Mofasa: A Step Change in Metal-Organic Framework Generation

Vaidotas Šimkus, Anders Christensen, Steven Bennett, Ian Johnson,
Mark Neumann, James Gin, Jonathan Godwin, Benjamin Rhodes

{vaidas,ben}@orbitalindustries.com
Orbital

Mofasa is an all-atom latent diffusion model with state-of-the-art performance for generating Metal-Organic Frameworks (MOFs). These are highly porous crystalline materials used to harvest water from desert air, capture carbon dioxide, store toxic gases and catalyse chemical reactions. In recognition of their value, the development of MOFs recently received a Nobel Prize in Chemistry.

In many ways, MOFs are well-suited for exploiting generative models in chemistry: they are rationally-designable materials with a large combinatorial design space and strong structure-property couplings. And yet, to date, a high performance generative model has been lacking. To fill this gap, we introduce Mofasa, a general-purpose latent diffusion model that jointly samples positions, atom-types and lattice vectors for systems as large as 500 atoms. Mofasa avoids handcrafted assembly algorithms common in the literature, unlocking the simultaneous discovery of metal nodes, linkers and topologies.

To help the scientific community build on our work, we release `MofasaDB`, an annotated library of hundreds of thousands of sampled MOF structures, along with a user-friendly web interface for search and discovery: https://mofux.ai/.

Discovering novel periodic crystal structures at scale is a grand challenge in materials science. In few places is this opportunity greater than with Metal-Organic Frameworks (MOFs)—highly porous materials critical for next-generation climate technologies [55].

Traditionally, MOFs are viewed as modular compositions of metal nodes and organic linkers arranged in a specific network topology. While new materials can be found by traversing the combinatorial space of known building blocks, discovering truly novel chemistry requires more than just recombining existing components—we must adopt an *all-atom* design approach.

All-atom generative models present a promising path forward to discover truly novel chemistries due to their unconstrained nature [22, 62, 47]. Yet, they have failed, so far, to scale to large crystal systems containing many atoms in the unit cell. This is odd, because generative models of text, images, and—most pertinently—biomolecules scale well as a function of dimensionality [26, 29].

This motivates us to find a scalable recipe for all-atom generation of large crystal structures. Our method takes ADiT [47] as an initial template, and re-engineers the architecture, parameterization, training loss, and sampling algorithm. In this sense, our approach is proudly incremental: we eschew methodological novelty in favor of thorough engineering. However, once implemented correctly, the *results* are far from incremental, and it is these results that constitute the bulk of this report.

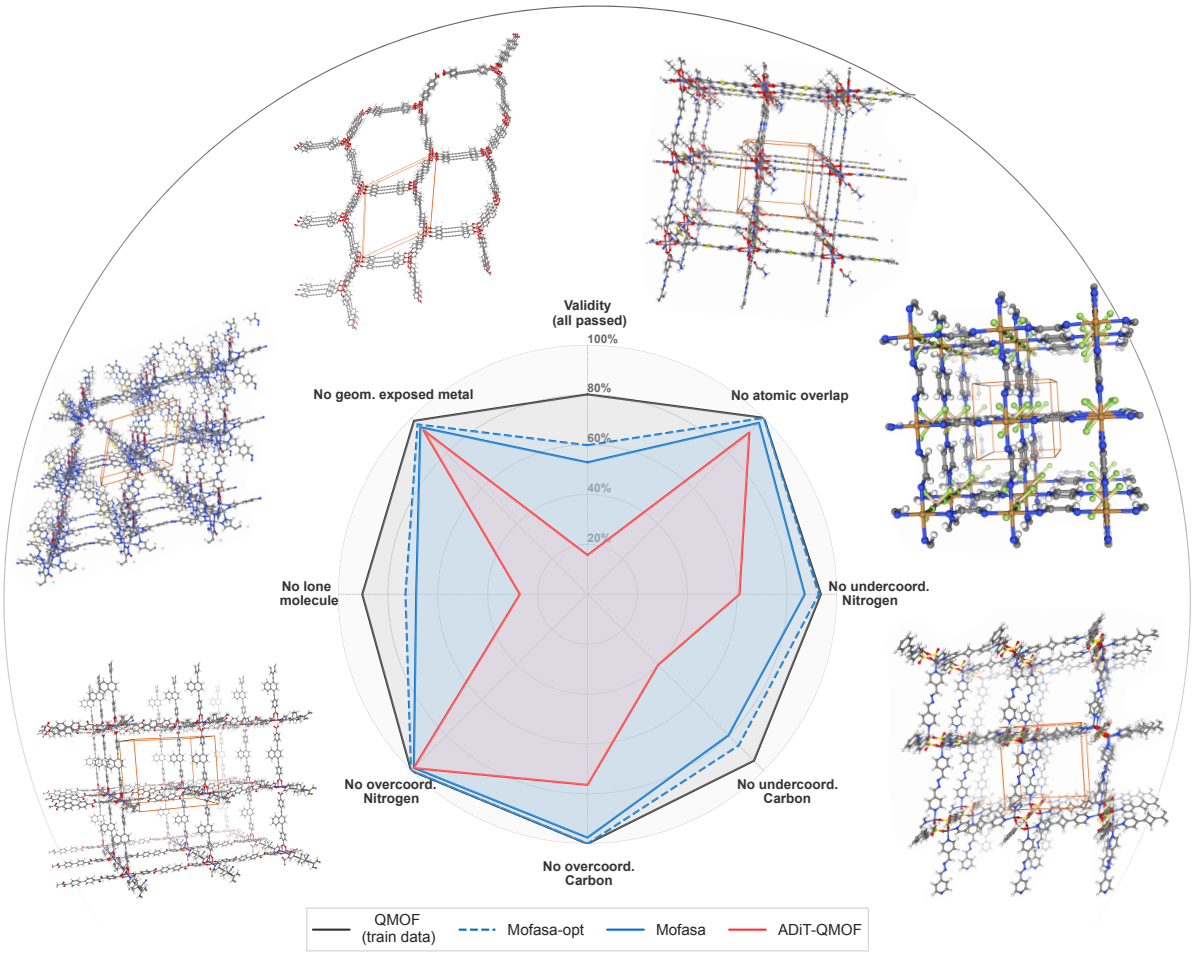

Figure 1: **Validation of geometric structure with MOFChecker**. Mofasa demonstrates a step-change improvement over the leading all-atom baseline, ADiT [47], increasing overall MOFChecker validity by 3.8× from 15.7% to 59.9%.

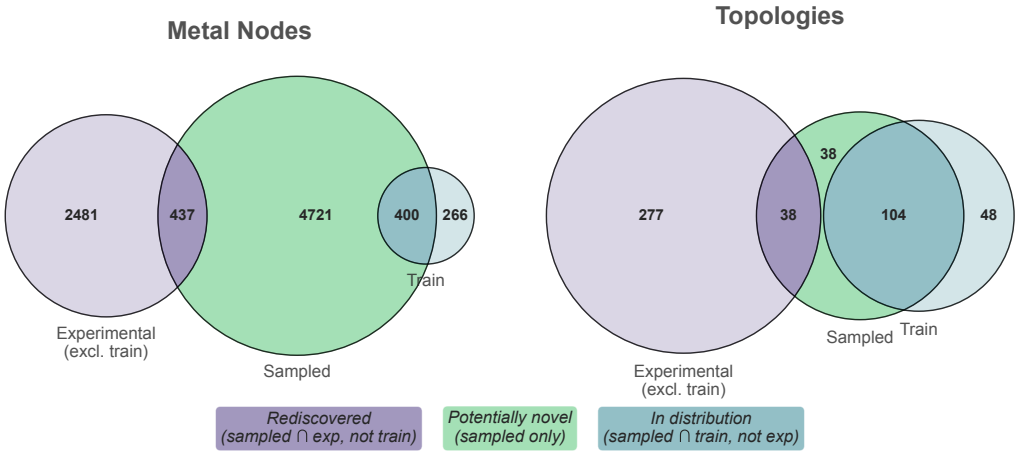

Figure 2: **Rediscovery and novelty analysis.** Mofasa demonstrates strong generalization by *rediscovering* 437 nodes + 38 topologies absent from the training set. Beyond generating known chemistry, the model also generates MOFs with chemistries unseen in experimental databases.

Our main contributions are:

- High-fidelity generation (53–62% `MOFChecker` validity[1]) of entire 3D crystal structures (positions, atom-types, lattice) for MOFs up to 500 atoms. On QMOF, we push SoTA forward by > 3× (see Figure 1).
- From a total of 100k samples, 40–46% are *valid, novel and unique* as measured by `MOFid` + `MOFChecker`. Importantly, the training set (QMOF) imposes an upper limit of 70%.
- A *rediscovery rate* of 8.5% and 50.0% for nodes and topologies, respectively. Mofasa rediscovers a significant number of metal nodes and topologies that are not present in the training data, but are present in other experimental databases.
- An open source database, `MofasaDB`, containing ~200k generated structures annotated with a large set of descriptors and properties to enable screening by composition, types of nodes and linkers, topology, space group, porosity and more (see Appendix A).
- A scalable data-agnostic architecture. Without modification, Mofasa can be jointly trained on MOFs, other porous materials classes, organic molecular crystals, and gas-phase organics + transition metal complexes.

1 Results

1.1 Comparison to state-of-the-art

A unified benchmark for MOF generation is lacking. As a consequence, many recent works [44, 56, 61] use disparate metrics and datasets, rendering direct comparison infeasible. Nonetheless, we believe ADiT [47], MOFFlow-2 [49] and MOF-BFN [45] are a fair representation of the state-of-the-art. All these works report `MOFChecker` validity [46], a rigorous metric that assesses structural and compositional validity of MOFs by penalizing flaws such as floating atoms and incorrect coordination (see Tables 4 and 5 for a full breakdown of the criteria).

`MOFChecker` validity scores for Mofasa (trained on different datasets) are shown in Table 1, along with reference values for the training data and baseline methods. Mofasa-opt refers to model samples that have been geometry-optimized with an MLIP (see Appendix D).

Model	**Sample count (n)**	**Validity (%)**	**Training dataset**		
			Name	**Count (n)**	**Validity (%)**
†ADiT	1k	15.7	QMOF [17]	14k	80.0
Mofasa	202k	52.9	QMOF [17]	14k	80.0
Mofasa-opt	202k	**59.9**	QMOF [17]	14k	80.0
†MOFFlow-2	10k	38.8	BW [8, 11]	157k	100.0
†MOF-BFN	1k	32.3	BW [8, 11]	324k	–
Mofasa	10k	**44.8**	BW [23]	250k	85.8
Mofasa	250k	52.2	Exp. [17, 23, 64]	49k	84.2
Mofasa-opt	225k	**62.8**	Exp. [17, 23, 64]	49k	84.2

Table 1: **Comparison of MOFChecker validity (%) across datasets.** † indicates baseline models, and **bold** (Mofasa/Ours). Right hand side shows the training dataset and its statistics for each model. In this table and elsewhere, Experimental (Exp.) = QMOF [17] + CoRE-MOF-2024 (computation-ready splits) [64] + ARC-MOF DB12 & DB14 [23]. See more details in Appendix B.

[1]Exact percentage varies with training set and whether samples are relaxed with an MLIP.

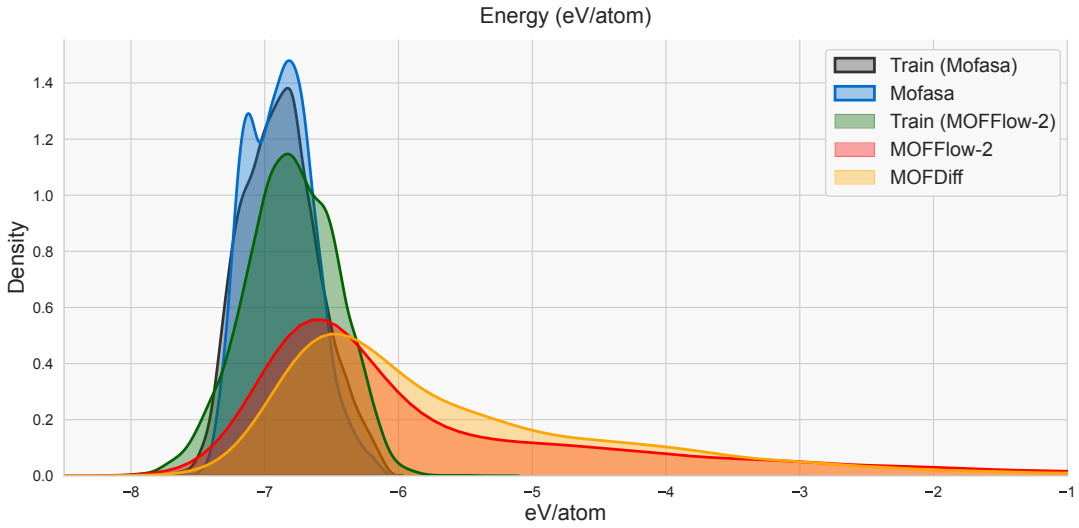

Figure 3: **Potential energy histograms on the Boyd-Woo (BW) [8, 11] dataset.** Note that, Mofasa (black) and MOFFlow-2 (red) are trained on slightly different subsets of BW (see Appendix B) and the Mofasa energies (black, blue) were computed with `Orb-v3-con-inf-omat`[58] vs UMA [60] for the MOFFlow-2 and MOFDiff. Nonetheless, the trend is clear: Mofasa is much better at matching the energy distribution of its training data.

One significant limitation of `MOFChecker` is that it is discrete and brittle. Small, structural deviations can cause physically reasonable structures to be flagged as invalid. One way to gauge the magnitude of structural errors is to study the energies of the samples using MLIPs, which are fast DFT surrogate models. In Figure 3, we plot the energy per atom of Mofasa samples (and its training set) against the samples and training set of MOFFlow-2 and MOFDiff [32]. It is evident that Mofasa is substantially better at following the true distribution of energies. Interestingly, this performance gap is not clear from the `MOFChecker` scores in Table 1, where Mofasa's 44.8% validity is only modestly higher than MOFFlow-2's 38.8%. This discrepancy underscores the need for distributional coverage evaluations alongside geometric checks.

1.2 Validity, novelty, and uniqueness

Validity scores and distributional similarities can be trivially maximized by a model that strictly memorizes the training data. To assess generalization, standard practice is to compute a Valid, Novel, and Unique (VNU) score. Here, a sample is considered *novel* if it does not match any data point in the training set, and *unique* if it does not match any other sample in a fixed set of generated samples. What, however, should count as a 'match'? This is a matter of an ongoing debate with many pitfalls [54, 48]. We do not claim to resolve all such pitfalls here and our definition of "novelty" should be interpreted with care.

We leverage prior efforts to uniquely identify MOFs using `MOFid` [12]. `MOFid` decomposes a 3D MOF into building blocks: metal nodes, organic linkers, network topology, and catenation. The final identifier is a string representation of these four components that is both convenient and chemically-informed. One drawback is incomplete coverage: `MOFid` only succeeds for ~85% of systems in QMOF, often failing on chemically valid structures such as 2D or rod-like MOFs [such as MOF-74, see 12] and is sensitive to geometric perturbations (see Tables 7 to 9).

Figure 4 shows Mofasa's "success rates" for a range of sample sizes and different combinations of VNU conditions. Importantly, at 100,000 samples, we see that:

- Mofasa obtains ~68% (maximum ~84%) NU score (`MOFid exists + novel + unique`).
- Mofasa(-opt) obtains 40% (46%) (maximum ~70%) VNU score (NU + `MOFChecker-valid`).

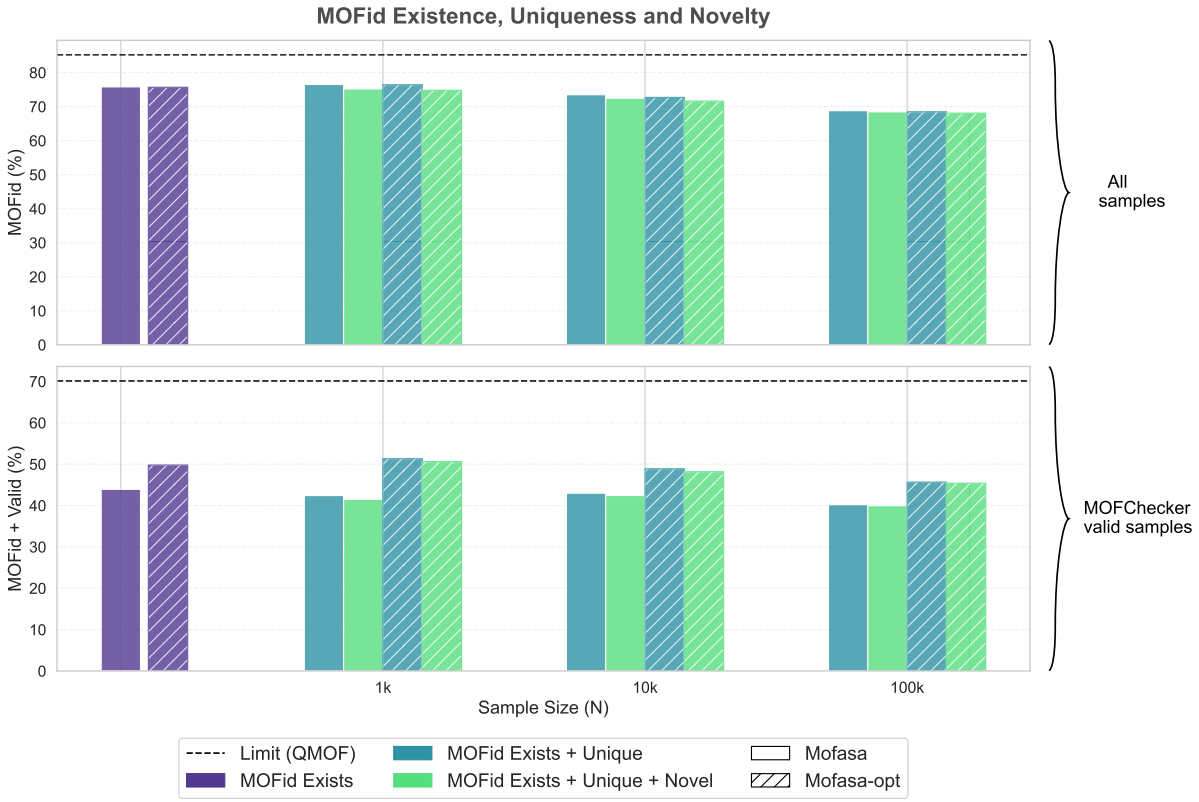

Figure 4: **Validity, Novelty, and Uniqueness (VNU) analysis.** Percentage of QMOF samples for which MOFids exists (purple) and is unique (teal) and is novel (light green). **Top row:** all samples without validity constraints; the maximum score is 84%. **Bottom row:** only MOFChecker valid systems; the maximum score is 70%. See Appendix Table 6 for full numerical results.

1.3 Rediscovery of unseen experimental nodes, linkers, and topologies

MOFs can be computationally designed without machine learning [36] using rule-based systems for assembling libraries of building blocks according to predefined topologies [16]. While these rule-based systems can be combined with generative models of building blocks [42, 61], they remain fundamentally constrained. The presupposition of a known topology introduces a degree of subjectivity [33], hinders the exploration of polymorphs [28], and creates a barrier to incorporating new sources of non-porous systems into the training data.

In contrast, Mofasa generates atoms directly in the full 3D crystal. Topologies and building blocks are not presupposed, they *emerge* from the generative process. This allows us to simultaneously discover new MOF "components" (nodes, linkers and topologies) as demonstrated in Table 2. The key figure in this table is the *rediscovery rate*, which is the fraction of unique and novel sampled components that exist in an experimental database, but *not* in the training data. See Figure 2 for a visual summary. Metal nodes have a rediscovery rate of 8.5% and topologies have a rediscovery rate of 50.0%, which proves Mofasa performs useful generalization. The rediscovery rate for linkers is very low (0.5%) which is expected given the enormous chemical diversity of possible linkers compared to the more constrained set of stable metal nodes and topologies.

1.4 Distributional realism of simple properties

Figure 5 shows a range of simple property histograms for `MofasaDB` (~200k samples from our QMOF model). The top row displays the distributions of lattice vector lengths (a, b, c), which

Table 2: **Analysis of unique MOF Components.** For each data source (experimental, training set, samples) we count the numbers of unique components. We then remove all components in the training set from the sampled components, to obtain unique + novel components. Finally, a unique and novel component is counted as *Rediscovered* if it exists in an experimental source.

Component	# Unique (Experiment)	# Unique (Train)	# Unique (Samples)	# Unique + Novel (Samples)	# Rediscovered (Rate %)
MOFid	34,873	11,014	134,800	134,351	182 (0.1%)
Nodes	3,584	666	5,558	5,158	437 (8.5%)
Linkers	15,188	6,302	111,047	109,831	538 (0.5%)
Linker combos	10,682	3,911	63,128	62,996	92 (0.1%)
Topologies	467	152	180	76	38 (50.0%)

are of ascending order in the Niggli-reduced representation. Mofasa tracks the QMOF training distribution well across all three dimensions, with a slight bias towards short a & b lengths. The second row visualizes the lattice angles (α, β, γ), where Mofasa is again reasonable, but overemphasizes ~90° angles. The third row examines symmetry and topology. We calculate spacegroups using `pymatgen` [6] at two different symmetry thresholds (symprec). The generated samples are 95% and 84% triclinic for 0.01 and 0.1 thresholds, which is substantially larger than the 54% and 48% in the data distribution. Surprisingly, geometry relaxation does not improve spacegroup symmetry matching, suggesting that the lack of diverse symmetries is a relatively fundamental problem, not simply a numerical precision issue.

The fourth row characterizes chemical complexity through the count of distinct nodes and linkers and the degree of catenation. In the distinct node/linker plots, the zero bin represents cases where the `MOFid` algorithm failed to identify a recognizable building block. The model overproduces failure cases and underproduces single node/linker systems. It also overproduces multivariate MOFs with 4+ linkers, but otherwise matches the real data reasonably well. The final two rows present porosity metrics calculated using Zeo++ [5] with a 1.86Å Nitrogen probe. These four distributions show that the model generates MOFs with fairly realistic internal voids, but skews towards overly small LCD/PLD values, which in turn means a higher percentage of systems do not allow the Nitrogen probe to enter (see percentages in legends of final row plots).

1.5 Dynamic behavior of sampled frameworks

To investigate the dynamic behavior of generated MOFs, we used a short three-step MD pipeline (see Appendix E) on 1,000 randomly selected samples from `MofasaDB`. These simulations indicate the structural flexibility of sampled structures above 0 K, helping us to understand dynamic behavior due to thermal effects that local geometry minimizations fail to identify. Our simulations used `orb-v3-con-inf-omat` [37, 58] + D3 [4], a combination that was shown to be highly accurate for MOFs in the zero-shot setting by Kraß et al. [50].

We report root mean square deviation (RMSD) of the initial and final frames of the NPT trajectory, as well as the volume drift between the first 10 ps and final 10 ps. Of the 1,000 samples, 3.6% did not converge after 1,000 geometry optimization steps and 4.7% exhibited rapid volume expansion, resulting in a failure of the MD run. The RMSD and volume drift histograms in Figure 6 (a-b) show that the majority of the samples do not exhibit a significant volume or atomic displacement, with 91% drifting by less than 10% in volume and 80.7% having < 2Å RMSD.

In Figure 6 (c), we highlight four representative copper-node systems, which demonstrate the wide range of coordination sites accessible to the generative model. The under-coordinated Structures 1 (T-shaped) and 2 (square planar) resemble less stable open metal sites, which

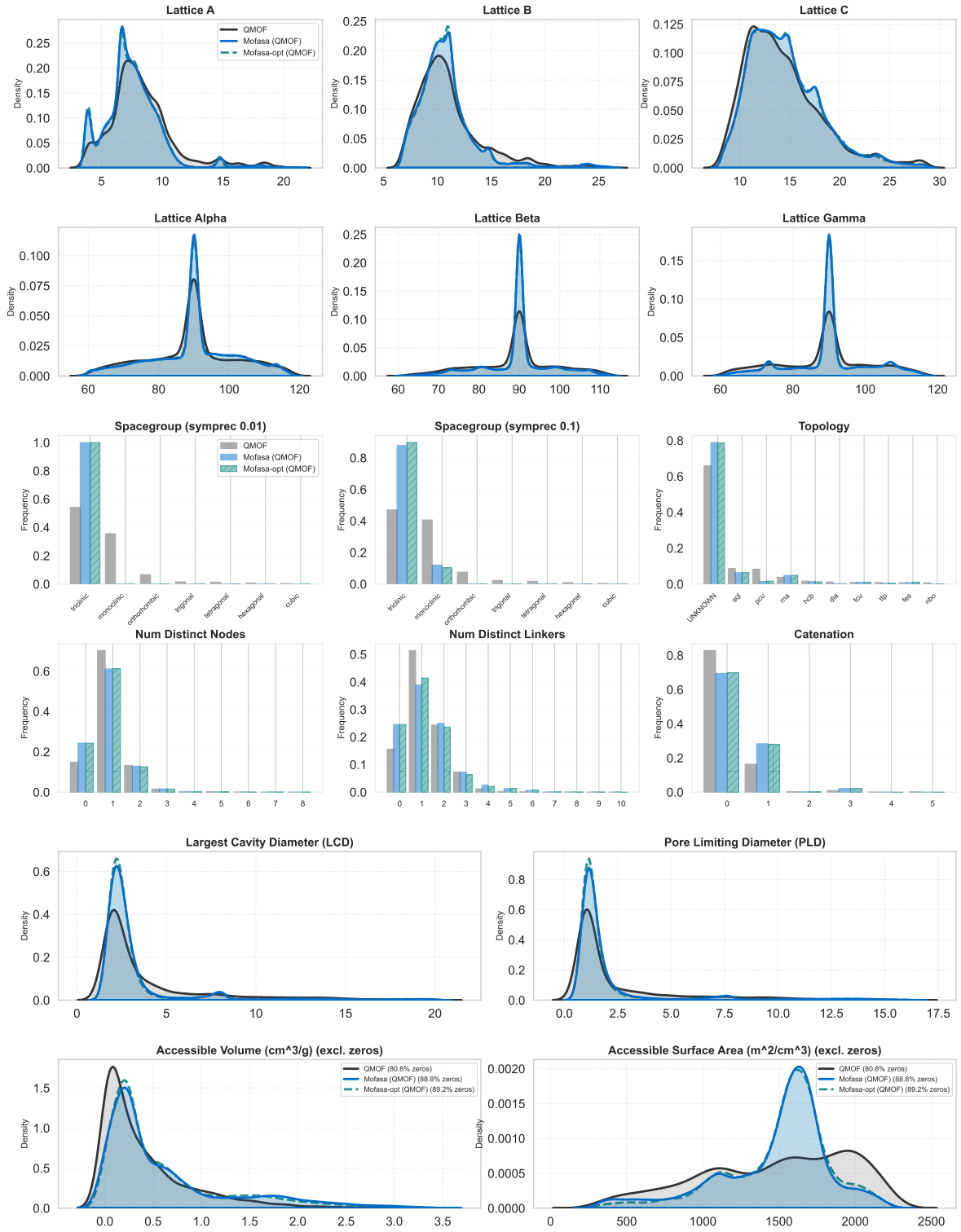

Figure 5: **Marginal distributions of simple properties of real data (QMOF, black) and generated samples (Mofasa, blue).** There is a significant amount of distributional overlap for all properties, with no signs of severe mode collapse. Nonetheless, several areas remain for improvement: generated MOFs are on average, slightly too small, too cubic, far too likely to be triclinic, more likely to fail `MOFid` identification (resulting in 0 nodes/linkers) and insufficiently porous as assessed by Zeo++ with a 1.86Å Nitrogen probe (note the legends of the final row, which state the percentage of systems for which zero volume/area is accessible).

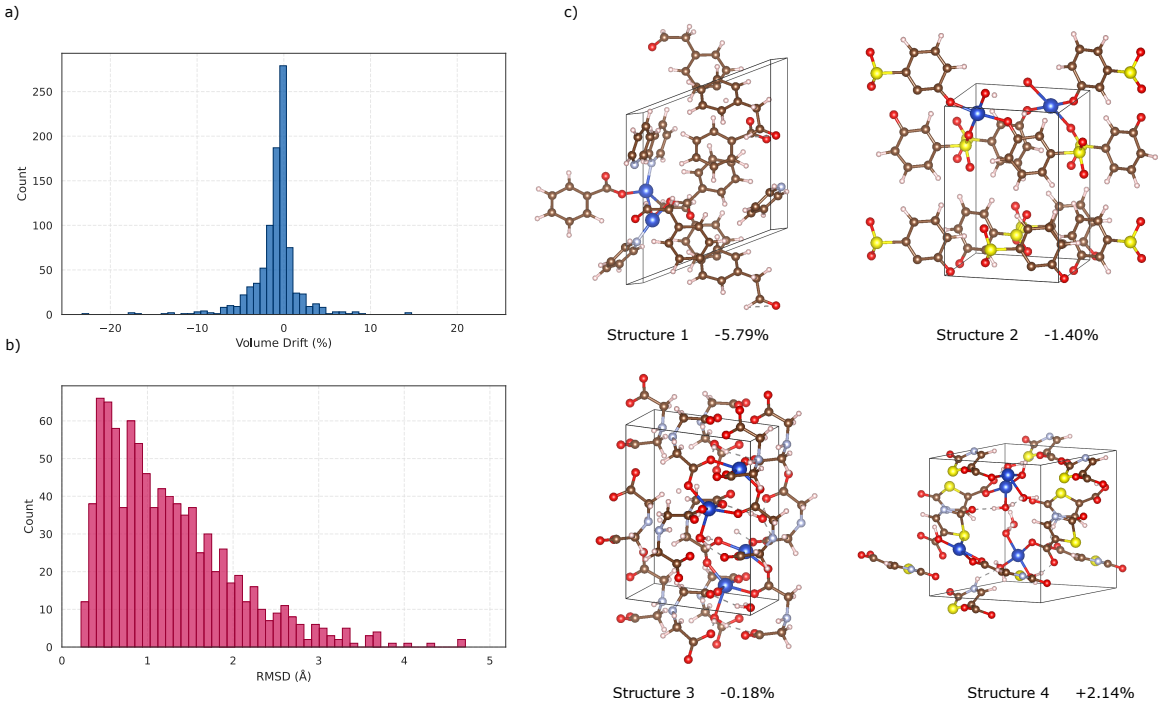

Figure 6: **Evaluation of dynamic stability.** Dynamic properties of 917 randomly selected samples from `MofasaDB`, calculated over a 50 ps MD trajectory. (a) Distribution of volume drift, calculated as the relative change in mean volume between the first and final 10 ps of the trajectory; (b) distribution of root mean square deviation (RMSD) between the initial and final frames; and (c) representative Cu-MOF trajectory endpoints with varying coordination geometries. The values below each structure indicate the volume drift percentage.

resulted in the largest contraction during the NPT run. In contrast, Structure 3 adopted the thermodynamically preferred square pyramidal geometry, characteristic of the prototypical copper paddle-wheel, and was the most rigid lattice. Finally, Structure 4 exhibited a distorted heteroleptic coordination environment, resulting in a volume expansion of 2.14%.

Altogether, the high stability rate and chemically intuitive structures observed in the copper-node examples underscore the physical plausibility and coordination diversity of the samples.

1.6 Conditional generation

All results presented so far pertain to unconditional samples. However, due to our multitask training setup (Section 3.1), Mofasa can accept several types of conditioning, including i) per-node atom-types (chemical formula) and ii) a set of molecular graphs (formula + bonds) for each node/linker fragment. Both types of conditioning provide useful information to the model, which ought to enhance sample quality.

In Figure 7, we confirm that conditioning increases `MOFChecker` validity across all system-sizes. We can distinguish three broad regimes: i) the $0-50$ atom range, where both types of conditioning boost performance significantly (+ 15 - 20 %) ii) the $50-250$ atom range, where unconditional < formula < formula+bonds, with $\sim$ 10% jumps between each iii) the $250-500$ atom range, which resembles the last case except with larger gaps of $15-20$ %. At 500 atoms, unconditional Mofasa decays to 5%, whilst formula + bonds remains steadily above 40%.

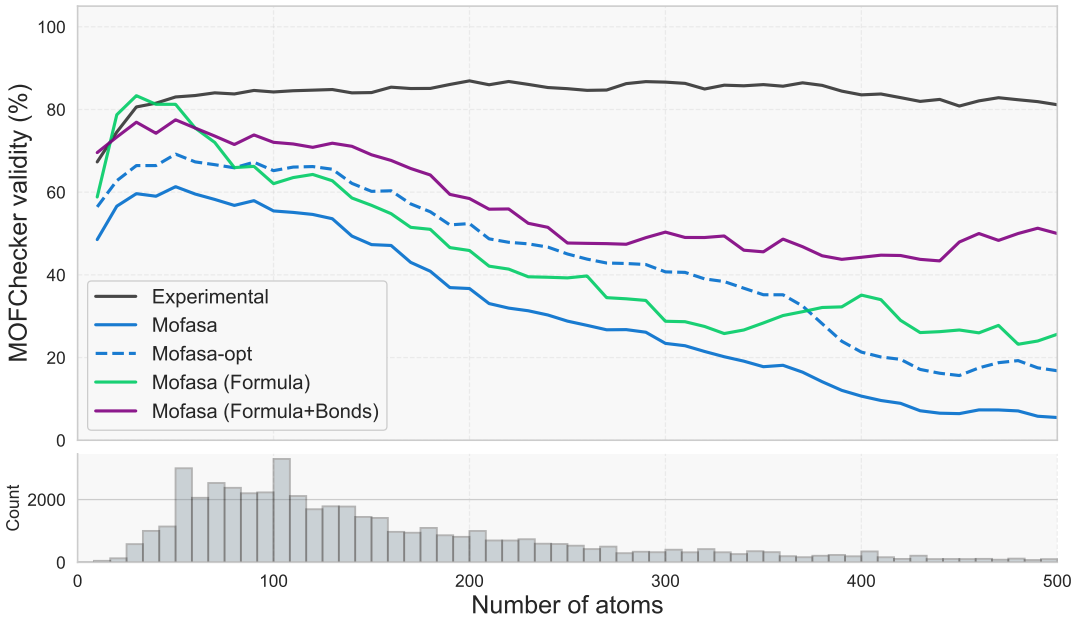

Figure 7: **MOFChecker validity vs system-size for different conditioning scenarios.** The more information that is conditioned on, the higher performance. The most informative conditioning (chemical formulas + bond-graphs) obtains > 40% validity at 500 atoms.

2 Significance for generative modeling of materials

Mofasa achieves high-fidelity generation—satisfying complex structural validity checks, exhibiting high degree of novelty, and maintaining high dynamic stability—all without relying on domain-specific compositional heuristics often used in MOF design. This success is significant not just for MOF discovery, but for the broader field of generative models in materials science.

Recent successful models such as MOFDiff [32], MOFFUSION [56] and MOF-BFN [45] have relied on a modular approach, treating MOFs as compositions of rigid building blocks. While this aligns with chemical intuition, it inherently constrains the model: architectures tailored to rigid building blocks are incompatible with non-modular domains. This incompatibility isolates MOF generation from the broader landscape of generative models for materials and hinders the development of a foundation model for atomic systems.

To unlock cross-domain transfer learning, we must operate at the atomic granularity. However, unified all-atom approaches, such as MatterGen [62], UniGenX [63] and ADiT [47], have, so far, not demonstrated scaling to large crystal systems like MOFs. They are often restricted to small crystal systems (<50 atoms) or suffer severe underfitting when modeling complex porous crystals. One might wonder if all-atom models suffer from a *data bottleneck*—that they require massive datasets to learn how to generate valid structures. Our results challenge this assumption by achieving state-of-the-art performance on systems up to 170 atoms using only ~14k structures in QMOF (and up to 500 atoms on 49k experimental structures).

This efficiency validates the all-atom approach as a viable backbone for future foundation models: if Mofasa can outperform domain-specific models, it is well-positioned to scale further. In the next section, we detail the implementation that enabled these results.

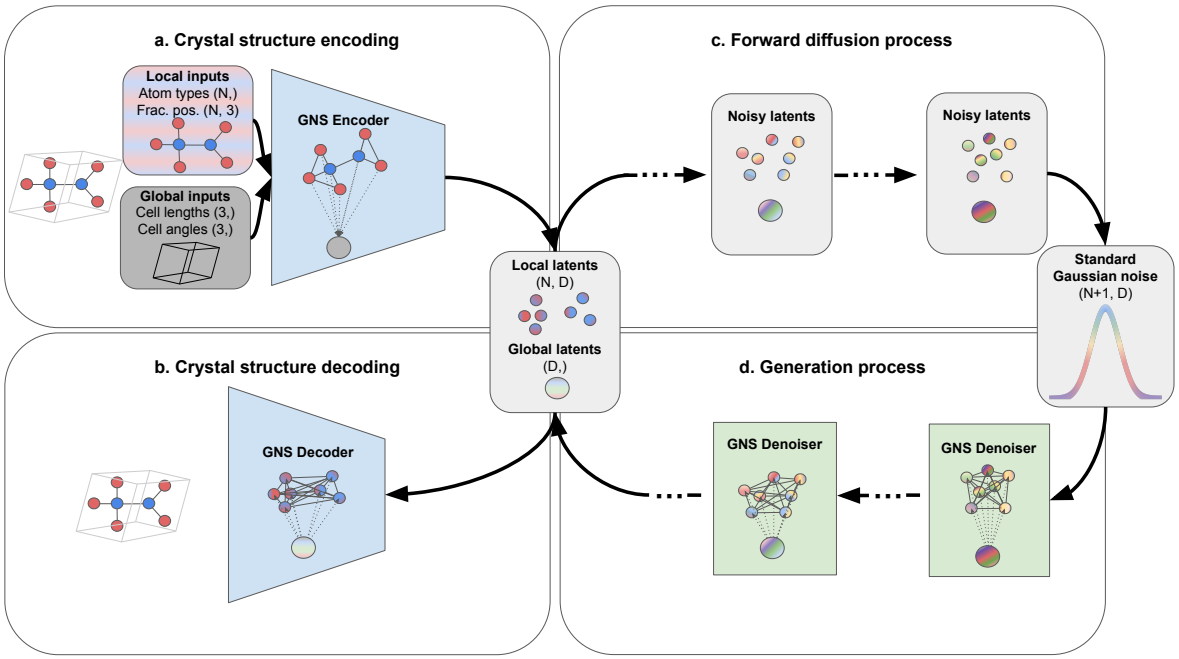

Figure 8: **Overview of Mofasa.** (a) The encoder maps the crystal structure to a continuous latent representation. (b) The decoder reconstructs the structure from the latent representation. (c) The forward process corrupts the latents with noise until it is indistinguishable from standard Gaussian noise. (d) The denoising model learns to reverse this process to generate clean latents.

3 An all-atom model for large crystal structure generation

Mofasa follows the standard latent diffusion framework [20], akin to recent generative approaches in materials science [47, 53]. As illustrated in Figure 8, the model consists of three main components: encoder, decoder, and a denoising model. These components interact across three distinct stages:

- **Autoencoding crystal structures (Panels a, b):** The encoder maps an input crystal structure (atomic types, fractional positions, and lattice parameters) into a continuous latent representation. This representation consists of *local latents*, encoding smoothed atomic neighborhoods, and *global latents*, capturing system-wide properties.
- **Latent diffusion (Panels a, c, d):** We train a denoising diffusion model to generate valid latent representations. During training, clean latents from the encoder are gradually corrupted by noise until they are indistinguishable from a standard Gaussian distribution (Panel c). A denoising model is trained to predict the "velocity" required to reverse this corruption (Panel d).
- **Generation (Panels b, d):** To generate novel crystal structures, we sample pure Gaussian noise, iteratively denoise it using the trained model to obtain a clean latent representation (Panel d), and then map it to the atomic domain using the decoder (Panel b).

3.1 Key Modeling Choices

While the full methodology is detailed in Appendix C, we highlight five important modeling decisions that enable high-fidelity generation.

Specialized processing for low- and high-level contexts. The encoder, the decoder, and the diffusion model share a common backbone: a hierarchical graph network simulator (GNS) [15]. Adapted from the Orb model [58], we leverage an architecture validated for fast, high-accuracy interatomic potential modeling. Importantly, the hierarchical message passing introduced here addresses the heterogeneity of crystal data, where local (atom-level) and global (system-level) features possess distinct properties necessitating specialized processing. This separation also enables directional information flow: the encoder uses it to aggregate atomistic details into the global latents, whereas the decoder and denoiser use it to broadcast global context back to local representations. This allows the model to simultaneously resolve fine-grained atomic details while maintaining precise long-range coherence.

Self-conditioning. Although the learned latent space is continuous, it encodes mixed discrete (atom types) and continuous (fractional positions) structural information. We find that self-conditioning—often used to bridge the gap between discrete and continuous diffusion—is also essential in this context, which we hypothesize is necessary for learning conditional dependencies between atom types (e.g., common bonding pairs) [57, Appendix A.3.2].

Tweaked cosine schedule. We use a cosine-based schedule with a log-SNR shift. This modification allocates significantly more training capacity (approx. 78% of diffusion timesteps) to the signal-dominated regime, which we found important for learning and generating fine-grained structural details.

Permutation symmetry breaking. Graph neural networks, such as our GNS backbone, are inherently permutation equivariant. However, the standard diffusion training objective regresses denoiser predictions against ordered targets, which introduces ambiguity: many permutations of the clean latents can result in the same noisy latents. The network cannot distinguish between permuted versions of the graph, yet the loss penalizes them differently based on the ordered targets. To resolve this ambiguity for *de novo* generation, we explicitly break permutation symmetry during training by conditioning the model on a canonical node ordering derived from the crystal graph topology. In contrast, we omit this ordering for conditional tasks (e.g., conformer generation or structure inpainting), as the conditioning signal provides sufficient context to implicitly resolve the symmetry.

Multi-task conditional training. To enable broad applicability, we train a single model for multiple tasks. By stochastically providing conditioning information, Mofasa learns to perform *de novo* generation, structure inpainting (e.g., generating linkers within a fixed node scaffold), and conditional generation based on chemical composition or bond topology.

4 Related work

Modular generation. This approach exploits the compositional structure of MOFs by generating individual building blocks—nodes, linkers, or topology—before assembling them into a full structure [38, 32, 56]. GHP-MOFAssemble [38] demonstrates this by generating linkers from molecular fragments using DiffLinker and combining them with pre-selected metal nodes into a fixed topology [34]. MOFDiff advances this via a coarse-grained (CG) diffusion process, where CG graph nodes, representing entire building blocks and their locations, are first generated via a diffusion model. Then, these CG nodes are replaced with atom-level details and assembled with a rigid assembly algorithm to determine the orientation of the otherwise fixed building block structure [32]. However, MOFDiff is restricted to a pre-determined library of rigid building blocks found in the training distribution, and hence hence has a significantly limited generalization. Addressing this limitation, MOFFlow-2 performs a similar two-step

Table 3: Comparison of generative models for Metal-Organic Frameworks and inorganic crystals.

Model	Approach	Resolution	Domain
Modular & Assembly-based			
GHP-MOFAssemble [38]	Diffusion (Linkers) + Rigid assembly	Building blocks	MOFs
MOFDiff [32]	Diffusion (CG) + Rigid assembly	Building blocks	MOFs
MOFFLOW-2 [49]	Autoregressive (Linkers) + Flow-matching (Assembly)	Building blocks	MOFs
MOFFUSION [56]	Latent diffusion (Mesh)	Implicit (Mesh)	MOFs
LLM & Agentic Approaches			
CrystaLLM [30]	Autoregressive	Text (CIF)	General
MatterGPT [31]	Autoregressive	Text (SLICES)	General
MOFGPT [40]	Autoregressive	Text (`MOFid`)	MOFs
Inizan et al. [44]	Agentic pipeline (LLM + Diffusion + Screening)	Hybrid (Text + SMILES + 3D)	MOFs
Direct All-Atom Generation			
CDVAE [22]	VAE + Diffusion	All-atom	General
MatterGen / UniMat [62, 39]	Diffusion	All-atom	General
FlowMM / CrystalFlow [35, 52]	Flow-matching	All-atom	General
ADiT / UAE-3D [47, 53]	Latent diffusion	All-atom	General
Mofasa (Ours)	**Latent diffusion**	**All-atom**	**General**

approach: first generating building block SMILES with an autoregressive model, then using a flow matching model infers the position and torsion angles for structural assembly [49]. Finally, MOFFUSION departs from modeling the discrete building blocks, instead using latent diffusion to first generate a continuous signed distance function that defines the geometry of the MOF, and then decoding these shapes into atomic structures [56].

All-atom *de novo* generation. A key requirement for crystal generation is the joint generation of atom types, positions, and lattice vectors. The seminal E(n)-equivariant diffusion model established the *all-atom* framework for jointly generating atomic composition and geometry of *molecules* directly in 3D space [19]. Simultaneously, CDVAE paved the way for a general-purpose all-atom approach to *inorganic crystal* generation [22]. Subsequently, MatterGen [62] improved generation fidelity by scaling the diffusion principles to a diverse, curated dataset of crystal structures up to 20 atoms. Simultaneously, UniMat [39] introduced a periodic-table inspired representation for molecular and crystal structures and used a diffusion model to generate small crystals. Flow-based model, such as FlowMM [35] and CrystalFlow [52] have also been used for modeling crystal structures, offering increased modeling flexibility and faster sampling. However, despite these advancements, the all-atom class of models remained primarily used on small inorganic materials (typically fewer than 60 atoms per unit cell).

Addressing this gap, ADiT proposed a unified latent diffusion-based framework for generating both periodic materials and non-periodic molecules [47, 53]. While promising for larger systems, ADiT focused on small systems (MP20, QM9), and its performance on complex porous crystals (QMOF) has been effectively limited (achieving 15.7% `MOFChecker` validity on QMOF). We adopt a similar diffusion-based framework as a conceptual foundation, re-engineering the model architecture, parameterization, and training objective. This yields state-of-the-art results on MOFs (achieving 52.9% validity on QMOF and 52.2% on a combination of experimental MOF DBs), while retaining a general-purpose capability applicable to diverse crystal modalities.

LLM-based generation. While diffusion-based approaches dominate 3D structure generation, autoregressive language models have also been applied to MOF crystal generation by treating structures as text sequences. Models such as CrystaLLM [30] and MatterGPT [31] are LLMs trained on crystallographic information files (CIF) and SLICES representations, respectively, learning the grammar of crystal structures. Similarly, MOFGPT uses an LLM to generate `MOFid` identifier sequences, incorporating a transformer-based property predictor and a reinforcement learning-based module for property-guided design [40]. Moving beyond direct sequence

generation, Inizan et al. [44] recently proposed a promising agentic pipeline where an LLM orchestrates linker SMILES generation while a separate diffusion model handles 3D structure generation. The generated structures are then geometry-optimized and screened, to ensure validity and synthesizability. By achieving a high-fidelity 3D generation directly, Mofasa offers a way to significantly reduce the cost of the post-processing steps, potentially accelerating agentic materials discovery by orders of magnitude.

5 Limitations and future work

While Mofasa achieves state-of-the-art performance in MOF generation, several limitations remain that present opportunities for future research.

An important constraint is the computational complexity of the architecture. The decoder and denoiser models operate on fully-connected graphs because, unlike in the encoder, the atomic positions are undefined during the early stages of generation, preventing the use of nearest neighbor methods typically used to construct sparse graphs. As a result, the memory requirement scales quadratically (N^2) with the number of atoms (N), making scaling beyond $N = 500$ atoms per unit cell difficult. Future work may investigate sparse graph approximations with long-range shortcut connections to maintain information flow while reducing the memory and computational overhead.

In addition, the GNS backbone architecture leaves room for optimization. We believe that explicit edge representations may be redundant. Moreover, the hierarchical message passing could be simplified to further reduce the number of computations.

Another key limitation is the requirement to specify the number of atoms (N) *a priori*. In *de novo* generation, this can be done by sampling N from the empirical training distribution. However, this poses a challenge for conditional generation, where specific property constraints (e.g., metal node composition) effectively alter the distribution of feasible atom counts. Currently, Mofasa does not have a mechanism to sample this conditional distribution.

Finally, while Mofasa is designed to be general-purpose, this study focused exclusively on Metal-Organic Frameworks. Investigating the model's performance and transfer learning between other material classes remains an important direction for future investigation.

6 Conclusion

We introduced Mofasa, an all-atom generative model based on latent diffusion. We evaluated the model on Metal-Organic Framework (MOF) generation—a domain of structurally complex porous materials—valued for carbon capture, gas separation, catalysis, and hydrogen storage. Mofasa achieves unprecedented generation fidelity, demonstrating high validity and dynamic stability. Notably, the model shows strong generalization, rediscovering metal nodes and topologies absent from the training distribution but observed in other experimental datasets.

Mofasa is the first model to successfully scale the all-atom approach to large crystal structures. We demonstrated the generation of complex unit cells containing up to 500 atoms, scaling well beyond prior inorganic crystal models (typically limited to < 60 atoms). Unlike tailored approaches that rely on MOF-specific decompositions, our method provides a more flexible framework, enabling the discovery of more novel chemistries.

Ultimately, these results extend beyond the MOF domain. By eliminating the need for domain-specific heuristics while maintaining high structural fidelity at scale, Mofasa validates the all-atom diffusion approach as a robust path toward future foundation models capable of transfer learning across the full range of materials.

References

[1] R. W. Grosse-Kunstleve, N. K. Sauter, and P. D. Adams. *Numerically Stable Algorithms for the Computation of Reduced Unit Cells*. In: *Acta Crystallographica Section A*. Foundations of Crystallography 60.1 (2004), pp. 1–6. ISSN: 0108-7673. DOI: 10.1107/S010876730302186X (cit. on p. 22).

[2] Erik Bitzek, Pekka Koskinen, Franz Gähler, Michael Moseler, and Peter Gumbsch. *Structural relaxation made simple*. In: *Physical review letters* 97.17 (2006), p. 170201 (cit. on p. 27).

[3] Tommi Junttila and Petteri Kaski. *Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs*. In: *Workshop on Algorithm Engineering and Experiments (ALENEX 2007)*. Society for Industrial and Applied Mathematics, 2007, pp. 135–149. DOI: 10.1137/1.9781611972870.13 (cit. on p. 26).

[4] Stefan Grimme, Jens Antony, Stephan Ehrlich, and Helge Krieg. *A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu*. In: *The Journal of chemical physics* 132.15 (2010) (cit. on pp. 6, 27).

[5] Thomas F Willems, Chris H Rycroft, Michaeel Kazi, Juan C Meza, and Maciej Haranczyk. *Algorithms and tools for high-throughput geometry-based analysis of crystalline porous materials*. In: *Microporous and Mesoporous Materials* 149.1 (2012), pp. 134–141 (cit. on pp. 6, 32).

[6] 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*. In: *Computational Materials Science* 68 (2013), pp. 314–319. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2012.10.028 (cit. on pp. 6, 19, 32).

[7] Peter G. Boyd and Tom K. Woo. *A Generalized Method for Constructing Hypothetical Nanoporous Materials of Any Net Topology from Graph Theory*. In: *CrystEngComm* 18.21 (2016), pp. 3777–3792. ISSN: 1466-8033. DOI: 10.1039/C6CE00407E (cit. on p. 19).

[8] Peter G. Boyd, Yongjin Lee, and Berend Smit. *Computational Development of the Nanoporous Materials Genome*. In: *Nature Reviews Materials* 2.8 (2017), p. 17037. ISSN: 2058-8437. DOI: 10.1038/natrevmats.2017.37 (cit. on pp. 3, 4, 19).

[9] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. *Neural Discrete Representation Learning*. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. arXiv: 1711.00937v2 (cit. on p. 24).

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks*. In: *International Conference on Learning Representations (ICLR)*. 2018. DOI: 10.48550/arXiv.1710.10903. arXiv: 1710.10903 (cit. on p. 20).

[11] Peter G. Boyd, Arunraj Chidambaram, Enrique García-Díez, Christopher P. Ireland, Thomas D. Daff, Richard Bounds, Andrzej Gładysiak, Pascal Schouwink, Seyed Mohamad Moosavi, M. Mercedes Maroto-Valer, Jeffrey A. Reimer, Jorge A. R. Navarro, Tom K. Woo, Susana Garcia, Kyriakos C. Stylianou, and Berend Smit. *Data-Driven Design of Metal–Organic Frameworks for Wet Flue Gas CO2 Capture*. In: *Nature* 576.7786 (2019), pp. 253–256. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1798-7 (cit. on pp. 3, 4, 19).

[12] Benjamin J. Bucior, Andrew S. Rosen, Maciej Haranczyk, Zhenpeng Yao, Michael E. Ziebel, Omar K. Farha, Joseph T. Hupp, J. Ilja Siepmann, Alán Aspuru-Guzik, and Randall Q. Snurr. *Identification Schemes for Metal–Organic Frameworks To Enable Rapid Search and Cheminformatics Analysis*. In: *Crystal Growth & Design* 19.11 (2019), pp. 6682–6697. ISSN: 1528-7483. DOI: 10.1021/acs.cgd.9b01050 (cit. on pp. 4, 19, 35).

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. In: *Neural Information Processing Systems (NeurIPS)*. 2020. DOI: 10.48550/arXiv.2006.11239. arXiv: 2006.11239 (cit. on pp. 24, 25).

[14] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. *Handling Incomplete Heterogeneous Data Using VAEs*. In: *Pattern Recognition* 107 (2020). ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107501. arXiv: 1807.03653 (cit. on p. 22).

[15] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. *Learning to Simulate Complex Physics with Graph Networks*. In: *International Conference on Machine Learning (ICML)*. arXiv, 2020. DOI: 10.48550/arXiv.2002.09405. arXiv: 2002.09405 (cit. on pp. 11, 20).

[16] Sangwon Lee, Baekjun Kim, Hyun Cho, Hooseung Lee, Sarah Yunmi Lee, Eun Seon Cho, and Jihan Kim. *Computational Screening of Trillions of Metal–Organic Frameworks for High-Performance Methane Storage*. In: *ACS Applied Materials & Interfaces* (2021). DOI: 10.1021/acsami.1c02471 (cit. on p. 5).

[17] Andrew S. Rosen, Shaelyn M. Iyer, Debmalya Ray, Zhenpeng Yao, Alán Aspuru-Guzik, Laura Gagliardi, Justin M. Notestein, and Randall Q. Snurr. *Machine Learning the Quantum-Chemical Properties of Metal–Organic Frameworks for Accelerated Materials Discovery*. In: *Matter* 4.5 (2021), pp. 1578–1597. ISSN: 2590-2385. DOI: 10.1016/j.matt.2021.02.015 (cit. on pp. 3, 19).

[18] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. *SoundStream: An End-to-End Neural Audio Codec*. In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 30 (2021). DOI: 10.48550/arXiv.2107.03312. arXiv: 2107.03312 (cit. on p. 22).

[19] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. *Equivariant Diffusion for Molecule Generation in 3D*. In: *International Conference on Machine Learning (ICML)*. 2022. DOI: 10.48550/arXiv.2203.17003. arXiv: 2203.17003 (cit. on pp. 12, 22).

[20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. DOI: 10.48550/arXiv.2112.10752. arXiv: 2112.10752 (cit. on pp. 10, 20, 24).

[21] Tim Salimans and Jonathan Ho. *Progressive Distillation for Fast Sampling of Diffusion Models*. In: *International Conference on Learning Representations (ICLR)*. 2022. DOI: 10.48550/arXiv.2202.00512. arXiv: 2202.00512 (cit. on p. 25).

[22] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. *Crystal Diffusion Variational Autoencoder for Periodic Material Generation*. In: *International Conference on Learning Representations (ICLR)*. 2022. DOI: 10.48550/arXiv.2110.06197. arXiv: 2110.06197 (cit. on pp. 1, 12, 23).

[23] Jake Burner, Jun Luo, Andrew White, Adam Mirmiran, Ohmin Kwon, Peter G. Boyd, Stephen Maley, Marco Gibaldi, Scott Simrod, Victoria Ogden, and Tom K. Woo. *ARC-MOF: A Diverse Database of Metal-Organic Frameworks with DFT-Derived Partial Atomic Charges and Descriptors for Machine Learning*. In: *Chemistry of Materials* (2023). DOI: 10.26434/chemrxiv-2022-mvr06 (cit. on pp. 3, 20).

[24] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. *Analog Bits: Generating Discrete Data Using Diffusion Models with Self-Conditioning*. In: *International Conference on Learning Representations (ICLR)*. 2023. DOI: 10.48550/arXiv.2208.04202. arXiv: 2208.04202 (cit. on p. 25).

[25] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. *TabDDPM: Modelling Tabular Data with Diffusion Models*. In: *International Conference on Machine Learning (ICML)*. 2023 (cit. on p. 22).

[26] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S.

Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. *De Novo Design of Protein Structure and Function with RFdiffusion*. In: *Nature* 620.7976 (2023), pp. 1089–1100. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06415-8 (cit. on p. 1).

[27] Minkai Xu, Alexander Powers, Ron Dror, Stefano Ermon, and Jure Leskovec. *Geometric Latent Diffusion Models for 3D Molecule Generation*. In: *International Conference on Machine Learning (ICML)*. 2023. DOI: 10.48550/arXiv.2305.01140. arXiv: 2305.01140 (cit. on p. 22).

[28] Yizhi Xu, Joseph M. Marrett, Hatem M. Titi, James P. Darby, Andrew J. Morris, Tomislav Friščić, and Mihails Arhangelskis. *Experimentally Validated Ab Initio Crystal Structure Prediction of Novel Metal–Organic Framework Materials*. In: *Journal of the American Chemical Society* 145.6 (2023), pp. 3515–3525. ISSN: 0002-7863. DOI: 10.1021/jacs.2c12095 (cit. on p. 5).

[29] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. *Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3*. In: *Nature* 630.8016 (2024), pp. 493–500. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07487-w (cit. on p. 1).

[30] Luis M Antunes, Keith T Butler, and Ricardo Grau-Crespo. *Crystal structure generation with autoregressive large language modeling*. In: *Nature Communications* 15.1 (2024), p. 10570 (cit. on p. 12).

[31] Yan Chen, Xueru Wang, Xiaobin Deng, Yilun Liu, Xi Chen, Yunwei Zhang, Lei Wang, and Hang Xiao. *MatterGPT: A generative transformer for multi-property inverse design of solid-state materials*. In: *arXiv preprint arXiv:2408.07608* (2024) (cit. on p. 12).

[32] Xiang Fu, Tian Xie, Andrew S. Rosen, Tommi Jaakkola, and Jake Smith. *MOFDiff: Coarse-grained Diffusion for Metal-Organic Framework Design*. In: *International Conference on Learning Representations (ICLR)*. 2024. DOI: 10.48550/arXiv.2310.10732. arXiv: 2310.10732 (cit. on pp. 4, 9, 11, 12).

[33] Lawson T Glasby, Joan L Cordiner, Jason C Cole, and Peyman Z Moghadam. *Topological characterization of metal–organic frameworks: a perspective*. In: *Chemistry of Materials* 36.19 (2024), pp. 9013–9030 (cit. on p. 5).

[34] Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. *Equivariant 3D-conditional Diffusion Model for Molecular Linker Design*. In: *Nature Machine Intelligence* 6.4 (2024), pp. 417–427. ISSN: 2522-5839. DOI: 10.1038/s42256-024-00815-9 (cit. on p. 11).

[35] Benjamin Kurt Miller, Ricky T. Q. Chen, Anuroop Sriram, and Brandon M. Wood. *FlowMM: Generating Materials with Riemannian Flow Matching*. In: *International Conference on Machine Learning (ICML)*. 2024. DOI: 10.48550/arXiv.2406.04713. arXiv: 2406.04713 (cit. on p. 12).

[36] Peyman Z Moghadam, Yongchul G Chung, and Randall Q Snurr. *Progress toward the computational discovery of new metal–organic framework adsorbents for energy applications*. In: *Nature Energy* 9.2 (2024), pp. 121–133 (cit. on p. 5).

[37] Mark Neumann, James Gin, Benjamin Rhodes, Steven Bennett, Zhiyi Li, Hitarth Choubisa, Arthur Hussey, and Jonathan Godwin. *Orb: A Fast, Scalable Neural Network Potential*. Tech. rep. arXiv:2410.22570. arXiv, 2024. arXiv: 2410.22570 (cit. on pp. 6, 20, 22, 23, 27).

[38] Hyun Park, Xiaoli Yan, Ruijie Zhu, Eliu A Huerta, Santanu Chaudhuri, Donny Cooper, Ian Foster, and Emad Tajkhorshid. *A generative artificial intelligence framework based on a molecular diffusion model for the design of metal-organic frameworks for carbon capture*. In: *Communications Chemistry* 7.1 (2024) (cit. on pp. 11, 12).

[39] Sherry Yang, KwangHwan Cho, Amil Merchant, Pieter Abbeel, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. *Scalable Diffusion for Materials Generation*. 2024. DOI: 10.48550/arXiv.2311.09235. arXiv: 2311.09235 (cit. on p. 12).

[40] Srivathsan Badrinarayanan, Rishikesh Magar, Akshay Antony, Radheesh Sharma Meda, and Amir Barati Farimani. *MOFGPT: Generative Design of Metal-Organic Frameworks Using Language Models*. In: *Journal of Chemical Information and Modeling* (2025). DOI: 10.48550/arXiv.2506.00198. arXiv: 2506.00198 (cit. on p. 12).

[41] Orion Cohen, Janosh Riebesell, Rhys Goodall, Adeesh Kolluru, Stefano Falletta, Joseph Krause, Jorge Colindres, Gerbrand Ceder, and Abhijeet S Gangan. *TorchSim: An efficient atomistic simulation engine in PyTorch*. In: *AI for Science* 1.2 (2025), p. 025003 (cit. on p. 27).

[42] Chenru Duan, Aditya Nandy, Sizhan Liu, Yuanqi Du, Liu He, Yi Qu, Haojun Jia, and Jin-Hu Dou. *Building-Block Aware Generative Modeling for 3D Crystals of Metal Organic Frameworks*. 2025. DOI: 10.48550/arXiv.2505.08531. arXiv: 2505.08531 (cit. on p. 5).

[43] Christopher Fifty, Ronald G. Junkins, Dennis Duan, Aniketh Iyengar, Jerry W. Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. *Restructuring Vector Quantization with the Rotation Trick*. In: *International Conference on Learning Representations (ICLR)*. 2025. DOI: 10.48550/arXiv.2410.06424. arXiv: 2410.06424 (cit. on p. 22).

[44] Theo Jaffrelot Inizan, Sherry Yang, Aaron Kaplan, Yen-hsu Lin, Jian Yin, Saber Mirzaei, Mona Abdelgaid, Ali H Alawadhi, KwangHwan Cho, Zhiling Zheng, et al. *System of agentic AI for the discovery of metal-organic frameworks*. In: *arXiv preprint arXiv:2504.14110* (2025) (cit. on pp. 3, 12, 13).

[45] Rui Jiao, Hanlin Wu, Wenbing Huang, Yuxuan Song, Yawen Ouyang, Yu Rong, Tingyang Xu, Pengju Wang, Hao Zhou, Wei-Ying Ma, Jingjing Liu, and Yang Liu. *MOF-BFN: Metal-Organic Frameworks Structure Prediction via Bayesian Flow Networks*. In: *Neural Information Processing Systems (NeurIPS)*. 2025 (cit. on pp. 3, 9, 19, 20).

[46] Xin Jin, Kevin Maik Jablonka, Elias Moubarak, Yutao Li, and Berend Smit. *MOFChecker: a package for validating and correcting metal–organic framework (MOF) structures*. In: *Digital Discovery* (2025) (cit. on pp. 3, 34).

[47] Chaitanya K. Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary W. Ulissi. *All-Atom Diffusion Transformers: Unified Generative Modelling of Molecules and Materials*. In: *International Conference on Machine Learning (ICML)*. 2025. DOI: 10.48550/arXiv.2503.03965. arXiv: 2503.03965 (cit. on pp. 1–3, 9, 10, 12, 19).

[48] Mikkel Juelsholt. *Continued Challenges in High-Throughput Materials Predictions: MatterGen Predicts Compounds from the Training Dataset*. 2025. DOI: 10.26434/chemrxiv-2025-mkls8 (cit. on p. 4).

[49] Nayoung Kim, Seongsu Kim, and Sungsoo Ahn. *Flexible MOF Generation with Torsion-Aware Flow Matching*. In: *Neural Information Processing Systems (NeurIPS)*. 2025. DOI: 10.48550/arXiv.2505.17914. arXiv: 2505.17914 (cit. on pp. 3, 12, 19, 20).

[50] Hendrik Kraß, Ju Huang, and Seyed Mohamad Moosavi. *MOFSimBench: Evaluating Universal Machine Learning Interatomic Potentials In Metal–Organic Framework Molecular Modeling*. In: *arXiv preprint arXiv:2507.11806* (2025) (cit. on pp. 6, 27).

[51] Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, sriniker, Peter Gedeck, Gareth Jones, Eisuke Kawashima, Schneider Nadine, Dan Nealschneider, Andrew Dalke, tadhurst-cdd, Matt Swain, Brian Cole, Samo Turk, Aleksandr Savelev, Niels Maeder, Alain Vaucher, Maciej Wójcikowski, Hussein Faara, Ichiru Take, Rachel Walker, Vincent F. Scalfani, Daniel Probst, Kazuya Ujihara, Axel Pahl,

Guillaume Godin, and Juuso Lehtivarjo. *RDKit: Open-source Cheminformatics.* Zenodo. 2025. DOI: 10.5281/zenodo.17746401 (cit. on p. 33).

[52] Xiaoshan Luo, Zhenyu Wang, Qingchang Wang, Xuechen Shao, Jian Lv, Lei Wang, Yanchao Wang, and Yanming Ma. *Crystalflow: A flow-based generative model for crystalline materials*. In: *Nature Communications* 16.1 (2025) (cit. on p. 12).

[53] Yanchen Luo, Zhiyuan Liu, Yi Zhao, Sihang Li, Hengxing Cai, Kenji Kawaguchi, Tat-Seng Chua, Yang Zhang, and Xiang Wang. *Towards Unified and Lossless Latent Space for 3D Molecular Latent Diffusion Modeling*. In: *Neural Information Processing Systems (NeurIPS)*. 2025. DOI: 10.48550/arXiv.2503.15567. arXiv: 2503.15567 (cit. on pp. 10, 12).

[54] Masahiro Negishi, Hyunsoo Park, Kinga O. Mastej, and Aron Walsh. *Continuous Uniqueness and Novelty Metrics for Generative Modeling of Inorganic Crystals*. In: *AI for Accelerated Materials Discovery (AI4Mat) Workshop (NeurIPS)*. 2025. DOI: 10.48550/arXiv.2510.12405. arXiv: 2510.12405 (cit. on p. 4).

[55] Nobel Prize Outreach. *Press release: The Nobel Prize in Chemistry 2025*. NobelPrize.org. Accessed: 2025-11-25. 2025. URL: https://www.nobelprize.org/prizes/chemistry/2025/press-release/ (cit. on p. 1).

[56] Junkil Park, Youhan Lee, and Jihan Kim. *Multi-Modal Conditional Diffusion Model Using Signed Distance Functions for Metal-Organic Frameworks Generation*. In: *Nature Communications* 16.1 (2025), p. 34. ISSN: 2041-1723. DOI: 10.1038/s41467-024-55390-9 (cit. on pp. 3, 9, 11, 12).

[57] Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. *CANDI: Hybrid Discrete-Continuous Diffusion Models*. 2025. DOI: 10.48550/arXiv.2510.22510. arXiv: 2510.22510 (cit. on pp. 11, 25).

[58] Benjamin Rhodes, Sander Vandenhaute, Vaidotas Simkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann. *Orb-v3: Atomistic Simulation at Scale*. Tech. rep. arXiv:2504.06231. arXiv, 2025. arXiv: 2504.06231 (cit. on pp. 4, 6, 11, 19, 20, 22, 27, 33).

[59] Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. *Rao-Blackwell Gradient Estimators for Equivariant Denoising Diffusion*. In: *Neural Information Processing Systems (NeurIPS)*. 2025. DOI: 10.48550/arXiv.2502.09890. arXiv: 2502.09890 (cit. on p. 26).

[60] Brandon M Wood, Misko Dzamba, Xiang Fu, Meng Gao, Muhammed Shuaibi, Luis Barroso-Luque, Kareem Abdelmaqsoud, Vahe Gharakhanyan, John R Kitchin, Daniel S Levine, et al. *UMA: A Family of Universal Models for Atoms*. In: *arXiv preprint arXiv:2506.23971* (2025) (cit. on p. 4).

[61] Xiaoli Yan, Nathaniel Hudson, Hyun Park, Daniel Grzenda, J. Gregory Pauloski, Marcus Schwarting, Haochen Pan, Hassan Harb, Samuel Foreman, Chris Knight, Tom Gibbs, Kyle Chard, Santanu Chaudhuri, Emad Tajkhorshid, Ian Foster, Mohamad Moosavi, Logan Ward, and E. A. Huerta. *MOFA: Discovering Materials for Carbon Capture with a GenAI- and Simulation-Based Workflow*. 2025. DOI: 10.48550/arXiv.2501.10651. arXiv: 2501.10651 (cit. on pp. 3, 5).

[62] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, Han Yang, Hongxia Hao, Jielan Li, Ryota Tomioka, and Tian Xie. *MatterGen: A Generative Model for Inorganic Materials Design*. In: *Nature* (2025). DOI: 10.48550/arXiv.2312.03687. arXiv: 2312.03687 (cit. on pp. 1, 9, 12).

[63] Gongbo Zhang, Yanting Li, Renqian Luo, Pipi Hu, Zeru Zhao, Lingbo Li, Guoqing Liu, Zun Wang, Ran Bi, Kaiyuan Gao, Liya Guo, Yu Xie, Chang Liu, Jia Zhang, Tian Xie, Robert Pinsler, Claudio Zeni, Ziheng Lu, Yingce Xia, Marwin Segler, Maik Riechert, Li Yuan, Lei Chen, Haiguang Liu, and Tao Qin. *UniGenX: Unified Generation of Sequence and Structure with Autoregressive Diffusion*. 2025. DOI: 10.48550/arXiv.2503.06687. arXiv: 2503.06687 (cit. on p. 9).

[64] Guobin Zhao, Logan Brabson, Saumil Chheda, Ju Huang, Haewon Kim, Kunhuan Liu, Kenji Mochida, Thang Pham, Prerna Prerna, Gianmarco Terrones, Sunghyun Yoon, Lionel Zoubritzky, François-Xavier Coudert, Maciej Haranczyk, Heather Kulik, Mohamad Moosavi, David Sholl, Ilja Siepmann, Randall Snurr, and Yongchul Chung. *CoRE MOF DB: A Curated Experimental Metal-Organic Framework Database with Machine-Learned Properties for Integrated Material-Process Screening*. 2025. DOI: 10.26434/chemrxiv-2024-nvmnr-v2 (cit. on pp. 3, 20).

Appendix A: MofasaDB

To help the scientific community build on top of this work, we provide a dataset of 201,926 raw and Orb-optimized structures generated by Mofasa-QMOF (trained on the QMOF (<170 atoms) subset). To enable property-based screening, all samples include the pre-computed properties detailed in Appendix H. For each sample, we additionally include latent embeddings from the `orb-v3-direct-20-omat` MLIP [58]. The data can be accessed at at https://huggingface.co/datasets/Orbital-Materials/MofasaDB and is released under the CC-BY 4.0 license.

Additionally, we also make available an interactive web interface for exploring and screening the unoptimized database; available at https://mofux.ai/. The exploration is supported by the included latents, allowing interactive search in the `Orb` latent space, which clusters similar MOFs as demonstrated in F.

Appendix B: Training datasets & pre-processing

Here, we detail the datasets used to train Mofasa and the relevant baselines. All crystal structures used to train Mofasa were standardized to their primitive unit cells using `pymatgen` [6]. Additionally, we applied the `metal-oxo` algorithm from the `MOFid` package [12] to decompose the MOF structures. This generated atom-level labels (metal nodes, bridges, organic linkers, and solvents) which were used for conditional training and downstream analysis.

ADiT + QMOF. The ADiT baseline [47] was trained on the Quantum MOF (QMOF) database [17], which contains 20k experimentally synthesized MOFs with structures relaxed via DFT. The authors filtered the dataset to include only structures with fewer than 150 atoms in the unit cell (corresponding to the 81st percentile of the full distribution). This resulted in a final training set of approximately 14k samples after a training/validation/test split.

Mofasa + QMOF. To compare against the ADiT baseline, we trained Mofasa on QMOF systems with up to 170 atoms in the unit cell (corresponding to 85th percentile). The data was randomly split 80/10/10% into training, validation, and test sets, resulting in a final training dataset of approximately 14k structures.

MOFFlow-2 + BW. The MOFFlow-2 baseline [49] was trained on the Boyd-Woo (BW) database [8, 11], consisting of 358k total hypothetical MOFs constructed using the ToBasCCo assembly method [7]. The authors used `MOFid` to decompose MOFs into the building blocks, discarded any structures with more than 20 building blocks, and filtered out all systems that failed `MOFChecker` criteria. This resulted in a training dataset of 157k samples with 100% `MOFChecker` validity.

MOF-BFN + BW. The MOF-BFN baseline [45] also used the BW database. In contrast to MOFFlow-2 they only filtered out structures with more than 200 building blocks. The authors did not report the exact final training set size.

Mofasa + BW. For comparison with MOFFlow-2 [49] and MOF-BFN [45] we use Mofasa on ARC-MOF DB0 split [23], which consists of 263k systems after additional ARC-MOF structure checks, including metal oxidation states, atom overlaps, unrealistically small unit cells, and over-coordination. We further filter out systems that have more than 500 atoms in the unit cell. After a 95/4/1% training/validation/test split, we trained Mofasa on 238k systems.

Mofasa + Experimental. We define the "Experimental" dataset as a combination of several available experimental sources: QMOF, the computatioSn-ready split of CoRE-MOF-2024 [64], and the DB12 and DB14 subsets from ARC-MOF [23]. This combinations results in a total of 49k systems with an average `MOFChecker` validity of 84.2%. When training Mofasa we only use systems up to 500 atoms from the Experimental dataset.

Appendix C: Building Mofasa

We adopt the latent diffusion framework [20] involving an autoencoder and a denoising diffusion model. The autoencoder learns a continuous latent representation from a crystal structure's mixed categorical and continuous features, while the diffusion model learns to generate these latent representations. The autoencoder and diffusion model components share a common graph neural network (GNN) backbone, which is an adaptation of an established architecture previously validated in machine-learned interatomic potential models [37, 58] for fast and accurate molecular simulations.

C.1 Hierarchical GNS backbone

The shared backbone architecture for both autoencoder and diffusion models is an extended version of the Graph Network Simulator (GNS) [15]. This architecture incorporates two key modifications: (1) the addition of a graph attention mechanism [10], and (2) a hierarchical message passing scheme. This hierarchical scheme is motivated by the heterogeneity of atomistic structural data, where structural features at the *local* and *global* levels have fundamentally distinct properties, hence necessitating specialized processing. We enable this by defining the graph components as distinct local and global sets, where local components may represent individual atoms (or atomic neighborhoods) and global components capture broader system properties like lattice parameters.

We denote the sets of local nodes, local edges, global nodes, and global edges as $\boldsymbol{V}_{\mathrm{L}} = \{\boldsymbol{v}_i^{\mathrm{L}}\}$, $\boldsymbol{E}_{\mathrm{L}} = \{\boldsymbol{e}_{j,k}^{\mathrm{L}}\}$, $\boldsymbol{V}_{\mathrm{G}} = \{\boldsymbol{v}_l^{\mathrm{G}}\}$, and $\boldsymbol{E}_{\mathrm{G}} = \{\boldsymbol{e}_{m,n}^{\mathrm{G}}\}$, respectively. All edges are directional. While local edges are restricted to connecting local nodes, global edges are allowed to cross-communicate between the global and local contexts, handled by a dedicated neural network distinct from the local message passing networks. Precise construction of these sets will be introduced in subsequent sections.

The processing of the GNS is outlined in Algorithm 1, with the key modification being the message passing stage. Each message passing iteration consists of two sequential steps: a local step, where information is exchanged exclusively between local nodes via local edges; and a global step, which processes a joint graph of global and local nodes. This global step aggregates atom-level information to update the global context and, simultaneously, broadcasts system-level context to the local nodes connected via the directed global edges. The message computation also involves an edge-attention mechanism, adopted directly from Orb [37, Section 2.2].

Algorithm 1 Hierarchical GNS processing

Inputs:

Features: Local nodes $\boldsymbol{V}_{\mathrm{L}}^0$ and edges $\boldsymbol{E}_{\mathrm{L}}^0$, and global nodes $\boldsymbol{V}_{\mathrm{G}}^0$ and edges $\boldsymbol{E}_{\mathrm{G}}^0$.
Embedders: Local embedding MLP, $\textsc{Embed}_{\mathrm{L}}(\cdot)$, and global embedding MLP, $\textsc{Embed}_{\mathrm{G}}(\cdot)$.
Message passing networks: Local edge and node networks, $\phi_{\mathrm{E,L}}^t(\cdot)$ and $\phi_{\mathrm{N,L}}^t(\cdot)$, and global edge and node networks, $\phi_{\mathrm{E,G}}^t(\cdot)$ and $\phi_{\mathrm{N,G}}^t(\cdot)$ for $t \in \{1, \ldots, T\}$.
Edge-attention functions: $\psi_1^t(\cdot)$ and $\psi_2^t(\cdot)$ for $t \in \{1, \ldots, T\}$.
Read-out layers: Local read-out MLP, $\textsc{Read}_{\mathrm{L}}(\cdot)$, and global read-out MLP, $\textsc{Read}_{\mathrm{G}}(\cdot)$.

def $\textsc{ComputeDeltas}(\boldsymbol{V}, \boldsymbol{E}; \phi_{\mathrm{E}}, \phi_{\mathrm{N}}, \psi_1, \psi_2)$
  for each $\boldsymbol{e}_{i,j} \in \boldsymbol{E}, \boldsymbol{v}_i \in \boldsymbol{V}, \boldsymbol{v}_j \in \boldsymbol{V}$: ▷ Compute edge residuals for each edge
    $\Delta \boldsymbol{e}_{i,j} \leftarrow \phi_{\mathrm{E}}([\boldsymbol{e}_{i,j}, \boldsymbol{v}_i, \boldsymbol{v}_j])$
  for each $\boldsymbol{v}_i \in \boldsymbol{V}$: ▷ Compute node residuals for each node
    $\boldsymbol{m}_i^1 \leftarrow \sum_{j \in \mathcal{R}(i)} \psi_1(\boldsymbol{e}_{i,j}) \Delta \boldsymbol{e}_{i,j}$; $\boldsymbol{m}_i^2 \leftarrow \sum_{j \in \mathcal{S}(i)} \psi_2(\boldsymbol{e}_{j,i}) \Delta \boldsymbol{e}_{j,i}$
    $\Delta \boldsymbol{v}_i \leftarrow \phi_{\mathrm{N}}([\boldsymbol{v}_i, \boldsymbol{m}_i^1, \boldsymbol{m}_i^2])$
  return $\{\Delta \boldsymbol{v}_i\}_i, \{\Delta \boldsymbol{e}_{i,j}\}_{i,j}$

1. Embedding Stage
$\boldsymbol{V}_{\mathrm{L}}^1, \boldsymbol{E}_{\mathrm{L}}^1 \leftarrow \textsc{Embed}_{\mathrm{L}}(\boldsymbol{V}_{\mathrm{L}}^0, \boldsymbol{E}_{\mathrm{L}}^0)$; $\boldsymbol{V}_{\mathrm{G}}^1, \boldsymbol{E}_{\mathrm{G}}^1 \leftarrow \textsc{Embed}_{\mathrm{G}}(\boldsymbol{V}_{\mathrm{G}}^0, \boldsymbol{E}_{\mathrm{G}}^0)$

2. Message Passing Loop
for $t = 1$ **to** T
  # Local Step: Update local components
  $\Delta \boldsymbol{V}_{\mathrm{L}}^t, \Delta \boldsymbol{E}_{\mathrm{L}}^t \leftarrow \textsc{ComputeDeltas}(\boldsymbol{V}_{\mathrm{L}}^t, \boldsymbol{E}_{\mathrm{L}}^t; \phi_{\mathrm{E,L}}^t, \phi_{\mathrm{N,L}}^t, \psi_{1,\mathrm{L}}^t, \psi_{2,\mathrm{L}}^t)$
  $\boldsymbol{V}_{\mathrm{L}}^{t+1} \leftarrow \boldsymbol{V}_{\mathrm{L}}^t + \Delta \boldsymbol{V}_{\mathrm{L}}^t$; $\boldsymbol{E}_{\mathrm{L}}^{t+1} \leftarrow \boldsymbol{E}_{\mathrm{L}}^t + \Delta \boldsymbol{E}_{\mathrm{L}}^t$

  # Global Step: Update global components, incorporating local node info
  $\boldsymbol{V}_{\mathrm{Joint}}^t \leftarrow \boldsymbol{V}_{\mathrm{G}}^t \cup \boldsymbol{V}_{\mathrm{L}}^{t+1}$
  $\Delta \boldsymbol{V}_{\mathrm{Joint}}^t, \Delta \boldsymbol{E}_{\mathrm{G}}^t \leftarrow \textsc{ComputeDeltas}(\boldsymbol{V}_{\mathrm{Joint}}^t, \boldsymbol{E}_{\mathrm{G}}^t; \phi_{\mathrm{E,G}}^t, \phi_{\mathrm{N,G}}^t, \psi_{1,G}^t, \psi_{2,G}^t)$
  $\boldsymbol{V}_{\mathrm{G}}^{t+1} \leftarrow \boldsymbol{V}_{\mathrm{G}}^t + \Delta \boldsymbol{V}_{\mathrm{Joint|G}}^t$; $\boldsymbol{E}_{\mathrm{G}}^{t+1} \leftarrow \boldsymbol{E}_{\mathrm{G}}^t + \Delta \boldsymbol{E}_{\mathrm{G}}^t$

  # Broadcast global information to connected local nodes
  $\boldsymbol{V}_{\mathrm{L}}^{t+1} \leftarrow \textsc{ScatterAdd}(\boldsymbol{V}_{\mathrm{L}}^{t+1}, \Delta \boldsymbol{V}_{\mathrm{Joint|L}}^t)$

3. Read-out Stage
return $\textsc{Read}(\boldsymbol{V}_{\mathrm{L}}^{T+1}), \textsc{Read}(\boldsymbol{V}_{\mathrm{G}}^{T+1})$

C.2 Learning to represent crystal systems

A crystal structure is defined in terms of its periodically repeating unit cell containing N atoms. To ensure a unique representation across infinitely-many arbitrary unit cell choices, we standardize the structures using primitive cell reduction and Niggli reduction [1]. We represent this standardized structure with the tuple $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$, where $\boldsymbol{A} = \{a_i\}_{i=1}^N \in \mathbb{A}^N$ denotes the atomic types, $\boldsymbol{F} = \{\boldsymbol{f}_i\}_{i=1}^N \in [0,1)^{N\times 3}$ represents the fractional positions relative to the unit cell, and $\boldsymbol{L} = (a, b, c, \alpha, \beta, \gamma) \in \mathbb{R}^3_{>0} \times [\frac{\pi}{3}, \frac{2\pi}{3}]^3$ is the rotationally-invariant lattice representation consisting of three length parameters and three angles describing the basis vectors.

While the above representation is rotationally invariant, the fractional positions depend on the arbitrary choice of the unit cell origin. To ensure the model learns robust representations, we use data augmentation. For every training sample, we shift all fractional positions by a random vector $\boldsymbol{u} \sim \mathcal{U}[0,1)^3$, obtaining the augmented sample $\boldsymbol{S}' = (\boldsymbol{A}, (\boldsymbol{F} + \boldsymbol{u}) \bmod 1, \boldsymbol{L})$, where the modulo operator corresponds to the periodic boundary condition and ensures that the fractional positions remain in $[0,1)^3$. For conciseness in this paper we will use $\boldsymbol{S}$ and $\boldsymbol{S}'$ interchangeably, since they represent the same system.

Learning to directly generate mixed categorical and continuous data, such as crystal structures, presents significant challenges [e.g., 14, 19, 25, 27]. To mitigate these issues, we train an autoencoder that maps the crystal structure $\boldsymbol{S}$ to a continuous latent representation $\boldsymbol{Z}$ using an encoder $\textsc{Enc}(\cdot)$, and reconstructs it via a decoder $\textsc{Dec}(\cdot)$.

C.3 Mapping crystal systems to latent representations

The encoder $\textsc{Enc}(\cdot)$ maps a crystal structure $\boldsymbol{S}$ to a continuous latent representation $\boldsymbol{Z}$ using the GNS architecture defined in Section C.1. We adapt the implementation and input featurization from Orb [37, 58], a state-of-the-art MLIP, leveraging its ability to produce robust representations for modeling potential energy and atomic forces.

Feature construction. We construct the input local node features $\boldsymbol{V}_\text{L}$ by concatenating the one-hot representations of the atomic types $\boldsymbol{A}$ with sinusoidal embeddings of the fractional coordinates $\boldsymbol{F}$. The input local edges $\boldsymbol{E}_\text{L}$ are then constructed using a unit cell-aware nearest neighbor scheme; the resulting edge displacement vectors are encoded using Bessel basis functions. To incorporate lattice information in the local context, we concatenate the lattice parameters, encoded via radial basis functions, to both the local node and edge feature vectors. Finally, we initialize a single global node $\boldsymbol{V}_\text{G}$ using the encoded lattice parameters and create directed global edges pointing from every local node to this global node.

Encoder processing. The GNS message passing maps these inputs to the latent representation $\boldsymbol{Z}$. The local message passing layers, which incorporate distance-smoothed attention [37], learn a smooth, localized representation of atomic structure, denoted $\boldsymbol{Z}_\text{L} \in \mathbb{R}^{N\times D}$. On the other hand, the global message passing steps aggregate system-level context from both local and global nodes into a global latent representation $\boldsymbol{Z}_\text{G} \in \mathbb{R}^D$. Together, these define the continuous latent representation as $\boldsymbol{Z} = (\boldsymbol{Z}_\text{L}, \boldsymbol{Z}_\text{G})$, which we learn to generate using a diffusion model in Section C.7.

C.4 Regularizing the representation space

To ensure the latent space is structured and informative for the diffusion model, we regularize representations with a bottleneck layer before the decoder. We set the latent dimensionality to $D = 4$. We use separate bottleneck layers for the two components: for the local latents $\boldsymbol{Z}_\text{L}$ we use residual vector quantization [18] with the rotation trick [43], whereas for the global

latent vector $\boldsymbol{Z}_{\mathrm{G}}$, we apply a Gaussian KL-divergence bottleneck. The regularized features after the bottleneck are denoted as $\widetilde{\boldsymbol{Z}} = \textsc{Bottleneck}(\boldsymbol{Z})$. Empirically, we observed that the model performance is robust to the specific choice of bottleneck configuration.

C.5 Mapping latent representations to crystal systems

Similar to the encoder, the decoder model $\textsc{Dec}(\cdot)$ uses the GNS architecture to reconstruct a crystal structure $\hat{\boldsymbol{S}}$ from the latent representation $\widetilde{\boldsymbol{Z}}$.

Feature construction. We initialize the input node features using the regularized latents $\widetilde{\boldsymbol{Z}}$ obtained from the bottleneck layer. In contrast to the sparse nearest-neighbor graph used in the encoder, the decoder constructs a fully-connected graph for the local context: local edges are created from each local node to every other node, initialized with the concatenated latent vectors of the connected pair. Additionally, the directed global edges are initialized from the single global node to every local node.

Decoder processing. Decoder message passing uses the edge-attention mechanism [37], omitting the distance-based smoothing used in the encoder, as spatial positions are not yet defined at this stage. The local message passing steps evolve the local features to recover atom-level details, while the global steps broadcast system-level context to the local nodes. Finally, dedicated read-out heads produce the reconstruction parameters: the fractional coordinates $\hat{\boldsymbol{F}}$ and atom type probabilities $\hat{\boldsymbol{P}}_A$ are predicted from local nodes (where the discrete atom type is given by $\hat{\boldsymbol{A}} = \{\arg\max_j \hat{P}_i^j\}_i$), and the lattice parameters $\hat{\boldsymbol{L}}$ are predicted from the global node.

C.6 Training the autoencoder

We train the autoencoder to reconstruct the input crystal system $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$ by minimizing a weighted sum of reconstruction losses and regularization losses. Given the decoder outputs $(\hat{\boldsymbol{P}}_A, \hat{\boldsymbol{F}}, \hat{\boldsymbol{L}}) = \textsc{Dec}(\widetilde{\boldsymbol{Z}})$ for latents $\widetilde{\boldsymbol{Z}} = \textsc{Bottleneck}(\boldsymbol{Z})$ with $\boldsymbol{Z} = \textsc{Enc}(\boldsymbol{S})$, we define the objectives as follows.

Reconstruction losses. We use the average cross-entropy loss to predict the discrete atom types $\hat{\boldsymbol{A}}$:

$$\mathcal{L}_A(\boldsymbol{A}, \hat{\boldsymbol{P}}_A) = \frac{1}{N} \sum_{i=1}^{N} \textsc{CrossEntropy}(a_i, \hat{\mathbf{P}}_i). \tag{1}$$

For fractional positions $\hat{\boldsymbol{F}}$ we use mean squared error (MSE):

$$\mathcal{L}_F(\boldsymbol{F}, \hat{\boldsymbol{F}}) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{f}_i - \hat{\boldsymbol{f}}_i \right\|_2^2. \tag{2}$$

To predict the lattice parameters $\hat{\boldsymbol{L}} = (\hat{a}, \hat{b}, \hat{c}, \hat{\alpha}, \hat{\beta}, \hat{\gamma})$ we also use the MSE after normalization. We normalize the lattice vector lengths by the cube root of the atom count, $N^{1/3}$, following Xie et al. [22, Appendix B.1], and apply a logarithmic transformation, before computing the loss:

$$\mathcal{L}_L(\boldsymbol{L}, \hat{\boldsymbol{L}}) = \frac{1}{3} \left\| \log\left(\frac{(a,b,c)}{N^{1/3}}\right) - \log\left(\frac{(\hat{a},\hat{b},\hat{c})}{N^{1/3}}\right) \right\|_2^2 + \frac{1}{3} \left\| (\alpha,\beta,\gamma) - (\hat{\alpha},\hat{\beta},\hat{\gamma}) \right\|_2^2. \tag{3}$$

Regularization. As discussed, we use different regularization methods for the local and global latent spaces. The local latents $\boldsymbol{Z}_{\mathrm{L}}$ use a residual vector quantization (VQ) bottleneck. We implement this using the `vector_quantize_pytorch` Python package[2], where the codebook $\boldsymbol{C}$ is updated using exponential moving average, and the encoder outputs are regularized to stay close to the codebook vectors via the standard commitment loss [9], ensuring the latent space does not grow during training:

$$\mathcal{L}_{\mathrm{L}}^{\mathrm{Reg}}(\boldsymbol{Z}_{\mathrm{L}}) = \frac{1}{N}\sum_{i=1}^{N} \textsc{Commitment}(\boldsymbol{z}_{\mathrm{L},i}, \boldsymbol{C}). \tag{4}$$

The global latents $\boldsymbol{Z}_{\mathrm{G}}$ are regularized via the KL-divergence:

$$\mathcal{L}_{\mathrm{G}}^{\mathrm{Reg}}(\boldsymbol{Z}_{\mathrm{G}}) = \frac{1}{N} D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{Z}_{\mathrm{G}}^{\mu}, \mathrm{diag}(\boldsymbol{Z}_{\mathrm{G}}^{\sigma})) \parallel \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})), \tag{5}$$

where the global latents correspond to the variational posterior parameters $(\boldsymbol{Z}_{\mathrm{G}}^{\mu}, \boldsymbol{Z}_{\mathrm{G}}^{\sigma}) = \boldsymbol{Z}_{\mathrm{G}}$, and $\mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$ denotes the standard Gaussian prior.

Total objective. The full training objective is the weighted sum of these components:

$$\mathcal{L}_{\mathrm{AE}}(\boldsymbol{S}, \hat{\boldsymbol{S}}, \boldsymbol{Z}) = \begin{pmatrix} \mathcal{L}_{A}(\boldsymbol{A}, \hat{\boldsymbol{A}}) \\ \mathcal{L}_{\mathrm{F}}(\boldsymbol{F}, \hat{\boldsymbol{F}}) \\ \mathcal{L}_{L}(\boldsymbol{L}, \hat{\boldsymbol{L}}) \\ \mathcal{L}_{\mathrm{L}}^{\mathrm{Reg}}(\boldsymbol{Z}_{\mathrm{L}}) \\ \mathcal{L}_{\mathrm{G}}^{\mathrm{Reg}}(\boldsymbol{Z}_{\mathrm{G}}) \end{pmatrix}^{\top} \cdot \begin{pmatrix} 1 \\ 300 \\ 1 \\ 1 \\ 10^{-4} \end{pmatrix}. \tag{6}$$

C.7 Learning to generate latent representations of crystal structures

We use latent diffusion [20] to model the distribution of crystal structure latent representations. Once trained, the model generates samples of the latents $\boldsymbol{Z}$, which are mapped back to the atom domain using the decoder: $\hat{\boldsymbol{S}} = \textsc{Dec}(\textsc{Bottleneck}(\boldsymbol{Z}))$.

We formulate the generative process using a denoising diffusion probabilistic model (DDPM) [13] operating on the continuous latent space $\boldsymbol{Z} \in \mathbb{R}^{(N+1)\times D}$. By modeling crystal structures in this continuous space, we avoid the challenges posed with directly generating mixed categorical and continuous data in the atom domain.

Since the dimensionality of the latent space depends on the system size (comprising N local nodes and 1 global node), the diffusion process is explicitly conditional on the number of atoms N. During inference, N is sampled from the empirical distribution of the training data. The diffusion framework is defined by two processes: a fixed forward process that gradually corrupts the data structure by adding noise, and a learnable reverse process that learns to generate data from pure noise.

Data preparation. The training targets are generated on-the-fly by passing the crystal systems $\boldsymbol{S}$ through the frozen encoder $\textsc{Enc}(\cdot)$. To help the model learn a translationally invariant distribution, we apply the random translation augmentation described in Section C.2 to the input structures before encoding. Moreover, to stabilize diffusion model training, we standardize the latent representations to have a zero mean and unit variance using running statistics. For conciseness, we re-use $\boldsymbol{Z}$ to denote the *standardized* latents in the remainder of this section.

[2] https://github.com/lucidrains/vector-quantize-pytorch/

Forward process. We define the forward process that transforms the clean latent representation $\boldsymbol{Z}_0$ (obtained from the encoder) into a standard Gaussian distribution over a sequence of timesteps $t \in \{0, \ldots, T\}$ (we set $T = 100$k during training, and $T = 4$k during sampling). The noisy sample at timestep t, denoted $\boldsymbol{Z}_t$, is sampled as:

$$\boldsymbol{Z}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{Z}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \qquad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}), \tag{7}$$

where $\boldsymbol{\epsilon}$ has the same dimensionality as the latents. The noise schedule $\bar{\alpha}_t$ follows a cosine-based schedule [13]:

$$\bar{\alpha}_t = \text{sigmoid}\left(\log \text{SNR}(t)\right), \quad \text{with} \quad \log \text{SNR}(t) = -\log \tan^2\left(\frac{t\pi}{2T}\right) + s \tag{8}$$

where we introduce a shift hyperparameter $s = 2$ in the log signal-to-noise ratio (SNR) domain, ensuring that the diffusion model spends more time in the regime dominated by signal than noise (with $s = 2$ this corresponds to 78% of the diffusion timesteps). As $t \to T$, the SNR ratio approaches zero, ensuring that the latents $\boldsymbol{Z}_T$ are indistinguishable from pure Gaussian noise.

Reverse process. The reverse diffusion process generates latents $\hat{\boldsymbol{Z}}_0$ from pure noise $\hat{\boldsymbol{Z}}_T$. Here, a denoising model is trained to reverse the corruption step-by-step. We use the v-prediction ("velocity") parameterization [21]. The denoising model predicts $\boldsymbol{v}_t$, which represents the velocity in the latent space, defined as a linear combination of the clean latents and the noise:

$$\boldsymbol{v}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{Z}_0 \tag{9}$$

This parametrization improves training stability and convergence, particularly because the target remains well-defined at $t = T$ (where SNR = 0), where the standard $\boldsymbol{\epsilon}$-prediction objective is generally unstable due to SNR = 0 [21].

Self-conditioning. We implement self-conditioning [24] to improve training convergence and sample quality. During training, with probability of 0.5, we condition the denoising model on a noisier sample from a later stage in the diffusion process. We sample a time offset $\delta t \sim \mathcal{U}[1, \ldots, \lfloor T * 0.002 \rfloor]$ and define an auxiliary timestep $t' = \min(t + \delta t, T)$. The model is then conditioned as $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{Z}_t, t, \boldsymbol{Z}_{t'}, t')$, where

$$\boldsymbol{Z}_{t'} = \sqrt{\frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\boldsymbol{Z}_t + \sqrt{1 - \frac{\bar{\alpha}_{t'}}{\bar{\alpha}_t}}\boldsymbol{\epsilon}', \quad \text{with} \quad \boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0}, \mathbf{1}). \tag{10}$$

This random sampling of the auxiliary timestep t' is different from the original implementation and allows the model to generalize to varying diffusion schedules and timestep granularity during sampling. Empirically, we found self-conditioning to be essential for model performance. We hypothesize that this necessity arises from the nature of the latent space: although the representations are continuous, they encode inherently discrete structural information. As recently suggested by Pynadath et al. [57], self-conditioning enables learning conditional dependencies, such as, atom bonding pairs, proving critical in this mixed latent space.

Training objective. The training objective for the denoising model $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{Z}_t, t, \boldsymbol{Z}_{t'}, t')$ is the expected MSE:

$$\mathcal{L}_{\text{Diff}} = \mathbb{E}_{t, \boldsymbol{Z}_0, \boldsymbol{\epsilon}, t', \boldsymbol{Z}_{t'}, \boldsymbol{\epsilon}'} \left\| \boldsymbol{v}_t - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{Z}_t, t, \boldsymbol{Z}_{t'}, t') \right\|_2^2, \tag{11}$$

where $t \sim \mathcal{U}[0, \ldots, T]$, $\boldsymbol{Z}_0$ are systems $\boldsymbol{S}$ from the empirical distribution encoded using the fixed encoder, $\boldsymbol{\epsilon}$ is noise from standard Gaussian distribution that has the same shape as $\boldsymbol{z}_0$, and the self-conditioning information t', $\boldsymbol{Z}_{t'}$ and $\boldsymbol{\epsilon}'$ is constructed as described above.

Denoising GNS model. We use the same GNS backbone described in Section C.1 for the denoising model $v_\theta(\boldsymbol{Z}_t, t, \boldsymbol{Z}_)$. We concatenate the $\boldsymbol{Z}$ with a positional embedding of the time index t and pass them into the GNS, which predicts the velocity v_t.

Permutation symmetry breaking for *de novo* generation. The GNS backbone is inherently permutation-equivariant; that is, permuting the input node ordering results in an equivalent permutation of the output features. The standard diffusion objective in Equation (11) regresses these order-dependent model predictions against ordered targets. However, in high noise regimes, the correspondence between the noisy latents and the clean data becomes ambiguous, as multiple node orderings could have generated the same noisy latents. This creates a difficult optimization task since an order-agnostic denoising model cannot effectively match the fixed target ordering. While recent works propose matching-based objectives to ensure a permutation-invariant loss [59], we choose a simpler approach: we explicitly break the symmetry within the denoising model itself. We achieve this by concatenating embeddings of the node order index to the node latent features before passing them to the GNS. Empirically, we observe that this order-conditioning is important for performance in the *de novo* setting. In contrast, it is less significant for conditional generation, where the conditioning signal often provides sufficient structural context to implicitly resolve the symmetry.

To maximize the efficacy of this symmetry breaking, it is important that the node order is consistent across data. We achieve this by pre-processing the data to enforce a canonical ordering derived from the crystal graph's topology. Specifically, we construct the bond graph of the system and determine the node order via the BLISS graph coloring algorithm [3], using atom types as the initial node colors to distinguish chemically distinct atoms. For MOFs, we make a further adjustment: we use the `MOFid` algorithm first to decompose the structure into its constituent building blocks (linkers, solvents, metal nodes and bridges) and apply the graph canonicalization algorithm within each component independently.

During *de novo* sampling, we condition the denoising model on an index sequence $[1, \ldots, N]$. Since this sequence relies only on the system size N and provides no prior information regarding the crystal's chemistry or topology, the generative process is unconditional.

Conditional generation. A key advantage of the direct mapping between the latents $\boldsymbol{Z} \in \mathbb{R}^{(N+1)\times D}$ and atoms of a system is that it facilitates fine-grained conditioning. We can inject atom- and bond-level information by concatenating embeddings of the desired properties, such as atomic types or spatial positions, directly to the corresponding node features. Similarly, connectivity constraints can be incorporated by augmenting the edge vectors with embeddings indicating the presence or absence of chemical bonds.

To enable flexible conditional generation alongside *de novo* tasks, we train Mofasa across diverse conditional tasks. For each training sample, we randomly mask specific subsets of structural information to simulate various generation scenarios. With a probability of 0.25, we fix the positions and atom types of all MOF substructures (nodes, linkers, or solvents) except for one randomly selected component, effectively training the model for structure inpainting tasks (e.g., generating linkers within a fixed node scaffold). Independently, we condition on the system's chemical compositions (atom types) and bond topology with a 0.25 probability each. Whenever composition or bonds are provided, we also supply information about clustering of the fragments that identify distinct molecular fragments and components; otherwise, this clustering information is provided stochastically. This multi-task training objective enables a single model to perform *de novo* generation, conformer generation, and partial structure inpainting. Importantly, when conditioning on structural information, the atom order index is not provided.

Appendix D: Geometry optimization with Orb

We perform geometry optimizations with the sum of two independent MLIPs: `orb-v3-direct-20-omat`, which was trained at the PBE level of theory and a 3-layer model with the same basic architecture, but trained to predict an additive D3 (zero) term. We run both models at `float32-highest` precision. These models are available from the `orb-models` package [37, 58]. Whilst D3 can be computed analytically, existing implementations are inefficient and create a bottleneck.

We use the Fréchet cell FIRE optimizer [2] implemented in `TorchSim` [41] with a 0.05 eV/Å max force convergence criterion and a maximum of 500 iterations. With these settings, 84% of `MofasaDB` and 98% of QMOF converge.

Appendix E: Three-step molecular dynamics pipeline for dynamic stability

To compute the dynamic properties of MOFs in Section 1.5 we used a three-step MD pipeline with similar settings to those reported by Kraß et al. [50] The pipeline was designed to rapidly assess the thermal stability of generated MOF samples at 300 K.

All steps used `orb-v3-con-inf-omat` combined with D3 dispersion corrections[58, 4]. For these 1,000 samples, we used the torch D3 implementation from the `torch-dftd` library. The first step of the pipeline was a coarse geometry optimization using the LBFGS optimizer in ASE (maximum force threshold of 0.05 eV/Å, maximum of 1,000 steps), followed by an NVT equilibration step using Langevin dynamics at 300 K (timestep 1 fs, friction coefficient of 0.01 fs^{-1}, 1 ps duration). Finally, the unit cell was allowed to relax isotropically in the NPT ensemble using MTK dynamics via the `IsotropicMTKNPT` ASE implementation. This final simulation was run for 50 ps at 300 K with a timestep of 1 fs, a thermostat timescale (*tdamp*) of 100 fs, and a barostat timescale (*pdamp*) of 1,000 fs. To quantify stability, the change in volume was calculated as the difference in mean cell volume between the first 10 ps and the final 10 ps of the simulation, while RMSD values were computed based on the deviation of atomic coordinates between the first and final frames.

Appendix F: ORB latent embeddings of MofasaDB samples

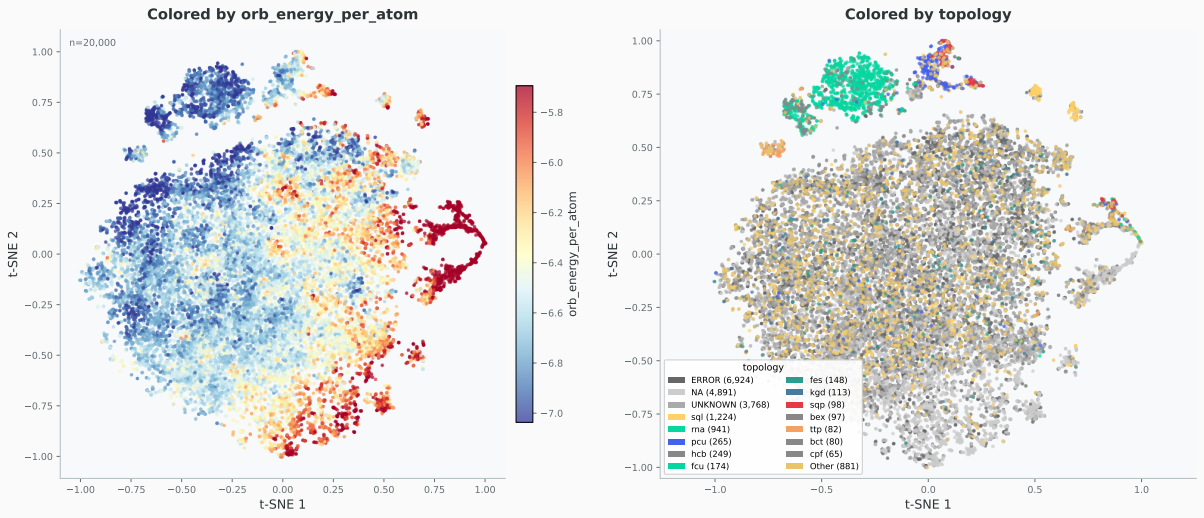

Figure 9: t-SNE plot of `orb-v3-direct-20-omat` latent embeddings (aggregated by mean) of 20,000 MofasaDB samples. Samples are colored according to their inferred *total potential energy per atom* (eV/atom, left) and *topology* (right).

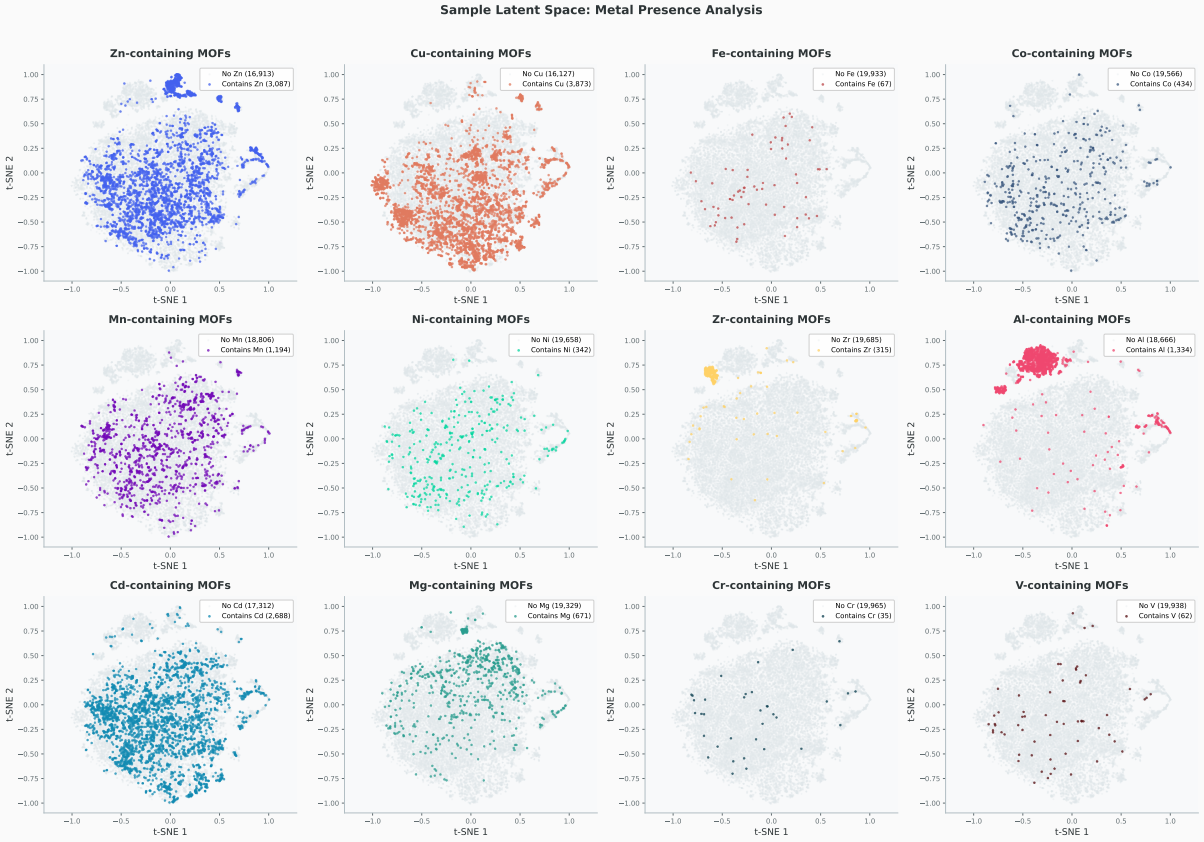

Figure 10: t-SNE plot of `orb-v3-direct-20-omat` latent embeddings (aggregated by mean) of 20,000 MofasaDB samples. Samples are colored according to the presence of specific metals in the metal nodes.

Appendix G: Extended results

Test (%)	QMOF	Mofasa-opt	Mofasa	ADiT-QMOF	ADiT Joint
Has carbon ↑	100.0	100.0	98.4	100.0	100.0
Has hydrogen ↑	99.8	99.9	98.4	99.6	100.0
Has atomic overlap ↓	0.0	0.0	2.8	8.3	10.8
Has overcoord. C ↓	0.0	0.0	2.5	23.6	34.3
Has overcoord. N ↓	0.0	0.0	1.6	1.5	1.6
Has overcoord. H ↓	0.0	0.0	2.4	1.0	3.6
Has undercoord. C ↓	5.6	14.3	20.1	60.0	72.1
Has undercoord. N ↓	6.5	7.3	12.9	39.1	39.9
Has undercoord. rare earth ↓	0.0	0.2	2.1	0.4	0.8
Has metal ↑	100.0	99.2	97.7	100.0	99.4
Has lone molecule ↓	9.7	27.1	31.3	72.9	83.2
Has high charge ↓	1.5	1.6	3.5	0.9	2.5
Has suspicious terminal oxo ↓	0.0	0.5	2.5	2.6	5.8
Has undercoord. alkali ↓	0.1	1.0	3.0	1.0	6.4
Has geom. exposed metal ↓	1.7	3.7	5.3	7.0	9.6
Validity rate (all passed) ↑	80.1	59.8	52.9	15.7	10.2

Table 4: A breakdown of the `MOFChecker` validity criteria for Mofasa and ADiT trained on the QMOF database. We observe that Mofasa-opt displays fewer violations of the "Has carbon/hydrogen/metal" checks. However, this cannot happen since geometry optimization cannot remove or add atoms. This is an artifact of `MOFChecker` failures. Whenever `MOFChecker` fails to compute for a system, all criteria default to "False". As `MOFChecker` failures were less frequent for optimized systems, fewer artificial violations were recorded.

Test (%)	Experimental	Mofasa-opt (Exp.)	Mofasa (Exp.)
Has carbon ↑	99.6	99.5	98.8
Has hydrogen ↑	98.2	97.9	97.2
Has atomic overlap ↓	0.0	0.0	1.7
Has overcoord. C ↓	0.0	0.0	2.4
Has overcoord. N ↓	0.0	0.0	1.0
Has overcoord. H ↓	0.0	0.0	2.1
Has undercoord. C ↓	3.3	17.1	24.7
Has undercoord. N ↓	4.1	6.9	13.5
Has undercoord. rare earth ↓	0.1	0.3	1.5
Has metal ↑	100.0	99.1	98.3
Has lone molecule ↓	6.5	18.3	22.6
Has high charge ↓	1.4	0.6	1.7
Has suspicious terminal oxo ↓	0.8	1.7	3.6
Has undercoord. alkali ↓	0.1	0.9	1.9
Has geom. exposed metal ↓	1.3	3.5	5.8
Validity rate (all passed) ↑	84.2	62.8	52.2

Table 5: A breakdown of the `MOFChecker` validity criteria for Mofasa trained on the Experimental database. We observe that Mofasa-opt displays fewer violations of the "Has carbon/hydrogen/metal" checks. However, this cannot happen since geometry optimization cannot remove or add atoms. This is an artifact of `MOFChecker` failures. Whenever `MOFChecker` fails to compute for a system, all criteria default to "False". As `MOFChecker` failures were less frequent for optimized systems, fewer artificial violations were recorded.

Dataset	N	E	EV	EU	EU-R	ENU	ENU-R	EVU	EVU-R	EVNU	EVNU-R
Mofasa (QMOF)	1k	77.1	42.8	76.5	89.8	75.2	88.3	42.4	60.5	41.5	59.2
Mofasa (QMOF)	10k	75.7	44.0	73.5	86.2	72.5	85.1	43.0	61.3	42.4	60.5
Mofasa (QMOF)	100k	75.8	43.9	68.8	80.7	68.5	80.3	40.2	57.3	40.0	57.0
Mofasa-opt (QMOF)	1k	77.0	51.7	76.4	89.7	74.8	87.8	51.3	73.2	50.6	72.2
Mofasa-opt (QMOF)	10k	75.3	50.2	72.7	85.3	71.6	84.1	48.9	69.7	48.2	68.7
Mofasa-opt (QMOF)	100k	75.7	49.8	68.4	80.3	68.1	79.9	45.7	65.1	45.4	64.8
Mofasa (Exp)	1k	82.3	46.9	77.6	84.2	73.6	79.9	43.1	54.7	40.1	50.9
Mofasa (Exp)	10k	82.7	46.7	71.8	77.9	69.8	75.7	38.4	48.7	36.8	46.8
Mofasa (Exp)	100k	82.9	46.5	67.6	73.3	66.9	72.6	35.5	45.0	34.9	44.4
Mofasa-opt (Exp)	1k	84.6	57.0	78.4	85.1	74.0	80.3	52.0	66.0	48.7	61.8
Mofasa-opt (Exp)	10k	83.5	55.0	72.6	78.7	70.5	76.5	46.3	58.8	44.8	56.8
Mofasa-opt (Exp)	100k	83.4	56.0	67.6	73.3	66.9	72.6	43.7	55.5	43.2	54.8

Table 6: **MOFid: Existence (E), Validity (V), Novelty (N) and Uniqueness (U)** for a range of sample-sizes (N). Importantly, the real data (QMOF or "Experimental") does not have 100% E or V, and so we also report rescaled (-R) percentages, dividing by the maximum possible score (which is a function of both dataset and metric).

Transition	Count	Percentage
ERROR → ERROR	107,967	53.5%
Valid → Valid	38,781	19.2%
UNKNOWN → UNKNOWN	30,441	15.1%
ERROR → UNKNOWN	5,865	2.9%
UNKNOWN → ERROR	5,731	2.8%
ERROR → Valid	3,937	1.9%
Valid → ERROR	3,857	1.9%
UNKNOWN → Valid	2,920	1.4%
Valid → UNKNOWN	2,427	1.2%

Table 7: Breakdown of how topologies of generated samples in MofasaDB change after geometry optimization with `orb-v3-direct-20-omat`. Notably, approximately 6.4% of samples change between Valid to ERROR/UNKNOWN states. This suggests that topology inferred via `MOFid` is sensitive to small geometry perturbations and should be interpreted with caution.

Status	Count	Percentage
Same Topology	33,762	87.1%
Changed Topology	5,019	12.9%
Total	**38,781**	**100.0%**

Table 8: Stability of the inferred MofasaDB topologies (i.e. topologies that were not labeled as `UNKNOWN`, `ERROR`, or `NA`) by `MOFid` pre- and post-geometry optimization. Shows that about 12.9% of inferred topologies changed after geometry optimization.

Rank	Transition			Count
1	sqp	→	pcu	311
2	sql	→	pcu	238
3	bct	→	fcu	184
4	dia	→	sqp	177
5	cpf	→	rna	135
6	hcb	→	sql	135
7	sql	→	hcb	122
8	sdb	→	ttp	85
9	sql	→	sqp	72
10	pcu	→	sql	72
11	hcb	→	dia	60
12	dia	→	pcu	57
13	sql	→	fes	55
14	hcb	→	sqp	54
15	fes	→	sql	51

Table 9: Top 15 topology changes in MofasaDB pre- and post-geometry optimization with `orb-v3-direct-20-omat`.

Appendix H: Glossary of computed properties

Table 10: MofasaDB Property Glossary

Key	Unit	Description
Pore Geometry (Zeo++ [5], default: N_2 probe radius of 1.86 Å)		
`lcd`	Å	Largest Cavity Diameter
`pld`	Å	Pore Limiting Diameter (narrowest channel point)
`dif`	Å	Diameter of Included sphere along Free path
`number_of_channels`	—	Count of distinct connected channel systems
`number_of_pockets`	—	Count of isolated pores (inaccessible to probe)
Volume Properties (Zeo++ [5], default: N_2 probe radius of 1.86 Å)		
`av_volume_fraction`	—	Accessible volume fraction of unit cell
`av_cm3_per_g`	cm^3/g	Accessible pore volume per gram
`nav_volume_fraction`	—	Non-accessible (pocket) volume fraction
`nav_cm3_per_g`	cm^3/g	Non-accessible volume per gram
`channel_volume_fraction`	—	Fraction of void volume in channels
`pocket_volume_fraction`	—	Fraction of void volume in pockets
Surface Area Properties (Zeo++ [5], default: N_2 probe radius of 1.86 Å)		
`asa_m2_per_cm3`	m^2/cm^3	Accessible surface area per unit volume
`asa_m2_per_g`	m^2/g	Accessible surface area per gram (cf. BET)
`nasa_m2_per_cm3`	m^2/cm^3	Non-accessible surface area per unit volume
`nasa_m2_per_g`	m^2/g	Non-accessible surface area per gram
`channel_surface_area_fraction`	—	Fraction of surface area in channels
`pocket_surface_area_fraction`	—	Fraction of surface area in pockets
Crystal Symmetry (Pymatgen [6])		
`spacegroup`	str	Crystal system from space group analysis at symprec=0.01 (e.g., "cubic")
`spacegroup_v2`	str	Crystal system from space group analysis at symprec=0.1 (more tolerant)
`symprec_0.01/pointgroup`	str	Point group symbol (Hermann-Mauguin notation)
`symprec_0.01/spacegroup`	str	Space group symbol (Hermann-Mauguin notation)
`symprec_0.01/spacegroup_number`	int	International Tables space group number (1-230)
`symprec_0.01/spacegroup_crystal`	str	Crystal system name
`symprec_0.1/pointgroup`	str	Point group symbol (at looser tolerance)
`symprec_0.1/spacegroup`	str	Space group symbol (at looser tolerance)
`symprec_0.1/spacegroup_number`	int	Space group number (at looser tolerance)
`symprec_0.1/spacegroup_crystal`	str	Crystal system name (at looser tolerance)

Continued on next page

Table 10 – continued from previous page

Key	Unit	Description
ORB Properties [58]		
`orb_energy_per_atom`	eV/atom	ORB-predicted total potential energy per atom
`orb_max_force`	eV/Å	Maximum atomic force magnitude
`orb_latent_{0-4}_{pool}`	—	GNN latent embeddings (dim=256); `pool` ∈ {`graph`, `nodes_and_bridges`, `linkers`, `bound_solvent`, `free_solvent`}
MOF Fragment Properties		
`{component}_formulas`	List[str]	Chemical formulas per fragment; `component` ∈ {`nodes_and_bridges`, `linkers`, `bound_solvent`, `free_solvent`}
`linkers_smiles`	List[str]	Full SMILES strings for each linker fragment
`linkers_simple_smiles`	List[str]	Simplified scaffold SMILES (no stereochemistry)
Linker Molecular Descriptors [51]		
`linkers_smiles_used`	List[str]	Which SMILES string was successfully parsed for each linker (original, fixed, or simple)
`linkers_smiles_standardized`	List[str]	Neutralized, canonical tautomer SMILES
`linkers_morgan_ecfp{4,6}[_std]`	—	Morgan fingerprints (2048-bit); `_std` = standardized
`linkers_morgan_count_sum`	List[int]	Sum of Morgan fingerprint bit counts (molecular complexity proxy)
`linkers_morgan_count_sum_max`	List[int]	Maximum count in Morgan fingerprint (indicates highly represented substructures)
`linkers_morgan_count_sum_std`	List[int]	Sum of counts for standardized fingerprints
`linkers_morgan_count_sum_max_std`	List[int]	Maximum count for standardized fingerprints
`linkers_rotatable_bonds`	List[int]	Number of rotatable bonds per linker (flexibility metric)
`linkers_ring_count`	List[int]	Number of rings per linker
`linkers_coordination_site_count`	List[int]	Total number of potential metal coordination sites per linker
`linkers_coordination_site_breakdown`	List[Dict]	Breakdown by coordination site type
`linkers_carboxylate_count`	List[int]	Number of carboxylate groups (-COO^-/-COOH)
`linkers_pyridine_count`	List[int]	Number of aromatic nitrogen sites
`linkers_imidazole_n_count`	List[int]	Number of imidazole/triazole NH groups
`linkers_primary_amine_count`	List[int]	Number of primary amine groups (-NH_2)
`linkers_secondary_amine_count`	List[int]	Number of secondary amine groups (-NH-)

Continued on next page

Table 10 – continued from previous page

Key	Unit	Description
linkers_tertiary_amine_count	List[int]	Number of tertiary amine groups (-N<)
linkers_phosphonate_count	List[int]	Number of phosphonate groups
linkers_sulfonate_count	List[int]	Number of sulfonate groups
linkers_phenolic_oh_count	List[int]	Number of phenolic hydroxyl groups
linkers_alcoholic_oh_count	List[int]	Number of alcoholic hydroxyl groups
linkers_thiol_count	List[int]	Number of thiol groups (-SH)
linkers_nitrile_count	List[int]	Number of nitrile groups (-C≡N)
Validation Metrics		
no_atom_too_close	bool	True if all interatomic distances are reasonable
smact_valid	bool	True if SMACT charge-balance check passes
MOFChecker Properties [46]: used for validation		
mofchecker_valid	bool	Overall validity flag (True if passes all checks)
mofchecker_no_carbon	bool	True if structure contains no carbon atoms
mofchecker_no_hydrogen	bool	True if structure contains no hydrogen atoms
mofchecker_no_metal	bool	True if structure contains no metal atoms
mofchecker_has_atomic_overlaps	bool	True if atoms are too close (clashing)
mofchecker_has_lone_molecule	bool	True if structure contains disconnected fragments
mofchecker_has_overcoordinated_c	bool	True if any carbon has too many bonds
mofchecker_has_overcoordinated_n	bool	True if any nitrogen has too many bonds
mofchecker_has_overcoordinated_h	bool	True if any hydrogen has too many bonds
mofchecker_has_undercoordinated_c	bool	True if any carbon has too few bonds
mofchecker_has_undercoordinated_n	bool	True if any nitrogen has too few bonds
mofchecker_has_undercoordinated_rare_earth	bool	True if any rare earth metal is undercoordinated
mofchecker_has_undercoordinated_alkali_alkaline	bool	True if alkali/alkaline earth metal is undercoordinated
mofchecker_has_suspicious_terminal_oxo	bool	True if incorrect terminal oxo groups exist
mofchecker_has_geometrically_exposed_metal	bool	True if metal has unusual coordination geometry
mofchecker_has_high_charges	bool	True if computed partial charges are unusually high
MOFChecker Properties [46]: not used for validation		
mofchecker_has_oms	bool	True if structure has Open Metal Sites

Continued on next page

Table 10 – continued from previous page

Key	Unit	Description
mofchecker_has_3d_connected_graph	bool	True if framework is 3D-connected
mofchecker_graph_hash	str	Hash of the full structure graph (atoms + bonds)
mofchecker_undecorated_graph_hash	str	Hash of graph with hydrogen atoms removed
mofchecker_decorated_scaffold_hash	str	Hash of framework scaffold with decorations
mofchecker_undecorated_scaffold_hash	str	Hash of bare framework scaffold
mofchecker_symmetry_hash	str	Hash encoding symmetry information
MOFid Properties [12]		
mofid	str	Full MOFid identifier string. Format: {nodes}.{linkers} MOFid-v1.{topology}.cat{n}. Value is "UNKNOWN" if MOFid could not be computed.
mofkey	str	MOFKey identifier (a hash-based representation). Format: {hash}.{topology}.MOFkey-v1.{short_code}. Value is "UNKNOWN" if MOFKey could not be computed.
nodes	str	Concatenated SMILES strings of all distinct metal nodes (.-separated). Value is "UNKNOWN" if not available.
linkers	str	Concatenated SMILES strings of all distinct organic linkers (.-separated). Value is "UNKNOWN" if not available.
num_distinct_nodes	int	Number of chemically distinct metal node types in the MOF
num_distinct_linkers	int	Number of chemically distinct organic linker types in the MOF
topology	str	Three-letter RCSR topology code (e.g., "pcu", "dia", "fcu"). Value is "UNKNOWN" if topology could not be determined.
topology_v2	str	Alternative topology assignment (may differ from primary if ambiguous)
catenation	int	Catenation number (degree of interpenetration). 0 = non-catenated, n = n-fold catenated