computational resources.

These improvements are attributed to the departure from graph-based architectures. Graph neural networks inherently involve sparse operations, which are not easily expressed in the dense linear algebraic form favored by modern accelerators. Consequently, graph-based models typically exhibit lower hardware utilization due to sparse operations, which lack the extensive optimization and backend support available with dense-matrix operations [35]. By forgoing graph representations and adopting architectural paradigms widely developed in natural-language processing and computer vision—where such operations benefit from extensive backend and library support—ADAPT achieves markedly higher computational throughput.

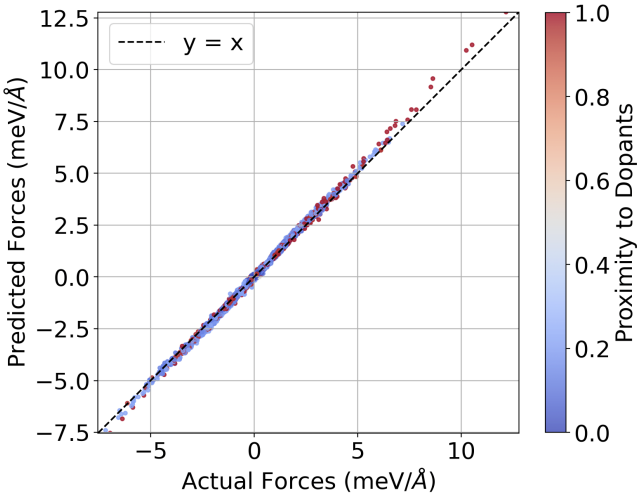
Energy Prediction. MACE generates energy predictions concurrently with force predictions within the same forward pass, yielding identical timing characteristics for both quantities. ADAPT trains an additional energy-predictor model, which required 1.93 compute hours on a single NVIDIA A100 GPU. Model training was conducted for 400 epochs, with the duration of a single epoch being 29 seconds on the same hardware. When including this cost, training both ADAPT models takes a total of 4.92 A100 hours, which is still more than 138× faster than MACE.

[6] Under the assumption of no additional inductive biases.

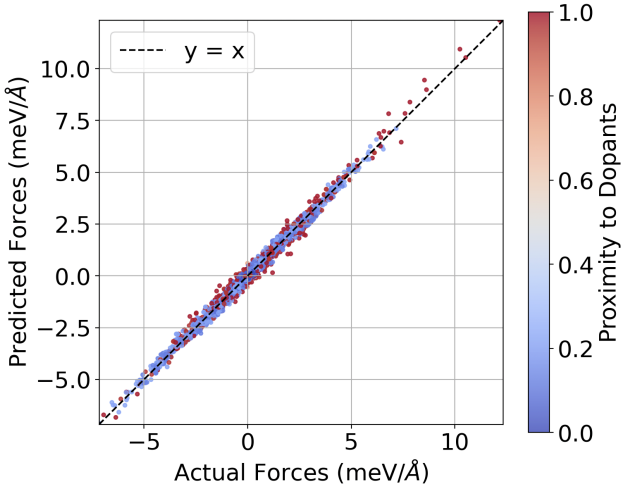
[7] The authors successfully trained Small ADAPT on a personal laptop.

Table 2: Comparison With State-Of-The-Art On 100 Test Structures

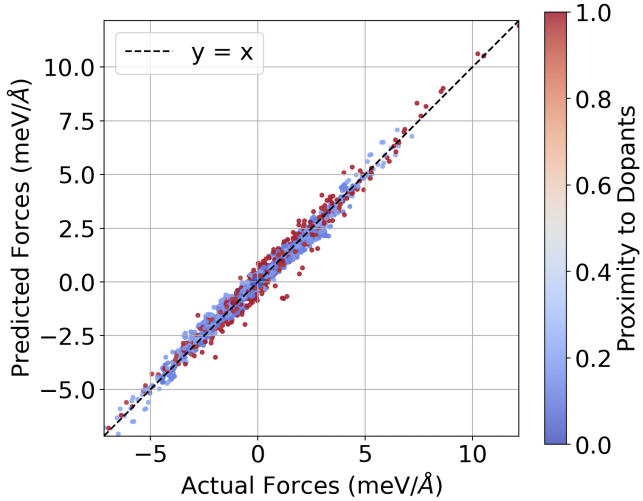
Architecture	Total Force $\mathcal{L}_2$ Error	Component Force $\mathcal{L}_2$ Error	Energy MAE Error (eV/structure)
ADAPT Small	**8.1165**	**0.00013**	**0.5782**
ADAPT Large	12.3891	0.00019	–
MACE Retrained	12.5908	0.00019	1.3129
MACE MP0a Large	43.1345	0.00066	6.1012
MACE MPA-0 Medium	27.0463	0.00041	2.0478
MACE OMAT-0 Medium	18.7016	0.00028	3.2232
MatterSim 1M	22.2803	0.00034	1.7430
MatterSim 5M	23.7851	0.00036	0.8289



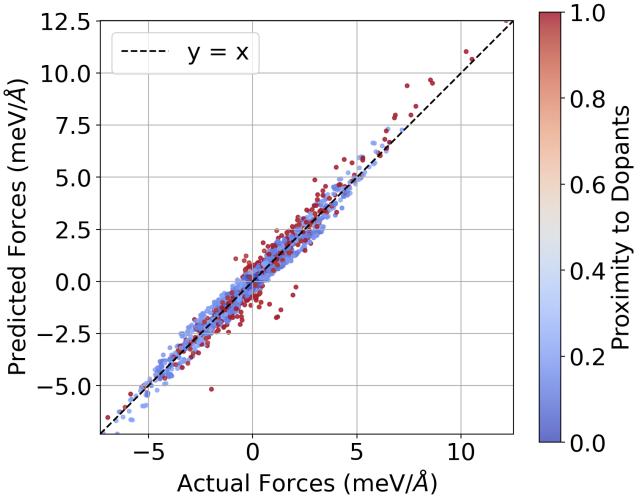
(a) ADAPT Model



(b) Retrained MACE

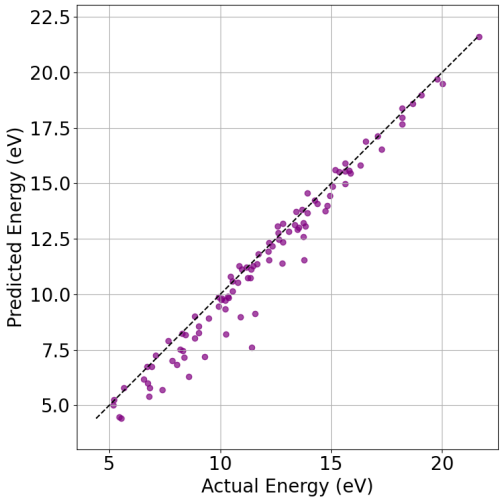


(c) MACE OMAT-0 Medium

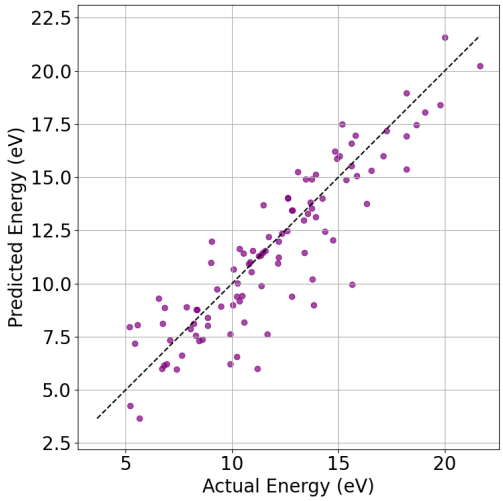


(d) Pretrained MatterSim 5M

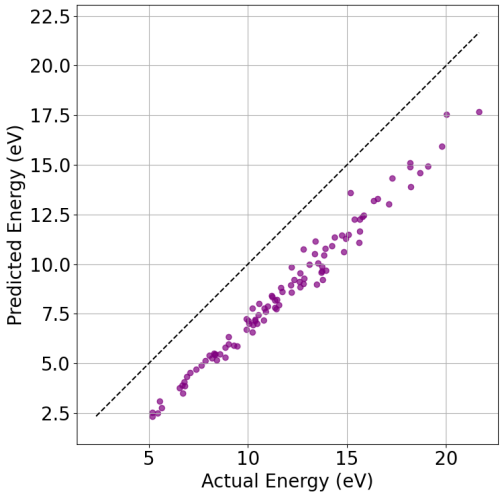
Figure 4: Scatter plots of predicted vs. actual forces across test structures. Adherence to the line $y = x$ is ideal.



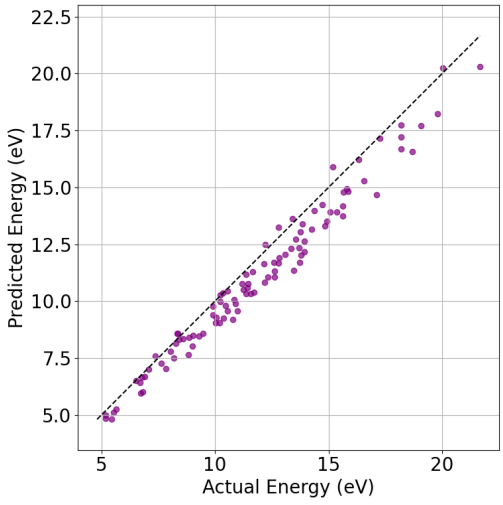
(a) Small ADAPT Model



(b) Retrained MACE



(c) MACE OMAT-0 Medium



(d) Pretrained MatterSim 5M

Figure 5: Scatter plots of predicted vs. actual energies across test structures. Adherence to the line $y = x$ is ideal.

3 Discussion

On the Use of Separate Models. ADAPT employs separate models for force and energy prediction, a design choice that carries several practical advantages. First, when only one quantity is required, the corresponding model can be deployed independently, reducing both runtime and memory consumption. This efficiency is relevant for practitioners working on local workstations or clusters with limited hardware capacity. Second, the separation increases modularity: force and energy predictors can be updated or retrained independently, allowing the integration of datasets without both quantities present, and enabling incremental model refinements without retraining the entire system.

This modularity, however, comes at a cost. Some applications —such as the FIRE optimizer [36]— require both forces and energies simultaneously. In such cases, a joint model is often more parameter-efficient [37] because it enables learning a shared representation across tasks. Moreover, multi-task formulations can leverage the physical correlations between forces and energies, potentially improving generalization when sufficient training data is available.

Architectural considerations also play a role in the two-model system. Unlike conventional neural networks, which allow outputs to be flexibly defined, Transformer architectures are inherently structured around token-to-token transformations. In ADAPT, where tokens correspond to atoms, the energy of the structure constitutes a non-token, global output. Accommodating this mismatch requires additional mechanisms. Extensive prior literature on this issue has yielded two main strategies: *i*) the introduction of "special" tokens representing global properties [38, 39], and *ii*) the use of specialized output heads appended to the model [40].

Given the limited training data available for silicon defects, it is not surprising [41–43] that a simpler MLP with residual connections outperformed a Transformer decoder in this setting—see Table 1. Nonetheless, the authors expect that, with sufficient force and energy data, Transformer architectures augmented with specialized heads may provide a more scalable and accurate solution. The design of such heads remains an active area of research, and identifying architectures that best balance modularity, efficiency, and accuracy is an open problem.

Coordinates vs. Graphs. Graph neural networks (GNNs) are the default backbone for modern MLFFs [3–6, 8] where atoms define nodes, and atomic bonds or proximity determine edge placement. By encoding geometric priors (permutation, rotation, and translation invariance), they incorporate strong inductive biases that improve data efficiency [1, 44–46] and have been argued to stabilize relaxation trajectories [9].

Representing continuous atomic interactions using discrete graph topologies introduces mismatches that can limit accuracy, especially in defects where long-range effects and precise geometries are important. GNNs inherently restrict interactions to local regions, relying on network depth to propagate forward information that is outside the interaction radius. This approach often leads to over-smoothing and over-squashing [47, 48], where long-range signals degrade rapidly as depth increases. Bulk crystal far from the defect core can substantially shape local defect structures. While long-range influences are less critical in many other chemical systems, neglecting them in crystalline materials can cause large errors. The poor performance of GNNs on large periodic systems—an issue especially relevant in modeling crystalline defects—has been noted [9, 13]. Adding long-range interactions into graph architectures [6, 9] often leads to significant cost in computation and model complexity. Thus, we arrive at the motivation for using an alternative MLFF strategy for modeling crystal defects in ADAPT, and a need recognized in [9, 13] as well.

With the advent of Transformer architectures and growing datasets, it is now feasible to move away from hard-coded geometric priors and instead focus on explicit representations of global distances and angles. ADAPT employs a Transformer encoder (Section 2.1, Appendix B) with full, unmasked self-attention, enabling *all-to-all* comparisons between atoms at each layer. This approach directly captures non-bonded and long-range interactions without depending on depth-based message passing. Although the model lacks explicit geometric equivariances, permutation invariance is inherent to unmasked attention, and experiments show that translational and rotational invariances can be learned sufficiently well from data. The importance of global attention is underscored in Table 3: restricting attention to local neighborhoods—as in GNNs—drastically degrades performance.

Table 3: Full vs Local Interaction.

Allowed Interactions (%)	Total $\mathcal{L}_2$ Loss
1.46	13.16*
18.7	13.61†
51.3	11.13†
100	8.11*

Radius is the percentage of every-to-every interactions allowed during training and inference. Interactions are controlled in Attention via Key-Structural Masks (Appendix C). Lower scores mean less error.
Note: * training converged after 80 epochs; † training ran for 200 epochs until convergence.

Accurate Representation of Geometries. Graphs excel at capturing connectivity, but do not inherently encode exact distances or angles. To handle this deficiency, many GNN variants supplement node and edge features with geometric data [1, 3, 4, 8, 9]; however, such information must still be passed iteratively from neighbor to neighbor, which can introduce truncation and discretization errors—an effect that compounds with increasing path lengths between atoms.

By contrast, a coordinate-based approach gives direct access to precise pairwise distances and angles for all atoms in a single computation step. This approach not only avoids approximations from multi-hop propagation, but also preserves geometric detail across all interaction scales.

Limitations and Future Directions. The ADAPT architecture is not inherently limited to defect prediction. However, it remains an open problem to determine ADAPT's applicability to diverse chemical structures. Additionally, Transformers typically require substantial quantities of data [41–43], making ADAPT unsuitable for tasks with limited training data.

Future directions include *i*) enforcing physical invariances algorithmically within both the architecture and the loss; *ii*) extending training beyond silicon to encompass a wider class of defects and materials; and *iii*) developing holistic force-field models that integrate broader physical context to further improve predictive fidelity.

3.0.1 First-principles simulation of defect trajectories

The DFT trajectories dataset contains both simple and complex defect in silicon, which correspond to the our previous work [49, 50]. The complex defects are in substitutional-interstitial configuration. The defect elements in the dataset spans most of the periodic table besides the noble gas, rare-earth, and the ones that are difficult to implantable, giving in total 56 elements [50]. In this work, we only focus on the charge neutral defect and extract 264,627 number of single-point calculations from the relaxation trajectories. The high-throughput defect computations were performed using the automatic workflows that are implemented in atomate software package [51–53]. The first-principles calculations were performed using Vienna Ab-initio Simulation Package (VASP) [54, 55] with the projector augmented wave (PAW) method [56]. All the calculations were spin-polarized at the Perdew-Burke-Erzhenhoff (PBE) level. Defect atoms are embedded in a Si supercell with 216 atoms. 520 eV cutoff energies were used for the plane-wave basis and the Brillouin zone was

sampled with single Γ. All the defect structures were optimized at a fixed volume until the ionic forces are smaller than 0.01 eV/Å.

4 Acknowledgments and Availability

4.1 Code and Data Availability

The datasets generated and/or analyzed during the current study are available in the [insert name of repository] repository, [insert persistent URL to datasets].
The underlying code and training/validation datasets for this study are available in the GitHub repository: ADAPT-released and can be accessed via this link [released after publication].

4.2 Acknowledgments

This study was funded by NSF grant CCF-2212558. The funder played no role in study design, data collection, analysis and interpretation of data, or the writing of this manuscript. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors, and do not necessarily reflect the views of the sponsoring entities.

4.3 Competing Interests

All authors declare no financial or non-financial competing interests.

References

1. Bronstein, M. M., Bruna, J., Cohen, T. & Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021).
2. Reiser, P. *et al.* Graph neural networks for materials science and chemistry. *Communications Materials* **3,** 93 (2022).
3. Batatia, I., Kovacs, D. P., Simm, G., Ortner, C. & Csányi, G. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in neural information processing systems* **35,** 11423–11436 (2022).
4. Deng, B. *et al.* CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence* **5,** 1031–1041 (2023).
5. Yang, H. *et al.* Mattersim: A deep learning atomistic model across elements, temperatures and pressures. *arXiv preprint arXiv:2405.04967* (2024).
6. Frank, J. T., Unke, O. T., Müller, K.-R. & Chmiela, S. A Euclidean transformer for fast and stable machine learned force fields. *Nature Communications* **15,** 6539 (2024).
7. Poltavsky, I. & Tkatchenko, A. Machine learning force fields: Recent advances and remaining challenges. *The journal of physical chemistry letters* **12,** 6551–6564 (2021).
8. Chen, C. & Ong, S. P. A universal graph deep learning interatomic potential for the periodic table. *Nature Computational Science* **2,** 718–728 (2022).
9. Frank, J. T., Unke, O. T. & Müller, K.-R. So3krates: Equivariant attention for interactions on arbitrary length-scales in molecular systems. *arXiv preprint arXiv:2205.14276* (2022).
10. Rahman, M. H. *et al.* Accelerating defect predictions in semiconductors using graph neural networks. *APL Machine Learning* **2** (2024).
11. Xiang, X., Soh, D. & Dunham, S. Exploration of deep learning models for accelerated defect property predictions and device design of cubic semiconductor crystals. *The Journal of Physical Chemistry C* **128,** 8821–8829 (2024).
12. Mosquera-Lois, I., Kavanagh, S. R., Ganose, A. M. & Walsh, A. Machine-learning structural reconstructions for accelerated point defect calculations. *npj Computational Materials* **10,** 121 (2024).
13. Yan, Q., Kar, S., Chowdhury, S. & Bansil, A. The case for a defect genome initiative. *Advanced Materials* **36,** 2303098 (2024).
14. Li, Q., Han, Z. & Wu, X.-M. *Deeper insights into graph convolutional networks for semi-supervised learning* in *Proceedings of the AAAI conference on artificial intelligence* **32** (2018).
15. Yang, Z. *et al.* Modeling crystal defects using defect informed neural networks. *npj Computational Materials* **11,** 229 (2025).
16. Lopez-Rojas, A. D. & Cruz-Villar, C. A. Neural networks as an approximator for a family of optimization algorithm solutions for online applications. *Neural Computing and Applications* **36,** 3125–3140 (2024).
17. Amos, B. *Tutorial on amortized optimization* 2025. arXiv: `2202.00665 [cs.LG]`. `https://arxiv.org/abs/2202.00665`.
18. Qiu, R., Sun, Z. & Yang, Y. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems* **35,** 25531–25546 (2022).

19. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

20. Zhou, C. *et al.* A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *International Journal of Machine Learning and Cybernetics,* 1–65 (2024).

21. Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

22. Abramson, J. *et al.* Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630,** 493–500 (2024).

23. Webster, J. J. & Kit, C. *Tokenization as the initial phase in NLP* in *COLING 1992 volume 4: The 14th international conference on computational linguistics* (1992).

24. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* **2,** 303–314 (1989).

25. Khakhar, A. & Buckman, J. Neural regression for scale-varying targets. *arXiv preprint arXiv:2211.07447* (2022).

26. Lee, J.-H., Lee, C. & Kim, C.-S. *Learning multiple pixelwise tasks based on loss scale balancing* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 5107–5116.

27. Jha, D. *et al.* Elemnet: Deep learning the chemistry of materials from only elemental composition. *Scientific reports* **8,** 17593 (2018).

28. Liang, Y. *et al.* A universal model for accurately predicting the formation energy of inorganic compounds. *Science China Materials* **66,** 343–351 (2023).

29. Zhang, L., Han, J., Wang, H., Car, R. & E, W. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters* **120,** 143001 (2018).

30. Müller, J. On the space-time expressivity of ResNets. *arXiv preprint arXiv:1910.09599* (2019).

31. Baggenstos, J. & Salimova, D. Approximation properties of residual neural networks for Kolmogorov PDEs. *arXiv preprint arXiv:2111.00215* (2021).

32. Moghaddam, M. M., Parand, K. & Kheradpisheh, S. R. Advanced Physics-Informed Neural Network with Residuals for Solving Complex Integral Equations. *arXiv preprint arXiv:2501.16370* (2025).

33. Noorizadegan, A., Cavoretto, R., Young, D.-L. & Chen, C.-S. Stable weight updating: A key to reliable PDE solutions using deep learning. *Engineering Analysis with Boundary Elements* **168,** 105933 (2024).

34. Kashinath, K. *et al.* Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A* **379,** 20200093 (2021).

35. Liang, S. *et al.* EnGN: A high-throughput and energy-efficient accelerator for large graph neural networks. *IEEE Transactions on Computers* **70,** 1511–1525 (2020).

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

37. Zhang, Y. & Yang, Q. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering* **34,** 5586–5609 (2021).

38. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* 2019. arXiv: 1810.04805 [cs.CL]. https://arxiv.org/abs/1810.04805.

39. Alayrac, J.-B. *et al. Flamingo: a Visual Language Model for Few-Shot Learning* 2022. arXiv: 2204.14198 [cs.CV]. https://arxiv.org/abs/2204.14198.

40. Ouyang, L. *et al. Training language models to follow instructions with human feedback* 2022. arXiv: 2203.02155 [cs.CL]. https://arxiv.org/abs/2203.02155.

41. Liu, Y. *et al.* Efficient training of visual transformers with small datasets. *Advances in Neural Information Processing Systems* **34,** 23818–23830 (2021).

42. Zhu, H., Chen, B. & Yang, C. Understanding why vit trains badly on small datasets: An intuitive perspective. *arXiv preprint arXiv:2302.03751* (2023).

43. Zhang, Y., Warstadt, A., Li, H.-S. & Bowman, S. R. When do you need billions of words of pretraining data? *arXiv preprint arXiv:2011.04946* (2020).

44. Ko, T. W. & Ong, S. P. Data-efficient construction of high-fidelity graph deep learning interatomic potentials. *npj Computational Materials* **11,** 65 (2025).

45. Kiechle, J. *et al. Graph Neural Networks: A Suitable Alternative to MLPs in Latent 3D Medical Image Classification?* in *International Workshop on Graphs in Biomedical Image Analysis* (2024), 12–22.

46. Oliva, M., Banik, S., Josifovski, J. & Knoll, A. *Graph Neural Networks for Relational Inductive Bias in Vision-based Deep Reinforcement Learning of Robot Control* 2022. arXiv: 2203.05985 [cs.LG]. https://arxiv.org/abs/2203.05985.

47. Giraldo, J. H., Skianis, K., Bouwmans, T. & Malliaros, F. D. *On the trade-off between over-smoothing and over-squashing in deep graph neural networks* in *Proceedings of the 32nd ACM international conference on information and knowledge management* (2023), 566–576.

48. Rusch, T. K., Bronstein, M. M. & Mishra, S. *A Survey on Oversmoothing in Graph Neural Networks* 2023. arXiv: 2303.10993 [cs.LG]. https://arxiv.org/abs/2303.10993.

49. Xiong, Y. *et al.* Computationally Driven Discovery of T Center-like Quantum Defects in Silicon. *Journal of the American Chemical Society* **146,** 30046–30056 (Nov. 2024).

50. Xiong, Y. *et al.* High-throughput identification of spin-photon interfaces in silicon. *Science Advances* **9,** eadh8617. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.adh8617. https://www.science.org/doi/abs/10.1126/sciadv.adh8617 (2023).

51. Jain, A. *et al.* Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL materials* **1,** 11002 (2013).

52. Mathew, K. *et al.* Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Computational Materials Science* **139,** 140–152. ISSN: 0927-0256. https://www.sciencedirect.com/science/article/pii/S0927025617303919 (2017).

53. Ong, S. P. *et al.* Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science* **68,** 314–319. ISSN: 0927-0256. https://www.sciencedirect.com/science/article/pii/S0927025612006295 (2013).

54. Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54,** 11169–11186. https://link.aps.org/doi/10.1103/PhysRevB.54.11169 (16 Oct. 1996).

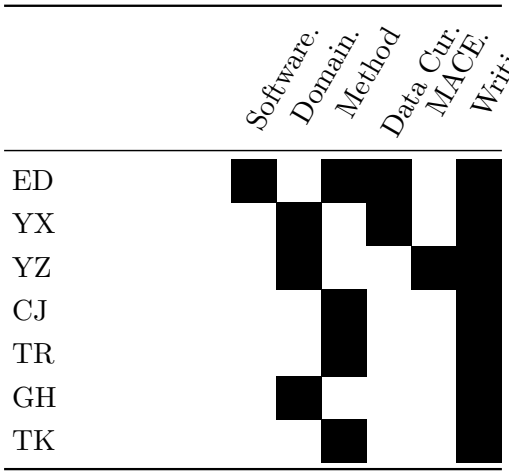
55. Kresse, G. & Furthmüller, J. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science* **6,** 15–50. ISSN: 0927-0256. https://www.sciencedirect.com/science/article/pii/0927025696000080 (1996).

56. Blöchl, P. E. Projector augmented-wave method. *Phys. Rev. B* **50,** 17953–17979. https://link.aps.org/doi/10.1103/PhysRevB.50.17953 (24 Dec. 1994).

57. Deng, J., Guo, J., Xue, N. & Zafeiriou, S. *Arcface: Additive angular margin loss for deep face recognition* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 4690–4699.

A Individual Contributions

Table 4: Author contributions by role (filled = contributed)



Roles: Software.=Creation of project software and documentation; Domain.=Domain Knowledge; Method. = Design of MLFF architecture; Data Cur.=Data Curation; MACE=Training of MACE; Writing=Writing and Editing.

B Architecture Details and Hyperparameters

Transformer Details. A full writeup of the mathematics of Scaled Dot-Product Attention and Transformers can be found at the following links:

- Attention: https://evandramko.github.io/files/attention.pdf
- Transformers: https://evandramko.github.io/files/transformer.pdf

Hyperparameters.

- ADAPT: We define the "small" model size by: [$d_{model} = 256$, $d_{ff} = 512$, #-layers= 8, #-heads= 8, dropout rate = 0.05] trained for 80 epochs. The "large" model size is: [$d_{model} = 512$, $d_{ff} = 1024$, #-layers= 8, #-heads= 8, dropout rate = 0.05] trained for 750 epochs.
- MACE: The retrained version of MACE (v0.3.14, PyTorch 2.6.0) uses: num_interactions=2, num_channels=256, max_L=2, correlation=3, r_max=5.0, trained for 300 epochs on single precision (float32).

B.1 Evaluation At Different Levels

While $\mathcal{L}_2$ error is the conventional standard for comparing force predictions, we find that it is insufficient to fully capture the dynamics of point defects in crystals. To perform a more appropriate comparison, we use two complementary levels. (i) *Model level* (MLFF): accuracy of force and energy predictions. (ii) *Predictor level*: quality of the final relaxed structure obtained by running a geometry optimizer with the MLFF.

Model-Level Evaluation of Forces: When comparing candidate models, in addition to the loss scores (see Section 2.1.2), we also consider the average angle and magnitude errors separately. We use the dot product to calculate the angular error in degrees via[8]

$$angle(\mathbf{y}, \widehat{\mathbf{y}}) = \arccos\left(\frac{\mathbf{y} \cdot \widehat{\mathbf{y}}}{||\mathbf{y}||_2 \cdot ||\widehat{\mathbf{y}}||_2}\right) \cdot \frac{180}{\pi},$$

and we calculate the difference in magnitudes via

$$mag(\mathbf{y}, \widehat{\mathbf{y}}) = |\|\mathbf{y}\|_2 - \|\widehat{\mathbf{y}}\|_2|$$

These results help to determine whether the model is genuinely learning the underlying dynamics or artificially minimizing error by predicting uniformly negligible forces—knowing that in reality, most of them will be close to zero.[9] From a domain perspective, it is often more important to predict the direction (angle) of the force correctly than its exact magnitude. Although this angular-magnitude metric is differentiable and theoretically usable as a loss function for the MLFF, in practice it is difficult to balance the angular and magnitude components effectively. Empirical results show that angular-loss functions are often brittle and require significant engineering effort to implement reliably [57]—a result borne out in our own experiments. In contrast, using a weighted mean-squared-error (MSE) loss is simpler, more robust, and yields strong performance at both the MLFF and Predictor (Structural-Relaxation) levels, making it the preferred choice. However, we did use the angle-prediction performance of models to compare and rank different training runs and different hyperparameter choices for our models.

Evaluation of Energy: The total energy of the crystal is represented with a single number, making evaluation very easy. We use the common $\mathcal{L}_2$ distance metric.

Evaluation of Predictor (Figure 3): In order to evaluate the final result of the full relaxation procedure, we use the well known SOAP and delta Q metrics. Other checkers (such as those which check bond lengths) are also viable, although we do not use them in this work.

C Masking in Attention

When restricting interactions in Attention, we apply *masks* to the attention logit matrix

$$QK^{\mathsf{T}} \in \mathbb{R}^{B \times H \times T \times T},$$

where B is the batch size, H the number of heads, and T the sequence length (number of tokens). Masking is applied along the **Key dimension** (the columns), so that certain tokens cannot be attended to. We use two types of masks:

1. **Padding mask.** To enable batching, all sequences are padded.[10] Padding tokens must not affect the model's output, so we mask them out of the attention computation.

2. **Restricted visibility (local radius).** To study the effect of limiting each token's visible neighborhood, we compute a restricted attention mask. Allowed interactions are precomputed from the $\mathcal{L}_2$ distances between raw coordinates, and then the same mask is applied to every attention step in the forward pass.

[8] In practice, we clamp the $\cdot$ $\arccos(\cdot)$ to ensure that arccos is always operating on valid values. This detail is omitted for clarity in the provided formula.

[9] In practice, many implementations of different models tended to produce near-zero results for all forces, and then stop improving.

[10] Padding means appending dummy tokens, typically all zeros, to make every sequence the same length.

Key masking mechanism. After computing $QK^\top$, all disallowed positions are replaced with `-inf`. During the row-wise softmax, these entries become zero, ensuring that they cannot contribute, regardless of the values in V. Consequently, masked tokens never influence the update of valid tokens. Query values at masked positions can be arbitrary ("nonsense" numbers),[11] but they cannot affect non-padded tokens.

[11] Some implementations explicitly zero them out after each attention layer for safety and clarity.