Fine-Tuned Language Models Generate Stable Inorganic Materials as Text

Nate Gruver[1] **Anuroop Sriram**[2] **Andrea Madotto**[2]
Andrew Gordon Wilson[1] **C. Lawrence Zitnick**[2] **Zachary Ulissi**[2]
[1]NYU [2]Meta FAIR

Abstract

Deep learning models have drastically accelerated materials discovery by accelerating predictive computational simulations like density functional theory (DFT). Large open computational materials databases such as the Materials Project or OQMD contain O(10^6) known structures, and it is now straightforward to search those databases for materials with exciting properties. However, these databases are limited to experimentally known materials or candidates discovered in high-throughput computational campaigns. Many state-of-the-art engineering advances in solar photovaltaics, battery electrodes, and catalysts are made by discovering materials with outstanding properties that have not yet been discovered. Generative models are a natural solution to expand families of interest through sampling. While popular methods are typically constructed from variational autoencoders or diffusion models, we propose fine-tuning large language models for generation of stable materials. While unorthodox, fine-tuning large language models on text-encoded atomistic data is simple to implement yet reliable, with around 90% of sampled structures obeying physical constraints on atom positions and charges. Using energy of hull calculations from both learned ML potentials and gold-standard DFT calculations, we show that our strongest model (fine-tuned LLaMA-2 70B) can generate materials predicted to be metastable at about twice the rate (49% vs 28%) of CDVAE, a competing diffusion model. Because of text prompting's inherent flexibility, our models can simultaneously be used for unconditional generation of stable material, infilling of partial structures and text-conditional generation. Finally, we show that language models' ability to capture key symmetries of crystal structures improves with model scale, suggesting that the biases of pretrained LLMs are surprisingly well-suited for atomistic data.

1 Introduction

The training objective for autoregressive large language models (LLMs) is simple–just predict the next word–but their abilities are increasingly general. Despite being pre-trained solely on text, large language models have been applied to a wide variety of tasks, including image generation [6], image compression [10], and robotic planning [13]. From these results, it's evident that pre-trained LLMs can act as general-purpose priors, identifying the most salient patterns in input data [17]. In fact, LLMs can be viewed as a compression of human reasoning available in the form of internet text [37]. More effective compression of complex phenomena leads naturally to stronger abilities to compress downstream data and extrapolate the most generalizable patterns, often with orders of magnitude less examples than models trained from scratch [5].

While fine-tuned LLMs have so far been particularly impactful for language and vision data, two modalities that are core to human cognition, their strengths as universal priors also make them promising for scientific data modalities (e.g. small molecules, proteins, or crystalline materials),

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

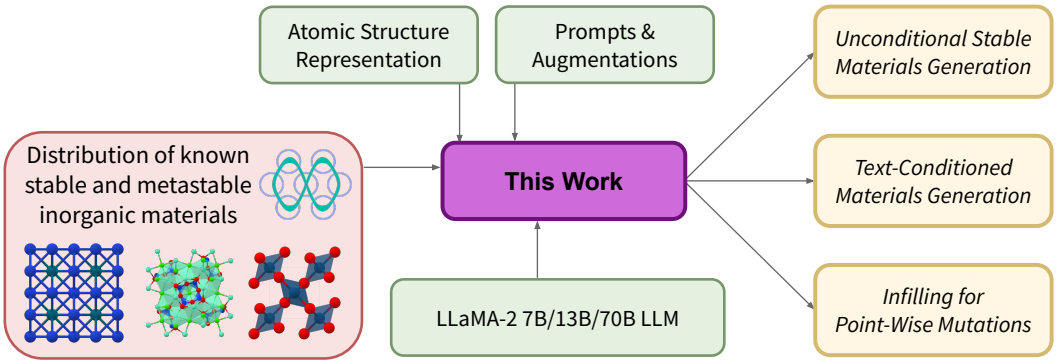

Figure 1: Overview of our approach to enable unconditional stable materials generation, text-condition materials generation, and structural infilling tasks. Base LLaMA2 models are fine-tuned using a variety of crystal atomic structure representations and task prompts, and trained on a database of known inorganic materials ([30].

where data can be small, multi-modal, and challenging for non-experts to interpret. LLMs have the potential to unite datasets within a strong pre-trained backbone and add an interpretable interface through text, which enables scientists to more easily specify desired properties and constraints. Lowering the barrier to expert-level atomistic design has the potential to unlock breakthroughs ranging from novel therapeutic drugs [4] to improved carbon capture [7] and beyond.

In this work, we show that fine-tuned LLMs can generate stable three-dimensional crystals as text (Figure 1). Our method is incredibly simple: first, encode crystals as new-line separated strings and combine with text instructions, then perform parameter efficient fine tuning (PEFT) on a base LLM (LLaMA-2) with a multitask curriculum and translation augmentations (Section 4). We evaluate our method with Materials Project data [22], comparing against an invariant diffusion model and a sequence model trained from scratch. Using both learned ML potentials and gold-standard DFT calculations, we show that our method can generate materials predicted to be stable at much higher rates than baseline methods. To understand the success of our fine-tuning approach compared to more domain-specific approaches, we probe the learned symmetry properties of our model, proposing a new metric for language models trained on atomistic data and examining the effect of model scale on learned invariance. Going beyond unconditional generation, We also show that our LLMs have other useful abilities within materials design, such as text-conditional generation and infilling, which can be used to optimize the properties of existing materials.[1]

2 Related Work

There are two central challenges in applying generative models to crystals and related atomistic modalities. The first challenge is that atoms are intrinsically both discrete and continuous objects, as they are associated with both an element identity and a position in three dimensional space. In the case of crystals, there are also continuous parameters that define a crystal's unit cell, the repeated pattern that is tiled infinitely in every direction (somewhat like a convolutional filter in computer vision). The second key challenge is the prevalence of symmetries in atomistic data. A unit cell, for example, is the common representation for crystals because it easily captures translation invariance, the fact that atoms can be shifted and wrapped around the unit cell while still representing the same underlying structure. Symmetries can pose challenges to deep learning models because they entail constraints on the functions that neural networks must learn.

Diffusion models Xie et al. [44] introduced crystal diffusion variational autoencoder (CDVAE) to directly deal with both of these challenges. CDVAE uses several individual generative models for discrete and continuous components that share a continuous (VAE) latent space. The chemical composition is reconstructed from this latent space using a language modeling head, while atom positions are generated with a denoising diffusion model [20]. Since CDVAE, Jiao et al. [23] has also extended diffusion processes to be used not just for the atom positions but also the lattice

[1] https://github.com/facebookresearch/crystal-llm

parameters and element identities. Both of these diffusion models were designed with a careful eye towards symmetries and are built on top of graph neural networks with strict invariance/equivariance properties.

Language models Recently Flam-Shepherd & Aspuru-Guzik [14] demonstrated an alternative approach to continuous denoising models and architectural invariances. Instead of treating discrete and continuous modalities separately, as in CDVAE, Flam-Shepherd & Aspuru-Guzik [14] uses sequences of discrete tokens to represent everything, including the digits of atomic coordinates. With all data encoded as tokens, standard language modeling methods designed for text can be applied with little to no modification. The simplicity of this method also makes it simple to adapt to many different kinds of molecular structures, including small molecules, protein binding pockets, and, of course, crystals. In lieu of architectural symmetries, augmentations of the training data are used to encourage learning known invariances. Flam-Shepherd & Aspuru-Guzik [14] demonstrates that language models trained from scratch on many common molecular datasets actually outperform popular domain-specific models, including CDVAE, in their ability to capture valid element compositions and high-level statistics of the training data. Similarly, Antunes et al. [3] also use language models to generate crystal structures as discrete sequences by training from scratch on millions of CIF strings.

LLMs as initializations or priors While language models can be trained from scratch for molecular modalities, several recent papers suggest that text-pretrained language models can be incredibly useful as initializations for non-text modalities. Chang et al. [6], for example, use a pre-trained T5 model to create a text-conditional image generator by extending the vocabulary with image tokens. Similarly, Delétang et al. [10] show that a pre-trained Chinchilla 70B can act as a more effective image compressor than domain-specific compression algorithms, and in the molecular space, Krause et al. [25] showed that LLMs fine-tuned on protein sequences can be used to generate antibodies with improved function.

Our work In this work, we show that pretrained LLMs are also incredibly useful for understanding and generating 3-dimensional atomic structures. By using a pre-trained LLM, we can achieve high rates of validity without crystal-specific tokenization [14] or millions of auxiliary structures [3]. Unlike models designed narrowly for crystal structures and symmetries [44], our model can also be easily extended to multiple crystal generation tasks and, in the future, to other atomistic modalities without any changes to the underlying model or training procedure. Building on the basic observations made by Flam-Shepherd & Aspuru-Guzik [14], we show that larger models, which are often more effective compressors of data, demonstrate improved ability to learn symmetries from the training data and augmentation.

3 Background

Language Modeling Autoregressive language models are trained on large datasets of sequences, $\mathcal{U} = \{U_1, U_2, \dots U_i, \dots, U_N\}$, where $U_i = \{u_1, u_2, \dots, u_j, \dots, u_{n_i}\}$ and u_i belong to a vocabulary $\mathcal{V}$. The parameters of the language model encode an autoregressive distribution, in which the probability of each symbol is only dependent on the previous symbols in the sequence, $p\left(U_i; \theta\right) = \prod_{j=1}^{n_i} p\left(u_j \mid u_{0:j-1}; \theta\right)$. The parameters, θ, are learned by maximizing the probability of the entire dataset $p(\mathcal{U}; \theta) = \prod_{i=1}^{N} p(U_i; \theta)$. Each conditional distribution $p(u_j \mid u_{0:j-1}; \theta)$ is a categorical distribution over the vocabulary size $|\mathcal{V}|$, and therefore each loss term resembles the standard cross-entropy loss. Models are often compared using *perplexity*, which is the exponent of the length-normalized cross entropy loss, $\text{PPL}(u_i) = 2^{\text{CE}(u_i)/n}$. All common autoregressive models use the transformer architecture with causal self-attention.

To sample new sequences from the learned model, each conditional distribution is sampled sequentially. Samples, however, are rarely drawn from the unmodified conditional distributions. Instead the sampling procedure is typically modulated with *temperature* (τ) and *nucleus size* (p) hyperparameters. Temperature serves to flatten the conditional distributions to uniform (high temperature) or collapse them around their maximal probabilities (low temperature). Nucleus size limits the tokens that can be sampled based on the cumulative distribution function, clipping out values that contribute very little mass. A nucleus of $p\left(0 < p \leq 1\right)$ corresponds to keeping tokens to cumulatively contribute $p\%$ of the total probability, and discarding the rest.

Tokenization To train language models on text datasets, strings are converted into sequences of tokens. Most modern LLMs rely on the SentencePiece [26] tokenizer which implements *byte pair encoding* (BPE) [16], a method that allows tokenization at the sub-word level. BPE has two appealing properties for tokenization. First, BPE performs an intial phase of compression on input strings by assigning common substrings their own token, making overall sequence lengths shorter. Second, BPE ensures that no new character is ever out of vocabulary, handling one aspect of generalization beyond the train set. One unique downside of BPE tokenization, however, is the default tokenization of numbers. BPE typically breaks numbers into irregular substrings instead of individual digits. In response, Touvron et al. [39] constrain the LLaMA-2 tokenizers to individual digits for numbers, which has been shown to dramatically improve performance on arithmetic tasks [29]. We chose to use LLaMA models in this work in order to leverage their natural representation of numbers, which facilitates learning simple functions acting on 3D coordinates and the invariances explored in Section 5.

Prompting Because large language models are simply trained to complete text, they can be used to sample from complex conditional distributions with "prompting", in which a text string is appended to data, with information that might be useful in controlling generation. Although simple at face value, prompting can encompass a surprisingly broad range of tasks, including conditioning based on a composable set of constraints and infilling partial sequences included within the prompt. Prompts can be either be manually formatted [5], or can be automatically learned using soft-prompt tuning [28]. Several recent papers have shown that careful design of prompts improves the ability of LLMs to follow instructions [41], reason [42], or verify factual claims [12].

Crystal structures and energy prediction Periodic materials are defined by a unit cell repeated infinitely along all three dimensions (Figure 2). The unit cell comprises a lattice (parallelepiped) with side lengths (l_1, l_2, l_3) and angles $(\theta_1, \theta_2, \theta_3)$. Within the lattice, there are N atoms, each specified by an element identity, e_i, and set of 3d coordinates (x_i, y_i, z_i) which can be absolute or fractional (specified as a percentage of the unit cell side lengths). Therefore a bulk material can be fully described by the tuple

$$C = (l_1, l_2, l_3, \theta_1, \theta_2, \theta_3, e_1, x_1, y_1, z_1, ..., e_N, x_N, y_N, z_N)\,. \tag{1}$$

For a given set of environmental conditions, every crystal has a corresponding energy that describes how likely it will occur in a particular configuration. Configuration with unfavorable electrostatic interactions from unlike atomic positions, such as highly overlapping atoms, are typically high energy. The gold standard for energy prediction is density functional theory (DFT), which provides tractable approximations to governing quantum mechanical equations that describe the energy and time evolution of a system. DFT, however, is still often prohibitively expensive, often scaling $O(n^3)$ with the system size, which has motivated development of deep learning potentials to approximate DFT solutions [27].

Stability of hypothetical materials (E_{hull}) The composition of a crystal also impacts its energy, as different elements have different geometries and charge properties. Certain stoichiometries, or

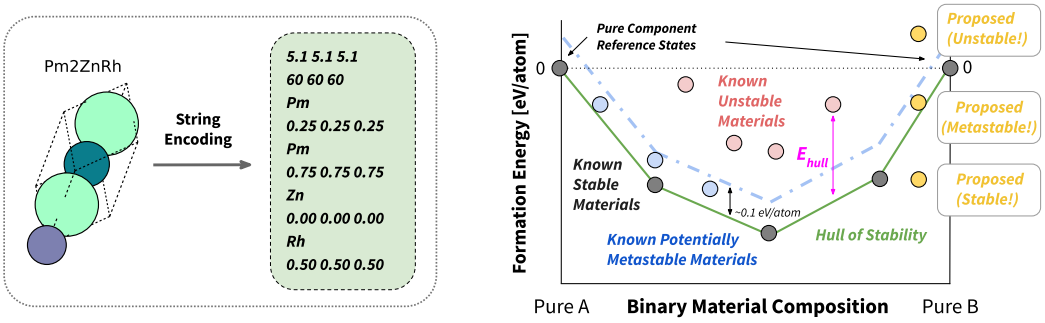

Figure 2: (**left**) We convert the crystal lattice, atom identities, and atom positions into strings. The model is trained to generate a structures conditioned on the text prompt, which might contain additional information about the composition, properties, or a starting structure to modify. (**right**) Energy above hull (E_{hull}) quantifies the stability of a material. A crystal with $E_{\text{hull}} < 0.1$ will be energetically favorable both in its structure and composition.

ratios of elements, are naturally favored, and a composition of elements A and B with constituent parts A_xB_y can dissociate into the composition A_cB_d if it is energetically favorable. Because of the effect of composition, the energy of a crystal is typically a two dimensional concept captured by the energy hull, which is the minimum observed configuration energy for a given composition. For a crystal to be low-energy and stable, and therefore give rise to a practically useful material, it must have a small *energy above hull (E_{hull})*, the distance from the energy hull for the crystals elemental composition (Figure 2). Crystals with $E_{\text{hull}} < 0$ are considered stable and by definition have lower energy than the known minimum (which has $E_{\text{hull}} = 0$). Crystals with $E_{\text{hull}} < 0.1$ eV/atom are often *metastable* and likely to be practical useful ([36]).

4 Method

Our approach to generating stable materials is pleasingly simple. We take a pre-trained LLM, which has useful biases towards generalizable patterns, and fine-tune it on crystal string representations. Because language models can also ingest text, we can condition the model's generations on text descriptions. The flexibility of language models also allows us to solve other tasks, such as infilling, through small modifications to the input formatting. Though we focus solely on crystal structures in this work, our method itself is general purpose and could be easily extended to proteins, nucleic acids, or small molecules. We include a more detailed discussion of how general text-pretraining impacts our method in Appendix A.5.

String formatting and tokenization We convert the crystal tuple C (Equation 1) using fixed precision numbers. An example of crystal string formatting is shown in Figure 2. We represent lattice lengths with one decimal place (2-3 digits) and lattice angles as integers (1-3 digits). Fractional coordinates are always represented with two digits. 3D coordinates are combined with spaces and all other crystal components are combined with newlines. We deliberately chose LLaMA-2 models because they are both state-of-the-art in overall performance among open-source models and because they tokenize numbers as individual digits by default. Notably, it is therefore impossible to create one token per full number, as Flam-Shepherd & Aspuru-Guzik [14] do in their best performing model (further discussion in Appendix A.1). Instead, we rely on the extensive pretraining of LLaMA-2 models to instill useful biases over numerical operations [29].

Prompt design To train a model that can be used for many tasks, including unconditional generation, text-conditional generation, and infilling, we adopt an instruction tuning framework [41]. The full input given to the model consists of a prompt followed by the string-formatted crystal (Figure 2). In the most basic case, the prompt indicates that the model should generate bulk materials represented as a lattice and atoms. The prompt can also be expanded to include a desired composition or material properties, or to include a starting structure, in the case of infilling. For infilling, the prompt includes the string-formatted crystal with every instance of a randomly chosen element replaced with [MASK], and the model is trained to generate the identity of the masked element at the end of the sequence. During training all three tasks are included through random sampling, with two thirds generation and one third infilling (details in Appendix A.2). As in instruction tuning, the prompt is given as input to the model but does not contribute to the generative loss function. The model is only penalized for its predictions on the crystal string or masked element.

Generation Prompt	**Infill Prompt**
<s>Below is a description of a bulk material. [The chemical formula is Pm2ZnRh]. Generate a description of the lengths and angles of the lattice vectors and then the element type and coordinates for each atom within the lattice: [Crystal string]</s>	<s>Below is a partial description of a bulk material where one element has been replaced with the string "[MASK]": [Crystal string with [MASK]s] Generate an element that could replace [MASK] in the bulk material: [Masked element]</s>

Blue text is optional and included to enable conditional generation, while purple text stands in for string encodings of atoms.

Augmentations Crystals structures are symmetric under translational. All atomic coordinates can be shifted modulo the lattice boundaries without changing the resulting material structure. Similarly, the ordering of atoms within the lattice is irrelevant to the underlying material (permutation invariance). Prior work on diffusion generative models guarantee these symmetries as invariance or equivariance constraints on the model architecture [44, 23]. To encourage translation invariance in our language models, we apply random uniform translations to the fractional coordinates. We chose not to augment the ordering of atoms because these variables often contained valuable information, for example grouping set of elements together for placement in the lattice (discussion in Appendix A.1).

5 Experiments

We explore several uses of language models in crystal generative modeling. First, in order to compare with prior work, we show that fine-tuned LLMs can be used for unconditional generation of novel materials and that the resulting materials correspond to stable relaxed structures under the predictions of an ML potential and DFT. We then show that LLMs can also be used for text-conditional generation and to propose small changes to existing materials.

Datasets and models For consistency with prior work ([44], [15]) we used MP-20 [22], a dataset of 45231 materials, when training for unconditional generation. All structures in MP-20 are stable, and therefore an effective generative model trained on MP-20 should tend to propose new crystals that are at least metastable. For text-conditioned generation, we train with all forms of prompting (Section 4) on a collection of 120,000 crystals from Materials Project (Appendix A.3). The collection includes basic property information, such as the space group number, band gap, E_{hull} and the chemical formula. All of our experiments were conducted with LLaMA-2 models (7B 13B, and 70B) [38, 39] through the Transformers library [43] and PyTorch [32]. In order to train on small number of GPUs we use 4-bit quantization [11] and Low-Rank Adapters (LoRA) [21]. We provide the full hyperparameters and training details in Appendix A.4.

Evaluation For basic evaluation of the LLM samples, we use the validity and diversity metrics introduced by Xie et al. [44]. Structural validity is determined by non-overlapping atomic radii (overlapping taken to be both atoms within half a radius of each other), while compositional validity captures the net charge of the structure (only structures with net neutral total charge are valid). Diversity is computed as pairwise distance between samples under featurizations of the structure and composition from Matminer [40, 44].

While useful for sanity checking models, simple validity metrics only reflect a subset of our real-world priorities in generating novel materials. Arguably the most important property that we hope to assess in samples is their predicted stability, which we can approximate by predicting the energy of relaxed structures. Using known materials and energy calculations from Materials Project we construct the ground truth energy convex hull and then calculate the approximate energy above hull, $\hat{E}_{\text{hull}}$. We chose two methods to estimate material stability:

- **ML potential**: M3GNet [8] provides energy, force, and stress approximations for crystal unit cells. For each sample we first run a relaxation using force and stress approximations then use the energy of the final structure.
- **DFT**: We run a relaxation using the Density Functional Theory code VASP [19] with INCAR settings chosen by Pymatgen [31]. DFT is the more accurate, but also much more computationally intense, of the two options.

In both cases, results are compatible with Materials Project values [22] (Appendix B.1). Because DFT is prohibitively expensive for many use cases (often hours per calculation), we only use it to double-check results obtained with ML potentials, and we only run VASP calculations on materials that have already been predicted as metastable by M3GNet ($<$0.1 eV/atom $\hat{E}_{\text{hull}}$). The use of a M3GNet surrogate model is not perfect as many structures in Figure 3 (right) have energies above the expected 0.1 eV/atom threshold, but the structures are largely close to the hull compared to the broader distribution of materials generated.

Unconditional generation We sample 10,000 structures from each fine-tuned LLaMA model, parsing a CIF from the generated string. We reject the sample and draw another if a CIF cannot be

Table 1: Benchmark comparison of fine-tuned LLaMA-2 models with existing generative models for inorganic materials. Following prior work [44], we report validity, which captures physical constraints, as well as coverage and property metrics, which capture alignment between the ground truth and sampling distribution. We add stability and diversity checks, which count the percentage of samples estimated to be stable by M3GNet [8] and DFT [19] (precise details in Appendix B.2). We also quantify the diversity among the predicted metastable materials (M3GNet). LLaMA excels at generating valid structures and generates a high percentage of stable materials, while maintaining diversity.

Method	Validity Check		Coverage		Property Distribution		Metastable	Stable	Diversity	
	Structural↑	Composition↑	Recall↑	Precision↑	wdist (ρ)↓	wdist (N_{el})↓	M3GNet ↑	DFT† ↑	Structural↑	Composition↑
CDVAE	**1.00**	0.867	0.991	0.995	0.688	1.43	28.8%	5.4%	0.777	15.0
LM-CH	0.848	0.835	0.9925	0.9789	0.864	0.13	n/a	n/a	n/a	n/a
LM-AC	0.958	0.889	0.996	0.9855	0.696	0.09	n/a	n/a	n/a	n/a
LLaMA-2										
7B ($\tau=1.0$)	0.918	0.879	0.969	0.960	3.85	0.96	35.1%	6.7%	0.806	16.4
7B ($\tau=0.7$)	0.964	0.933	0.911	0.949	3.61	1.06	35.0%	6.2%	0.827	13.7
13B ($\tau=1.0$)	0.933	0.900	0.946	0.988	2.20	0.05	33.4%	8.7%	0.865	14.1
13B ($\tau=0.7$)	0.955	0.924	0.889	0.979	2.13	0.10	**38.0%**	14.4%	**1.03**	15.9
70B ($\tau=1.0$)	0.965	0.863	0.968	0.983	1.72	0.55	**35.4%**	10.0%	**1.028**	15.9
70B ($\tau=0.7$)	**0.996**	**0.954**	0.858	0.989	0.81	0.44	**49.8%**	10.6%	**0.706**	13.6

† Fraction of structures that are first predicted by M3GNet to have $E_{\text{hull}}^{\text{M3GNet}} < 0.1$ eV/atom, and then verified with DFT to have $E_{\text{hull}}^{\text{DFT}} < 0.0$ eV/atom.

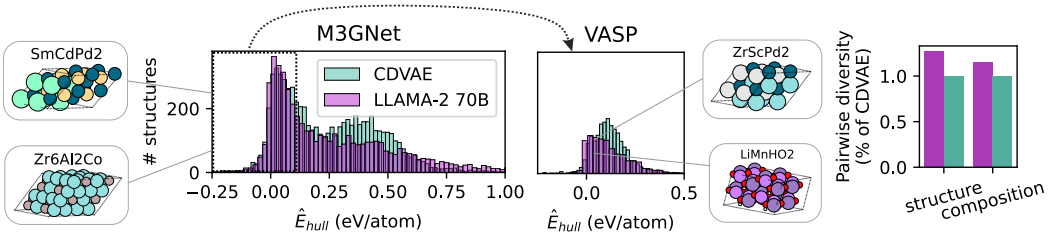

Figure 3: Stability of LLaMA samples compared to CDVAE [44]. Fine-tuned LLaMA-2 70B generates a higher rate of metastable ($\hat{E}_{\text{hull}} < 0.1$) and stable materials than CDVAE, using estimates of $\hat{E}_{\text{hull}}$ from both M3GNet [8] and VASP [19]. Because of computational cost, we only run VASP on structures predicted to be stable by M3GNet. Stable materials generated by LLaMA are also more diverse (as quantified by Matminer featurization [40]) than stable samples from CDVAE. We include sampled stable structures, shown as (2,2,2) supercells, which display a high-degree of regularity and understanding of three-dimensional space.

parsed from the sampled string, which guarantees all samples can be interpreted as crystals but does not guarantee validity of the resulting crystal. We show the validity and predicted stability ([44]) of the resulting structures in Figure 3, which shows that LLMs can achieve near-perfect rates of structural and compositional validity. Hyper-parameters like temperature and nucleus size can be used to trade-off validity and stability of samples with their coverage (Figure 4). LLaMA-2 70B strikes an effective balance, generating high rates of stable materials with good coverage and diversity. By defaut, generation is completed unconstrained and therefore the model can hallucinates imaginary elements, for example "Ln," a common abbreviation for Lanthanide (Figure 4), but the problem can be easily avoided with constrained generation [43].

Symmetry learning As crystal structures have translational symmetry, ideally our model's likelihood should be invariant to translations. We propose *Increase in Perplexity under Transformation (IPT)* as metric for assessing the invariance of language models to continuous group transformations. For a transformation group G with group elements g and group action t, we define IPT for an input s to be,

$$\text{IPT}(s) = \mathbb{E}_{g \in G}[\text{PPL}(t_g(s)) - \text{PPL}(t_{g^*}(s))]$$

where

$$g^* = \arg\min \text{PPL}(t_{g^*}(s))$$

and PPL is the perplexity of the sequence assigned by the language model (Section 3). In our case G is the group of translation, where each g is a distance to translate by, and t_g is the mapping that decode the string, translates the coordinates (wrapping them around the boundary), and re-encodes the string.

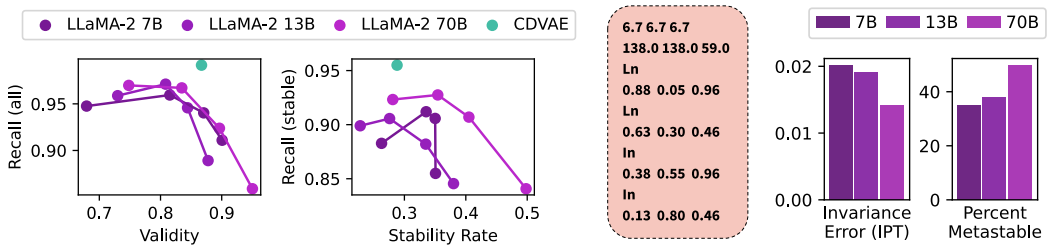

Figure 4: (**left**) Validity and rate of stability depend on sampling hyper-parameters. Lowering the temperature or restricting the nucleus size leads to significant improvements in validity/stability but incurs a cost to coverage of a held-out test set (recall). Fine-tuned LLaMA-2 70B displays the best trade-off between coverage and stability, generating materials that are both stable and diverse. (**center**) One failure mode in high temperature sampling is hallucination of elements (e.g. "Ln", an abbreviation for Lanthanide). (**right**) Translation invariance on test data and ability to generate stable materials increase in proportion. Larger models learn invariances from augmentations more effectively during training, likely as a result of their preference for abstract and compressible patterns.

IPT captures the degree to which transformations change a language model's compression ability. Good understanding of group transformations and invariance in the data should lead to minimal change in the perplexity of a transformed sequence. We can approximate IPT by sampling many values of g (e.g. 20), picking g^* as the minimum among those values, and computing a sample mean. Figure 4 shows the mean IPT of 500 random crystals from the test set, for each of the three LLaMA model sizes. We see that invariance, indicated by low error, increases alongside a model's ability to generate metastable structures, and is one way to understand their good generative performance. We include additional details about our IPT calculation in Appendix B.3.

Text-conditioned generation Extending our method to text-conditional generation is as simple as including additional information in the prompt, with a small amount of additional text (Figure 4). We explore conditioning on spacegroup number, composition, and E_{hull}, as these properties are easy to verify (at least approximately) in silico. We assess the model's ability to perform conditional generation by comparing the intended condition with labels obtained from an in-silico oracle for the constraint. For the chemical formula, we simply parse the composition from the generated CIF. For space group determination, we use pymatgen's SpacegroupAnalyzer with a precision of 0.2 angstroms [31]. For stability, we use M3GNet to estimate E_{hull} as before. Using the oracle's labels, we then compute the percentage of cases in which the condition was properly met (Figure 5). The model is able to generate a material with the correct composition the majority of the time but becomes less reliable as the number of atoms in the chemical formula increases. Space group conditioning is more challenging, as it requires precise control and understanding of 3D structure, but the observed 24% is impressive when considering the 230 possible space groups. Generating stable/unstable structures as a binary task is the most challenging, likely because the training dataset is predominantly stable compounds and stability is defined only in reference to existing compounds. Stability is most easily controlled by modulating sampling hyperparameters.

Infilling Existing Materials In many practical settings, sampling and filtering materials from scratch is unnecessary. Good starting materials are often known, and manufacturing processes are easier to adapt to related compositions than develop completely from scratch by making small edits to their composition–often referred to as *template methods*) [24, 33]. To emulate a typical template method, we construct a lookup table that maps each element to elements that have a similar atom radius when in the same oxidation state (code in Appendix C). We choose an element uniformly at random and swap it with a random element chosen from the table. The resulting structure is then relaxed using M3GNet [8]. To improve this strategy using our fine-tuned LLM, we used the infilling prompt (Section 4) to obtain a distribution over elements (modulated with temperature τ) which we use instead of a uniform distribution over swaps. To evaluate our mutation procedure, we sample 3000 structures randomly from the test set and generate perform one mutation-relaxation step for each, using both uniform and language model-guided sampling. In Figure, 5 we show the percentage of stable compounds and diversity in the stable compounds for the uniform baseline and LLaMA-2

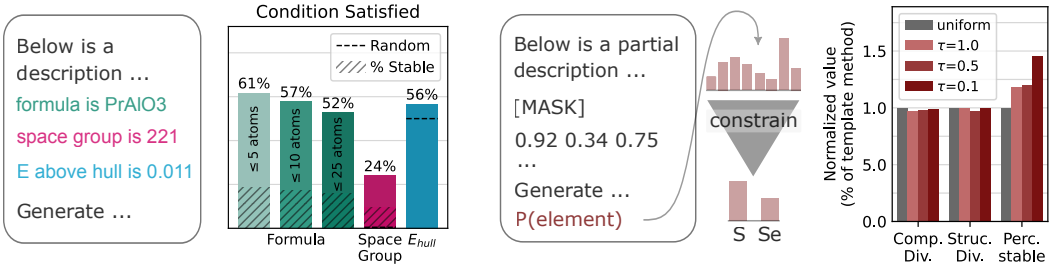

Figure 5: Text-conditional generation and infilling of existing structures with fine-tuned LLMs. (**left**) Including composition or property information (sampled from a hold-out set) in the text prompt leads to a high rate of samples with the desired composition/property (space group or stability). We bin stability as $\hat{E}_{\text{hull}} < 0.1$ (metastable) and $\hat{E}_{\text{hull}} > 0.1$ (unstable) for simplicity. Complex formulas and space groups challenge the model, but the samples are correct at a rate that facilitates practical use. We also show the rate of samples that both satisfy the condition and are predicted to be metastable by M3GNet. (**right**) Using the infilling prompt we can select mutations to existing materials. LLaMA-2 70B proposes a distribution over elements, which we constrain using knowledge of atom radii and charge interactions. We sample mutations with temperature τ and relax the results structure with M3GNet. When we apply this mutation procedure, we obtain more stable materials per mutation, with negligible changes to the overall diversity of the stable materials.

70B with different temperature values. LLaMA-2 70B proposes elements that lead to stable structures at a higher rate than the baseline template method without sacrificing diversity.

6 Discussion

By generating a high rate of plausible stable materials (verified by DFT), we have demonstrated LLMs can be state-of-the-art generative models for atomistic domains with direct application of parameter-efficient instruction tuning and minimal task-specific modeling choices. This approach to generative modeling opens the door to multitask capabilities within a single sampling paradigm and multimodal training on atoms and text (e.g. to extract knowledge from a large corpus of scientific papers). We also advocate for the use of evaluation metrics (e.g. E_{hull}) for generative models that are more closely tied to the downstream task of generating stable or metastable materials. The space of all hypothetical materials is combinatorially large (consider all the ways to pack 20 arbitrary elements into a box), but only a small subset of materials will actually be stable or metastable. Models that can directly generate near-stable structures make all downstream tasks far easier, and increases the likelihood the generative models may be useful for day-to-day tasks in materials discovery.

Limitations Our method shares the limitations of the underlying generative models. LLMs are sensitive to precise details of the chosen prompt and the tokenization strategies, particularly in how tokenization effects processing of numbers. Hallucination of unphysical chemical elements or structures has been observed, though fortunately is easy to check and filter. Text-conditioning has the potential to tap latent conceptual understanding in the underlying LLM, but training LLMs that successfully leverage scientific and chemistry literature is a major outstanding challenge. Lastly, training the largest of our LLMs can be prohibitively expensive for some computational budgets. Despite this, inference from all LLMs is often highly tractable when compared to baseline methods, as we show in Appendix B.4.

Future directions There is substantial room for improvement in conditional generation, which could ideally be used to generate materials with desired properties directly. While we did not pursue alternative sampling strategies in depth, approaches like classifier-free guidance [34] or variants of PPLM [9] might be useful in combination with fine-tuned LLMs for improved conditional generation. These methods could also be combined with primitives from Bayesian optimization for sample-efficient and uncertainty-aware design [35, 18].

References

[1] NVIDIA A100 GPU Benchmarks for Deep Learning. https://lambdalabs.com/blog/nvidia-a100-gpu-deep-learning-benchmarks-and-architectural-overview.

[2] LLaMA 2 on Amazon Sagemaker, a Benchmark. https://huggingface.co/blog/llama-sagemaker-benchmark.

[3] Luis M Antunes, Keith T Butler, and Ricardo Grau-Crespo. Crystal structure generation with autoregressive large language modeling. *arXiv preprint arXiv:2307.04340*, 2023.

[4] Carrie Arnold. Inside the nascent industry of ai-designed drugs. *Nature medicine*, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

[7] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 2021. doi: 10.1021/acscatal.0c04525.

[8] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table. *Nature Computational Science*, 2(11):718–728, 2022.

[9] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[10] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[11] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022.

[12] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

[13] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

[14] Daniel Flam-Shepherd and Alán Aspuru-Guzik. Language models can generate molecules, materials, and protein binding sites directly in three dimensions as xyz, cif, and pdb files. *arXiv preprint arXiv:2305.05708*, 2023.

[15] Daniel Flam-Shepherd, Kevin Zhu, and Alán Aspuru-Guzik. Atom-by-atom protein generation and beyond with language models. *arXiv preprint arXiv:2308.09482*, 2023.

[16] Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38, 1994.

[17] Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023.

[18] Nate Gruver, Samuel Stanton, Nathan C Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. *arXiv preprint arXiv:2305.20009*, 2023.

[19] Jürgen Hafner. Ab-initio simulations of materials using vasp: Density-functional theory and beyond. *Journal of computational chemistry*, 29(13):2044–2078, 2008.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[21] J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

[22] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

[23] Rui Jiao, Wenbing Huang, Peijia Lin, Jiaqi Han, Pin Chen, Yutong Lu, and Yang Liu. Crystal structure prediction by joint equivariant diffusion on lattices and fractional coordinates. In *Workshop on"Machine Learning for Materials"ICLR 2023*, 2023.

[24] Scott Kirklin, James E Saal, Bryce Meredig, Alex Thompson, Jeff W Doak, Muratahan Aykol, Stephan Rühl, and Chris Wolverton. The open quantum materials database (oqmd): assessing the accuracy of dft formation energies. *npj Computational Materials*, 1(1):1–15, 2015.

[25] Ben Krause, Subu Subramanian, Tom Yuan, Marisa Yang, Aaron Sato, and Nikhil Naik. Improving antibody affinity using laboratory data with language model guided design. *bioRxiv*, 2023.

[26] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://api.semanticscholar.org/CorpusID:52051958.

[27] Janice Lan, Aini Palizhati, Muhammed Shuaibi, Brandon M Wood, Brook Wander, Abhishek Das, Matt Uyttendaele, C Lawrence Zitnick, and Zachary W Ulissi. Adsorbml: Accelerating adsorption energy calculations with machine learning. *arXiv preprint arXiv:2211.16486*, 2022.

[28] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, abs/2101.00190, 2021.

[29] Tiedong Liu and Bryan Kian Hsiang Low. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201*, 2023.

[30] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[31] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

[33] James E Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and Christopher Wolverton. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (oqmd). *Jom*, 65:1501–1509, 2013.

[34] Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023.

[35] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *International Conference on Machine Learning*, pp. 20459–20478. PMLR, 2022.

[36] Wenhao Sun, Stephen T Dacek, Shyue Ping Ong, Geoffroy Hautier, Anubhav Jain, William D Richards, Anthony C Gamst, Kristin A Persson, and Gerbrand Ceder. The thermodynamic scale of inorganic crystalline metastability. *Science advances*, 2(11):e1600225, 2016.

[37] Ilya Sutskever. An observation on generalization. Workshop on Large Language Models and Transformers, 2023. URL https://www.youtube.com/watch?v=AKMuA_TVz3A&ab_channel=SimonsInstitute.

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[39] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[40] Logan Ward, Alexander Dunn, Alireza Faghaninia, Nils ER Zimmermann, Saurabh Bajaj, Qi Wang, Joseph Montoya, Jiming Chen, Kyle Bystrom, Maxwell Dylla, et al. Matminer: An open source toolkit for materials data mining. *Computational Materials Science*, 152:60–69, 2018.

[41] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *ArXiv*, abs/2201.11903, 2022.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[44] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Crystal diffusion variational autoencoder for periodic material generation. *arXiv preprint arXiv:2110.06197*, 2021.

Appendix

Table of Contents

A Training Details

A.1 Numerical Formatting

Notably, our approach to tokenization is distinctly different from prior work on modeling atomic structures with language models. Instead of using a special vocabulary and training models from scratch, we use LLaMA-2's existing tokenizer. This choice allows us to easily process both encoded crystals and text data. In early experiments, we tried out many other approaches, including fine-tuning LLaMA-2 models with additional tokens specific to crystal data. These methods were more challenging to train and didn't lead to any improvements over using a shared tokenizer. We include a set of example training losses below:

	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Special Crystal Tokens	0.783	0.693	0.623	0.611	0.588
Shared Tokenization	0.457	0.432	0.424	0.401	0.385

There are many important decisions involved both in text formatting (e.g the choice of fractional or absolute coordinates) and augmentation of the input data (e.g. translation or permutation augmentations on coordinates). As a simple example, we provide average validity numbers (using low temperature sampling) from earlier experiments on LLaMA-2 7B models trained with different formatting styles

Setting	Structural Validity	Compositional Validity
Fractional coords	91.4%	83.2%
Absolute coords	90.8%	80.5%
No permutations	92.5%	82.9%
With permutations	89.2%	81.7%

A.2 Training with Stochastic Prompts

In order to enable multi-task use of the fine-tuned LLMs, we train on a stochastically generate prompt. Two thirds of the time we provide the model with a generation task, in which the prompt consists of a basic instruction to generate a bulk material as a lattice and atom positions. We randomly sample a set of properties from the available descriptors of a given crystal and add any chosen ones (if any) to the prompt, using a small amount of wrapper text. The remaining one third of the time, we provide

use the sampled crystal to construct and infilling task. We choose on element randomly from the set of elements in the composition and we construct a prompt that contain the string encoding of the crystal with this element replaced with [MASK]. The model then generates the replaced element as text following the prompt.

A.3 Extended Materials Project Dataset

To facilitate text-conditional generation, we extend the original CDVAE training dataset with materials from Materials Project [22] as of April 2023. We filter out crystal with more than 30 atoms in the unit cell, which slow down training with minimal benefit to model performance, leaving a training set that contains 127609 crystal structures. The original validation and test splits are left unchanged and all test/validation points are removed from the new training set.

A.4 Training Hyperparameters and Details

We provide the training details per model:

- **LLaMA-2 7B**: Batch size of 256 for 65 epochs with a cosine annealed learning rate of 0.0005. LoRA rank 8 and alpha 32.
- **LLaMA-2 13B**: Batch size of 256 for 44 epochs with a cosine annealed learning rate of 0.0005. LoRA rank 8 and alpha 32.
- **LLaMA-2 70B**: Batch size of 32 for 21 epochs with a cosine annealed learning rate of 0.0005. LoRA rank 8 and alpha 32.

Limitations around available compute lead to our use of differing batch sizes and total number of epochs for each model. Ideally, we would train all models with the largest batch sized used among all models and would train all models for the same number of epochs (the maximum used by any model). At the same time, we wanted to properly demonstrate the full potential of all model sizes and therefore chose to present results for the best model we were able to train at each model size.

A.5 Role of Text Pretraining

Text pretraining is essential to our method for two reasons.

1. It would be impractically expensive or computationally infeasible to train models with up to 70B parameters from scratch on our data. Using a pretrained model with LoRA [21] offers the benefits of model scale while maintaining tractability and limiting overfitting, as the actual number of trainable parameters can be relatively small.
2. Pretraining on text data yields a model that can be conditioned on text for free, and text conditioning opens up a huge new realm of exciting possibilities, like conditioning samples on desired properties. It would be challenging to achieve a similar result from scratch without significantly expanding the size of the dataset (to improve general text understanding) and without essentially training a general-purpose language model in the process.

To better understand the first point, let's quickly review the exact details of the finetuning procedure. We are using low-rank adapters (LoRA), as opposed to end-to-end finetuning, and this means we are adding a small number of additional parameters to an existing, frozen model. The easiest way to see the difference between this approach and training a model from scratch (as in [14]) is to compare the training loss over the first few epochs of training.

Model	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
GPT-2 (from scratch)	0.946	0.878	0.807	0.757	0.740
LLaMA-13B (LoRA)	0.457	0.432	0.424	0.401	0.385
LLaMA-70B (LoRA)	0.402	0.344	0.325	0.305	0.296

If we attempt to run LoRA finetuning with randomly initialized parameters for the LLaMA-2 7B model we observe an immediate and significant difference in the training losses:

Model	1 Iter	0.33 Epochs	0.66 Epochs	1 Epoch
Random	13.46	1.53	0.81	0.78
Pre-trained	1.57	0.47	0.41	0.39

While LoRA finetuning is tractable because 99.95% of the model is frozen, finetuning a LLaMA-2 model end-to-end in half-precision would require at least 4 times as many GPUs, making it infeasible for all but a handful of researchers. When using LoRA, even though the base models are large the number of trainable parameters is very small. In fact, the LLamA-2 7B model has less trainable parameters than one of the baseline methods we compared (CDVAE) [44]. The number of trainable parameters for each of our models and the baseline models is shown below:

Model	Trainable parameters (millions)	Percentage of total
CDVAE [44]	4.5	100%
LM-CH/AC [14]	1-100	100%
LLaMA-2 7B	3.5	0.05%
LLaMA-2 13B	6.5	0.05%
LLaMA-2 70B	35	0.05%

B Model Evaluation

B.1 Evaluation with ML potentials and DFT

Approximating E_{hull} from the energies of known materials in Materials Project requires a consistent correction scheme. We touch on some of the details here.

M3GNet Importantly, M3GNet was trained on the total energy of VASP calculations in the Materials Project dataset, so the results were expected to be consistent with the correction schemes and absolute energies in Section 5.

VASP To be consistent with the Materials Project settings (e.g. the PBE functional, DFT/DFT+U as appropriate, consistent pseudopotentials, etc). We did a single relaxation for every candidate structure using the default parameters in MPRelaxSet [31]. VASP relaxations were run using the GPU-accelerated VASP6 code.

In both situations, the total energies were corrected using the MP2020 compatibility scheme, which was important to maintain consistency when calculating formation energies, and allow the use of varying functionals (DFT/DFT+U) for different materials.

B.2 Stability Checks and Percentages

To calculate the percentage of metastable compounds, we take all samples and remove samples that are invalid under the basic structure and composition checks. We then run relaxations with M3GNet and obtain the final relaxation energies. The final percentage takes into account both the rate of validity (used to perform the initial filtering), and the rate of compounds with $\hat{E}_{\text{hull}} < 0.1$, as determined by the convex hull calculation using the M3GNet relaxation energy. To calculate the VASP percentage, we select materials determined to be metastable M3GNet and run VASP with default setting. We then report the percentage of the materials with $\hat{E}_{\text{hull}} < 0.0$.

B.3 Increase in Perplexity under Transformation (IPT)

We calculate IPT for each model using 500 test datapoints and 20 randomly translation sampled as fraction coordinates from a uniform distribution per dimension. The translations themselves are implemented in PyMatgen and respect periodic boundary conditions [31]. In order to combine the IPT values in a meaningful way across different datapoints, we normalize their values by the mean perplexity over transformations. Thus datapoints which happen to have large perplexity, and therefore naturally large potential changes in perplexity, do not drown out points with small perplexity.

B.4 Sampling Speed

Although LLMs might seem like computational overkill at face value, if we perform a detailed analysis, we can see that deploying a LLaMA model for large-scale sampling of materials actually has comparable computational overhead to competing approaches. We can use LLaMA-2 13B and 70B as the models for our method and AWS as the deployment environment. In a recent benchmark on a cloud instance with 8 A100 GPUs (ml.p4d.12xlarge) [2], LLaMA-2 13B achieved 0.416 hr/1M tokens and LLaMA-2 70B achieved 0.864 hr/1M tokens. One crystal is around 100 tokens on average, so the throughput for 10,000 crystals is the same as for 1M tokens. For comparison, we use CDVAE and its recorded runtimes for generating 10,000 crystals on a single RTX2080 Ti GPU [44]. To obtain the final numbers, we adjust for the number of GPUs (8) and a 2x improvement from RTX2080 Ti to A100 GPUs [1].

Model	Hours / 10,000 crystals	Hours / 10,000 metastable (M3GNet) crystals
CDVAE	0.363	1.260
LLaMA-13B	0.416	1.094
LLaMA-70B	0.864	1.728

We see that LLaMA-2 13B actually has a comparable computational overhead to prior work, and LLaMA-2 70B is only slightly higher. When considering the rate of stable materials generated by each method, we see that LLaMA-2 13B actually has a higher throughput than CDVAE and with noticeably higher diversity. We think LLMs are particularly exciting because their ability to understand physical laws improved consistently with the performance of the base model, capping out at overall validity (structure and composition) rates near 90%.

C Template Method Baseline

We provide code in Listing 1 implementing construction of the physically-inspired element swap table. This table is used by both the template method and the LLM-guided sampling method to constrain search to elements that are physically plausible. Listing 2 shows our implementation of a basic template method with uniform sampling. The LLM-guided procedure is mostly identical, except with uniform sampling of the swap element changed for sampling from a distribution obtained from the LLM with an infilling prompt (and modulated with temperature parameter τ)

```
import os
import random
import pandas as pd
import numpy as np
from pymatgen.core import Element
from pymatgen.core.structure import Structure
from m3gnet.models import Relaxer

def find_similar_elements(target_element, elements, tolerance=0.1):
    similar_elements = []
    for state, radius in target_element.ionic_radii.items():
        for el in elements:
            if state in el.ionic_radii:
                radius_diff = abs(radius - el.ionic_radii[state])
                if radius_diff < tolerance and el.symbol !=
    target_element.symbol:
                    similar_elements.append((el.symbol, state,
    radius_diff))
    return sorted(similar_elements, key=lambda x: x[2])

def make_swap_table():
    elements = [Element(el) for el in Element]

    swap_table = {}

    for el in elements:
        swap_table[el.symbol] = [
            x[0] for x in find_similar_elements(el, elements)
```

```
        ]

    return swap_table
```

Listing 1: Self contained code to construct the template method table which can be used to proposed mutations for local optimization around an existing material. The same table can be used in tandem with a language model to provide sampling constraints (i.e. eliminate elements which are very physically unlikely).

```
def propose_new_structures(cif_str, swap_table, max_swaps=1):
    struct = Structure.from_str(cif_str, fmt="cif")

    elements = [el.symbol for el in struct.species]
    swappable_elements = [
        el for el in elements if el in swap_table and len(swap_table[el]) > 0
    ]

    num_possible_swaps = sum([len(swap_table[el]) for el in swappable_elements])
    num_swaps = min(num_possible_swaps, max_swaps)

    relaxer = Relaxer()
    new_bulks = []
    for _ in range(num_swaps):
        old_el = random.choice(swappable_elements)
        possible_new = swap_table[old_el]
        new_el = random.choice(possible_new)

        new_bulk = struct.copy()
        new_bulk.replace_species({old_el: new_el})

        relax_results = relaxer.relax(new_bulk)
        final_structure = relax_results['final_structure']
        final_relaxed_energy = relax_results['trajectory'].energies[-1]

        new_bulks.append(dict(
            cif=final_structure.to(fmt="cif"),
            energy=final_relaxed_energy
        ))

    new_bulks = pd.DataFrame(new_bulks)
    return new_bulks
```

Listing 2: Self contained code implementing a template method with uniform sampling. Our language model procedure is essentially the same but replaces uniform sampling with logits from a prompted language model. This language model can use the context from the rest of the crystal structure to propose a mutation instead of choosing a mutation completely at random.