Neural Scaling of Deep Chemical Models

Nathan C. Frey[1*], Ryan Soklaski[1], Simon Axelrod[2,3], Siddharth Samsi[1], Rafael Gómez-Bombarelli[2], Connor W. Coley[4,5] and Vijay Gadepally[1]

[1]Lincoln Laboratory, Massachusetts Institute of Technology, 244 Wood Street, Lexington, 02421, MA, USA.
[2]Department of Materials Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, 02139, MA, USA.
[3]Department of Chemistry and Chemical Biology, Harvard University, 12 Oxford Street, Cambridge, 02138, MA, USA.
[4]Department of Chemical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, 02139, MA, USA.
[5]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, 02139, MA, USA.

*Corresponding author(s). E-mail(s): ncfrey@mit.edu;
Contributing authors: ryan.soklaski@ll.mit.edu; simonaxelrod@g.harvard.edu; ssamsi@mit.edu; rafagb@mit.edu; ccoley@mit.edu; vijayg@mit.edu;

Abstract

Massive scale, both in terms of data availability and computation, enables significant breakthroughs in key application areas of deep learning such as natural language processing (NLP) and computer vision. There is emerging evidence that scale may be a key ingredient in scientific deep learning, but the importance of physical priors in scientific domains makes the strategies and benefits of scaling uncertain. Here, we investigate neural scaling behavior in large chemical models by varying model and dataset sizes over many orders of magnitude, studying models with over one billion parameters, pre-trained on datasets of up to ten million datapoints. We consider large language models for

generative chemistry and graph neural networks for machine-learned interatomic potentials. To enable large-scale scientific deep learning studies under resource constraints, we develop the Training Performance Estimation (TPE) framework to reduce the costs of scalable hyperparameter optimization by up to 90%. Using this framework, we discover empirical neural scaling relations for deep chemical models and investigate the interplay between physical priors and scale. Potential applications of large, pre-trained models for "prompt engineering" and unsupervised representation learning of molecules are shown.

Keywords: chemistry, deep learning, graph neural networks, hyperparameter optimization, neural scaling

1 Introduction

The "unreasonable effectiveness" of deep learning [1] in domains such as computer vision (CV) and natural language processing (NLP) relies on the ability of deep neural networks (DNNs) to leverage ever-increasing amounts of compute, data, and model capacity. Large-scale models, including Bidirectional Encoder Representations from Transformers (BERT) [2] and DALL-E [3], have been so successful at synthesizing information from large datasets via self-supervised pre-training and performing a variety of downstream tasks with little to no finetuning that most state-of-the-art (SOTA) models in NLP and CV are adapted from a small set of large, pre-trained models [4]. Naturally, we might expect that massive model and dataset scaling will be a prerequisite to achieving out-sized success for deep learning in science. Recent work such as AlphaFold [5], the Open Catalyst Project [6, 7], and ChemBERTa [8] indicate that larger datasets and models, pre-training, and self-supervised learning – all key ingredients in CV and NLP – unlock new capabilities for deep learning in chemistry. However, unlike in CV and NLP, the path to scaling deep chemical networks and the potential benefits are unclear. Chemical deep learning can incorporate physics-based priors that may ameliorate the steep resource requirements seen in other fields [9–12]. Moreover, because of the heterogeneity and complexity of chemical space [13] and molecular machine learning (ML) tasks [14, 15], training general and robust models that perform well on a wide variety of downstream tasks remains a pressing challenge [8, 16, 17]. The enormity of chemical space and heterogeneity of these tasks motivates investigations of large-scale models in chemistry, because such models are well-suited to unlabeled, multi-modal datasets [3, 4]. Recently, neural scaling laws [18, 19] have emerged as a way to characterize the striking trends of improved model performance over many orders of magnitude with respect to model size, dataset size, and compute; however, these experiments require immense computational resources and rely on well-known, domain-specific model training procedures that do not apply outside of traditional deep learning application areas.

With the inordinate costs of developing and deploying large models [20], it is difficult to investigate neural scaling behaviors of scientific deep learning models, which require expensive hyperparameter optimization (HPO) and experimentation. Architectures and hyperparameters that work well for small models and small datasets do not transfer to larger scales [21]. This presents a risk that scientific deep learning will become increasingly inaccessible as resource demands increase. Techniques for accelerating neural architecture search (NAS) and hyperparameter transfer such as training speed estimation (TSE) [22] and μTransfer [21] could accelerate the development of large-scale scientific deep learning models, where rapid advances in architecture design and complex data manifolds prevent the easy transfer of parameters and settings used in CV and NLP. To investigate the capabilities of deep chemical models across resource scales, practical and principled approaches are needed to accelerate hyperparameter transfer and characterize neural scaling.

In this paper, we develop strategies for scaling deep chemical models and investigate neural scaling behavior in large language models for generative chemical modeling and graph neural networks (GNNs) for machine-learned interatomic potentials. We introduce *ChemGPT*, a generative pre-trained transformer for autoregressive language modeling of small molecules. We train ChemGPT models with over one billion parameters, using datasets of up to ten million unique molecules. We also examine large, invariant and equivariant GNNs trained on trajectories from molecular dynamics and investigate how physics-based priors affect scaling behavior. To overcome the challenges of hyperparameter tuning at scale in new domains, we extend techniques for accelerating neural architecture search to reduce total time and compute budgets by up to 90% during HPO and neural architecture selection. We identify trends in chemical model scaling with respect to model capacity and dataset size, and show the performance improvements seen with increasing scale. We demonstrate the capability to tune ChemGPT's outputs via "prompt engineering" and sampling strategies. Pre-trained ChemGPT models are also robust, self-supervised representation learners that generalize to previously unseen regions of chemical space and enable embedding-based nearest-neighbor search. The scaling strategies and results enable immediate, significant improvements to model performance, as well as computational and data efficiency for deep chemical models. Our results provide motivation and practical guidance for scaling studies in scientific deep learning, as well as many fruitful new research directions at the intersection of massive scale and physics-informed deep learning.

2 Main

In this section we describe the aspects of the workflow developed in this paper, summarized graphically in Figure 1. We define neural scaling and the model architectures considered here, which are chosen specifically for their likelihood

to exhibit interesting scaling behavior. Then we introduce strategies to enable scaling large chemical models and investigations of scaling behavior.

Neural scaling. For large language and computer vision models trained to convergence with sufficient model parameters and/or data, performance is characterized by empirical scaling laws where the loss scales as a power-law [18] of the form

$$L(R) = \alpha R^{-\beta} \tag{1}$$

for coefficient α, scaling exponent β, and resource R. R is the number of model parameters, dataset size, or compute. β measures the slope of the power-law and indicates the scaling efficiency of the model with respect to a scaling factor, R. The power-law trends break down in "resolution-limited" regimes [23], indicating that the model (dataset) size is insufficient for the given amount of data (model parameters).

Neural scaling presents a best-case scenario for model performance with increasing resources, and allows for optimal allocation of fixed budgets, e.g., to decide whether longer training, more data, or larger models will be most efficient for improving performance. Comparing neural scaling exponents also provides a fundamental metric for measuring resource efficiency across model architectures. Investigations into neural scaling in the NLP domain have revealed general conclusions about overfitting, sensitivity to architectural choices, transfer learning, and sample efficiency [18]. These factors are equally or more important in scientific deep learning applications, where rapid advances are being made in specialized architecture development, and it is often unclear how architectures will perform beyond the small benchmark datasets that are commonly available in scientific settings.

Large chemical language models. Strings are a simple representation for molecular graphs [24], thereby making sequence-based ML models a natural choice for working with chemical data. Following the demonstrated performance improvements of Transformer-based models with increasing model and dataset sizes [8, 18, 23], we designed a large generative language model for chemistry called *ChemGPT* to investigate the impact of dataset and model size on pre-training loss. ChemGPT is a Generative Pre-trained Transformer 3 (GPT3)-style model [25, 26] based on GPT-Neo [27, 28] with a tokenizer for Self-referencing embedded strings (SELFIES) [24, 29] representations of molecules. SELFIES enforce chemical validity and are straightforward to tokenize, but ChemGPT can easily be used with simplified molecular-input line-entry system (SMILES) strings as well [30]. For chemical language modeling, a set of molecules $(x_1, x_2, ..., x_n)$ is represented with each molecule as a sequence of symbols $(s_1, s_2, ..., s_n)$. The probability of a sequence, $p(x)$ is factorized as the product of conditional probabilities [31]:

$$p(x) = \prod_{i=1}^{n} p(s_i | s_1, ..., s_{i-1}). \tag{2}$$

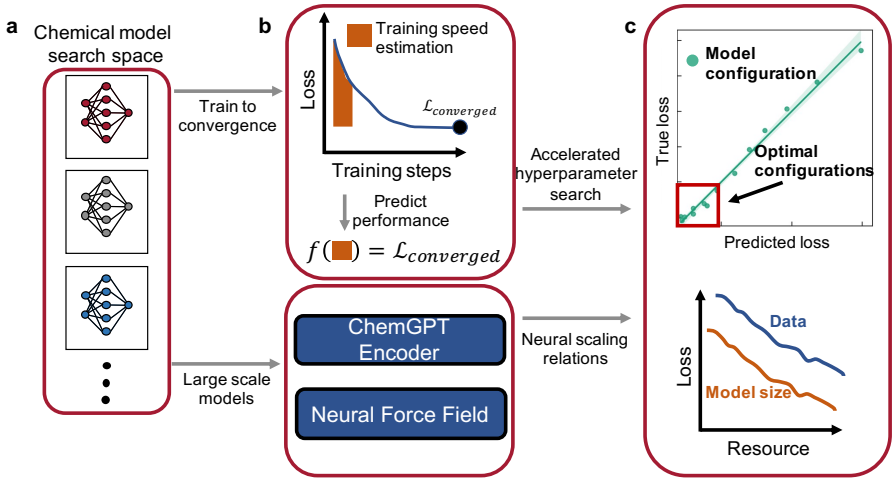

Fig. 1 Training performance estimation uses training speed to accelerate model selection and hyperparameter optimization, enabling the discovery of neural scaling relations for deep chemical models. Over a domain of (**a**) model candidates, final, converged model performance is (**b**) predicted from only a few initial epochs of training for large-scale models. (**c**) Non-optimal model architectures and hyperparameter configurations are identified early in training, allowing for efficient selection of the ideal architecture and hyperparameters. The model with the best hyperparameters is then trained with varying model and dataset sizes to discover neural scaling relations.

ChemGPT uses the Transformer [32] architecture with a self-attention mechanism to compute conditional probabilities, estimate $p(x)$, and sample from it to generate new molecules. ChemGPT is pre-trained on molecules from PubChem [33] with a causal language modeling task, where the model must predict the next token in a sequence, given the previous tokens. ChemGPT models of up to one billion non-embedding parameters are trained on up to ten million molecules, whereas typical chemical generative models have less than one million parameters and are trained on less than one million samples [30].

Graph neural network force fields (GNNFFs). For many tasks in chemistry, molecular geometry and 3D structure is essential and string-based representations are not sufficient. Neural force fields (NFFs) are graph neural networks (GNNs) that take molecular geometries as inputs, described by a set of atomic numbers $(Z_1, ..., Z_n | Z_i \in \mathbb{N})$ and Cartesian coordinates $(\vec{r}_1, ..., \vec{r}_n | \vec{r}_i \in \mathbb{R}^3)$. The NFF with parameters θ, f_θ, predicts a real-valued energy $\hat{E} = f_\theta(X)$ for an atomistic configuration X. The NFF produces energy-conserving atomic forces by differentiating the energies with respect to the atomic coordinates,

$$\hat{F}_{ij} = -\frac{\partial \hat{E}}{\partial r_{ij}} \tag{3}$$

for atom i and Cartesian coordinate j. Typically, the network is trained by minimizing the loss $\mathcal{L}$ computed from the average mean squared error for a

mini-batch of size N,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}[\alpha_E\|E_i - \hat{E}_i\|^2 + \alpha_F\|\mathbf{F}_i - \hat{\mathbf{F}}_i\|^2] \tag{4}$$

where α_E and α_F are coefficients that determine the relative weighting of energy and force predictions during training [34]. For scaling experiments we use the L1 loss or mean absolute error,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}[\alpha_E\|E_i - \hat{E}_i\| + \alpha_F\|\mathbf{F}_i - \hat{\mathbf{F}}_i\|] \tag{5}$$

which we empirically find to show more robust convergence behavior.

In this work we consider four flavors of NFFs: SchNet [35], PaiNN [36], Allegro [10], and SpookyNet [37]. This series of models represents increasingly physics-informed model architectures, from models with internal layers that manipulate only E(3) invariant quantities (SchNet) to those that use E(3) equivariant quantities (PaiNN, Allegro, SpookyNet), strictly local models with learned many-body functions and no message passing (Allegro), and physically-informed via empirical corrections (SpookyNet). The power and expressivity of these GNNs can be defined in terms of their capacity [38],

$$c = d * w \tag{6}$$

where d is depth (number of layers or convolutions [35]) and w is width (the embedding dimension or number of basis functions employed by each convolution). Capacity is a simple parameter to vary during neural scaling experiments, because model size is not a strictly useful scaling parameter for GNNs [38]. Typical evaluations of NFFs consider training dataset sizes of less than 1,000 3D geometries of a single chemical species, which leads to insensitivity to model capacity because of the simplicity of the learning task [17]. Here, we consider up to 100,000 training geometries (corresponding to 4.5 million force labels) and GNNs with millions of trainable parameters.

Accelerating hyperparameter optimization with training performance estimation. Because model hyperparameters, including learning rates and batch sizes, are essential for achieving optimal model performance and are non-transferable between different domains and model/dataset sizes [21], we need efficient strategies for scalable HPO in deep chemical models. We adapt Training Speed Estimation (TSE) [22], a simple technique for ranking computer vision architectures during neural architecture searches, to accelerate HPO and model selection for ChemGPT and GNNs. We call this method "Training Performance Estimation" (TPE), as it uses training speed to more generally enable performance estimation across a wide range of applications. TPE generalizes TSE to HPO for new deep learning domains (Large Language Models [LLMs], GNNs) and can be used to directly predict converged loss, in

addition to rank ordering different architectures. HPO typically involves training tens or hundreds of networks and using random search and/or Bayesian optimization to identify optimal hyperparameters. For optimal performance, the process must be repeated when considering new datasets or distribution shift.

By calculating the "training speed" from only the first few epochs of training, the converged model performance is predicted and optimal hyperparameters are identified using only a small fraction of the total training budget. For example, networks that require 100 epochs to train to convergence are trained for only 10–20 epochs, and the final performance is predicted using TPE to identify the best performing networks, thereby saving 80–90% of the total training budget.

Training speed is estimated by summing the training losses of each mini-batch during the first T epochs of training. After training the network for T epochs with B training steps per epoch, TSE is defined as

$$TSE = \sum_{t=1}^{T} \left(\frac{1}{B} \sum_{i=1}^{B} \mathcal{L}(f_{\theta(t,i)}(\mathbf{X}_i), \mathbf{y}_i) \right), \tag{7}$$

for a loss function $\mathcal{L}$ and a neural network $f_{\theta(t,i)}$, with parameters θ at epoch t and mini-batch i. $(\mathbf{X}_i, \mathbf{y}_i)$ is a tuple of inputs and labels in the i-th mini-batch. TSE is correlated with the converged performance of the network and can be used to rank networks early in training to yield substantial compute savings. Given a sufficient number of networks (5–10) that are trained to convergence, a linear regression of the form

$$L = m \cdot TSE + b \tag{8}$$

is fit with parameters m and b and the calculated TSE values to predict the converged loss, L. This allows predictions of converged network loss for partially-trained networks evaluated during HPO based on its TSE values. In our experiments, we noted that L is monotonic in TSE, meaning that Equation 8 is not needed to simply choose the best hyperparameters. The TSE values computed after a small number of epochs are sufficient for ranking model configurations and finding the optimal ones. Although leveraging Equation 8 requires training some small number of networks to convergence in order to fit the parameters, it provides the benefit of being able to predict the expected performance of new hyperparameter choices.

3 Results

Training performance estimation accelerates hyperparameter optimization for new datasets, models, and scales. To conduct extensive scaling experiments, we first need to find reasonable hyperparameters and

training settings. Unlike for NLP and CV, there are no default model architectures, datasets, tasks, hyperparameter settings, or training settings for large-scale chemical deep learning. Simply transferring empirical results from other deep learning domains or smaller scale experiments will lead to suboptimal results [21]. Whereas large models and datasets are standard in traditional deep learning application areas, to investigate scaling in deep chemical models, we must lay the groundwork for large-scale experiments. To this end, we first tackle the problem of accelerating HPO in general settings, for new model architectures, heterogeneous datasets, and at scales that have not been previously investigated.

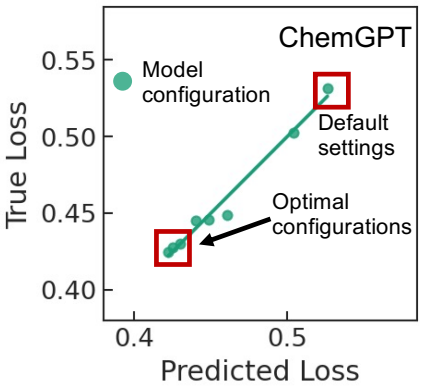

Fig. 2 Optimal models are identified early in training with training performance estimation and training of non-optimal models is stopped to save 80%+ total compute consumption. ChemGPT final validation loss (cross-entropy for causal language modeling) predicted from 20% of training budget using training performance estimation. Model configurations are determined through a grid search of different batch sizes and learning rates. Models are trained on two million molecules from Molecular Sets (MOSES).

Figure 2 shows the results of TPE for ChemGPT models trained on two million molecules from the Molecular Sets (MOSES) [39] dataset. MOSES is significantly smaller than PubChem and is representative of datasets on which chemical generative models are typically trained [30, 39]. Here, we use MOSES to demonstrate how optimal settings for a chemical LLM such as ChemGPT can be quickly discovered using TPE. To enable scaling experiments, we are mainly concerned with settings related to the learning dynamics (e.g., batch size and learning rate), that will significantly impact large-scale training and fluctuate depending on the type of model and the characteristics of the dataset. To demonstrate the effectiveness of TPE, we initialize ChemGPT with the default learning rate and batch size for causal language modeling in HuggingFace. We then vary the learning rate and batch size and train models with different hyperparameters for 50 epochs. Figure 2 shows the true loss after 50 epochs versus the predicted loss using TPE after only 10 epochs. $R^2 = 0.98$ for the linear regression (Equation 8), and Spearman's rank correlation $\rho = 1.0$.

With only 20% of the total training budget, we are able to identify model configurations that significantly outperform the default settings from HuggingFace. The procedure is easily repeatable for new datasets and enables accelerated HPO.

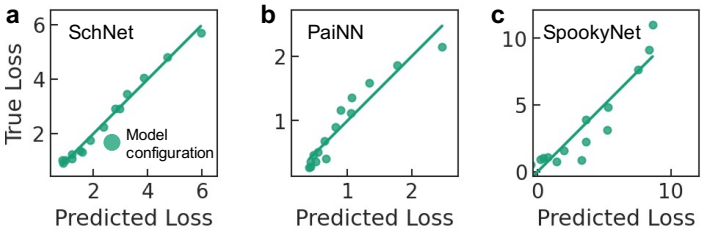

Fig. 3 Optimal models are identified early in training with training performance estimation and training of non-optimal models is stopped to save 80%+ total compute consumption. Neural force field (**a** SchNet, **b** PaiNN, **c** SpookyNet) model performance – measured via Equation 5 – predicted from <= 20% of training budget using training performance estimation. Model configurations are determined through a grid search of different batch sizes and learning rates. Models are trained on 10,000 frames from the revised MD-17 dataset.

Table 1 Goodness of fit metrics for graph neural network model performance using linear regression from TPE.

Model	R^2	Spearman's ρ
SchNet	0.99	0.99
PaiNN	0.91	0.97
SpookyNet	0.86	0.92

While training procedures for large language models like ChemGPT are well established, scaling NFFs to larger datasets and more expressive models requires new, scalable training procedures [17]. Large-batch training through data parallelism is one method for accelerating training, but there are known limitations and correct batch sizes vary widely for different domains [40]. This problem is particularly acute for NFFs, where each datapoint actually contains $3N+1$ labels for energies and atomic forces, where N is the number of atoms, creating a large effective batch size with large variance within each mini-batch. Hence, it has been observed that small batch sizes (even mini-batches of 1) work well across different NFF architectures [9, 37]. TPE provides a method for quickly evaluating the speed-accuracy trade off for different combinations of batch size and learning rate, which are interdependent and must be varied together to enable large-batch training.

TPE performs equally well for GNNs. We repeat the TPE procedure, varying the learning rate and batch size, for SchNet, PaiNN, and SpookyNet, training on 10,000 frames (1,000 frames/molecule) from the revised MD-17 [41]

dataset of 10 small organic molecules. Using only 20% of the total training budget, we achieve excellent predictive power (Figure 3) with TPE for SchNet and PaiNN. The variance in model performance using the entire training budget is significant, indicating the importance of proper HPO.

Because SpookyNet is a complex architecture that includes non-local interactions and empirical corrections, it exhibits slow convergence and the training speed is less correlated with the model performance than SchNet and PaiNN. However, the rank ordering of model configurations for SpookyNet from TPE is still robust (Spearman's $\rho = 0.92$), which allows for discarding non-optimal model configurations early in training, representing significant computational savings. The goodness of fit metrics for linear regressions using TPE are given in Table 1.

Neural scaling quantifies the performance improvements of large chemical models with increasing model and dataset sizes. Next, with a strategy in place to efficiently scale up experiments using TPE, we investigate neural scaling in ChemGPT and NFFs. For each model, we perform TPE to identify good hyperparameter choices that are predicted to perform well over a range of model and dataset sizes. Then, we systematically vary the dataset size (d) and model size (m) and perform exhaustive experiments to determine the converged loss, $L(m, d)$. For efficiency and to isolate scaling behavior, we fix hyperparameters from TPE as m and d are varied, but strictly speaking the optimal hyperparameters will change as m and d vary [21]. Due to computational resource limitations, we train ChemGPT models for a fixed number of epochs (10) to determine the loss.

Figure 4 shows the pre-training loss as a function of model and dataset size over many orders of magnitude. Models are trained in a self-supervised, causal language modeling setting and evaluated on next-token prediction for a fixed validation set. Surprisingly, no limitations in performance improvement are seen with increasing scale. The pre-training loss monotonically improves with increasing dataset size up to nearly 10 million molecules. Furthermore, for a fixed data budget, increasing model size provides monotonic improvements to the pre-training loss until the model reaches 1B+ non-embedding parameters. This indicates that even for small datasets, much larger models than were previously considered for deep generative modeling [30] may be useful for pre-training. For the largest dataset considered here, diminishing returns to performance improvements are seen for models above 100 million non-embedding parameters. Interestingly, greater performance improvements are seen with increasing model sizes for smaller datasets than larger ones. For the largest dataset considered, model performance saturates quickly beyond 100 million parameters. However, for the smallest dataset considered, performance plateaus for model sizes between $10^5 - 10^7$ parameters and then improves considerably. This indicates that for a fixed, small pre-training data budget, significant improvements are possible simply by scaling up the model size. Irrespective of model size, increasing dataset size provides continuous performance improvements with no evidence of diminishing returns for the dataset sizes considered here.

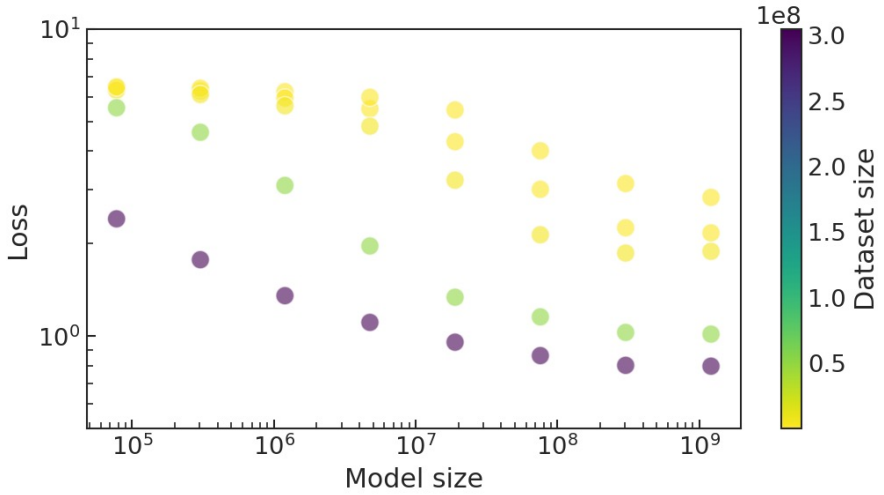

Fig. 4 Neural scaling of ChemGPT model performance (test loss) as a function of model (number of non-embedding parameters) and dataset (number of tokens) size. ChemGPT is pre-trained on up to 10 million molecules (300 million tokens) from PubChem. Performance improvements are seen for models up to one billion non-embedding parameters and continuous improvements are observed with increasing pre-training dataset size.

Depending on the dataset size, regimes of power-law-like scaling behavior are seen for different ranges of model sizes. Power-law scaling is graphically identifiable as an approximately straight line fit of loss versus model size on a log-log plot. For larger datasets, power law scaling is observed for smaller model sizes. For example, the largest dataset exhibits approximate power law scaling for models between 10^5 and 10^7 non-embedding parameters (Figure A.1). Conversely, for smaller datasets power law scaling is observed for larger models and over a more limited range of model sizes. The smallest dataset exhibits approximate power law scaling for models between 10^7 and 10^8 non-embedding parameters (not shown).

The break down in power-law scaling is indicative of "resolution-limited" neural scaling [23], where the model is sufficiently large but the dataset is not, or vice-versa. Identifying these resolution-limited regimes from the neural scaling relations allows us to understand in general terms whether model performance is limited by data availability or model capacity. The scaling exponent β is equal to 0.17 ± 0.01 for the largest dataset (Figure A.1), after discarding the three largest models from the power law fit. $\beta = 0.30 \pm 0.01$ for the next largest dataset (Figure A.2). The scaling exponent quantifies the performance improvements due to increasing model size, for a fixed data budget. A larger value of β corresponds to a steeper slope and better performance with increasing data/model size.

Graph neural network (GNN) interatomic potentials exhibit robust neural scaling behavior. The potential benefits of large-scale GNNs

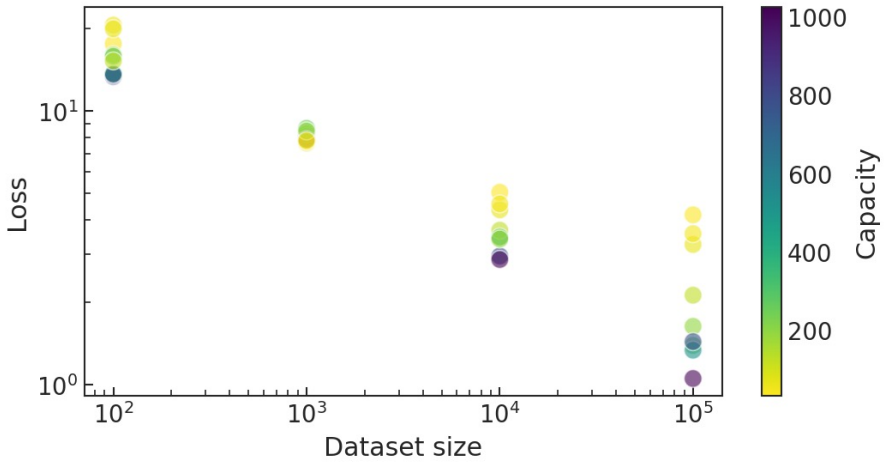

Fig. 5 Neural scaling of PaiNN model performance (test loss) as a function of model capacity (depth * width) and dataset size (number of geometries). PaiNN is trained to predict atomic forces from density functional theory calculations on small organic molecules from the ANI-1x dataset. Performance improvements are seen for models with greater capacity and continuous improvements are observed with increasing dataset size.

are less clear than for LLMs, as are the relevant parameters to vary, due to the inequivalence of depth and width for GNNs [38] and additional parameters beyond notions of model size that impact performance, e.g., nearest-neighbor cutoff in graph construction. To simplify GNN scaling experiments, here we vary GNN capacity (depth * width) by systematically changing network width and the number of convolutions (depth). We train GNNs to predict atomic forces from the ANI-1x dataset [42], the largest publicly available dataset of energies and forces for small molecules. NFF models are trained with a learning rate scheduler that reduces the learning rate every 50 epochs without improvement in the validation loss, until the learning rate reaches 10^{-7}. The loss is an L1 loss (Equation 5), shown in Figure 5 over four orders of magnitude of dataset size.

The neural scaling results for the equivariant GNN, PaiNN (Figure 5), show monotonic improvements to the loss with increasing dataset size. For a fixed dataset size, the converged loss is strongly correlated with the total training time (compute) and model capacity. Other than for 10^4 datapoints (for which some small models reach convergence quickly), the converged loss has a Spearman correlation coefficient $\rho >= 0.88$ with the model capacity and $\rho >= 0.75$ with the total training time. This means that the best models are those with optimal capacity that are able to train the longest without the validation loss plateauing. The optimal capacity and depth versus width change with the dataset size, i.e., the ideal GNN capacity is dataset-size dependent, and these choices can significantly impact the converged loss. These effects may also be artifacts of random initialization that would diminish with repeated trials. Interestingly, there is a stark change at 10^4 datapoints – the converged loss is

then nearly perfectly rank correlated with model capacity (Spearman's $\rho >= 0.93$). This might indicate that substantial overlap exists between the training and validation set, such that higher capacity models are merely exhibiting better memorization than lower capacity models. In these experiments, the validation set is constructed from unseen geometries and seen species (chemical species are the same in the training and validation sets). Repeating these experiments with a hold-out set of unseen chemical species will reveal if the same trend holds, which would indicate that rather than memorizing, the network is achieving generalization to new chemistries.

We observe similar trends in neural scaling for the invariant GNN, SchNet (Figure A.3, although the equivariant GNNs, PaiNN and Allegro (Figure A.4), exhibit significantly better scaling efficiency. A comparison of neural scaling between SchNet, PaiNN, and Allegro for models with fixed capacity (Equation 6), $c = 64$ (4 layers, width 16), is shown in Figure A.5. Over many orders of magnitude of dataset size, PaiNN and Allegro exhibit significantly greater sample efficiency, quantified by the calculated scaling exponents (Table A.1). That is, not only do the equivariant GNNs achieve better performance for a given data budget, they achieve increasingly greater performance gains given more training data. This is due to the models' equivariance, which is known to produce greater sample efficiency [9, 10], but it is interesting to note that this trend persists to much larger and more chemically diverse datasets than were previously considered, which typically include only $10^2 - 10^3$ molecular geometries from a single molecular species.

Training performance estimation and neural scaling enable significant improvements to model performance, and computational and data efficiency. Next, we briefly highlight the practical outcomes and usages of TPE and neural scaling as enabling technologies for scalable scientific deep learning. Based on the results presented above, TPE can be used in conjunction with any HPO routine to enable aggressive early stopping and accelerate HPO without sacrificing model performance. Clearly, the benefits of this approach become more pronounced in chemical and biological applications, where new network architectures must be continuously retrained, optimized, and evaluated on heterogeneous datasets.

Similarly, neural scaling provides practical ways to improve model performance and efficiency. Given an unlimited data and computation budget, the minimum loss in the neural scaling plot and corresponding model can be used. For example, the 300 million parameter ChemGPT model trained on 300 million tokens minimizes the loss in Figure 4. Likewise, the PaiNN model with capacity $\approx$ 1000 trained on 10^5 frames minimizes the loss in Figure 5. This may be valuable for pre-trained models that are designed to be reused and finetuned, where the training cost is amortized over many downstream applications. However, for many scientific applications, greedily optimizing for the minimum loss is not practical or even necessary. From the neural scaling results, identifying regions with the steepest slope allows for optimal and efficient allocation of resources. For example, for large chemical

language models, the greatest performance improvements (Figure 4) are seen for large data budgets when scaling up small models (10^5 parameters). For small data budgets, more rapid performance improvements are seen when scaling up medium-sized models (10^7 parameters). For NFFs, there are diminishing returns with increasing dataset sizes for low capacity models, while high capacity models show rapid performance improvements with increasing dataset size (Figure 5). The benefits from scaling model and dataset sizes should therefore be balanced against the increased computational costs in order to find the most computationally- and data-efficient opportunities for performance improvement. Beyond optimizing resource allocation to achieve better model performance, our results on large chemical models suggest potentially new capabilities of these models, which we will explore next.

Large chemical language model outputs are tunable via prompt engineering. There is a vast space of potential applications for a pre-trained LLM for chemistry, including but not limited to downstream tasks like property prediction [8] and distribution learning [30]. Here, to demonstrate potentially unique capabilities of ChemGPT, we focus on two applications: 1) prompt engineering and 2) representation learning. Prompt engineering is an emerging field in generative LLMs, wherein training a model is only the beginning of the process, and the "quality" and diversity of outputs is tunable based on the sampling strategy and sequences used to condition the generation. In contrast to smaller, simpler generative models, the outputs of GPT-style models are highly tunable. The goal here is not to exhaustively investigate prompt engineering and representation learning, which have massive scope, but instead to demonstrate interesting capabilities of ChemGPT and provoke new research questions.

We consider choosing a molecule as the basis for generation (Figure 6a) and picking a scaffold from the molecule, or otherwise fixing some part of the molecule. This will be used as a conditioning sequence for generation, and represents a potential use case for a generative LLM: pre-training and/or fine-tuning the LLM on a particular region of chemical space and using its generative capabilities to explore the chemical space around a known hit or lead compound in a drug discovery campaign. Molecules are then generated using the conditioning sequence with either top-k/nucleus sampling or beam search. We may be interested in generating new samples that have similar properties to our original molecule, or in generating samples that preserve a scaffold or substructure but otherwise have significantly different properties.

We consider the distribution of molecular properties of samples from top-k sampling and beam search with reference to an original training molecule. We find that the property distributions of samples vary considerably with sampling parameters, which introduce many degrees of freedom in the generation process [43]. These include the number of beams, B, used in beam search, which explores a graph of generation possibilities by choosing samples that have the highest overall probability according to the model. Greedy search (not considered here) always selects the next token with the highest probability, hence

beam search is guaranteed to find higher probability samples than a purely greedy strategy. The softmax temperature, T, controls the sensitivity of sampling to low probability tokens. Lowering T makes the distribution sharper, decreasing the likelihood of low probability tokens. For chemical generation, this results in less random, more repetitive samples. Conversely, increasing T results in more random, less repetitive samples. In top-k sampling [44], the k most likely next tokens are considered at each step of generation. This limits the sampling to a small set of high likelihood tokens, while expanding the diversity of sampling beyond greedy and beam search. In top-p (nucleus) sampling, the smallest set of tokens with cumulative probability $> p$ are chosen. This dynamically changes the number of possible next tokens according to the changing probability distributions. We combine top-k and top-p sampling in our experiments, which avoids low probability tokens while providing more randomness and "surprising" outputs than beam search [43]. We also vary the number of generated samples, n_s, and the prompt length, l_{max} (Figure 6b).

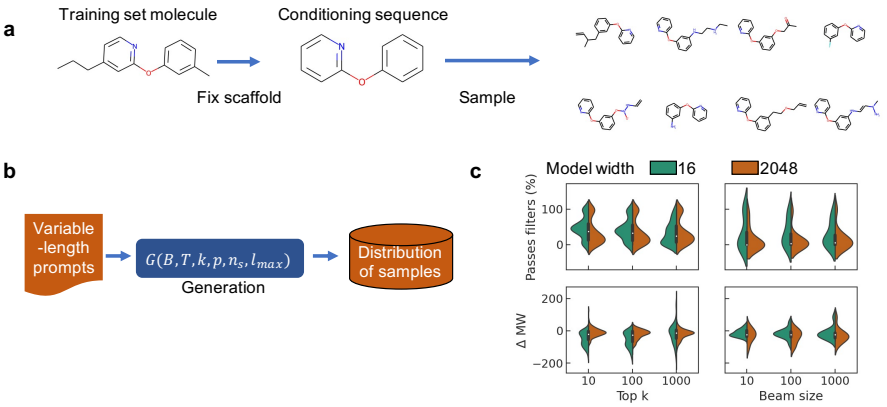

Fig. 6 ChemGPT outputs are controllable via sampling strategy and prompt engineering. (**a**) A scaffold or fragment from a molecule is fixed and used as a conditioning sequence for ChemGPT generation. (**b**) n_s samples are generated from a distribution of prompts with varying sampling strategy parameters (B, T, k, p) and prompt lengths (l_{max}). (**c**) Different model sizes and sampling strategies produce tunable distributions of samples.

To better understand the effects of sampling strategy, we repeat generation for ten different molecules randomly chosen from PubChem10M, and take prompts of length $l_{max} \in (5, 10, 20, l-3)$ tokens for each molecule, where l is the length of the original training molecule (Figure 6c). For the smallest and largest pre-trained ChemGPT models, we then plot the distributions of the percentages of samples that pass MOSES filters and the difference in molecular weight of generated samples compared to the original molecules used for conditioning (Figure 6c). The smaller model (green shading in Figure 6c) generally outperforms the larger model (orange) in generating samples that pass filters, for both top-k and beam search, except for $k = 1000$. However, the larger model generates samples that are more uniformly distributed with respect to

molecular weight. The k parameter in top-k sampling has a limited effect for the large model, but tends to shift the samples to a lower pass filter rate for the small model. Changes in the beam size can induce shifts from multi-modal to unimodal distributions of samples. Overall, the property distributions of samples are highly dependent on the sampling strategy, sampling parameters, and model size, suggesting that the complexity of ChemGPT introduces many new and important degrees of freedom for generative modeling. Importantly, beam search is typically employed in sampling from chemical generative models and this restricts the diversity of outputs compared to top-k/nucleus sampling.

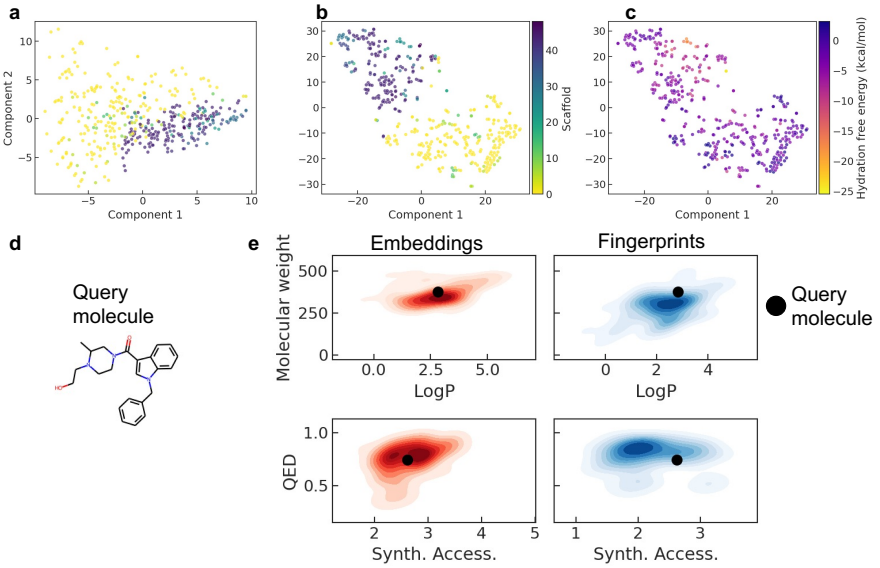

Fig. 7 ChemGPT embeddings can be used for unsupervised representation learning, clustering, and similarity search. (**a**) Principal component analysis and (**b**) t-SNE of ChemGPT embeddings for molecules in the FreeSolv dataset. Molecules exhibit clustering in the embedding space with respect to molecular scaffold and (**c**) hydration free energy, without finetuning on the dataset. (**d**) An example query molecule from the Enamine HTS Collection. (**e**) Contour plot of properties (molecular weight, LogP, quantitative estimate of druglikeness, and synthetic accessibility) of 100 nearest-neighbors identified from ChemGPT embeddings (red) and fingerprints (blue). Property values for the query molecule are shown by a black dot. Nearest neighbors identified from embeddings have closer mean and median property values to the query molecule than nearest neighbors from fingerprints.

Large chemical language models are self-supervised representation learners. A particularly exciting prospective application of large, pre-trained chemical models is to create a general representation learner that operates across diverse chemistries [45–47]. To demonstrate the representation learning capabilities of ChemGPT, we show the illustrative examples of clustering and nearest-neighbor similarity search using unsupervised embeddings. We choose a representative dataset from Therapeutic Data Commons [15] and

MoleculeNet [14], the FreeSolv [48] dataset of 642 druglike molecules and their hydration free energies in water.

We generate embeddings for the FreeSolv dataset from the hidden states in the last layer of ChemGPT models pre-trained on PubChem10M, *without fine-tuning on FreeSolv*. This is intended to simulate the usage of pre-trained ChemGPT models in general representation learning settings on new, previously unseen chemical spaces. Of course, the model could be fine-tuned on a target dataset to improve the quality of embeddings. We project the 1024-dimensional embeddings down to two dimensions using principal component analysis (PCA) (Figure 7a) and t-distributed stochastic neighbor embedding (t-SNE) ((Figure 7b). For both dimensionality reduction techniques, the embeddings cluster based on the molecular scaffolds. Additionally, the t-SNE embeddings are clustered with respect to the target property, hydration free energy (Figure 7c). The structure in the latent space suggests that the learned embeddings from ChemGPT are chemically meaningful and useful for reasoning about chemical spaces outside of the pre-training set. We also show (Figure A.6) unsupervised ChemGPT embeddings for the Tox21 [49] dataset, which is a binary classification task from toxicity measurements for 7,831 small molecules on 12 different targets. These embeddings cluster active and inactive compounds, again without finetuning on the dataset. The FreeSolv and Tox21 tasks show the relevance of unsupervised ChemGPT embeddings for both physicochemical and ADMETox tasks, where properties depend on intrinsic qualities of the molecules.

Learned embeddings can also be used for a common cheminformatics task: similarity search and retrieval. Given some query molecule (Figure 7d), the goal is to find "chemically similar" molecules that are nevertheless different and diverse with respect to the initial query. This workflow may be used to design a library of chemical matter for high-throughput virtual screening or assays, or to preserve desirable characteristics of a query molecule while "hopping" to new molecular scaffolds or discarding undesirable moieties. The molecules identified as similar depend sensitively on the similarity/distance metric used, and the molecular representation. We propose a general representation learning problem defined as returning a distribution of nearest neighbors that comprise a smooth manifold in property space. That is, regardless of the representation and distance metric, the objective is to identify molecules that are similar to the query molecule in property space. This framing transforms the abstract problem of defining "chemical similarity" into the more tractable, easily understood problem of identifying molecules with similar properties. For simplicity, we consider molecular weight (MW), partition coefficient (LogP), quantitative estimate of druglikeness (QED), and synthetic accessbility (SA) [50], although any accessible properties could be used.

To benchmark our similarity search performance, we compare to the traditional method of encoding molecules using Morgan fingerprints and computing similarities using a Tanimoto distance. As an illustrative example, we show a query molecule (Figure 7d) from the Enamine HTS Collection in property

space and the property distributions of its 100 nearest neighbors computed via ChemGPT embeddings (Figure 7e, red contour plots) and the fingerprint method (Figure 7e, blue contour plots). Again, ChemGPT is not fine-tuned on this dataset, the pre-trained model is able to generate "chemically meaningful" embeddings for Enamine HTS, which is a standard chemical library for high throughput screening. The ChemGPT embeddings are dense, real-valued, high-dimensional vectors, so a combination of dimensionality reduction and flexible choice of distance metric (L1, L2, cosine, etc.) yields a smooth and tunable distribution of nearest neighbors in property space. In this example, we use PCA to reduce the embedding dimension from 1024 to 100 and compute distances with a cosine similarity.

Although it is difficult to see by eye in the contour plots in Figure 7e, by computing the statistics of the nearest neighbor property distributions we find that the nearest neighbors identified from learned embeddings are closer to the query in property space than those from non-learned fingerprints, using the following equation,

$$\Delta_p = \|\overline{p}_{NN} - p_{query}\|. \tag{9}$$

We compute the mean of the property values, $\overline{p}_{NN}$ for properties p (LogP, synthetic accessibility, molecular weight, and QED), of the 100 nearest neighbors identified with embedding and fingerprint methods and calculate the difference (Δ_p) between the mean property value of the nearest neighbors and the query molecule p_{query}. We report these values in Table A.2. In this example, for all properties considered, the nearest neighbors from embeddings are closer to the query molecule in property space. The same trend is observed if we consider median rather than mean property values of nearest neighbors. For some applications, this may be desirable, for others the goal may be to discover nearest neighbors that are chemically similar, but with significantly different properties, e.g., avoiding activity cliffs.

4 Discussion

In this paper, we developed and applied strategies for scaling large chemical language models and GNN interatomic potentials. To enable the efficient scaling of deep chemical models under computational resource constraints, we introduced Training Performance Estimation (TPE), a generalization of Training Speed Estimation that significantly reduces the computational costs of hyperparameter optimization and model selection for both chemical models trained on large datasets and on GNN interatomic potentials. The use of TPE enabled large-scale experiments, training GPT-style chemical models with over one billion non-embedding parameters on nearly ten million molecules. It also made training tractable for invariant and equivariant GNNs with a wide range of model capacities on up to 100 thousand 3D molecular geometries (4.5 million force labels). We discovered empirical power law "neural scaling" behavior that quantifies how converged model loss depends on the scale of model and dataset size over many orders of magnitude. These results enable optimal

allocation of computational and data budgets for maximally efficient model performance improvements, and make scalable scientific deep learning more accessible to a broader community of researchers. Finally, we showed that the outputs of large chemical generative models are tunable via prompt engineering and sampling strategies, and that the model embeddings can be used for unsupervised representation learning and similarity search.

A key motivation for this work was to begin to investigate scientific deep learning at scale – using large models on massive datasets. Unlike areas such as natural language processing (NLP) and computer vision (CV), where scale has proven to be a key ingredient to recent breakthroughs, scientific domains are built on physics-based priors that impose high levels of structure on data generation processes. For this reason, it is unclear what, if any, benefits massive scale will confer to scientific deep learning. And although there is significant engineering effort required to train and deploy large-scale deep learning models, the study of such models is of inherent scientific interest because these models may display surprising, emergent characteristics that are not predictable by extrapolating from small scales. There is an opportunity in scientific deep learning to anticipate trends towards so-called foundational models, similar to those seen in NLP and CV, to ensure that such important investigations are not limited to a few extremely well-resourced research organizations.

Our work, specifically our findings with ChemGPT, suggests a "bittersweet lesson" for scientific deep learning – although incorporating principled physical priors and domain knowledge into scientific machine learning frameworks will continue to play an important role in this field, achieving sheer massive scale in model size and data diversity is likely to be a key component in some scientific deep learning breakthroughs. This finding presents exciting opportunities to further explore what techniques and lessons can be taken from traditional deep learning application domains. The neural scaling results indicate that traditional techniques for training large language models apply when scaling models using string-based representations of molecules. The additional data- and model-efficiency gained by careful incorporation of physical priors observed in the neural force field experiments suggests non-trivial interactions with typical scaling techniques. The trend towards larger models introduces challenges in any field, namely, increasing resource demands and environmental impact. We provide strategies and examples for scalable chemical deep learning here, but studies at massive scale remain inaccessible to most researchers. Likewise, strategies for reducing the environmental impact and carbon footprint of large-scale deep learning exist, but the engineering resources and hardware needed to do so are concentrated in large organizations [51, 52]. However, a growing body of research suggests that large models accelerate the optimization problem and finding of robust solutions, and that once trained, techniques like model distillation, quantization, and dataset pruning are effective at reducing model and dataset sizes while retaining performance.

The work presented here invites many directions for future research. Our goal is to enable neural scaling studies and investigate the potential benefits of scaling for chemical deep learning. Future work will investigate the complex relationships between pre-training performance improvements and downstream tasks. Large, pre-trained models can be fine-tuned on any smaller chemical space of interest to investigate the benefits of transfer learning and potential for generalization. Similarly, further study of the representations learned by large pre-trained models is warranted, including the proposal of new benchmarks for evaluating the quality of learned representations beyond downstream property prediction tasks. We were limited by computational and engineering constraints, but much larger chemical models and pre-training datasets clearly warrant investigation; this will require more advanced model parallelism approaches. Finally, the effects of physics-based priors on scaling behavior give a rich description of how the incorporation of physics, known empirical relationships, and other forms of knowledge into machine learning frameworks impact both learning quality and efficiency. Future work in this area is well-poised to yield fundamental advances in scientific machine learning.

5 Methods

In this section, we report settings for the experiments performed in this paper. All experiments described in this paper were conducted on NVIDIA Volta V100 graphics processing units (GPUs) with 32 GB of memory per node and 2 GPUs per node. All models were implemented in PyTorch [53] and trained with the Distributed Data Parallel (DDP) accelerator [54], the NVIDIA Collective Communication Library (NCCL), PyTorch Lightning [55] and `LitMatter` [56] for multi-GPU, multi-node training.

Large Language Models (LLMs). The ChemGPT model architecture is based on the GPT-Neo [27, 28] transformer implementation in HuggingFace [57]. The model has 24 layers, with variable width, w, where $w \in (16, 32, 64, 128, 256, 512, 1024, 2048)$ and w determines the model size. Model sizes range from 77,600 to 1,208,455,168 non-embedding parameters. The model is trained via stochastic gradient descent (SGD) with the AdamW [58] optimizer, using a learning rate of $2 * 10^{-5}$, a per-GPU batch size of 8, and a constant learning rate schedule with 100 warmup steps for scaling experiments. Models were trained for 10 epochs in a self-supervised manner, with a cross-entropy loss for causal language modeling.

The training dataset for scaling experiments is PubChem10M [8], a set of 10 million SMILES strings. 5% of the data is randomly sampled and held out as a fixed validation set of size 500,000 molecules. Variable training datasets with sizes 10^n, where $n \in (2, 3, 4, 5, 6)$, were used. The largest training dataset includes all molecules in PubChem10M, excluding the validation set. The maximum vocabulary size was 10,000 and the maximum sequence length was 512 tokens. SMILES strings were converted to SELFIES using version 1.0.4 of the SELFIES library [29]. SELFIES were tokenized by splitting individual strings

into minimally semantically meaningful tokens denoted by brackets, including start-of-string, end-of-string, and padding tokens. Dataset sizes range from 51,200 to 304,656,384 tokens.

Graph Neural Networks (GNNs). We train GNNs to predict the forces of molecular geometries. Force-only training ($\alpha_E = 0$ in Eq. 5) was used for neural scaling experiments to improve convergence and avoid issues with systematic drift in predicted energies, which we identified during the course of this work and plan to address in future work. We use the SchNet [59], PaiNN [36], Allegro [10], and SpookyNet [37] models. Model implementations are from the `NeuralForceField` repository [34, 60, 61] and the `Allegro` repository [10]. Model sizes (w in Equation 6) were varied between 16, 64, and 256, while the number of layers/convolutions (d in Equation 6) was chosen to be 2, 3, or 4. A 5 Å nearest-neighbor cutoff was used. All other model hyperparameters were set to default values from the original implementations. GNN models were trained with SGD using the Adam [62] optimizer.

A learning rate scheduler reduced the learning rate by 0.5× after 30 epochs without improvement in the validation loss, with a minimum learning rate of 10^{-7}. Early stopping was applied after 50 epochs without improvement in the validation loss, and training was capped at 1000 epochs. Initial learning rates of 10^{-3}, 10^{-4}, and 10^{-5}, and per-GPU batch sizes of 4, 8, 16, 32, and 64 were used during hyperparameter optimization experiments, while keeping the network architecture hyperparameters fixed. Models were trained for 50 epochs during hyperparameter optimization to approximate a full training budget, with a limited percentage of the total training budget used to calculate TSE.

The training dataset was assembled from ANI-1x [42, 63], which contains energies and forces from 5 million density functional theory calculations for small molecules. A fixed validation dataset of 50,000 frames was held out by random sampling. Different splits of training were taken with sizes 10^n where $n \in (2, 3, 4, 5, 6)$. Training datasets for TPE were assembled by randomly sampling 1,000 structures from molecular dynamics (MD) trajectories for each of the 10 molecules available in the revised MD-17 [41] dataset, for a total of 10,000 training samples. A validation dataset of equal size was constructed from the remaining geometries. Revised MD-17 is an updated version of the MD-17 [64] dataset, recomputed at the PBE/def2-SVP level of theory with strict convergence criteria to remove noise found in the original MD-17 dataset.

6 Data availability

PubChem data for pre-training large language models is available through DeepChem [65]. The Molecular Sets (MOSES) data is available through GitHub [39]. The Enamine HTS Collection is available here. The ANI-1x data for training neural force fields is available through Figshare [42]. The revised MD-17 dataset was accessed here.

7 Code availability

Code used to perform the experiments and Training Performance Estimation (TPE) reported in this paper is available via GitHub in the `LitMatter` repository [56]. Neural force field model code is available here and Allegro model code is available here. The GPT-Neo model that ChemGPT is based on is available here. PubChem10M tokenizers using SELFIES versions 1.0.4 and 2.0.0 are available through the `LitMatter` repository and the Hugging Face Hub. Because of the significant computational resources required to train large models and the value of those models, pre-trained model checkpoints for ChemGPT are available via the Hugging Face Hub. Pre-trained model checkpoints for PaiNN and Allegro are available through Figshare.

Supplementary information. Supplementary Figures A.1 - A.5. Supplementary Table A.1.

Acknowledgments. The authors acknowledge the MIT SuperCloud [66] and the Lincoln Laboratory Supercomputing Center for providing HPC and consultation resources that contributed to the research results reported within this paper. The authors acknowledge the MIT SuperCloud team: William Arcand, David Bestor, William Bergeron, Chansup Byun, Matthew Hubbell, Michael Houle, Mike Jones, Jeremy Kepner, Anna Klein, Peter Michaleas, Lauren Milechin, Julie Mullen, Andrew Prout, Albert Reuther, Antonio Rosa, and Charles Yee. The authors acknowledge Judy Marchese for proofreading.

Contributions. NCF, RGB, CWC, and VG conceptualized the project. NCF developed the methodology, performed the investigations, and made the visualizations. All authors contributed to analyzing the results, writing, and editing the manuscript. VG supervised the project.

Declarations

Funding. This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001, and by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator under Cooperative Agreement Number FA8750-19-2-1000. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering, or the United States Air Force. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Competing interests. The authors declare no competing interests.

Appendix A Extended Data

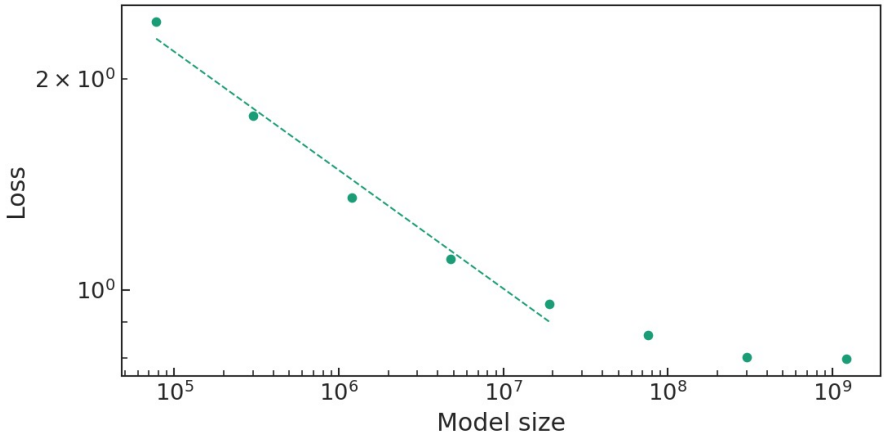

Fig. A.1 Calculating neural scaling power law for ChemGPT. Pre-training test loss versus model size for a dataset of 304,656,384 tokens. The power law fit breaks down for large model sizes, indicating a data-resolution limited regime. $R^2 = 0.98$ for the fitted region and the scaling exponent $\beta = 0.17 \pm 0.01$.

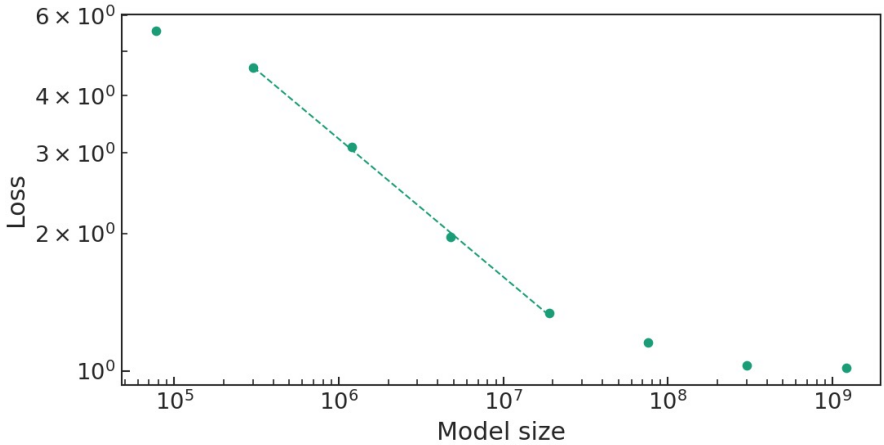

Fig. A.2 Calculating neural scaling power law for ChemGPT. Pre-training test loss versus model size for a dataset of 51,200,000 tokens. The power law fit breaks down for large model sizes, indicating a data-resolution limited regime. $R^2 = 0.99$ for the fitted region and the scaling exponent $\beta = 0.30 \pm 0.01$.

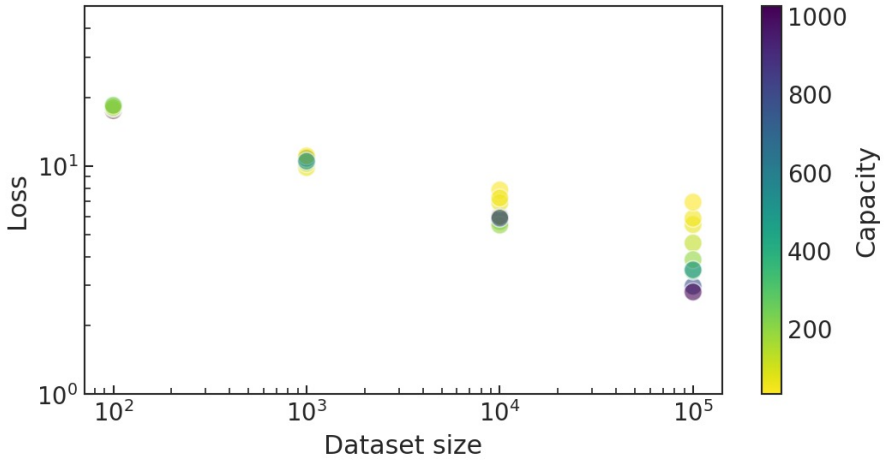

Fig. A.3 Neural scaling of SchNet model performance (test loss) as a function of model capacity and dataset size. SchNet is trained to predict atomic forces from density functional theory calculations on small organic molecules from the ANI-1x dataset. Performance improvements are seen for models with greater capacity and continuous improvements are observed with increasing dataset size.

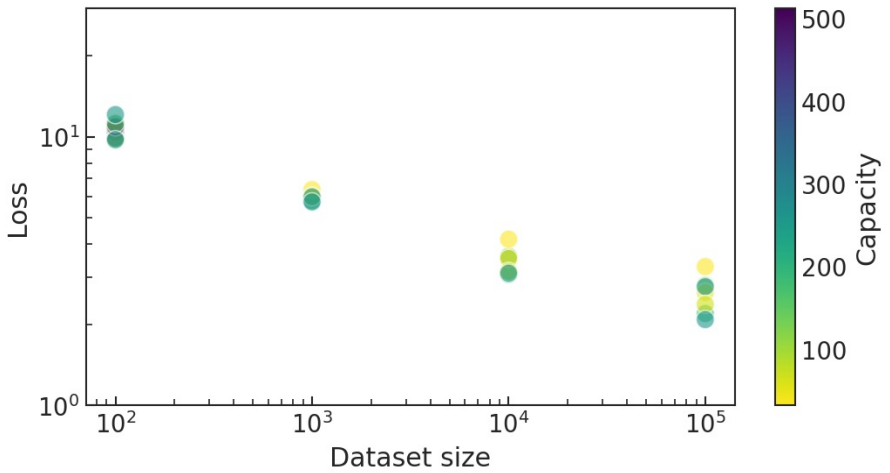

Fig. A.4 Neural scaling of Allegro model performance (test loss) as a function of model capacity and dataset size. Allegro is trained to predict atomic forces from density functional theory calculations on small organic molecules from the ANI-1x dataset. Performance improvements are seen for models with greater capacity and continuous improvements are observed with increasing dataset size. Due to GPU memory constraints, the capacity of Allegro models is capped at 512.

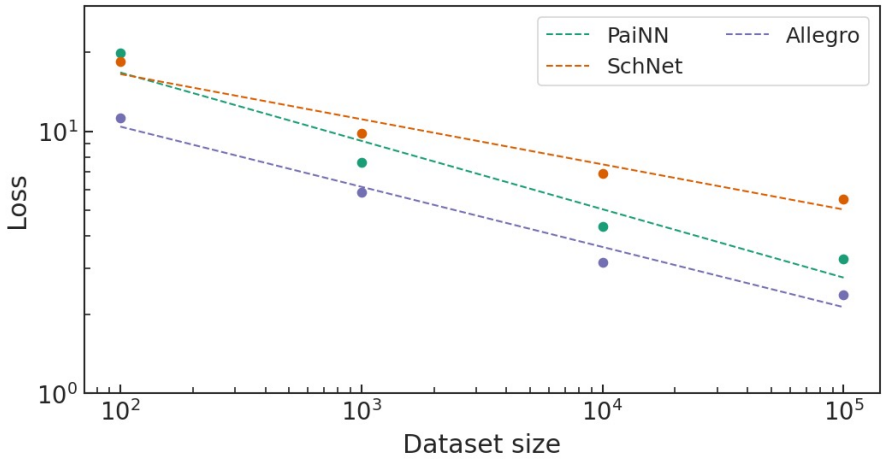

Fig. A.5 Calculating neural scaling power laws for neural force fields. Test loss versus dataset size for PaiNN, Allegro, and SchNet models with fixed capacity, 64.

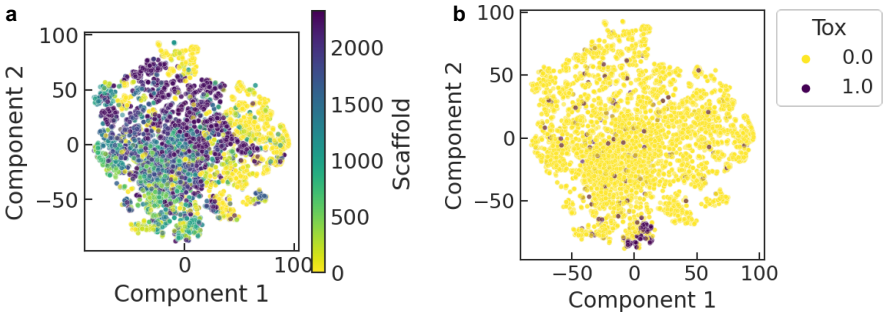

Fig. A.6 Unsupervised ChemGPT embeddings cluster the Tox21 dataset. t-SNE plots of ChemGTP embeddings for Tox21 showing clustering with respect to (**a**) molecular scaffold and (**b**) activity.

Table A.1 Power laws for neural force field scaling.

Model	R^2	**Scaling exponent** β
SchNet	0.95	0.17 ± 0.03
PaiNN	0.94	0.26 ± 0.05
Allegro	0.97	0.23 ± 0.03

Table A.2 Difference (smaller is better) between mean property values and query molecule property values for nearest neighbors from ChemGPT embeddings and fingerprints.

Property	Δ_p **embedding**	Δ_p **fingerprint**
LogP	**0.03**	0.53
Synth. Access.	**0.05**	0.41
Molecular Weight	**20.92**	93.14
QED	**0.00**	0.04

References

[1] Sejnowski, T.J.: The unreasonable effectiveness of deep learning in artificial intelligence. Proceedings of the National Academy of Sciences **117**(48), 30033–30038 (2020)

[2] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[3] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-Shot Text-to-Image Generation. arXiv (2021). https://doi.org/10.48550/ARXIV.2102.12092. https://arxiv.org/abs/2102.12092

[4] Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J.Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D.E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Kohd, P.W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X.L., Li, X., Ma, T., Malik, A., Manning, C.D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J.C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J.S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A.W., Tramèr, F., Wang, R.E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S.M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., Liang, P.: On the Opportunities and Risks of Foundation Models (2021)

[5] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., *et al.*: Highly accurate protein structure prediction with alphafold. Nature **596**(7873), 583–589 (2021)

[6] Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K., Heras-Domingo, J., Ho, C., Hu, W., *et al.*: Open catalyst 2020 (oc20) dataset and community challenges. ACS Catalysis **11**(10), 6059–6072 (2021)

[7] Sriram, A., Das, A., Wood, B.M., Goyal, S., Zitnick, C.L.: Towards training billion parameter graph neural networks for atomic simulations. arXiv preprint arXiv:2203.09697 (2022)

[8] Chithrananda, S., Grand, G., Ramsundar, B.: Chemberta: Large-scale self-supervised pretraining for molecular property prediction. arXiv preprint arXiv:2010.09885 (2020)

[9] Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J.P., Kornbluth, M., Molinari, N., Smidt, T.E., Kozinsky, B.: Se (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. arXiv preprint arXiv:2101.03164 (2021)

[10] Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C.J., Kornbluth, M., Kozinsky, B.: Learning local equivariant representations for large-scale atomistic dynamics. arXiv preprint arXiv:2204.05249 (2022)

[11] Trewartha, A., Walker, N., Huo, H., Lee, S., Cruse, K., Dagdelen, J., Dunn, A., Persson, K.A., Ceder, G., Jain, A.: Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. Patterns **3**(4), 100488 (2022)

[12] Kalinin, S.V., Ziatdinov, M., Sumpter, B.G., White, A.D.: Physics is the New Data. arXiv (2022). https://doi.org/10.48550/ARXIV.2204.05095. https://arxiv.org/abs/2204.05095

[13] Coley, C.W.: Defining and exploring chemical spaces. Trends in Chemistry **3**(2), 133–145 (2021)

[14] Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.: Moleculenet: a benchmark for molecular machine learning. Chemical science **9**(2), 513–530 (2018)

[15] Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C.W., Xiao, C., Sun, J., Zitnik, M.: Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. arXiv preprint arXiv:2102.09548 (2021)

[16] Pappu, A., Paige, B.: Making Graph Neural Networks Worth It for Low-Data Molecular Machine Learning (2020)

[17] Gasteiger, J., Shuaibi, M., Sriram, A., Günnemann, S., Ulissi, Z., Zitnick, C.L., Das, A.: How do graph networks generalize to large and diverse molecular systems? arXiv preprint arXiv:2204.02782 (2022)

[18] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural

language models. arXiv preprint arXiv:2001.08361 (2020)

[19] Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T.B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D.M., Schulman, J., Amodei, D., McCandlish, S.: Scaling Laws for Autoregressive Generative Modeling (2020)

[20] Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., Villalobos, P.: Compute Trends Across Three Eras of Machine Learning (2022)

[21] Yang, G., Hu, E.J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., Gao, J.: Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. arXiv preprint arXiv:2203.03466 (2022)

[22] Ru, B., Lyle, C., Schut, L., Fil, M., van der Wilk, M., Gal, Y.: Speedy performance estimation for neural architecture search. arXiv preprint arXiv:2006.04492 (2020)

[23] Bahri, Y., Dyer, E., Kaplan, J., Lee, J., Sharma, U.: Explaining Neural Scaling Laws (2021)

[24] Krenn, M., Ai, Q., Barthel, S., Carson, N., Frei, A., Frey, N.C., Friederich, P., Gaudin, T., Gayle, A.A., Jablonka, K.M., Lameiro, R.F., Lemm, D., Lo, A., Moosavi, S.M., Nápoles-Duarte, J.M., Nigam, A., Pollice, R., Rajan, K., Schatzschneider, U., Schwaller, P., Skreta, M., Smit, B., Strieth-Kalthoff, F., Sun, C., Tom, G., von Rudorff, G.F., Wang, A., White, A., Young, A., Yu, R., Aspuru-Guzik, A.: SELFIES and the future of molecular string representations. arXiv (2022). https://doi.org/10.48550/ARXIV.2204.00056. https://arxiv.org/abs/2204.00056

[25] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., *et al.*: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)

[26] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)

[27] Black, S., Leo, G., Wang, P., Leahy, C., Biderman, S.: GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. Zenodo. If you use this software, please cite it using these metadata. (2021). https://doi.org/10.5281/zenodo.5297715. https://doi.org/10.5281/zenodo.5297715

[28] Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al.: The pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027 (2020)

[29] Krenn, M., Häse, F., Nigam, A., Friederich, P., Aspuru-Guzik, A.: Self-referencing embedded strings (selfies): A 100% robust molecular string representation. Machine Learning: Science and Technology **1**(4), 045024 (2020)

[30] Skinnider, M.A., Stacey, R.G., Wishart, D.S., Foster, L.J.: Chemical language models enable navigation in sparsely populated chemical space. Nature Machine Intelligence **3**(9), 759–770 (2021)

[31] Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. Advances in Neural Information Processing Systems **13** (2000)

[32] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[33] Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B.A., Thiessen, P.A., Yu, B., *et al.*: Pubchem in 2021: new data content and improved web interfaces. Nucleic acids research **49**(D1), 1388–1395 (2021)

[34] Schwalbe-Koda, D., Tan, A.R., Gómez-Bombarelli, R.: Differentiable sampling of molecular geometries with uncertainty-based adversarial attacks. Nature communications **12**(1), 1–12 (2021)

[35] Schütt, K.T., Sauceda, H.E., Kindermans, P.-J., Tkatchenko, A., Müller, K.-R.: Schnet–a deep learning architecture for molecules and materials. The Journal of Chemical Physics **148**(24), 241722 (2018)

[36] Schütt, K., Unke, O., Gastegger, M.: Equivariant message passing for the prediction of tensorial properties and molecular spectra. In: International Conference on Machine Learning, pp. 9377–9388 (2021). PMLR

[37] Unke, O.T., Chmiela, S., Gastegger, M., Schütt, K.T., Sauceda, H.E., Müller, K.-R.: Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects. Nature communications **12**(1), 1–14 (2021)

[38] Loukas, A.: What graph neural networks cannot learn: depth vs width. arXiv preprint arXiv:1907.03199 (2019)

[39] Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V.,

Veselov, M., *et al.*: Molecular sets (moses): a benchmarking platform for molecular generation models. Frontiers in pharmacology **11**, 1931 (2020)

[40] McCandlish, S., Kaplan, J., Amodei, D., Team, O.D.: An Empirical Model of Large-Batch Training. arXiv (2018). https://doi.org/10.48550/ARXIV.1812.06162. https://arxiv.org/abs/1812.06162

[41] Christensen, A.S., von Lilienfeld, O.A.: On the role of gradients for machine learning of molecular energies and forces. Machine Learning: Science and Technology **1**(4), 045018 (2020)

[42] Smith, J.S., Zubatyuk, R., Nebgen, B., Lubbers, N., Barros, K., Roitberg, A.E., Isayev, O., Tretiak, S.: The ani-1ccx and ani-1x data sets, coupled-cluster and density functional theory properties for molecules. Scientific data **7**(1), 1–10 (2020)

[43] von Platen, P.: How to generate text: using different decoding methods for language generation with transformers. Hugging Face (2020)

[44] Fan, A., Lewis, M., Dauphin, Y.: Hierarchical neural story generation. arXiv preprint arXiv:1805.04833 (2018)

[45] Huang, B., Von Lilienfeld, O.A.: Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity. The Journal of Chemical Physics **145**(16), 161102 (2016)

[46] Chuang, K.V., Gunsalus, L.M., Keiser, M.J.: Learning molecular representations for medicinal chemistry: miniperspective. Journal of Medicinal Chemistry **63**(16), 8705–8722 (2020)

[47] Sabando, M.V., Ponzoni, I., Milios, E.E., Soto, A.J.: Using molecular embeddings in qsar modeling: does it make a difference? Briefings in bioinformatics **23**(1), 365 (2022)

[48] Mobley, D.L., Guthrie, J.P.: Freesolv: a database of experimental and calculated hydration free energies, with input files. Journal of computer-aided molecular design **28**(7), 711–720 (2014)

[49] Huang, R., Xia, M., Nguyen, D.-T., Zhao, T., Sakamuru, S., Zhao, J., Shahane, S.A., Rossoshek, A., Simeonov, A.: Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. Frontiers in Environmental Science **3**, 85 (2016)

[50] Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of cheminformatics **1**(1), 1–11 (2009)

[51] Patterson, D.: The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. TechRxiv (2022). https://doi.org/10.36227/techrxiv.19139645.v1. https://www.techrxiv.org/articles/preprint/The_Carbon_Footprint_of_Machine_Learning_Training_Will_Plateau_Then_Shrink/19139645/1

[52] Frey, N.C., Li, B., McDonald, J., Zhao, D., Jones, M., Bestor, D., Tiwari, D., Gadepally, V., Samsi, S.: Benchmarking Resource Usage for Efficient Distributed Deep Learning. arXiv (2022). https://doi.org/10.48550/ARXIV.2201.12423. https://arxiv.org/abs/2201.12423

[53] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc., ??? (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[54] Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., Chintala, S.: PyTorch Distributed: Experiences on Accelerating Data Parallel Training (2020)

[55] Falcon et al., W.: Pytorch lightning. GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning **3** (2019)

[56] Frey, N.C., Samsi, S., McDonald, J., Li, L., Coley, C.W., Gadepally, V.: Scalable geometric deep learning on molecular graphs. In: NeurIPS 2021 AI for Science Workshop (2021)

[57] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)

[58] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)

[59] Schütt, K.T., Kindermans, P.-J., Sauceda, H.E., Chmiela, S., Tkatchenko, A., Müller, K.-R.: SchNet: A continuous-filter convolutional neural network for modeling quantum interactions (2017)

[60] Axelrod, S., Gomez-Bombarelli, R.: Molecular machine learning with conformer ensembles. arXiv preprint arXiv:2012.08452 (2020)

[61] Axelrod, S., Shakhnovich, E., Gómez-Bombarelli, R.: Excited state, non-adiabatic dynamics of large photoswitchable molecules using a chemically transferable machine learning potential. arXiv preprint arXiv:2108.04879 (2021)

[62] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)

[63] Smith, J.S., Nebgen, B., Lubbers, N., Isayev, O., Roitberg, A.E.: Less is more: Sampling chemical space with active learning. The Journal of chemical physics **148**(24), 241733 (2018)

[64] Chmiela, S., Tkatchenko, A., Sauceda, H.E., Poltavsky, I., Schütt, K.T., Müller, K.-R.: Machine learning of accurate energy-conserving molecular force fields. Science advances **3**(5), 1603015 (2017)

[65] Ramsundar, B., Eastman, P., Walters, P., Pande, V., Leswing, K., Wu, Z.: Deep Learning for the Life Sciences. O'Reilly Media, ??? (2019). https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837

[66] Reuther, A., Kepner, J., Byun, C., Samsi, S., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Jones, M., Klein, A., Milechin, L., Mullen, J., Prout, A., Rosa, A., Yee, C., Michaleas, P.: Interactive supercomputing on 40,000 cores for machine learning and data analysis. In: 2018 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6 (2018). IEEE