Implementing the CGnets Deep-Learning Coarse-Grain Approach in the Julia Programming Language

Ananya Venkatachalam
Harvey Mudd College
(Project partner: Paco Navarro)
(Dated: May 5, 2025)

Link to GitHub repository where all code is located.

I. INTRODUCTION

In molecular simulations of biomolecules, fully atomistic models provide detailed insights but remain limited by computational expense, restricting access to biologically relevant timescales and length scales. To address this challenge, coarse-grained (CG) models reduce the degrees of freedom by grouping atoms into effective interaction sites, allowing longer simulations of larger systems. Seminal approaches to CG modeling, such as the bottom-up, multiscale CG approaches practiced by the Voth Group [1] and the Martini force field [2], have demonstrated success in reproducing structural and thermodynamic properties while making strong assumptions about the form of interactions.

However, traditional CG models often rely on fixed functional forms and pairwise potentials, limiting their ability to capture complex many-body interactions and high-dimensional free energy landscapes. In recent years, machine learning (ML) has emerged as a flexible and data-driven alternative capable of learning CG potentials directly from simulation data with fewer assumptions. Early ML-based contributions to the field of simulations include the use of deep neural networks [3] to model potential energy surfaces.

The CGnet framework introduced by Wang et al. in 2019 [4] represents a significant step forward in this direction, formulating CG force field learning as a supervised learning problem. CGnet learns to approximate the potential of mean force (PMF) by training neural networks to match forces derived from atomistic simulations, naturally incorporating many-body correlations and nonlinearities. Building on this concept, other works such as TorchMD-NET [5] have expanded the capabilities of ML-based CG models by leveraging attention mechanisms and transformers.

In this project, we implement a simplified version of CGnet, originally coded entirely in Python, in the Julia programming language. We train a neural network to learn the force field for a 2D toy potential using force matching. We demonstrate that a simplified form of the model not only captures the force landscape but is also able to reconstruct the associated free energy profile.

II. BACKGROUND

In classical molecular dynamics (MD), accurately capturing the thermodynamics and kinetics of molecular systems requires atomistic resolution, where every atom and its interactions are explicitly simulated. While this level of detail provides valuable insight, it comes at a high computational cost, restricting simulations to relatively short timescales (nanoseconds to microseconds) and small system sizes (nanometers). As shown in Fig. 1, atomistic MD occupies only a small region of the broader multiscale modeling landscape. To reach biologically or materially relevant regimes—such as microsecond protein folding events or mesoscale self-assembly—alternative methods are required.

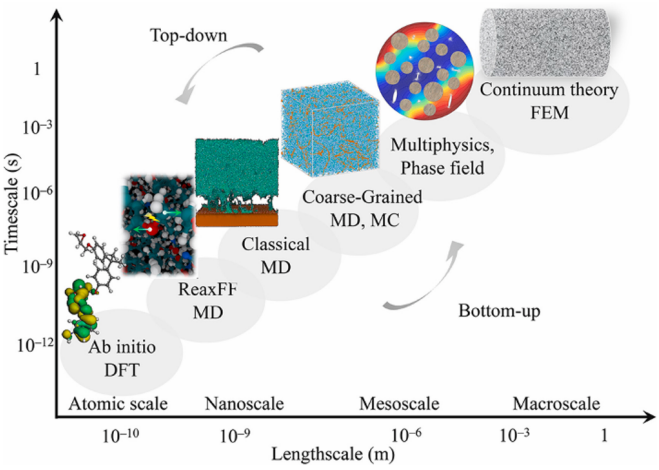


FIG. 1: Overview of modeling approaches across spatiotemporal scales in molecular simulations. Ab initio methods such as density functional theory operate at the atomic scale and femtosecond timescales, offering high accuracy at high cost. Classical and reactive MD span the nanoscale to mesoscale, while CG methods extend reach to longer timescales and larger systems by reducing resolution. At macroscopic scales, continuum theories and finite element methods dominate. Figure reproduced from [6].

CG models address this limitation by reducing the number of degrees of freedom in a system, typically by mapping groups of atoms into larger effective particles or "beads." This dimensionality reduction smooths out fast atomic fluctuations, enabling simulations over

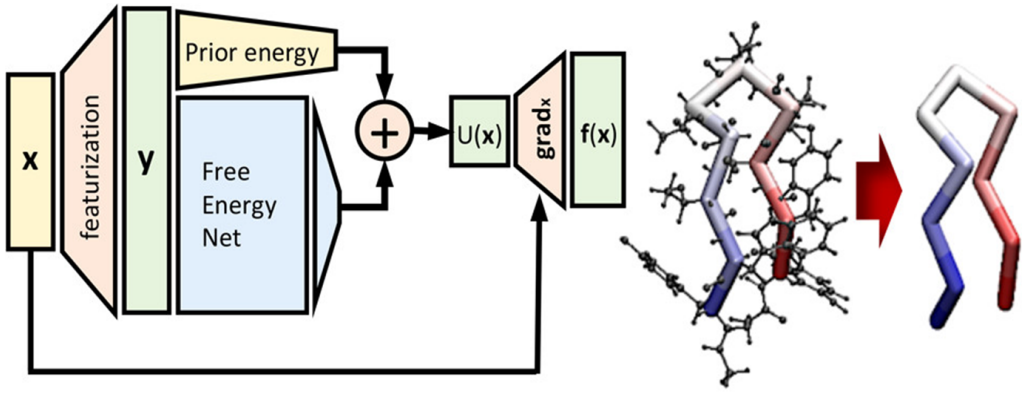


FIG. 2: **Left:**Schematic of the CGnet architecture for learning coarse-grained force fields. A low-dimensional CG configuration $\mathbf{x}$ is transformed via featurization into $\mathbf{y}$, which are then passed to a neural network that estimates the system's free energy $U(\mathbf{x})$. A physically motivated prior energy term can be included to impose baseline behaviors. The learned energy is differentiated with respect to $\mathbf{x}$ to produce forces $\mathbf{f}(\mathbf{x}) = -\nabla U(\mathbf{x})$, trained to match forces obtained from atomistic simulation. **Right:** a protein structure is mapped to a coarse-grained representation with fewer degrees of freedom. Figure reproduced from [4].

longer timescales and larger length scales. As a result, CG models are widely used to study large biomolecular complexes, soft matter systems, and long-timescale processes [7]. However, this abstraction introduces a central challenge: defining an effective interaction potential that faithfully reproduces the thermodynamic and dynamic properties of the original atomistic system.

To address the challenges of constructing accurate and expressive coarse-grained potentials, Wang et al. introduced the CGnet framework [4], which casts force field learning as a supervised machine learning problem. Rather than specifying a fixed functional form for the coarse-grained potential, CGnet learns the PMF directly from atomistic simulation data using a neural network. As shown in Fig. 2, CGnet takes as input a coarse-grained configuration $\mathbf{x}$ (e.g., the coordinates of CG beads), which is then transformed into descriptors $\mathbf{y}$ that encode structural information (e.g., pairwise distances or angular terms). These features are passed through a fully connected neural network, referred to as the Free Energy Net, which outputs a scalar energy value $U(\mathbf{x})$ representing the estimated free energy.

A. Force Matching as a Learning Objective

A central challenge in coarse-graining is determining an effective force field that reproduces the thermodynamics of the atomistic system in a reduced coordinate space. One widely used approach is force matching, where the objective is to learn a coarse-grained force function that reproduces the average forces acting on CG coordinates as derived from atomistic simulations. In CGnet, the model learns the PMF, $U(\mathbf{x})$, as a function of CG coordinates $\mathbf{x}$, from which the forces are obtained via

$$\mathbf{f}(\mathbf{x}) = -\nabla U(\mathbf{x}). \tag{1}$$

To evaluate CGnet's ability to learn such forces, the paper introduces a 2D toy model in which the underlying potential energy surface is known analytically. This potential includes nonlinear features in x, a harmonic well in y, and small oscillatory perturbations:

$$V(x,y) = \frac{1}{50}(x-4)(x+2)(x-2)(x+3) + \frac{1}{2}y^2 + \frac{1}{25}\sin\left(3(x+5)(x+6)\right). \tag{2}$$

Reference data is generated by sampling positions uniformly from the 2D domain and computing instantaneous forces via exact gradients of the potential.

The authors then train CGnet to learn the coarse-grained energy $U(x)$ using only the x-component of the force data. The CGnet model takes scalar x values as input and returns a scalar energy output; forces are then computed by differentiating the output with respect to the input using automatic differentiation. The loss function used during training is the force-matching loss, which minimizes the squared difference between these predicted forces and the reference forces derived from the potential:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{f}(x_i) - f_{\text{ref}}(x_i)\right)^2, \tag{3}$$

where $\hat{f}(x) = -\partial_x U(x)$.

This setup allows for a direct comparison between the learned force field and the exact mean force, as well as the ability to recover PMF via integration. Importantly, the network **never sees the true potential or its integrals during training—only the force values—yet**

still succeeds in reconstructing both the force and free energy landscape.

Fig. 3(c) illustrates this concept on the aforementioned 2D toy system. The orange points represent instantaneous forces from atomistic simulations, while the blue curve shows the average (mean) force. CGnet models learn smooth approximations to these forces that better capture the underlying mean force field than simple feature-based regression. As shown in panel (d), integrating the learned forces produces free energies that closely match the exact PMF, even when the network is trained only on force data. This demonstrates that CGnet is not only capable of recovering forces but also the associated thermodynamic landscape.

B. Invariances and Regularization

When learning coarse-grained models from data, it is key that the learned potential respects the symmetries and physical constraints of the system. For instance, a CG potential should be invariant under translations, rotations, and permutations of identical particles. Without careful design, neural networks may learn unphysical artifacts or overfit sparse training data, especially in poorly sampled regions of configuration space.

CGnet addresses this issue through a combination of architectural design and explicit regularization. Input features to the network are constructed to be invariant under rigid body motions—for example, using inter-particle distances or angles instead of raw coordinates. CGnet additionally introduces a prior energy term $U_{\text{prior}}(\mathbf{x})$ that encodes known physical constraints, such as bonded interactions or repulsive terms. This prior is added to the learned energy from the neural network:

$$U_{\text{total}}(\mathbf{x}) = U_{\text{prior}}(\mathbf{x}) + U_{\text{net}}(\mathbf{x}).$$

As shown in Fig. 3(c,d), unregularized models (green dashed lines) often extrapolate poorly, diverging in regions not covered by training data. In contrast, the regularized CGnet (cyan) extrapolates more smoothly and preserves realistic force behavior. This regularization leads to significantly improved free energy reconstruction, as visible in panel (d), aligning closely with the exact potential of mean force.

III. OUR IMPLEMENTATION: A 2D PROOF-OF-CONCEPT

To gain hands-on intuition for the CGnet framework and its learning dynamics, we implemented a simplified version in Julia focused on a 2D toy potential. Our goal was to build a minimal but extensible foundation for CGnet-style architectures that can later be adapted to higher-dimensional or molecular coarse-graining tasks.

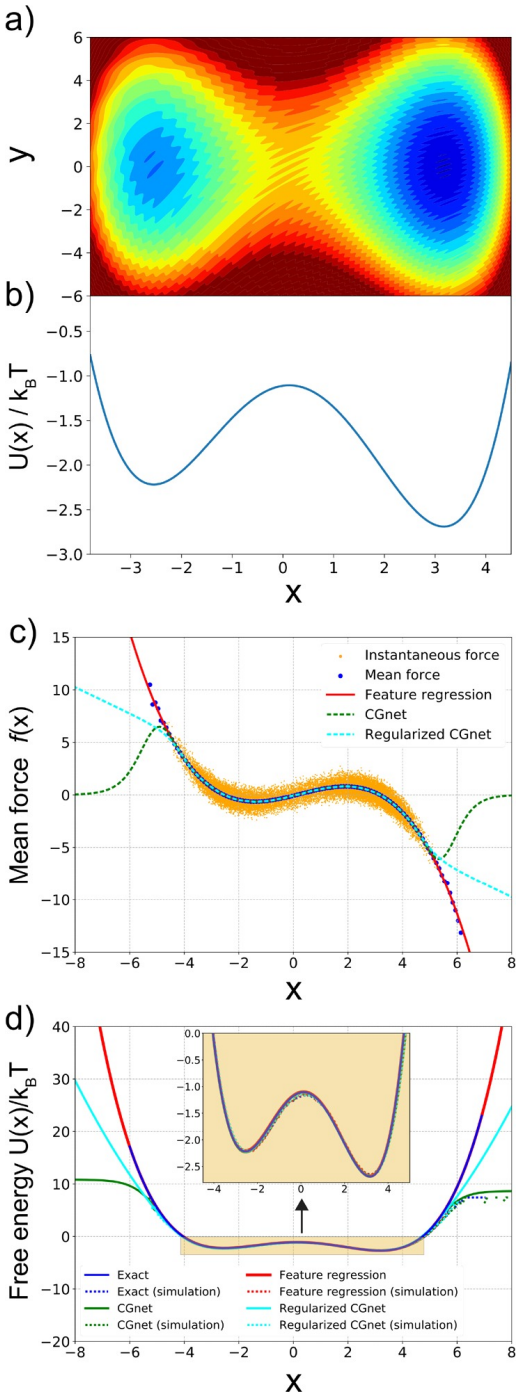


FIG. 3: Machine-learned coarse-graining of dynamics in a rugged 2D potential. (a) Two-dimensional potential used as a toy system. (b) Exact free energy along x. (c) Instantaneous forces and the learned mean forces using feature regression and CGnet models (regularized and unregularized) compared to the exact forces. (d) PMF along x predicted using feature regression, and CGnet models compared to the exact free energy. Free energies are also computed directly from histogramming simulation data, using the underlying 2D trajectory or simulations run with the feature regression and CGnet model. Figure reproduced from and caption adapted from [4].

To validate the CGnet framework and explore its learning dynamics in a controlled setting, we implemented a simplified version in Julia targeting a two-dimensional toy potential introduced in [4]. This potential, defined in Eq. 2 produces a free energy surface with global structure and local complexity. We coarse-grain over variable y to learn a one-dimensional PMF $U(x)$ that captures the equilibrium thermodynamics along x. To generate training data, we uniformly sample 5,000 points from the 2D domain $(x, y) \in [-5, 5]^2$. For each sample, we compute the exact instantaneous force in the x-direction, $f_x = -\frac{\partial V}{\partial x}$, using automatic differentiation. These values serve as ground-truth labels for supervised training. The model never sees the full potential $V(x, y)$, only the sampled x-coordinates and their corresponding forces.

Our neural network, implemented in Flux.jl, takes scalar x values as input and outputs a scalar energy $U(x)$, interpreted as the learned coarse-grained PMF. The force predictions $\hat{f}(x)$ are computed as gradients of this output using Zygote.jl (see Eq. 1). We then optimize the network by minimizing the force-matching loss in Eq. 3, which compares predicted forces to the reference values.

The network architecture consists of four fully connected layers with ReLU (Rectified Linear Unit) activation:

- `Dense(1 → 128)` with ReLU
- `Dense(128 → 128)` with ReLU
- `Dense(128 → 128)` with ReLU
- `Dense(128 → 1)` linear output

This structure follows the standard ML design of a hidden-layer multilayer perceptron with nonlinear activations for function approximation, followed by a linear output layer. ReLU activation functions are used for their computational efficiency and their ability to mitigate the vanishing gradient problem in deeper networks. We train the network using the ADAM optimizer with a learning rate of 0.0005 over 200 epochs, processing data in batches of 256 to prevent overfitting. The model and data are moved to GPU memory using Julia's CUDA.jl backend to accelerate training. After optimization, we evaluate the model's performance by comparing both its force predictions and the reconstructed PMF (via numerical integration) against the ground-truth results from the original potential. As a whole, this foundation closely mirrors the design in the CGnet paper.

IV. RESULTS AND EVALUATION

To evaluate the performance of our learned CGnet model, we compare the predicted force field and reconstructed potential of mean force (PMF) against exact results derived from the underlying 2D potential. First, we examine the loss function throughout training, shown in Fig. 4.

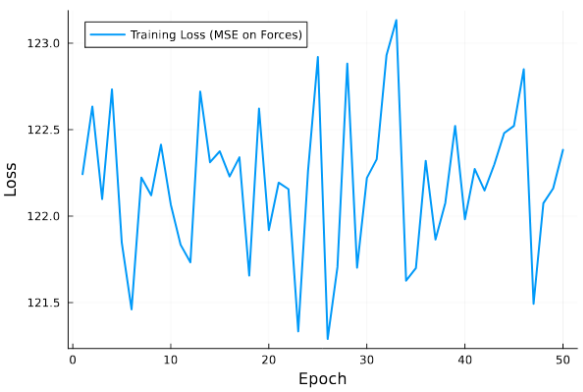


FIG. 4: Training loss (MSE on forces) over 50 epochs. The loss fluctuates heavily and does not decrease significantly, indicating poor convergence and potential failure to escape a local minimum.

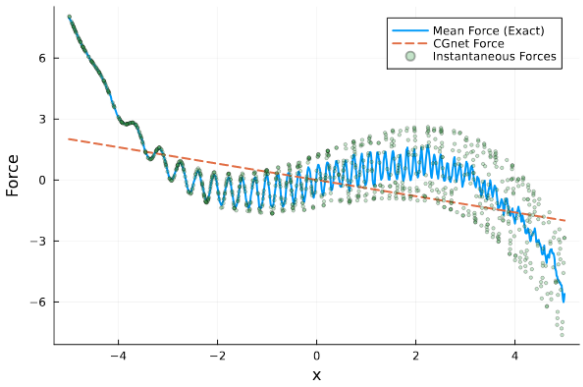
As shown in Fig. 4, the loss function plateaus quickly and fluctuates between values of approximately 121–123. This suggests that the model is not learning effectively or is stuck in a shallow local minimum. In addition, visualizing the predicted CGnet force field and PMF in Fig. 5 shows that both are significantly off from their true functions, respectively. While the general direction of the force field in Fig. 5a is preserved in noting that the function decreases overall as position increases, the fit misses almost all of the oscillatory features of the original function.

We attempted to rectify this by lowering the learning rate to 10^{-4}, increasing the number of epochs, changing the activation function to `tanh`, and even increasing the model depth and number of neurons, but these changes did not significantly improve convergence.

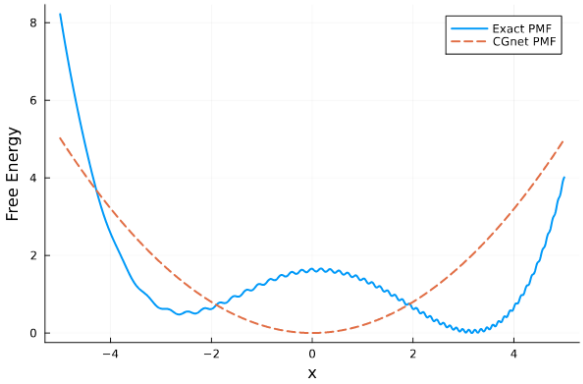
These results correspond to the CGnet implementation where the model only sees x values and their corresponding forces during training, never the full 2D potential $V(x, y)$. While this setup is faithful to the CGnet framework, it appears to make learning more difficult.

As a control, we tested a simpler version of our implementation where the model directly predicts forces from x, bypassing the energy gradient formulation. In this variant, we trained for 1000 epochs using a modified loss function that directly minimizes the difference between predicted and true forces rather than computing forces as gradients of the predicted energy. This version showed steady and linear loss convergence, as seen in Fig. 6.

Taken together, these results demonstrate that our simplified direct force prediction model captures the essential physics of the system with far greater fidelity than the original CGnet implementation in our hands. As illustrated in Figs. 6–8, the learned forces and reconstructed PMF closely mirror the exact quantities, showing comparable accuracy to the original CGnet benchmarks reported in Fig. 3.



(a) Comparison of predicted CGnet force field to the mean force from the true potential.



(b) Reconstructed PMF (via force integration from CGnet) versus the exact PMF derived from the full potential.

FIG. 5: Evaluation of CGnet performance on a 2D toy potential. The poor match in both subfigures further confirms the model's limited learning under this setup.

A. Analysis and Discussion

The contrast between our two training strategies—energy-based CGnet versus direct force prediction—highlights several key considerations in developing coarse-grained models. While the CGnet approach is more physically grounded through an explicit energy function, training such models presents practical challenges; specifically, the optimization landscape appears ill-conditioned, resulting in minimal reduction in training loss even after extensive epochs. We can immediately visually tell that this model does not produce the most accurate results based on the results. These results as a whole suggest that second-order gradient dependencies, the only difference between the two models, may introduce numerical instability or amplify gradient noise.

In contrast, the direct force prediction model circumvents these complications by optimizing directly on the force labels. This leads to a more stable and efficient training process, as reflected in the performance visual-

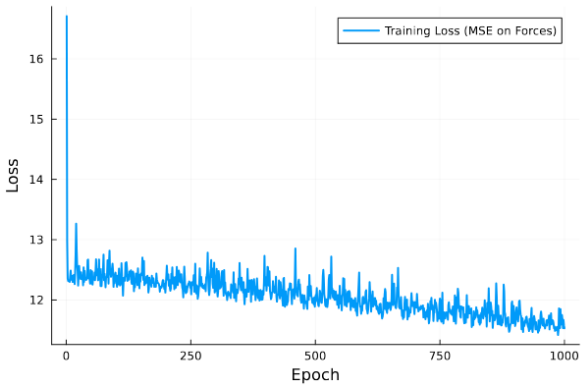


FIG. 6: Loss function over 1000 epochs in the simplified model directly trained to predict forces. The loss decreases steadily, demonstrating successful learning.

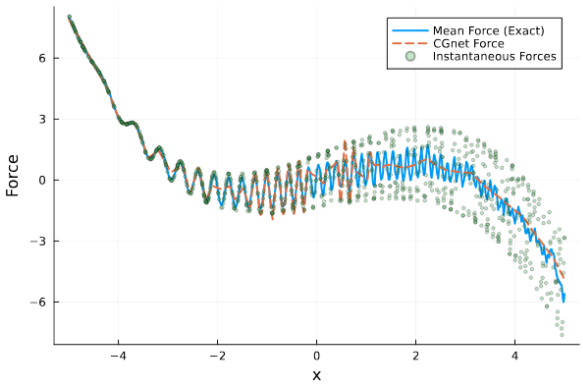


FIG. 7: Force field predicted by the directly trained model compared to the exact force. Much closer agreement is achieved compared to the original CGnet model.

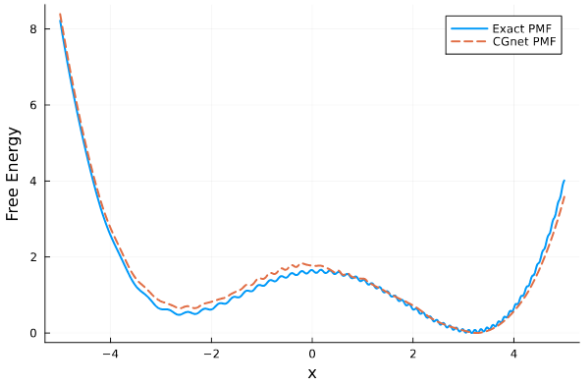


FIG. 8: Reconstructed PMF from the directly trained model shows strong agreement with the exact PMF, confirming that this model learns a good approximation of the true energy landscape.

ized in Figure 9. Subplot 9a shows that the training and validation losses both converge quickly and remain stable throughout, indicating a well-behaved learning dynamic and absence of overfitting. The validation loss remains consistently lower than the training loss, a sign of effective generalization.

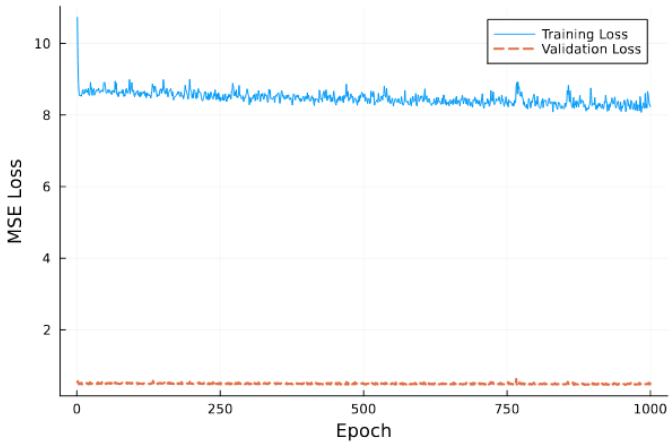
Moreover, the fidelity of force prediction is evident in Subplot 9b, where predicted forces closely follow the ground truth, with little deviation from the identity line. This demonstrates that the model has effectively learned the functional mapping from input features to force outputs. The residuals (defined as $f_{\text{true}} - f_{\text{pred}}$, shown in Subplot 9c, are also mostly centered around zero, with a modest spread, particularly near the tails of the distribution. This suggests that while the model performs reliably across the bulk of the data, there may still be systematic biases or reduced accuracy at the extremes—perhaps due to lower data density or more complex dynamics in those regimes.

In the future, we can explore hybrid strategies that combine the physical consistency of energy-based models with the training simplicity of force prediction. This may involve improved gradient estimation techniques (e.g., smoothing second-order derivatives), regularization schemes enforcing energy conservation, or pretraining force models to help energy-based optimization.

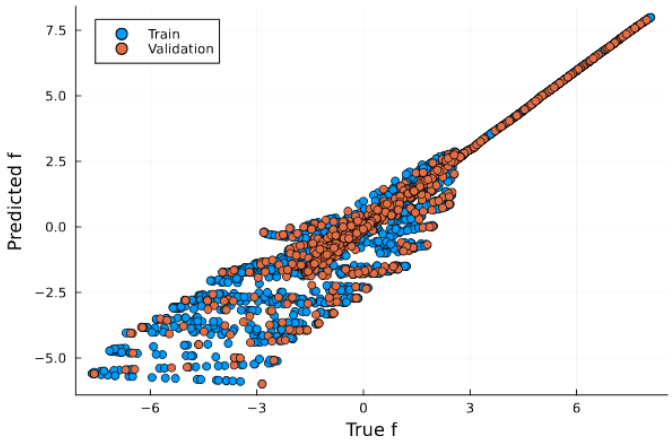
ACKNOWLEDGMENTS

We would like to acknowledge Prof. Sahakian for allowing us to use his GPU resources for our project.

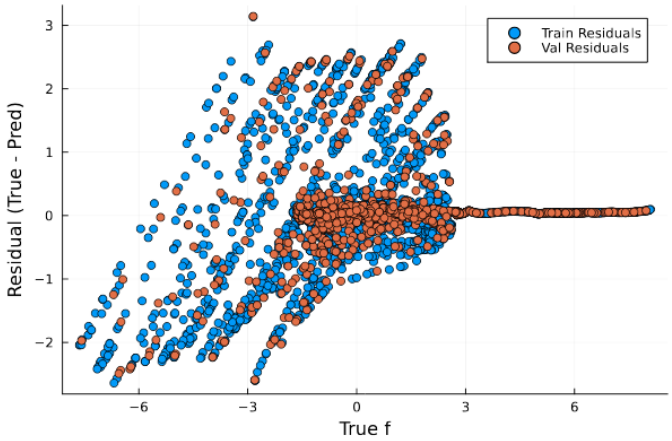
[1] W. G. Noid, J.-W. Chu, G. S. Ayton, V. Krishna, S. Izvekov, G. A. Voth, A. Das, and H. C. Andersen, The Journal of Chemical Physics **128**, 244114 (2008).
[2] S. J. Marrink, J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, J. Phys. Chem. B **111**, 7812 (2007).
[3] J. Behler and M. Parrinello, Phys. Rev. Lett. **98**, 146401 (2007).
[4] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. de Fabritiis, F. Noé, and C. Clementi, ACS Central Science **5**, 755 (2019).
[5] P. Thölke and G. De Fabritiis, Adv. Neural Inf. Process. Syst. (NeurIPS) (2022).
[6] K. Lin and Z. Wang, Communications Materials **4**, 66 (2023).
[7] K. Liebl and G. A. Voth, Journal of Chemical Theory and Computation (2023), 10.1021/acs.jctc.5c00063, article ASAP.



(a) Training and validation loss curves over 1000 epochs. While the training loss exhibits mild fluctuations, the validation loss remains consistently low, suggesting stable generalization.



(b) Scatter plot comparing predicted force values against ground truth. Points clustering along the diagonal line indicate accurate force prediction for both training and validation data.



(c) Residuals (difference between true and predicted forces) plotted against true values. Residuals are centered around zero, with slightly greater spread at extreme values, indicating model bias near the tails.

FIG. 9: Performance of the direct force prediction model across training loss, prediction accuracy, and residual distribution.