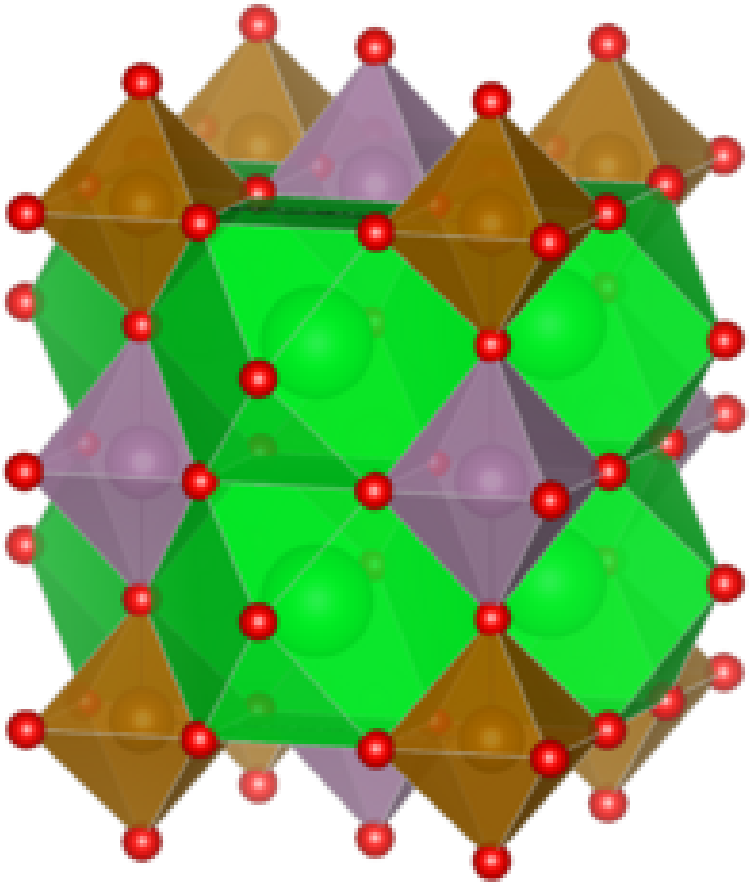
CHALMERS
UNIVERSITY OF TECHNOLOGY



Transformer-Based Crystal Structure Generation from OTC and Chemical Composition

Master's thesis in Physics

ANU PETER

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Transformer-Based Crystal Structure Generation from OTC and Chemical Composition

ANU PETER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Transformer-Based Crystal Structure Generation
from OTC and Chemical Composition
ANU PETER

Supervisor: Henrik Klein Moberg, Department of Physics, Chalmers University of Technology, Gothenburg, Sweden
Examiner: Anders Hellman, Department of Physics, Chalmers University of Technology, Gothenburg, Sweden

Master's Thesis 2026
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Transformer-Based Crystal Structure Generation from OTC and Chemical Composition
Anu Peter
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Multi-component oxides, composed of three or more elements, offer a vast combinatorial space of possible structures with tunable properties such as thermal stability, ion conductivity, and catalytic activity. Exploring this space using traditional trial-and-error methods is time-consuming and expensive.
This thesis investigates the use of a Transformer-based language model to generate Crystallographic Information Files (CIFs), which encode atomic positions, lattice parameters, and symmetry elements. The model is trained to learn relationships between structural features and material properties, allowing it to propose new CIFs representing potential novel crystal structures based on input descriptors like oxygen transfer capacity and composition.
The results show that the Transformer model can capture complex structural patterns and generate valid CIF sequences, demonstrating its potential as a data-driven tool to accelerate the discovery and design of multi-component oxides.

Acknowledgements

I would like to sincerely thank Anders Hellman, Professor of Chemical Physics, Physics, my supervisor, and examiner, for his invaluable guidance, continuous support, and encouragement throughout this project. His knowledge and feedback were crucial for my progress and greatly improved the quality of my work.
I am also deeply grateful to Henrik Klein Moberg for his guidance with the code and model architecture. His experience in AI and technical support was essential during the implementation phase.
Finally, I thank Rocío Mercado, Assistant Professor of Computer Science and Engineering at Chalmers University, for her guidance and for providing me with the opportunity to carry out this project.

Anu Peter
Gothenburg, January 2026

List of Acronyms

Below is the list of acronyms used throughout this thesis, presented in alphabetical order:

AI	Artificial Intelligence
CIF	Crystallographic Information File
CLC	Chemical looping combustion
CSD	Cambridge Structural Database
CSP	Crystal structure prediction
DFT	Density Functional Theory
DeCIFer	Decoder for Crystallographic Information Files
GPU	Graphics Processing Unit
HEO	High entropy oxides
MAE	Mean Absolute Error
MSE	Mean Squared Error
LM	Language Model
ML	Machine Learning
NLP	Natural Language Processing
OC	Oxygen Carrier
OTC	Oxygen Transfer Capability
PXRD	Powder X-ray diffraction
RMSE	Root Mean Squared Error

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Crystal structures determine how materials behave, and the arrangement of atoms inside a structure influences mechanical strength, electronic properties, thermal stability, and chemical reactivity. To describe these arrangements, researchers use the Crystallographic Information File (CIF) format, which stores cell parameters, symmetry elements, and atomic positions in a structured and consistent way. CIFs are widely used in crystallography databases and simulation tools.
As the demand for new functional materials grows, the ability to generate crystal structures becomes increasingly valuable. Traditional discovery is slow because the search space is enormous, and both experimental and computational screening are resource-intensive [1, 2] . Machine learning offers a new direction by learning structural patterns directly from large datasets of known materials and using them to propose new candidates.
Transformer-based language models have become powerful tools for sequence generation. Although originally developed for natural language, they are effective at learning long-range patterns in any structured sequence. Since CIFs can be viewed as sequences of tokens representing atoms, symmetries, and coordinates, Transformers are a promising choice for learning how to "write" new crystal structures. This thesis explores how a Transformer model can understand information stored in CIFs and generate new structures based on high-level material descriptors, such as oxygen transfer capacity (OTC) and chemical composition.

1.2 Problem Description

Even with thousands of known structures available, we still lack tools that can create new crystal structures directly from material properties. Existing methods often depend on human intuition or computationally heavy simulations. This thesis investigates a different approach: using a Transformer model to generate a complete CIF file from OTC, composition, and energy values.
The idea is to train a model that not only understands the formatting of a CIF but also learns how structural features relate to material properties. If successful, the model could propose crystal structures that match a desired OTC. However, generating CIFs is challenging. CIF sequences vary in length, contain many rare tokens, and require accurate predictions of atomic positions. Precision errors or sequence drift can easily break structural validity. GPU memory constraints further

limit the model size and batch size during training.
This project explores whether it is possible to overcome these challenges and move closer to automated property-driven structure generation.

1.3 Research Questions

To understand the potential and limitations of conditional CIF generation, this thesis focuses on three main questions:

1. Can a Transformer generate a valid CIF structure from OTC and composition?
2. Where does the model fail within the CIF sequence, and why?
3. How do the model's architecture and hyperparameters—such as sequence length, tokenization method, and GPU memory limits—affect the accuracy and validity of the generated CIFs?

These questions guide the evaluation of the model and help reveal what aspects are most important for improving CIF generation.

1.4 Scope and Limitations

To keep the project focused, several limitations apply:

- The dataset contains roughly 4632 CIFs, which is relatively small for training deep Transformer models.
- Only a specific group of compositions is included, limiting generalization to broader material families.
- The study focuses solely on Transformer architectures; diffusion models, graph networks, or symmetry-aware generative methods are not explored.
- Conditioning information is limited to OTC, composition, and energy. Other structural or thermodynamic properties are not considered.

These constraints define the boundaries of the study and help maintain a clear focus on evaluating Transformer models for conditional CIF generation.

2

Theory

This chapter presents the fundamental concepts necessary for understanding and interpreting this study.

2.1 Crystal Structures and CIF Files

The arrangement of atoms in a crystal, known as the crystal structure, fundamentally determines the physical and chemical properties of a material. Thermal stability, ion conductivity, and catalytic activity are all influenced by how atoms are positioned within the lattice. Understanding and representing these structures in a standardized format is crucial for experimental characterization, computational modeling, and data-driven material design [3].

Crystallographic Information Files (CIFs) provide a standardized format to describe crystal structures. Each CIF encodes three main components[4][5]:

1. Unit cell parameters: The lengths a, b, c and angles α, β, γ define the repeating unit of the crystal.

2. Symmetry information: The space group specifies the symmetry operations present in the structure.

3. Atomic positions: Fractional coordinates indicate the location of each atom within the unit cell.

A widely studied example is magnetite (Fe_3O_4), a cubic oxide with a spinel structure. Table 2.1 shows a simplified CIF snippet for magnetite, highlighting the key features of cell parameters, symmetry, and atomic positions[6].

Keyword / Column	Description	Example
_cell_length_a	Unit cell length along x	8.396 Å
_cell_length_b	Unit cell length along y	8.396 Å
_cell_length_c	Unit cell length along z	8.396 Å
_cell_angle_alpha	Angle	90°
_cell_angle_beta	Angle	90°
_cell_angle_gamma	Angle	90°
_symmetry_space_group_name_H-M	Space group	Fd-3m
_atom_site$_{l}abel$	Atom label	Fe1
_atom_site$_{t}ype_{s}ymbol$	Atom type	Fe
_atom_site$_{f}ract_{x}$	Fractional x coordinate	0.0000
_atom_site$_{f}ract_{y}$	Fractional y coordinate	0.0000
_atom_site$_{f}ract_{z}$	Fractional z coordinate	0.0000
_atom_site$_{l}abel$	Atom label	Fe2
_atom_site$_{t}ype_{s}ymbol$	Atom type	Fe
_atom_site$_{f}ract_{x}$	Fractional x coordinate	0.6250
_atom_site$_{f}ract_{y}$	Fractional y coordinate	0.6250
_atom_site$_{f}ract_{z}$	Fractional z coordinate	0.6250
_atom_site$_{l}abel$	Atom label	O1
_atom_site$_{t}ype_{s}ymbol$	Atom type	O
_atom_site$_{f}ract_{x}$	Fractional x coordinate	0.2600
_atom_site$_{f}ract_{y}$	Fractional y coordinate	0.2600
_atom_site$_{f}ract_{z}$	Fractional z coordinate	0.2600

Table 2.1: CIF snippet for magnetite (Fe_3O_4)

To complement the CIF data, Figure 2.1 shows a VESTA visualization of magnetite (Fe_3O_4). The 3D unit cell representation displays the positions of Fe and O atoms, illustrating how CIF information translates into actual atomic positions in the crystal lattice [7].

2.2 Chemical Looping and Oxygen Carriers

Chemical looping combustion (CLC) is an innovative combustion technology in which a solid oxygen carrier (OC) transfers oxygen from an air reactor to a fuel reactor. This process allows fuel oxidation to occur without direct contact between the fuel and atmospheric air, effectively reducing nitrogen oxide (NOx) emissions and improving overall combustion efficiency [8, 9].
An oxygen carrier is a solid material capable of reversibly incorporating and releasing oxygen during the CLC cycle[2]. In the air reactor, the OC is oxidized by oxygen from the air, while in the fuel reactor, it is reduced, transferring oxygen to the fuel to facilitate combustion. A key performance metric of an oxygen carrier is its oxygen transfer capacity (OTC), which quantifies the amount of oxygen that can be delivered per unit mass of the material. Higher OTC values indicate more efficient oxygen transfer, leading to more complete fuel oxidation and fewer cycles needed for combustion [10, 11],

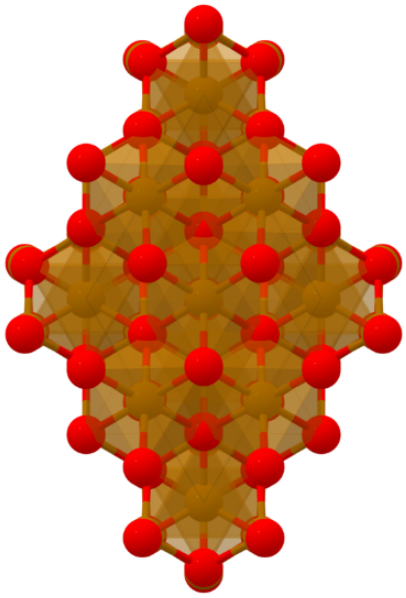


Figure 2.1: VESTA visualization of magnetite (Fe_3O_4). Fe atoms are shown in brown and O atoms in red.

High-entropy oxides (HEOs), which are multi-component oxides consisting of three or more elements, have shown significant promise as oxygen carriers. Their compositional complexity increases structural stability at high temperatures and enables high oxygen transfer capacity, making them ideal candidates for sustainable CLC processes. [12, 13]

A schematic representation of the CLC process is shown in Figure 2.2, highlighting:

- The flow of oxygen from the air reactor to the fuel reactor.
- The cyclic oxidation and reduction of the oxygen carrier.
- The concept of oxygen transfer capacity (OTC) along the cycle.

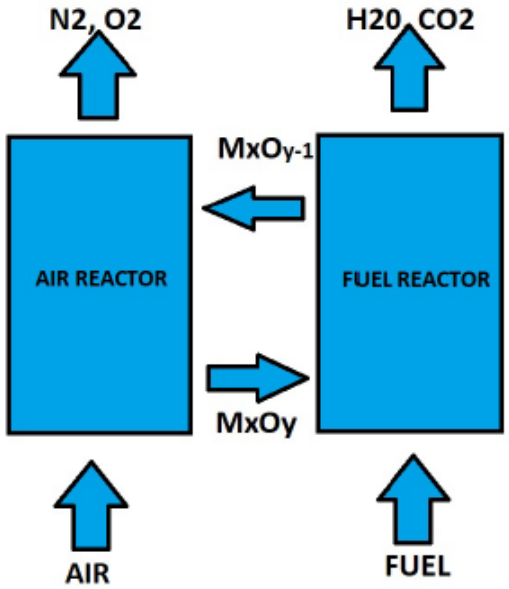


Figure 2.2: Schematic of chemical looping combustion. The oxygen carrier alternates between oxidized and reduced states, transferring oxygen from the air reactor to the fuel reactor [1].

In this study, Transformer-based language models are employed to generate Crystallographic Information Files (CIFs) for multi-component oxides. By learning the relationships between atomic arrangements and material properties such as OTC, the model can propose novel oxygen carrier candidates efficiently. This approach offers a data-driven route to accelerate the discovery and design of high-performance materials for chemical looping applications [2, 14].

2.3 Transformer Model: Architecture, Mechanisms, and Relation to CIF Generation

2.3.1 Introduction

The Transformer architecture marks a significant evolution in sequence modeling, replacing recurrent and convolutional mechanisms with a unified attention-based framework. First introduced by Vaswani et al. (2017) in *Attention Is All You Need*, the Transformer demonstrates that sequential dependencies can be captured solely through self-attention, enabling efficient parallelization and improved learning of long-range interactions. Its scalability, stability, and expressiveness have since established it as the foundational model for modern generative systems.[15]
In this thesis, a Transformer is adopted to generate **Crystallographic Information File (CIF)** sequences in an autoregressive manner. Since a CIF file contains structured crystallographic information—such as lattice parameters, symmetry operations, atom labels, and fractional coordinates—representing it as a token sequence

allows the entire generation task to be formulated as a *conditional next-token prediction problem*, analogous to natural-language modeling.

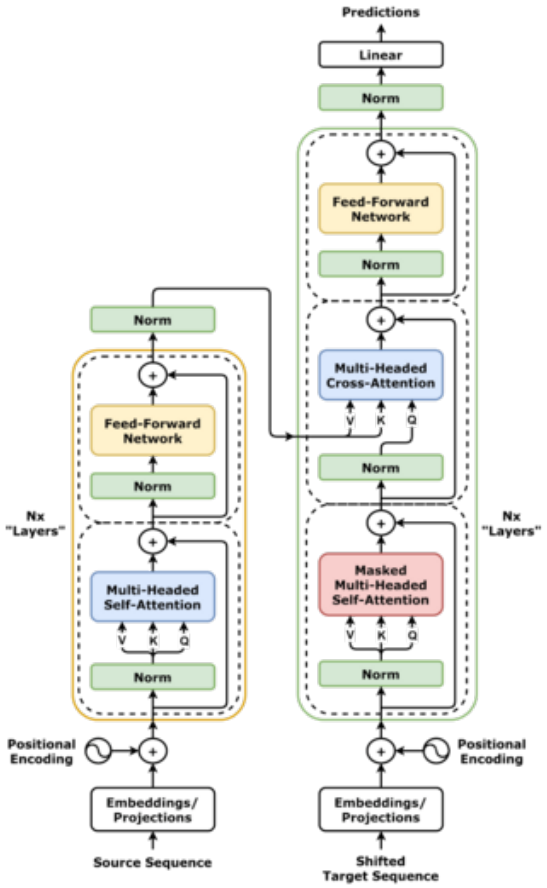


Figure 2.3: Transformer architecture with encoder and decoder. The encoder maps an input sequence into a continuous representation using stacked layers of multi-head self-attention and feed-forward networks with residual connections and layer normalization. Positional encodings preserve the order of tokens. The decoder generates the output sequence autoregressively, combining masked self-attention over previous outputs with cross-attention to the encoder representations, enabling the model to learn complex dependencies between input and output sequences.

2.3.2 Sequence Modeling and the Next-Token Prediction Task

Before introducing the Transformer architecture, it is important to outline the task it is designed to solve: predicting the next token in a sequence.
Sequence generation models learn statistical patterns in token order and structure. Given a sequence of tokens

$$(x_1, x_2, \ldots, x_t),$$

the model predicts the probability distribution of the next token x_{t+1}:

$$P(x_{t+1} \mid x_{\leq t}).$$

This is known as *autoregressive modeling*, where each new token depends on all previously generated tokens[16].
Consider the natural-language sequence:

"The cat sat on the . . . "

From its training data, a model may infer that the word *mat* commonly completes this phrase. It then outputs a probability distribution over potential next tokens:

Candidate Token	**Probability**
mat	0.82
floor	0.07
bed	0.04
chair	0.03

Table 2.2: Illustrative next-token prediction probabilities.

The model selects the token with the highest probability to extend the sequence:

"The cat sat on the mat"

The prediction process then continues iteratively until a designated end-of-sequence marker is reached.
In this thesis, the same autoregressive generation mechanism is applied to CIF sequences. Instead of words, the tokens represent crystallographic elements—unit-cell parameters, symmetry labels, atom identifiers, and fractional coordinates—allowing the Transformer to construct a CIF file token-by-token.

2.3.3 Tokenization of Input Sequences

Transformers operate on discrete *tokens*, not raw text. In natural language, tokens are typically words or subwords. In crystallography, tokens correspond to CIF keywords, numbers, atomic symbols, and structural markers.
Each token is assigned an integer ID and mapped to a vector embedding of dimension d_{model}:

$$x_t \xrightarrow{\text{token ID}} \text{ID}_t \xrightarrow{\text{embedding}} \mathbf{e}_t \in \mathbb{R}^{d_{\text{model}}}.$$

CIF snippet	Token ID	Embedding vector ($\mathbf{e}_t$)
_cell_length_a	1	[0.12, -0.34, 0.56, . . .]
5.430	2	[-0.21, 0.88, 0.15, . . .]
_cell_length_b	3	[0.09, -0.44, 0.72, . . .]
5.430	2	[-0.21, 0.88, 0.15, . . .]
loop_	4	[0.05, -0.12, 0.44, . . .]
Si	5	[0.23, -0.65, 0.77, . . .]
0.000	6	[-0.11, 0.49, 0.21, . . .]

Table 2.3: Example of CIF tokenization and embedding mapping

The token IDs represent unique discrete identifiers for each type of CIF component, while the embedding vectors are learned during training and capture semantic and

structural relationships between tokens. Repeated values such as ‘5.430‘ share the same token ID to reduce the vocabulary size. Special tokens such as ‘loop_‘ enable the model to detect block structures and repeated patterns in the CIF[17].

2.3.4 Embedding and Positional Encoding

After tokenization, each CIF token is first assigned a unique integer ID. However, the Transformer model cannot work directly with discrete IDs and requires continuous vector representations. This is achieved through *embeddings*, which convert tokens into dense vectors that capture semantic and structural information.

2.3.5 Token Embedding

Each token x_t is mapped to a d_{model}-dimensional vector using a learned embedding matrix $E \in \mathbb{R}^{|V| \times d_{\text{model}}}$, where $|V|$ is the size of the token vocabulary:

$$\mathbf{e}_t = E(x_t), \quad t = 1, 2, \ldots, T.$$

These embeddings allow the model to understand relationships between tokens[18]. For example, tokens representing similar fractional coordinates or atomic species that often appear in the same crystallographic context will have similar vector representations.

2.3.6 Positional Encoding

Since Transformers process all tokens in parallel and do not inherently know their order, positional information must be added explicitly. This is done through sinusoidal positional encodings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$

where *pos* indicates the token's position in the sequence and i indexes the dimension within the embedding vector. The final input for the Transformer is obtained by summing the token embedding and its positional encoding:

$$\mathbf{z}_t^{(0)} = \mathbf{e}_t + PE_t.$$

This combination provides the model with both the identity of the token and its position in the sequence, which is essential for learning structural and sequential patterns in CIF files, such as the order of keywords, symmetry blocks, and atomic coordinates[18].

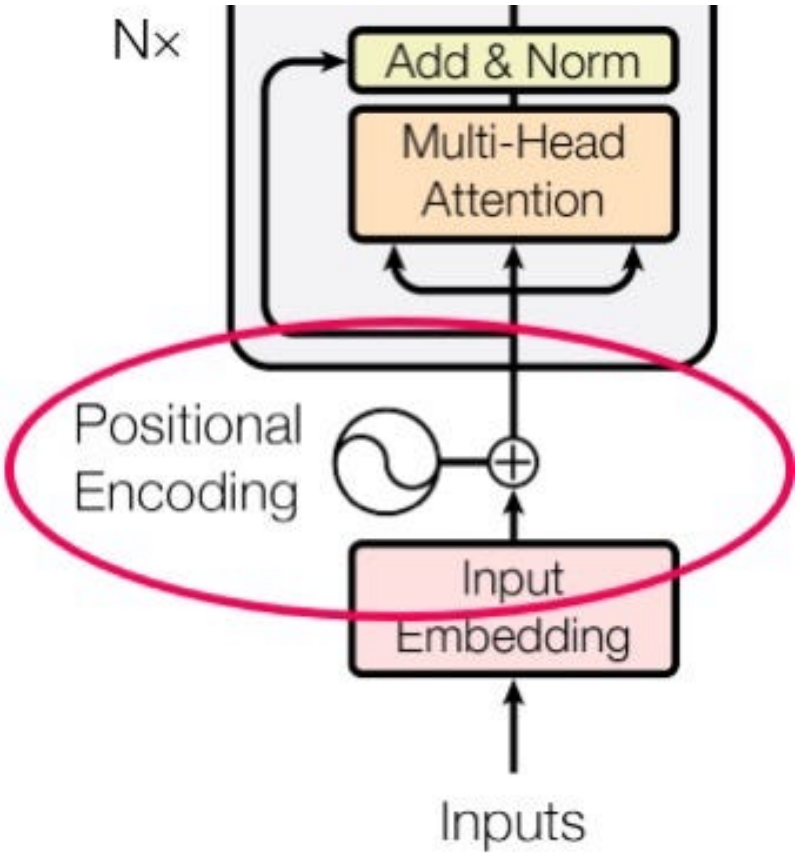


Figure 2.4: Workflow of input preparation for the Transformer: CIF tokens are mapped to IDs, converted into embeddings, augmented with positional encodings, and then fed into the Transformer.

2.3.7 Attention Mechanism

The core of the Transformer is the attention mechanism, which allows the model to capture relationships between all tokens in a sequence simultaneously. For CIF generation, attention enables the model to understand dependencies such as how unit cell parameters influence each other, how symmetry operations relate to atomic positions, and how element types correspond to fractional coordinates.
For each token, the model computes three vectors: queries (Q), keys (K), and values (V):

$$Q = XW^Q, \qquad K = XW^K, \qquad V = XW^V,$$

where X is the input representation (embedding + positional encoding), and W^Q, W^K, W^V are learned projection matrices. The attention between tokens is calculated via the scaled dot-product:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where d_k is the dimension of the key vectors. This computation assigns higher

weights to tokens that are more relevant to the current token, allowing the model to focus on the most important relationships in the sequence.

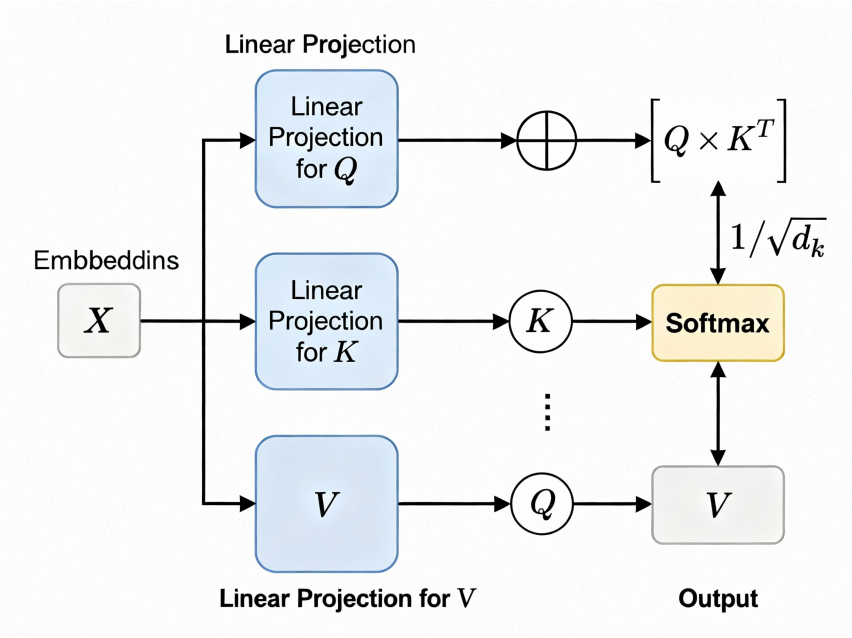


Figure 2.5: Each token embedding x_i is projected into a query (Q), key (K), and value (V) vector. Similarity scores are computed via the scaled dot-product $QK^T/\sqrt{d_k}$, and a softmax converts these scores into attention weights. These weights determine how strongly each token attends to others, and the final output is computed as a weighted sum of the value vectors.

2.3.8 Multi-Head Attention

A single attention head captures one type of relationship between tokens, but CIF sequences contain multiple simultaneous dependencies, including connections between lattice parameters, atomic positions, element types, and symmetry operations. Multi-head attention addresses this by computing several attention operations in parallel.
For each head i, the input embeddings with positional encodings are linearly projected into queries, keys, and values, and attention is computed independently:

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V),$$

where W_i^Q, W_i^K, W_i^V are learned matrices for the i-th head. The outputs of all heads are concatenated and projected to form the final representation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O.$$

This allows each head to specialize in capturing different relationships, for example, one may focus on element-to-coordinate dependencies, another on lattice angles, and a third on block ordering. Multi-head attention thus enables the model to better capture the complex patterns inherent in CIF sequences.

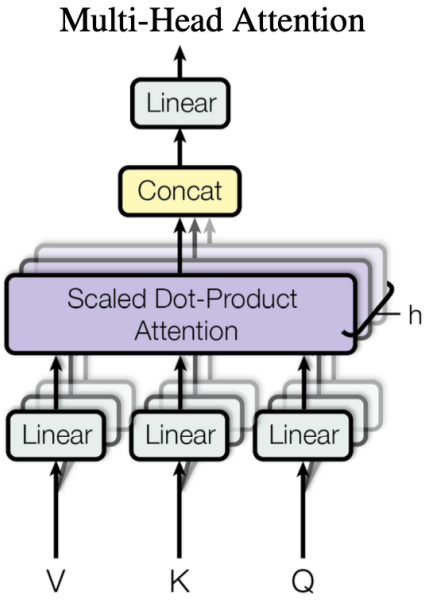


Figure 2.6: Illustration of multi-head attention. Input embeddings are projected into multiple queries, keys, and values, forming independent attention heads. Each head captures different relationships, and their outputs are concatenated and linearly projected to produce the final representation.

For structured formats like CIFs, these mechanisms are complementary. Self-attention enables the model to learn relationships between symmetry information, lattice parameters, and atomic positions, while positional encoding ensures the correct sequential ordering of fields. Multi-head attention allows simultaneous modeling of multiple structural dependencies. Together, these components allow the Transformer to generate coherent, valid CIF files in an autoregressive manner.

2.3.9 Feed-Forward Layers, Residual Connections, and Layer Normalization

After capturing relationships between tokens through the multi-head attention mechanism, each Transformer layer includes a position-wise fully connected feed-forward network (FFN)[18, 19] . This layer introduces nonlinearity and enhances the model's ability to process complex interactions. Unlike attention, which mixes information across tokens, the FFN operates independently on each token, transforming its context-aware representation into a higher-level feature space. The FFN is defined as:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2,$$

where W_1 and W_2 are learned weight matrices, b_1 and b_2 are bias vectors, and the ReLU activation introduces nonlinearity. In the context of CIF generation, this layer allows the model to refine information captured from attention, such as

combining dependencies between unit cell parameters, symmetry operations, and atomic coordinates into coherent representations.
To further improve training stability and facilitate learning in deep architectures, each sub-layer—including multi-head attention and the feed-forward network—is equipped with a residual connection followed by layer normalization[20]. The residual connection adds the original input of the sub-layer to its output, which helps gradients flow during backpropagation and prevents vanishing or exploding gradient issues[15]. Mathematically, for a sub-layer function Sublayer$(\cdot)$, the output with residual connection and layer normalization is:

$$\mathbf{z}'_t = \text{LayerNorm}(\mathbf{z}_t + \text{Sublayer}(\mathbf{z}_t)),$$

where $\mathbf{z}_t$ is the input token representation and $\mathbf{z}'_t$ is the normalized output.
For CIF generation, this mechanism ensures that each token retains its original information while also incorporating complex patterns learned from the attention and feed-forward layers. For example, a token representing `_atom_site_fract_x` maintains its initial embedding but also integrates contextual information from lattice parameters, symmetry operations, and other atomic positions.

2.3.10 Autoregressive Generation and Masked Attention

After processing token representations through multi-head attention and the feed-forward network, the Transformer decoder generates sequences autoregressively. During training, given a sequence $(x_1, \ldots, x_t)$, the model learns to predict the next token x_{t+1} based on all previous tokens[21] . To ensure the model does not attend to future positions, a causal mask is applied:

$$\text{Mask}(i, j) = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i. \end{cases}$$

This mask is added to the QK^T matrix before the softmax computation, preventing information leakage.

2.3.11 Next-Token Prediction Objective

For a vocabulary V, the model outputs logits $\ell_t \in \mathbb{R}^{|V|}$ for each position. Applying a softmax converts these into probabilities:

$$P(x_{t+1} \mid x_{\leq t}) = \text{softmax}(\ell_t).$$

The training objective is to minimize the negative log-likelihood:

$$L = -\sum_{t=1}^{T} \log P(x_{t+1} \mid x_{\leq t}).$$

2.3.12 Output Layer and Token Prediction

After processing through multiple Transformer layers—including multi-head attention, feed-forward networks, residual connections, and layer normalization—each token obtains a final context-aware representation. This representation encodes both the identity of the token and its relationships with all other tokens in the sequence. The output layer then maps these continuous vectors to a probability distribution over the vocabulary, allowing the model to predict the next token in an autoregressive manner. This is performed using a linear projection followed by a softmax function:

$$P(x_{t+1} \mid x_1, \ldots, x_t) = \text{softmax}(\mathbf{z}_t^{(L)} W^O + b^O),$$

where $\mathbf{z}_t^{(L)}$ is the representation of token t from the final Transformer layer, and W^O and b^O are the learned projection weights and biases.
In the context of CIF generation, this mechanism allows the model to sequentially predict crystallographic tokens, including keywords, numerical values, element symbols, and atomic coordinates, by leveraging the context provided by previously generated tokens.
In summary, the output layer converts the final context-aware token embeddings into vocabulary probabilities, iteratively selects the next token, and continues this process until a complete CIF file is generated. This ensures that the generated sequence is both structurally coherent and compliant with crystallographic conventions.

2.3.13 Training Objective: Cross-Entropy Loss

The Transformer is trained to generate a CIF file one token at a time. At each step, the model looks at all previous tokens and tries to predict the next one. This training setup is called *autoregressive learning*. For a sequence of tokens $(x_1, \ldots, x_T)$, the overall goal is to assign high probability to the correct next token. This idea can be written as

$$\mathcal{L} = \sum_{t=1}^{T-1} \log P(x_{t+1} \mid x_1, \ldots, x_t).$$

In practice, the model minimizes the *cross-entropy loss*, which is the negative log-likelihood of the correct next token:

$$\text{CE} = -\sum_{t=1}^{T-1} \log P(x_{t+1} = \hat{x}_{t+1}),$$

where $\hat{x}_{t+1}$ denotes the true next token from the training data.
Cross-entropy measures how different the predicted probability distribution is from the true token. A lower cross-entropy value means that the model predicts the next token more accurately. Overall, cross-entropy is a simple, stable, and effective objective for learning the generative structure of CIF sequences [19].

2.3.14 Autoregressive Generation of CIF Files

Once the model is trained, it generates CIF files one token at a time in an autoregressive manner. At each step t, the model looks at all the tokens generated so far

$(x_1, \ldots, x_t)$ and predicts a probability distribution over the next token in the vocabulary. Instead of always choosing the most likely token, the next token is randomly sampled from this distribution using stochastic sampling. This introduces variation in the generated sequences, allowing the model to produce multiple valid CIFs from the same starting point.[22, 23]
The generation process continues step by step until a special end-of-file token is reached. By predicting tokens in sequence, the model preserves the correct order of crystallographic information, such as listing cell parameters before symmetry operations and atomic coordinates after the atom labels. This approach ensures that the generated CIFs are complete and consistent, maintaining the correct relationships between lattice parameters, element types, symmetry operators, and atomic positions[3].

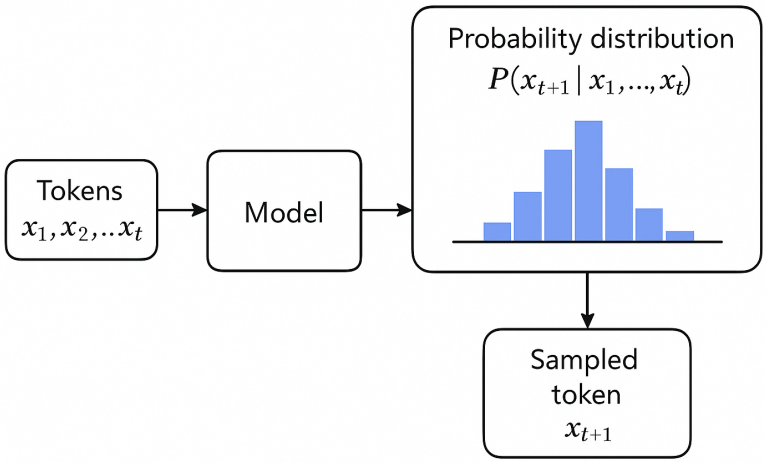


Figure 2.7: This diagram illustrates how the model generates CIF files autoregressively. At each step, the sequence of previously generated tokens $(x_1, \ldots, x_t)$ is fed into the Transformer model, which outputs a probability distribution over the vocabulary. The next token x_{t+1} is sampled from this distribution, allowing for diverse and valid crystallographic sequences. The process repeats until the end-of-file token is produced.

2.3.15 Summary of the CIF Generation Workflow

The complete CIF generation process integrates tokenization, embedding, positional encoding, multi-head attention, feed-forward transformation, residual normalization, output projection, and autoregressive decoding. Numerical and symbolic CIF information is converted into tokens, transformed into continuous representations, and processed across multiple Transformer layers to capture both local and global crystallographic patterns. The output layer predicts the next token in the sequence, and the model iteratively constructs a full CIF file using a chosen sampling strategy. This workflow enables a data-driven, end-to-end approach for generating crystallographic structures and forms the foundation of the CIFFormer model developed in

this thesis.

2.3.16 Model Architecture and Hyperparameters

The CIF generation model is based on a Transformer architecture with stacked layers that process sequences of atomic positions and features. Each layer consists of multi-head self-attention and feed-forward networks, which allow the model to capture both local and long-range dependencies within the crystal structures.
A separate pretrained encoder is used to provide additional input features for the CIF sequences. This encoder is frozen, meaning its weights are not updated during training, and its outputs are precomputed to reduce computational overhead. The pretrained encoder is therefore treated as a fixed feature extractor rather than a trainable component of the main model.
The key hyperparameters of the Transformer model are summarized using common notation:

- **Number of layers (L):** 8 layers in the Transformer used for CIF generation.
- **Number of attention heads (H):** 8 heads per layer, allowing the model to focus on multiple aspects of the sequence simultaneously.
- **Hidden embedding dimension ($d_{\mathbf{model}}$):** 512, determining the size of internal representations for each token.
- **Dropout rate ($p_{\mathbf{drop}}$):** 0.1, applied to attention and feed-forward layers to prevent overfitting.
- **Batch size (B):** 16 sequences per training step, balancing GPU memory usage and training stability.
- **Maximum sequence length ($n_{\mathbf{seq}}$):** 1750 tokens, accommodating CIF sequences with many atoms or symmetry operations.
- **Learning rate (η):** 0.001, with a decay schedule down to 0.00001 for stable convergence.
- **Maximum iterations ($N_{\mathbf{iter}}$):** 50,000 training steps.
- **Warmup iterations ($N_{\mathbf{warm}}$):** 100, gradually increasing the learning rate at the start of training.
- **Gradient clipping ($g_{\mathbf{clip}}$):** 1.0, to prevent instability from large gradients.
- **Conditional input size ($d_{\mathbf{cond}}$):** 2 features used to guide CIF generation.

These hyperparameters were selected to balance model performance, training stability, and the computational constraints of the available GPU resources[24, 19] .

2.3.17 Vocabulary and Tokenization Challenges in CIF Data

Tokenizing CIFs files is more complicated than tokenizing normal text because CIFs contain numbers, symmetry rules, and nested crystallographic information. The vocabulary needs to cover both textual elements, like `_symmetry_space_group` or element symbols, and numerical values for lattice constants, angles, and atomic coordinates. Since floating-point numbers cannot be directly represented as tokens, they are usually split into smaller sub-tokens or discretized, which makes sequences longer and the model more complex.

Additionally, CIF files follow a strict syntax: certain sections must appear in a specific order, and numerical values must be precise[6] . If tokenization is done poorly, it can cause information loss, inconsistent number representation, or even invalid crystal structures during generation. Because of these challenges, designing a careful tokenization strategy is essential. A good vocabulary helps the Transformer understand both the structure (grammar) and the numbers in CIF files, enabling it to generate valid and accurate crystal structures[4, 25].

2.4 Computational Complexity of Transformer Models

The Transformer architecture is highly effective but comes with significant computational and memory requirements, mainly due to the self-attention mechanism. Self-attention examines relationships between all pairs of tokens in a sequence. For a sequence of length n and hidden dimension d, this requires roughly $O(n^2 d)$ operations and $O(n^2)$ memory to store the attention weights. Here, the notation $O(n^2)$ indicates that the computational cost and memory usage grow approximately with the square of the sequence length. In other words, if the sequence length doubles, the number of computations and memory needed roughly quadruples. In CIF sequences, which can be long due to a large number of atoms or symmetry operations, this can become a major computational bottleneck during both training and inference.
Other factors, such as batch size and the number of layers, further increase memory usage, potentially limiting the sequence lengths or batch sizes that can be used effectively. These constraints may affect the model's expressiveness and the speed of convergence during training. Nevertheless, modern GPUs and careful management of computational resources make it possible to train Transformer-based models for CIF generation within practical limits.[18]

2.5 Overview of deCIFer

deCIFer is an autoregressive transformer model designed for crystal structure prediction (CSP) from powder X-ray diffraction (PXRD) data. The model generates Crystallographic Information Files (CIFs), which encode the atomic structure of a crystal, by conditioning on PXRD patterns. A small neural network first embeds the PXRD data into a learnable vector, which is prepended to the sequence of CIF tokens and used to guide the transformer decoder during generation. This allows deCIFer to produce crystal structures that are consistent with experimental diffraction measurements [26]
To efficiently handle CIF sequences of variable length, deCIFer uses a *sequence-packing strategy* along with *attention masking*, ensuring that each structure is processed independently while maintaining internal context. During training, PXRD patterns are augmented with simulated noise and peak broadening to mimic real experimental conditions, enhancing the model's robustness. Generated structures are evaluated based on agreement with reference PXRD patterns and structural

validity.
Figure 2.8 illustrates the overall workflow of deCIFer. PXRD data is first embedded and combined with tokenized CIF sequences, which are then fed into the transformer decoder. The autoregressive process predicts the CIF tokens sequentially, resulting in a valid crystal structure aligned with the input PXRD pattern. By integrating experimental data directly into the generative process, deCIFer bridges computational CSP and experimental diffraction analysis, providing a powerful tool for materials characterization and discovery.

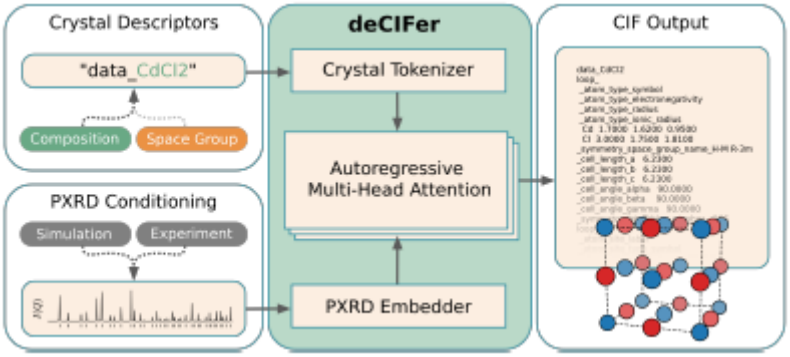


Figure 2.8: Schematic overview of deCIFer. The PXRD pattern is embedded and prepended to the tokenized CIF sequence, which is then processed by the transformer decoder to generate the CIF autoregressively.

3
Methods

This chapter outlines the methodology followed to generate CIF files using a Transformer model. It covers the preparation of data, model architecture, training procedures, and evaluation strategies to assess generation quality.

3.0.1 Dataset Overview and Exploration

The dataset used in this study contains 4,632 perovskite structures in CIF format, representing multicomponent oxides. These structures include 25 different elements mixed across the A and B sites. Each structure has information on its composition, energy, and oxygen transfer capacity (OTC), which are important for understanding stability and functionality.

To better understand the data,the elemental composition and co-occurrence patterns were analyzed using a combined figure showing a donut chart and a Pearson correlation heatmap. This illustrates which elements are most common and how frequently they appear together, providing insight into the structural combinations the model needs to learn (see Figure 3.1).

The OTC and energy distributions were also analyzed across all CIFs. The OTC histogram shows how oxygen transfer capacity varies among structures, highlighting those with particularly high or low values. The energy histogram illustrates the range of structural stability across the dataset (see Figure 3.2).

Finally, the **token number distribution** was analyzed to understand the variability in CIF sequence lengths. Most structures have a moderate number of tokens, but some very long sequences could challenge the Transformer during training (see Figure 3.3).

These analyzes provide a clear picture of the dataset and guide decisions regarding preprocessing, tokenization, and model design for the Transformer.

3.0.2 Preprocessing

The raw CIF files cannot be used directly for training and therefore require preprocessing. This step prepares the data in a consistent and clean form so that the Transformer can focus on learning structural patterns rather than addressing formatting issues.

Each CIF file is first checked to ensure that it represents a valid crystal structure. Structures with partial atomic occupancy are excluded by default, as incomplete occupancies introduce uncertainty in atomic positions. Oxygen transfer capacity

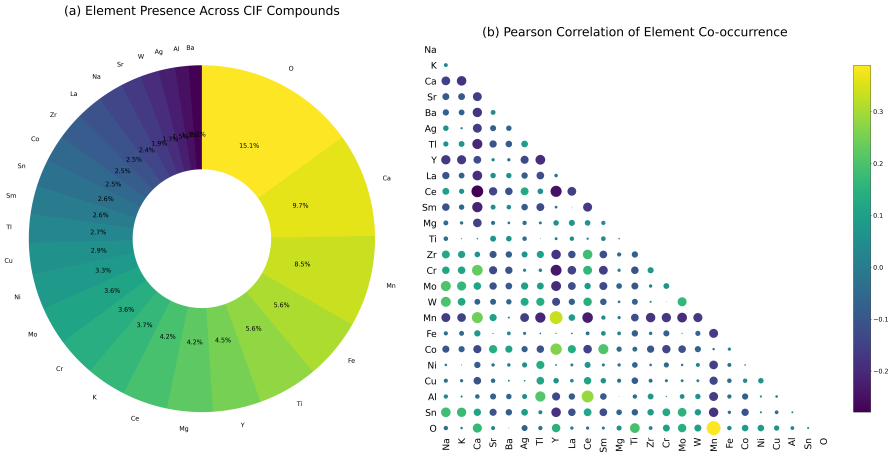


Figure 3.1: Combined donut chart and Pearson correlation heatmap showing element composition and co-occurrence across 4,632 CIFs.

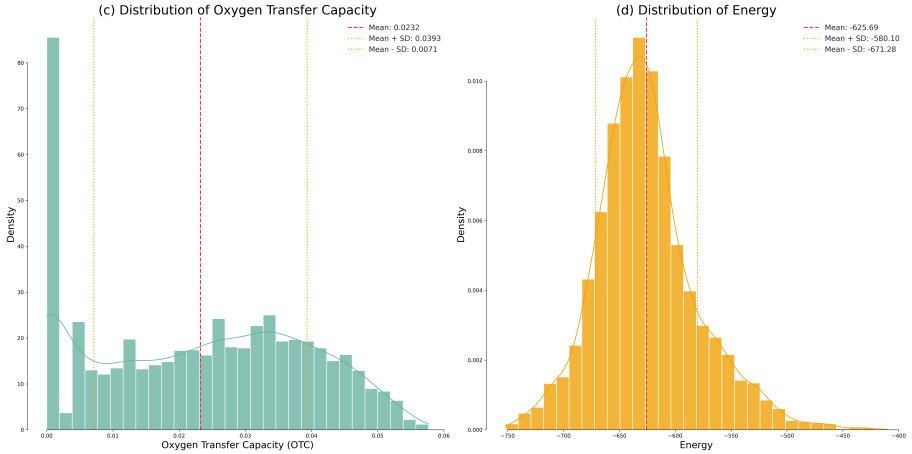


Figure 3.2: Combined histogram showing OTC values and energy values across the dataset.

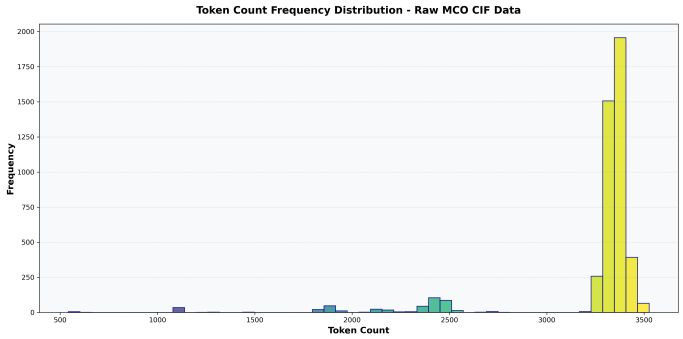


Figure 3.3: Histogram showing the number of tokens per CIF, illustrating sequence length variation.

(OTC) and energy values are extracted and normalized to place all structures on a comparable scale.
The CIF text is then simplified and standardized. Unnecessary header comments are

removed, numerical values are rounded to a fixed precision, and atomic information is written in a consistent format. Chemical composition, atomic species, and space group information are also extracted, as they are later used during training and dataset splitting.

Overall, preprocessing reduces noise in the data, limits extreme variations between structures, and produces a clean and uniform representation of CIF files.

3.0.3 Tokenization and Dataset Splitting

After preprocessing, each CIF file is converted into a sequence of discrete tokens using a customized tokenizer designed specifically for crystallographic data. Instead of relying on character-level or word-level tokenization, the tokenizer uses a fixed vocabulary that explicitly includes CIF keywords, element symbols, space-group labels, digits, and punctuation. This design helps preserve the syntactic structure of CIF files while keeping the representation interpretable.

Each token is mapped to a unique numerical identifier (token ID), which is the actual input to the Transformer model. Numerical values are not treated as single tokens; instead, digits and decimal points are tokenized separately. This allows the model to learn numerical patterns directly from the sequence structure rather than relying on predefined numeric embeddings. Tokens that do not belong to the predefined vocabulary are replaced with a special unknown token (`<unk>`). Space-group symbols are disambiguated by appending a suffix to avoid confusion with element names.

Table 3.1 shows a simplified example of how CIF text is tokenized and converted into token IDs.

Table 3.1: Example of CIF tokenization using the customized tokenizer. Each token is mapped to a numerical ID from a fixed vocabulary.

CIF Fragment	Token Sequence	Token IDs
`_cell_length_a 7.62`	`_cell_length_a, 7, ., 6, 2`	`128, 7, 94, 6, 2`
`Ba Ti O`	`Ba, Ti, O`	`64, 29, 7`
`data_sample`	`data_, sample`	211, 56
Pm^3m	`Pm_sg, 3, m`	`301, 3, 145`

Once tokenization is completed, token sequences are padded to a fixed maximum length to enable batch training. A special padding token is used so that shorter sequences do not affect model learning.

The tokenized dataset is then randomly divided into three subsets: 80% for training, 10% for validation, and 10% for testing. The training set is used to learn model parameters, the validation set is used to monitor performance and tune hyperparameters, and the test set is reserved for final evaluation. This split ensures a fair assessment of the model's ability to generalize to unseen crystal structures.

3.1 Training Setup and Conditional Integration

After tokenization and dataset splitting, the model is trained on sequences of token IDs representing crystal structures. Each structure is paired with conditional features, specifically Oxygen Transfer Capacity (OTC) and energy values. These features provide physical guidance to the model, helping it generate structures consistent with desired properties.

3.1.1 Conditional Features and Objective

Let $X = (x_1, x_2, \ldots, x_T)$ represent a token sequence of length T for a crystal structure, and let $c = [\text{OTC}, \text{Energy}]$ be the conditional vector associated with this structure.
The model aims to learn the **conditional probability distribution**:

$$P(X \mid c; \theta) = P(x_1, x_2, \ldots, x_T \mid c; \theta)$$

Using the **chain rule of probability**, this can be factorized as:

$$P(X \mid c; \theta) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_{t-1}, c; \theta)$$

Here:

- x_t is the token at position t.
- θ are the model parameters.
- c ensures that each predicted token considers the conditional features.

3.1.2 Dual Conditioning Approach

The model integrates conditional features using **encoder embeddings** and **prefix tokens**.

3.1.2.1 Encoder Conditioning

Conditional features c are passed through a pretrained encoder network f_{enc} to create a high-dimensional embedding:

$$e_c = f_{\text{enc}}(c) \in \mathbb{R}^{d_{\text{emb}}}$$

This embedding is added to each token embedding in the sequence:

$$\tilde{x}_t = x_t + e_c, \quad t = 1, \ldots, T$$

3.1.2.2 Prefix Conditioning

The conditional vector c is also converted into discrete tokens and prepended to the token sequence:[**?**, 21]

$$X_{\text{input}} = [c_{\text{tokens}}, x_1, x_2, \ldots, x_T]$$

This allows the transformer to process conditional information as part of the input sequence, influencing attention directly.

3.1.3 Precomputed Conditional Embeddings

To improve efficiency, embeddings e_c for all unique feature combinations are precomputed:

$$\{e_c \mid c \in \text{unique OTC-energy pairs}\}$$

During training, the model fetches these embeddings rather than recomputing them, reducing computation and memory cost.

3.1.4 Batch Size and Block Size

The model processes sequences in batches of size $B = 32$. Each batch is represented as a tensor:

$$X_{\text{batch}} \in \mathbb{R}^{B \times T \times d_{\text{emb}}}$$

Here $T = 1750$ is the **block size**, representing the maximum number of tokens per sequence (including prefix). Shorter sequences are padded with a special token so that all sequences in a batch have the same length.

3.1.5 Transformer Forward Pass

For each layer $l = 1, 2, \ldots, L$, the transformer computes **hidden states** $h_t^{(l)}$ for each token t:

$$h_t^{(l)} = \text{TransformerLayer}(h_1^{(l-1)}, \ldots, h_T^{(l-1)})$$

where $h_t^{(0)} = \tilde{x}_t$. The final layer outputs $h_t^{(L)}$, which contains contextualized representations for prediction.

3.1.6 Prediction and Loss

At each token position t, the model predicts a probability distribution over the vocabulary:

$$\hat{y}_t = \text{softmax}(W_o h_t^{(L)} + b_o)$$

The **cross-entropy loss** is used to measure the difference between predicted probabilities $\hat{y}_t$ and true token IDs y_t:

$$\mathcal{L} = -\frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \log \hat{y}_t^{(b)}[y_t^{(b)}]$$

Minimizing $\mathcal{L}$ ensures that the predicted sequence matches the true CIF tokens while respecting conditional features. [19]

3.1.7 Parameter Update

Model parameters θ are updated using AdamW optimizer with gradient clipping:[24]

$$\theta \leftarrow \theta - \eta \frac{\nabla_\theta \mathcal{L}}{\|\nabla_\theta \mathcal{L}\|_2 + \epsilon}$$

Gradient clipping ensures stable training by preventing exploding gradients.[15]

3.1.8 Training Pipeline Summary

The training pipeline can be summarized as follows:

1. Tokenized CIF sequences and conditional features c are prepared.
2. Conditional embeddings are obtained via encoder and prefix tokens.
3. Transformer processes the sequences and computes hidden states.
4. Probabilities for the next token are predicted.
5. Cross-entropy loss is computed.
6. Model parameters are updated.
7. Validation is performed and checkpoints are saved.

This setup ensures that the model learns the conditional distribution $P(X \mid c)$ effectively, enabling the generation of physically meaningful and structurally accurate crystal sequences.

3.1.9 Checkpointing in Training

Checkpoints are essential for saving and restoring the model's state during training and evaluation. They preserve progress and allow resumption, evaluation, or further analysis.

3.1.9.1 Purpose of Checkpoints

Checkpoints serve several important purposes:

- **Resume Training:** Allows continuation after interruptions without losing progress.
- **Evaluation:** Enables testing on unseen data using saved models.
- **Best Model Saving:** Keeps the model with the lowest validation loss for later use.
- **Reproducibility:** Ensures that results can be replicated for debugging or further experiments.

3.1.9.2 Contents of a Checkpoint

A typical checkpoint includes:

1. **Model State Dictionary:** Contains all trainable parameters, including embeddings, attention layers, feedforward layers, and normalization weights.
2. **Optimizer State:** Stores optimizer variables such as learning rate, momentum, and gradient history to allow consistent resumption of training.
3. **Training Configuration:** Includes model architecture details, hyperparameters, and dataset paths.

4. **Training Metrics:** Tracks the current iteration, best validation loss, and early stopping counters.
5. **Pretrained Encoder State :** Includes weights of any pretrained encoder integrated in the model.

Checkpoints are saved periodically during training:

- After every evaluation interval.
- If validation loss improves, the best-performing model is saved.
- If checkpoint is set to true, checkpoints are saved after every evaluation.

Saving is performed asynchronously to prevent blocking the training loop.
Checkpoints are loaded during evaluation or to resume training. The process includes:

1. Loading the checkpoint file using `torch.load` with appropriate device mapping.
2. Extracting the state dictionary of the model.
3. Rebuilding the model according to the saved configuration.
4. Loading the state dictionary into the model using a function `load_state_dict`.
5. Returning the fully restored model for evaluation or training continuation.

Checkpoints are saved as `.pth` files, which is the standard format for PyTorch models and supports compatibility across CPU and GPU devices.

3.1.10 CIF Generation

CIF generation refers to the process by which the trained Transformer model produces new crystal structures in the form of Crystallographic Information Files (CIFs). Each generated CIF represents a candidate multicomponent perovskite structure, including lattice parameters, atomic species, and fractional atomic positions.
The model generates CIFs in an *autoregressive* manner. Starting from an initial prompt, the model predicts one token at a time, where each prediction depends on all previously generated tokens. This allows the model to learn and reproduce the sequential structure of CIF files, including repeated patterns such as cell parameters, symmetry information, and atomic position blocks.
Generation is guided by conditional information provided to the model. This includes the chemical composition and, when applicable, target material properties such as oxygen transfer capacity (OTC) and energy. These conditions help steer the generation process toward structures with desired chemical and functional characteristics.
Producing valid CIFs is challenging due to their length and the strict syntax of CIF files. CIF sequences can contain hundreds or thousands of tokens, requiring the model to maintain long-range consistency. In addition, CIF files follow rigid formatting rules, and small token errors can lead to invalid or non-physical structures. The complexity of multicomponent perovskites further increases the difficulty, as multiple elements may share lattice sites with varying occupancies.
After generation, the produced CIF files are evaluated to assess their structural validity and chemical consistency. Valid CIFs provide insight into the model's ability to learn crystallographic rules and can be used to explore new perovskite compositions within a large and complex design space.

4
Results

This chapter presents the outcomes of CIF generation, model performance, and evaluation metrics.
All experiments presented in this chapter were conducted on the Alvis A100 GPU [27], which provided the computational resources required to train and evaluate the transformer-based models used in this work. This section presents the results of model training and architectural comparison and explains the rationale behind the final model choice used for structure generation and evaluation.

4.0.1 Model Architecture Exploration and Selection

The first set of experiments focused on analyzing the impact of model depth on training behavior and overall performance. Three transformer architectures with 2, 4, and 8 layers were initially considered. The objective was to investigate whether increasing model depth leads to improved learning or better generation quality for crystal structure data.
During training, the 8-layer transformer model could not complete the intended number of training iterations. The increased depth resulted in substantially higher GPU memory usage, which exceeded the available memory on the Alvis A100 during training. Consequently, the training process terminated prematurely. Since this model could not be trained under the same conditions as the others, it was excluded from further evaluation.
Both the 2-layer and 4-layer models were successfully trained without memory-related issues. Their training behavior was analyzed by comparing the loss curves over training iterations. The loss plots show that all models follow a similar convergence pattern, with no clear improvement in convergence speed or final loss value when increasing the number of layers.
These observations indicate that increased model depth does not provide a clear advantage for the given dataset and task. Considering training stability, memory efficiency, and comparable learning behavior, the 2-layer transformer model was selected for all subsequent experiments. This model was, therefore, used for CIF generation, checkpoint-based evaluation, and all further analyzes presented in this chapter.

4.0.2 Model Configuration and Training Setup

The final model used for CIF generation was a transformer with 2 layers and 2 attention heads. The embedding dimension was set to 512, and the block size was

1750 tokens. This allowed the model to handle long CIF sequences that contain structural information, lattice parameters, and atomic positions. The model had approximately 25.6 million trainable parameters, which were large enough to learn complex structural patterns while still fitting within the available GPU memory.

Training was performed with a batch size of 16. The AdamW optimizer was used because it is well-suited for transformer training and provides stable learning. The initial learning rate was set to 1×10^{-3} and was gradually reduced to a minimum learning rate of 1×10^{-5} as training progressed. A warm-up phase of 100 iterations was used at the beginning of training to make the optimization process more stable and to prevent large updates in the early stages.

The optimizer beta values were set to (0.9, 0.98), which control how past gradients are averaged during training. The model was trained for a maximum of 50,000 iterations. During this process, checkpoints were saved regularly so that CIF generation and evaluation could be performed at different stages of learning.

This configuration provided a good balance between model size, learning ability, and memory efficiency, which made it suitable for learning CIF structure patterns.

4.0.3 CIF Generation and Checkpoint Analysis

CIF files were generated from each of the 100 saved checkpoints. During analysis, the model generally produced valid atomic positions and structural features. However, after the final atomic position entries in many generated CIF files, invalid characters occasionally appeared. These trailing regions were ignored during evaluation to ensure that the analysis reflected only the meaningful structural information.

The generation success rate was tracked across checkpoints to observe how the proportion of valid CIF files evolved during training. In the early stages, the success rate was relatively low, with the minimum value around 50%. As training progressed, the model became more stable and a clear improvement was observed. After approximately 25,000 iterations, the success rate consistently remained above 90%, with the highest recorded value reaching 99.4%. This trend suggests that the model gradually learned to generate structurally valid CIF files more reliably as training continued.

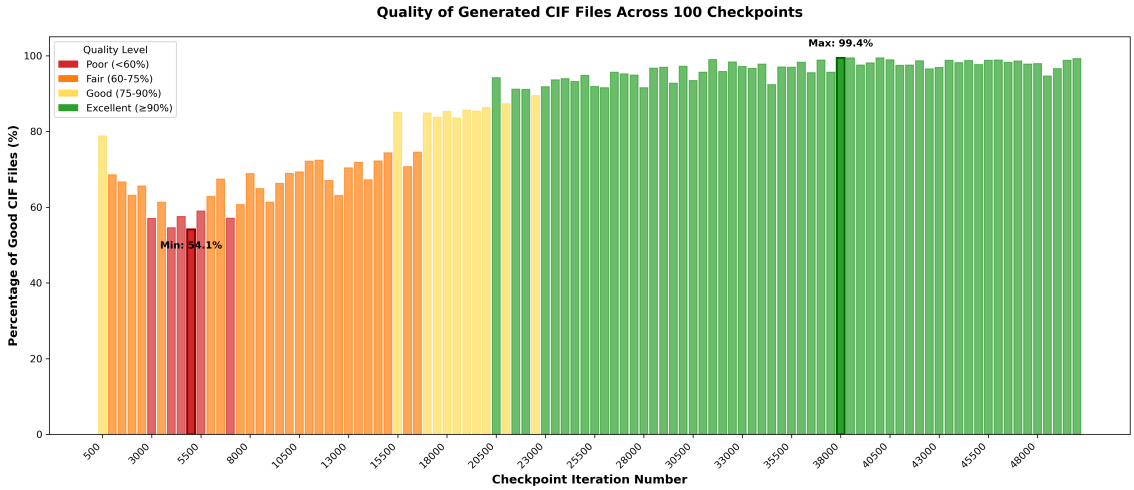


Figure 4.1: Percentage of valid CIF files generated at each checkpoint during training. The success rate shows a steady improvement, exceeding 90% after around 25,000 iterations and reaching a maximum of 99.4%, indicating increased generation stability over time.

4.0.4 OTC and Energy Range and Distribution Analysis

The generated CIF files were analysed with a focus on their OTC and energy values. The distributions of these properties from the generated structures closely resemble those from the raw test data. Both the range and the spread of values are well preserved, indicating that the model learned to generate outputs within physically meaningful limits and avoided producing extreme or unrealistic values.
Figure 4.2 shows the overall distribution of OTC and energy values, confirming that the generated data covers a similar range as the ground truth. Figure 4.3 compares generated and raw values in a scatter plot, revealing a surprisingly linear pattern. Although a more scattered relationship was initially expected, the linear trend indicates that the model maintains a consistent relationship between OTC and energy.
Overall, the generated values generally follow the range of the raw data. The observed linear pattern in the scatter plot is noted as a point for further consideration rather than a definitive conclusion about the model's behavior.

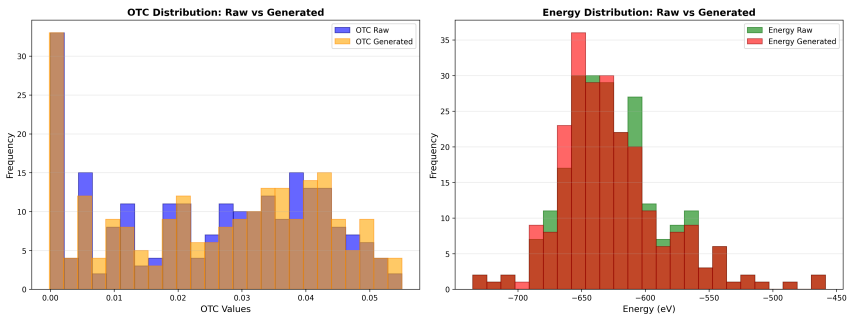


Figure 4.2: Distribution of OTC and energy values for raw test data and generated CIF files.

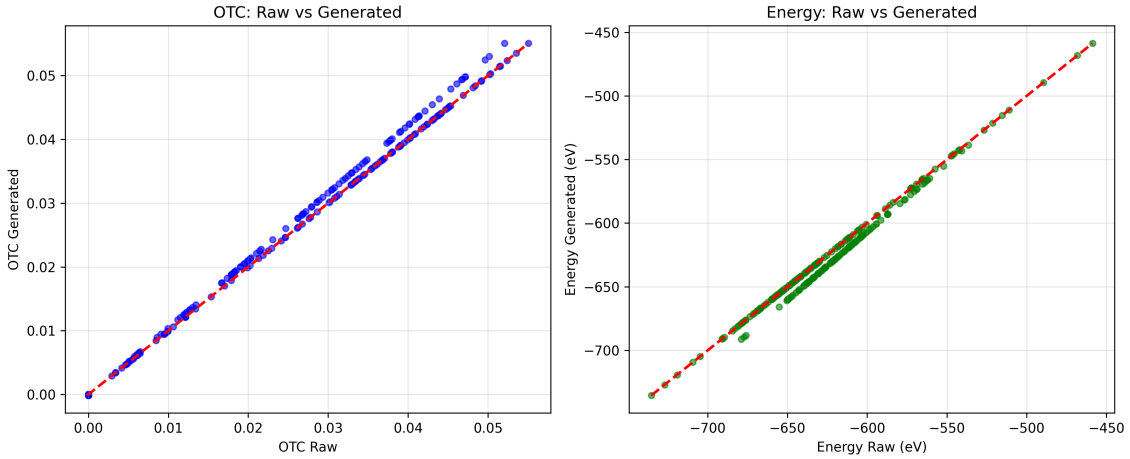


Figure 4.3: Scatter plot comparing raw and generated OTC and energy values.

4.0.5 Atomic Position Error Analysis

To evaluate how accurately the model predicts atomic positions, the Mean Absolute Error (MAE) and Mean Squared Error (MSE) were calculated by comparing the generated structures with the reference structures. These metrics were computed across all atomic coordinates and then plotted to observe the overall behavior.

The results show noticeable but moderate differences between the predicted and true atomic positions. This means the model does not reproduce the exact coordinates perfectly. However, this is expected, as predicting full atomic structures is a highly complex task involving strict spatial and chemical constraints. Despite the presence of errors, their magnitude remains within a reasonable range. This indicates that the generated structures are still physically meaningful and can provide useful guidance for materials design and structural exploration, even if they are not precise enough for exact structural determination.

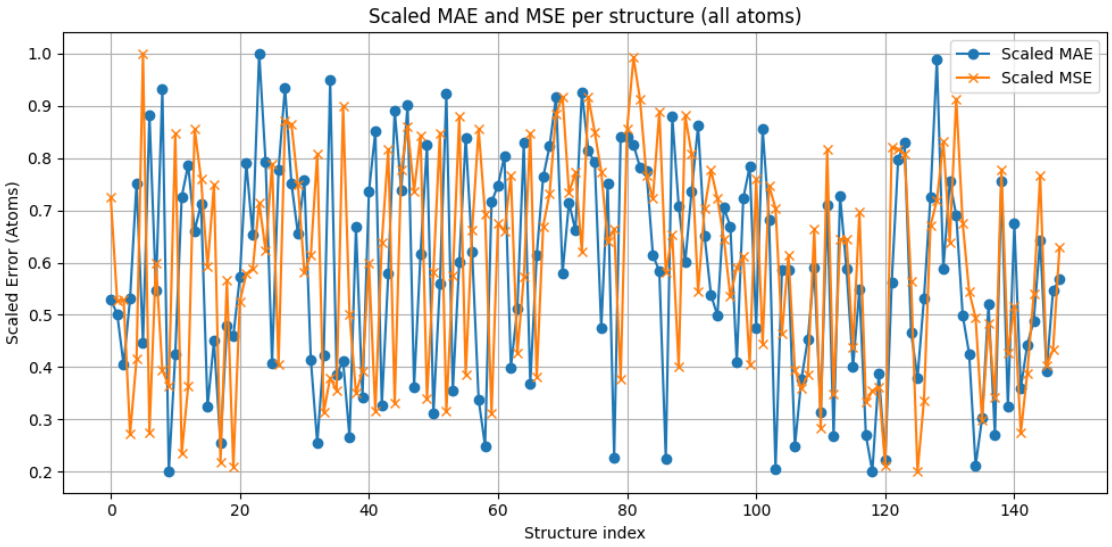


Figure 4.4: Mean Absolute Error (MAE) and Mean Squared Error (MSE) of predicted atomic positions compared to reference structures. The plot shows that, although deviations exist, the overall error remains within a reasonable range, indicating physically meaningful structure generation.

4.0.6 Element-Level Generation Behavior

To better understand the chemical factors influencing generation quality, the presence of individual elements was analyzed in both well-generated and poorly-generated CIF structures. The goal was to determine whether certain elements are more frequently associated with high-quality generations or with structural deviations. This was quantified using a *preference ratio*, which measures how often an element appears in high-quality structures relative to deviated ones.

The analysis shows that some elements are strongly associated with well-generated structures. These include yttrium (Y), lanthanum (La), samarium (Sm), strontium (Sr), barium (Ba), and cobalt (Co). For example, yttrium (Y) appears in good generations about 94% of the time, lanthanum (La) about 91%, and samarium (Sm) about 88%. This suggests that the model handles structures containing these elements more reliably. A possible reason is that these elements occur in more regular or well-represented structural environments in the training data.

In contrast, some elements are more frequently linked to deviations. Zirconium (Zr) appears in poorly-generated structures in about 85% of its occurrences, while chromium (Cr) and molybdenum (Mo) are found in deviated generations more than 83% of the time. Tin (Sn) shows a similar trend. This may indicate that structures containing these elements are more complex, less common in the dataset, or involve coordination environments that are more difficult for the model to learn.

Other elements, including iron (Fe), nickel (Ni), calcium (Ca), titanium (Ti), tungsten (W), silver (Ag), potassium (K), and thallium (Tl), show a more balanced behavior. They appear in both well-generated and deviated structures at similar rates, suggesting moderate and stable model performance for these chemical environments.

The element-wise trends are visualized in Figures 4.5 and 4.6, which show how structural deviation and energy-related differences vary across elements. These plots highlight that generation quality is not only a modeling issue but also closely linked to the underlying chemistry of the elements involved.

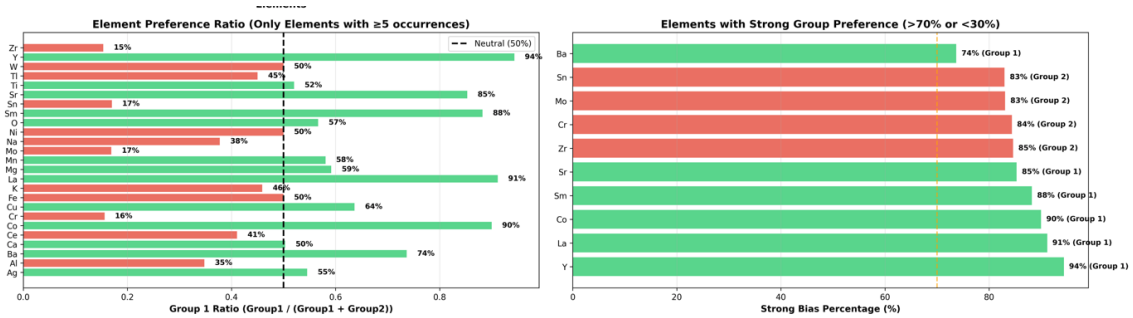


Figure 4.5: Element-wise analysis of structural deviation (OTC-related metric).

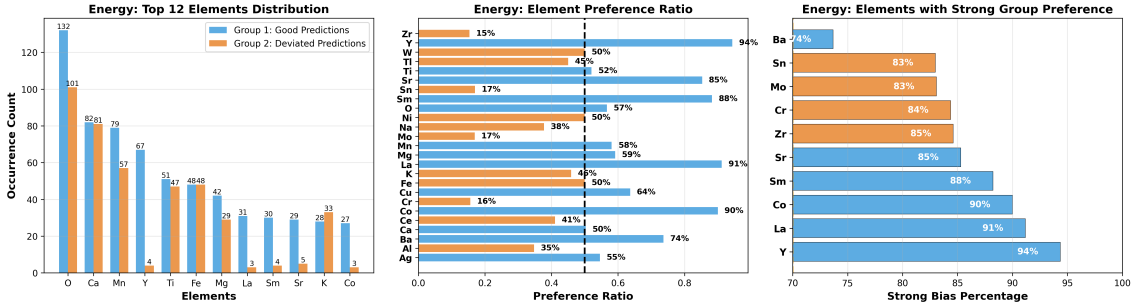


Figure 4.6: Element-wise analysis of energy-related differences between well-generated and deviated structures, illustrating how generation quality varies depending on chemical composition.

5
Conclusion

This work demonstrates that the Transformer model can successfully generate CIF files, highlighting its potential as a tool for materials design. The model is capable of learning structural patterns and producing crystal structures that are chemically reasonable rather than random, which is promising for complex systems like high-entropy oxides.
However, further analysis is necessary to fully assess the quality and correctness of the generated structures. While the outputs generally follow meaningful patterns, it remains important to determine whether the model is truly generating novel structures or primarily reproducing patterns observed in the training data. Understanding this distinction between learning and memorization is a key area for future work.
The results also indicate that the training data contains many similar structures. This limits the diversity of the generated outputs, suggesting that expanding the dataset to include more varied compositions could help the model explore a wider chemical space and improve the novelty of generated structures.
In addition, alternative sampling techniques could be explored during generation to enhance diversity while maintaining realistic and stable structures. Adjustments in the way outputs are selected may allow the model to produce a broader range of chemically valid materials.
Overall, the study shows that the Transformer-based approach is promising for generating crystal structures, but additional experimentation, validation, and optimization are required before it can be confidently applied to the discovery of new high-entropy oxide materials.

Bibliography

[1] J. Hildingsson, "Material exploration through active learning: A method to explore compositional space and find oxygen carriers for chemical looping applications, 2024.

[2] Y. De Vos et al., "Development of stable oxygen carrier materials for chemical looping processes—A review," *Catalysts*, vol. 10, no. 8, 926, 2020.

[3] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, "The Materials Project: A materials genome approach to accelerating materials innovation," *APL Materials*, vol. 1, no. 1, 011002, 2013.

[4] S. R. Hall, F. H. Allen, and I. D. Brown, "The crystallographic information file (CIF): a new standard archive file for crystallography," *Acta Crystallographica Section A*, vol. 47, pp. 655–685, 1991.

[5] I. D. Brown and B. McMahon, "The Crystallographic Information File (CIF)," *Data Science Journal*, vol. 5, pp. 174–177, 2006.

[6] International Union of Crystallography (IUCr), "CIF standard specification," IUCr Resources, 1991.

[7] K. Momma and F. Izumi, "VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data," *Journal of Applied Crystallography*, vol. 44, pp. 1272–1276, 2011.

[8] A. Lyngfelt, *Chemical-looping combustion of solid fuels – Status of development.* Fuel, 83, pp. 1459–1473, 2004.

[9] A. Lyngfelt, "Chemical looping combustion," in *Greenhouse Gas Issues*.

[10] J. Brorsson, H. K. Moberg, J. Hildingsson, J. Gastaldi, T. Mattisson, and A. Hellman, *Data-Efficient Design of High-Entropy Oxygen Carriers for Chemical Looping Using Active Learning*, ACS Materials Au, 2026.

[11] J. Brorsson, H. K. Moberg, J. Hildingsson, J. Gastaldi, T. Mattisson, and A. Hellman, "Material exploration through active learning – METAL," arXiv:2601.03933, 2026.

[12] I. Adánez-Rubio et al., "Use of a high-entropy oxide as an oxygen carrier for chemical looping," *Energy*, vol. 298, 131307, 2024.

[13] A. Sarkar et al., "High entropy oxides for reversible energy storage," Chemistry literature, 2018.

[14] C. Riley et al., "A High Entropy Oxide Designed to Catalyze CO Oxidation."

[15] R. Pascanu, T. Mikolov, and Y. Bengio, *On the Difficulty of Training Recurrent Neural Networks*, ICML, 2013.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI Technical Report, 2019.

[17] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," ACL, 2016.
[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention Is All You Need*, Advances in Neural Information Processing Systems, 2017.
[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," arXiv:1607.06450, 2016.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, NAACL, 2019.
[22] T. Shen, "Chemical looping combustion performance of high-entropy oxide oxygen carriers," Journal Article, 2025.
[23] J. Yang, "A comprehensive review of high entropy oxides: unique properties and applications," SciOpen, 2025.
[24] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*, ICLR, 2019.
[25] S. Gražulis et al., "Crystallography Open Database (COD): An open-access collection of crystal structures," *Nucleic Acids Research*, 2012.
[26] F. L. Johansen et al., "deCIFer: Crystal Structure Prediction from Powder Diffraction Data using Autoregressive Language Models," arXiv:2502.02189, 2025.
[27] Chalmers e-Commons, C3SE, & NAISS, *Alvis: National AI/ML GPU Cluster with NVIDIA A100 GPUs*, NAISS Resource Information, 2026. Available at: `https://www.naiss.se/resource/alvis/`

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY