Advancing AI-Driven Molecular Generation: Exploring Model Architectures, Decoding Strategies, and Tokenization Techniques

Mahomed Aadil Ally
University of Cape Town
Cape Town, South Africa
ALLMAH002@myuct.ac.za

Anri Lombard
University of Cape Town
Cape Town, South Africa
LMBANR001@myuct.ac.za

Gabriel Marcus
University of Cape Town
Cape Town, South Africa
MRCGAB004@myuct.ac.za

ABSTRACT

The discovery of novel therapeutic compounds is crucial for modern drug development, but traditional approaches struggle to efficiently explore the vast chemical space of potential drug-like molecules. This proposal aims to advance AI-driven molecular generation by conducting a comprehensive investigation of state-of-the-art techniques, focusing on tokenization methods, model architectures, and decoding strategies. We will compare different tokenization techniques for molecular string representations and assess their impact on generative model performance. We will also evaluate the performance of Transformer-based models, like SAFE-GPT, against State Space Model (SSM) architectures, such as MAMBA and hybrid models like Griffin, on molecular generation tasks using different-sized datasets. Finally, we will investigate various maximization-based and stochastic decoding strategies to determine the optimal approach for generating unique, diverse, and drug-like molecules. By systematically investigating these aspects, we aim to develop an efficient and adaptable base model for molecular generation.

1 INTRODUCTION AND MOTIVATION

Drug discovery is a time-consuming and expensive process, with traditional approaches often struggling to efficiently explore the vast chemical space of potential drug-like molecules [4, 22]. The identification of novel compounds with desirable pharmacological properties is a cornerstone of modern drug development, yet the challenges associated with navigating the immense landscape of possible molecular structures have hindered progress in this field.

In recent years, the application of machine learning, particularly generative AI models, has emerged as a promising strategy to accelerate the discovery of new therapeutic compounds [21, 22]. By learning the underlying distribution of known molecules and generating novel structures that adhere to this distribution, these models have the potential to revolutionize the early stages of drug discovery. Generative models can explore vast chemical spaces more efficiently than traditional methods, proposing novel compounds with a higher likelihood of possessing desired properties.

The advent of Transformer architectures [27], which have achieved state-of-the-art performance in natural language processing tasks, has further propelled the field of AI-driven molecular generation. The self-attention mechanism employed by Transformers allows them to capture long-range dependencies and learn complex patterns in molecular data. Adaptations of Transformer models, such as the Molecular Transformer [24] and MoLeR [16], have demonstrated the power of these architectures in predicting chemical reactions and optimizing molecules for specific properties.

More recently, the development of advanced molecular representations, such as the SAFE (Sequential Attention-based Fragment Embedding) format [17], has opened new avenues for molecular generation. The SAFE representation encodes molecules as strings of interconnected fragments, providing a more chemically intuitive and interpretable representation compared to traditional string-based formats like SMILES. SAFE-GPT, a Transformer-based model trained on the SAFE representation, has shown impressive results in generating diverse and novel molecules, highlighting the potential of combining state-of-the-art architectures with domain-specific representations.

Despite these advancements, several challenges remain in developing efficient and effective models for generating high-quality, diverse, and novel molecules. The computational complexity of Transformer models grows quadratically with the sequence length, limiting their ability to handle the long sequences often required to represent complex molecular structures. Furthermore, the choice of decoding strategy and tokenization technique can significantly impact the diversity and quality of the generated molecules, necessitating careful consideration and evaluation of these components.

In this project, we aim to address these challenges by conducting a comprehensive investigation of the various aspects influencing the performance of AI-driven molecular generation models. By systematically comparing different tokenization techniques, model architectures and decoding strategies, we seek to provide insights into the most effective approaches for generating diverse and novel molecules. The ultimate goal is to contribute to the development of more efficient and accurate tools for exploring chemical space, accelerating the discovery of new therapeutic compounds, and advancing the field of drug discovery.

2 RESEARCH MOTIVATION

2.1 Problem Statement

Despite the recent advancements in AI-driven molecular generation, several challenges remain in developing efficient and effective models for generating high-quality, diverse, and novel molecules. These challenges include:

(1) The quadratic computational complexity of self-attention in Transformer models, which limits their ability to handle long sequences representing complex molecular structures.
(2) The impact of decoding strategies on the diversity and quality of generated compounds, with the need to identify the most effective approach for molecular generation tasks.
(3) The role of tokenization techniques in the model's ability to learn meaningful representations of molecular strings.
(4) The trade-off between using large, diverse datasets and smaller, domain-specific datasets for training molecular generation models.

2.2 Research Questions

To tackle the aforementioned challenges, we propose the following research questions:

(1) How do State Space Models (SSMs) and hybrid models combining elements of SSMs and Transformers compare to Transformer-based models in terms of performance on molecular generation tasks when trained on datasets of varying sizes and specificity?
(2) What is the impact of different decoding strategies, such as stochastic and maximization-based methods, on the key metrics of generated molecules, including uniqueness, diversity, drug-likeness, and validity and are the results the same for both a large model and a smaller model.
(3) How do various tokenization strategies affect the performance of molecular generation models in terms of validity, uniqueness, diversity, and quantitative estimate of drug-likeness (QED)?
(4) How do constraining decoders affect the quality of the generated output in terms of the evaluation metrics?

By investigating these research questions, we aim to provide insights into the most effective model architectures, decoding strategies, and tokenization techniques for AI-driven molecular generation.

3 RELATED WORK

3.1 Molecular String Representations

SMILES [28] is a compact and human-readable string representation that encodes the molecular structure using a combination of characters and rules for connectivity. However, SMILES has some limitations, such as the lack of uniqueness (a single molecule can have multiple valid SMILES representations) and sensitivity to small perturbations, which can lead to invalid or unintended molecular structures when modified.

Self-Referencing Embedded Strings (SELFIES) [11] is another string representation that aims to address some of the limitations of SMILES. SELFIES uses a set of derivation rules to generate a unique and valid representation for each molecule, ensuring that any valid SELFIES string can be decoded into a valid molecular structure. However, SELFIES strings can be less human-readable and may not capture the inherent structure of molecules as effectively as other representations.

The SAFE representation [17] addresses these limitations by representing molecules as an unordered sequence of interconnected fragments, each associated with attachment points. SAFE strings are derived from SMILES by fragmenting the molecule using a specified set of rules and then encoding the fragments and their attachment points using a special syntax. This approach provides a more structured and interpretable representation of molecules compared to SMILES, enabling generative models to perform tasks such as fragment-based molecular design and scaffold modification more effectively.

SAFE offers several advantages over SMILES. It provides improved interpretability by clearly separating molecular fragments and their connectivity, making it easier to understand and analyze the generated molecules. The fragment-based nature of SAFE allows for more precise control over the molecular generation process, enabling tasks such as fragment-based molecular design and scaffold modification. SAFE's syntax and rules also help ensure the validity of the generated molecules, reducing the likelihood of producing invalid or chemically infeasible structures. However, SAFE does have some potential limitations. The SAFE representation may be less compact compared to SMILES for certain molecules, especially those with complex ring systems or highly branched structures. Additionally, the fragmentation rules used in SAFE (e.g., BRICS) might not always align with the desired chemical substructures or synthetically accessible building blocks for specific applications. Despite these limitations, SAFE's benefits in interpretability, controllability, and validity make it a promising alternative to SMILES for various molecular generation tasks.

3.2 SAFE-GPT

SAFE-GPT [17] is a Transformer-based (Appendix C.2, Figure **??**) generative model designed for molecular design tasks, utilizing the Sequential Attachment-based Fragment Embedding (SAFE) representation. SAFE encodes molecules as a sequence of interconnected fragments, enabling SAFE-GPT to capture structural and chemical properties more effectively than traditional string-based representations like SMILES. This approach improves the interpretability and controllability of the generated molecules.

The model architecture follows a decoder-only structure with task-specific adaptations. Fragment embeddings are used to represent the molecular fragments, and property conditioning allows the model to generate molecules with desired properties. Scaffold-based generation focuses on generating molecules with specific scaffolds, while reinforcement learning is employed to optimize the generated molecules towards a specific objective.

Although SAFE-GPT has shown impressive results in generating diverse and novel molecules, its performance heavily depends on the quality and diversity of the training data. The model's ability to generalize to other datasets or chemical spaces requires further evaluation. Moreover, like other Transformer-based models, SAFE-GPT may encounter challenges when generating molecules with complex spatial and geometric constraints due to the limitations of the self-attention mechanism, which has a quadratic computational complexity with respect to the sequence length.

3.3 Tokenization

3.3.1 Byte Pair Encoding (BPE) for Molecular Data. Inspired by the success of Byte Pair Encoding (BPE) [25] in open-domain natural language tasks like machine translation, researchers have adapted this data-driven subword tokenization technique for molecular SMILES. BPE iteratively merges the most frequent character pairs into new tokens, effectively creating a vocabulary of subword units that commonly occur together. In the context of molecular data, works like SmilesPE train the BPE algorithm on large datasets of SMILES strings to learn a vocabulary of multi-atom subword tokens [3].

3.3.2 WordPiece Tokenization. WordPiece [23] is a subword tokenization algorithm similar to BPE but with some differences in the merging criteria and vocabulary construction. While BPE focuses on merging the most frequent character pairs, WordPiece

considers all possible merges and selects the one that maximizes the likelihood of the training data. WordPiece has been explored for tokenizing molecular data in works like MolBERT [6], which employs this technique for pretraining a BERT-based model on SMILES strings.

3.3.3 DeepSMILES & Grammar-Based Tokenization. DeepSMILES introduced a rule-based tokenization scheme for SMILES that groups atoms and bonds into chemically meaningful subunits like rings, branches, atom types, and bond types [3]. Unlike atom-wise tokenization, this approach aims to capture higher-level structural and chemical information within each token, providing an interpretable tokenization that can potentially enhance the model's ability to learn meaningful representations of molecular motifs. Similar grammar-driven tokenization approaches have been explored in works like SMARTS Transformer [16], which employs a context-free grammar to tokenize SMILES into a hierarchical representation of functional groups and substructures. These methods leverage domain knowledge and chemical rules to guide the tokenization process, which improve the model's understanding of the underlying molecular structure.

3.4 Evaluation Metrics

Evaluating the quality and practicality of machine-generated molecules is a critical aspect of developing generative models for molecular design. Several metrics have been proposed to assess the performance of these models, each focusing on different aspects of the generated molecules.

3.4.1 Validity, Uniqueness, and Novelty. The most fundamental metrics for evaluating generative models are validity, uniqueness, and novelty. Validity measures the percentage of chemically valid structures according to a molecular parsing tool such as RDKit [14]. Given a set of generated molecules G, the validity is defined as:

$$\text{Validity}(G) = \frac{|m \in G : \text{is\_valid}(m)|}{|G|} \times 100 \tag{1}$$

where is_valid$(\cdot)$ is a function that determines the chemical validity of a molecule using a molecular parsing tool.

Uniqueness refers to the fraction of non-duplicate molecules within a set of generated compounds. It is calculated as:

$$\text{Uniqueness}(G) = \frac{|\text{unique}(G)|}{|G|} \times 100 \tag{2}$$

where unique$(\cdot)$ returns the set of unique molecules in G.

Novelty measures the percentage of generated molecules that are not present in the training dataset. It assesses the model's ability to generate novel molecules that expand beyond the training data. Novelty is calculated as:

$$\text{Novelty}(G) = \frac{|m \in G : m \notin D|}{|G|} \times 100 \tag{3}$$

where D is the training dataset.

These metrics provide a basic assessment of the generative models' performance and are widely reported in the literature. However, it's important to note that these metrics do not directly assess the quality or desirability of the generated molecules for a specific application, only their basic properties.

3.4.2 Diversity. Diversity measures the chemical diversity of the generated molecules. It can be quantified using various metrics, such as the Tanimoto similarity between the generated molecules and the training data, or the distribution of molecular properties like molecular weight, logP, and synthetic accessibility score.

One common approach to measure diversity is to calculate the average pairwise Tanimoto distance between the generated molecules based on their fingerprint representations (e.g., ECFP4) [20]. Given a set of generated molecules G and a fingerprint function $f(\cdot)$, the diversity is defined as:

$$\text{Diversity}(G) = \frac{2}{|G|(|G|-1)} \sum_{i=1}^{|G|} \sum_{j=i+1}^{|G|} \text{Tanimoto}(f(m_i), f(m_j)), \tag{4}$$

where $m_i, m_j \in G$ and Tanimoto$(\cdot,\cdot)$ is the Tanimoto similarity between two fingerprint representations.

A higher diversity score indicates that the generated molecules are more chemically diverse and cover a wider range of the chemical space. However, this diversity metric has limitations in fully capturing the scope of chemical diversity, as it depends on the choice of fingerprint representation and may not account for all the relevant structural and functional differences between molecules.

3.4.3 Quantitative Estimate of Druglikeness (QED). The Quantitative Estimate of Druglikeness (QED) [1] is a metric that assesses the druglikeness of a molecule based on its physicochemical properties. It integrates eight properties (molecular weight, logP, hydrogen bond donors, hydrogen bond acceptors, polar surface area, rotatable bonds, aromatic rings, and structural alerts) into a single score ranging from 0 (low druglikeness) to 1 (high druglikeness).

The QED score is calculated using a weighted geometric mean of the individual property scores:

$$\text{QED} = \left(\prod_{i=1}^{n} d(p_i)^{w_i} \right)^{\frac{1}{\sum_{i=1}^{n} w_i}}, \tag{5}$$

where $d(p_i)$ is the desirability score for property p_i, w_i is the weight assigned to property p_i, and n is the number of properties.

QED provides a quantitative estimate of a molecule's druglikeness and can be used to prioritize generated molecules that are more likely to possess drug-like properties. It has been shown to outperform rule-based methods like Lipinski's Rule of Five [15] in distinguishing drugs from non-drugs. However, QED is a general estimate and may not always align with the specific requirements for a given drug discovery project, which can vary depending on the target, indication, and desired pharmacokinetic properties.

3.4.4 Synthetic Accessibility Score (SAS). The Synthetic Accessibility Score (SAS) [5] is a metric that estimates the ease of synthesis of a molecule based on the complexity of its substructures. SAS is calculated by summing the complexity contributions of each atom in the molecule, along with additional factors such as the presence of chiral centers and the complexity of the molecular graph.

The SAS of a molecule m is defined as:

$$\text{SAS}(m) = \frac{1}{n} \sum_{i=1}^{n} c_i + \frac{1}{n} \sum_{i=1}^{n} r_i + s + t, \tag{6}$$

where n is the number of atoms in the molecule, c_i is the complexity of the i-th atom's substructure, r_i is the number of rings in the substructure, s is the overall complexity of the molecular graph, and t is the number of chiral centers.

SAS ranges from 1 (easy to synthesize) to 10 (difficult to synthesize). Incorporating SAS into the evaluation pipeline can help prioritize generated molecules that are more feasible to synthesize, thus increasing the practical utility of the generative models. However, SAS is an approximation of synthetic accessibility and may not capture all the nuances and challenges involved in actual chemical synthesis, such as reagent compatibility, reaction conditions, and purification steps.

Despite these limitations, validity, uniqueness, novelty, diversity, QED, and SAS remain widely used evaluation metrics for assessing the performance of molecular generative models.

4 PROCEDURES AND METHODS

We will focus on three key aspects: tokenization techniques, model architectures, and decoding strategies. The study will be conducted using the SAFE-GPT dataset [17] for training and evaluation, with additional experiments on the MOSES dataset [18] for comparative analysis.

4.1 Datasets

4.1.1 ZINC Dataset. We will use a subset of roughly 200 million molecules extracted from the ZINC20 database as our primary training dataset. ZINC (ZINC Is Not Commercial) is a curated collection of commercially available chemical compounds prepared for virtual screening. The molecules in the ZINC dataset are originally represented using the SMILES format. To ensure compatibility with the SAFE-GPT model and maintain consistency across the three subprojects, we will convert the SMILES representations to the SAFE format using the algorithm described in the original SAFE paper [17]. This conversion process will be performed as a preprocessing step before training the baseline SAFE-GPT model. The resulting SAFE-encoded ZINC dataset will be used to train the baseline model, which will serve as the starting point for the tokenization, architecture comparison, and decoder strategy experiments.

4.1.2 MOSES Dataset. We will conduct additional experiments on the Molecular Sets (MOSES) dataset [18] to provide a comparative analysis of the models' performance on a smaller, drug-discovery-focused dataset. The MOSES dataset contains 1.9 million drug-like molecules represented using the SMILES format refined from the ZINC Clean Leads collection and filtered out roughly 2.5 million zinc molecules based on some specific criteria. Rather than going into these specifics, the dataset is preferred because the original SAFE paper trained their smaller model on it, and thus to attempt reproduction of their results and extend it to comparison of other small model versions this dataset fits the bill. Since the molecules are encoded in the SMILES representation the dataset will be converted to the SAFE representation by applying the algorithm [17].

4.2 Tokenization Techniques

We will compare the performance of the SAFE-GPT model using three tokenization techniques: Byte Pair Encoding (BPE)[25], WordPiece tokenization[13, 23], and character-level tokenization [12]. BPE and WordPiece tokenizers will be trained on the SAFE-GPT dataset using the Hugging Face Tokenizers library. Character-level tokenization will be used as a baseline for comparison. Relevant hyperparameters for each tokenization technique will be explored and tuned, including:

- BPE/WordPiece: Vocabulary size, merging criteria, dropout, vocabulary pruning.
- SentencePiece: Model type, vocabulary size, smoothing techniques.
- Unigram Language Model: Objective function parameters, regularization techniques.

Unigram Language Model Tokenization. The Unigram Language Model tokenizer can be implemented using the Hugging Face Tokenizers library or a custom implementation based on the algorithm described by Kudo [12]. During training, the unigram language model will learn to segment SAFE strings into subword units by optimizing an objective function that balances the likelihood of the training data and the length of the encoded sequences.

The impact of each tokenization technique on the model's performance will be evaluated using the metrics described in the Related Works section.

4.3 Model Architectures

We will compare the performance of three model architectures for molecular generation: SAFE-GPT, MAMBA, and Griffin. For each architecture, we will train and evaluate two versions of the model, a smaller version and a larger version, to assess their performance in resource-limited settings and when scaled up to more powerful configurations. The specific configurations for the small and large versions of each model were chosen to provide a fair comparison across architectures while considering the computational constraints of our study.

4.3.1 SAFE-GPT. SAFE-GPT [17] is a Transformer-based generative model designed for molecular design tasks, utilizing the Sequential Attachment-based Fragment Embedding (SAFE) representation. We will use the SAFE-GPT code from the original paper as a starting point for our implementation and train two versions of the model:

- SAFE-GPT-20M: 6 layers, 8 attention heads per layer, hidden state size of 768, 20M parameters.
- SAFE-GPT: 12 layers, 12 attention heads per layer, hidden state size of 768, 60M parameters.

4.3.2 MAMBA. MAMBA [8] is a State Space Model (SSM) designed for efficient and scalable sequence modeling tasks, incorporating selective state spaces into a simplified end-to-end neural network design (Figure 1). The input sequence is projected to a higher-dimensional space, passed through an SSM layer with a gating mechanism that selectively updates or retains information from the hidden states based on the current input, and then projected back to the input space. Residual connections and layer normalization are used to preserve information and improve stability. This process is repeated for a stack of identical MAMBA blocks, allowing the model to capture complex dynamics in the input sequence. However, MAMBA's performance in generating molecules, including its ability to capture the highly nonlinear and multimodal nature of molecular data, still needs to be assessed. We will use the MAMBA

code from the original paper as a starting point and train two versions of the model:

- MAMBA-Small: 6 MAMBA blocks, hidden state size of 512, 20M parameters.
- MAMBA-Large: 12 MAMBA blocks, hidden state size of 1024, 60M parameters.

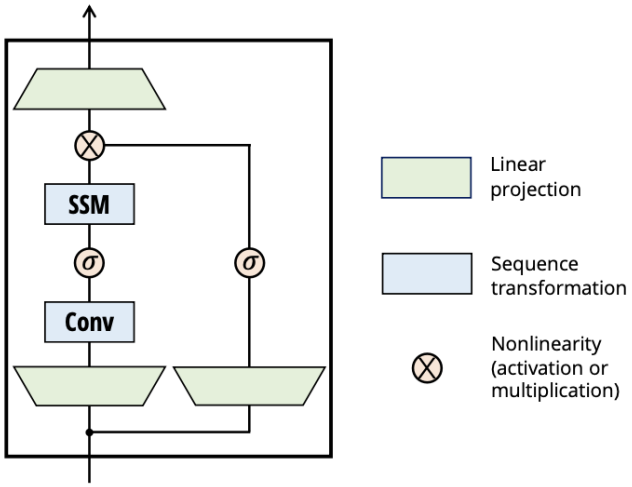

Figure 1: The MAMBA architecture, which incorporates selective state spaces into a simplified end-to-end neural network design [8].

4.3.3 Griffin. Griffin [2] is a hybrid model that combines SSMs with local attention to achieve a balance between efficiency and expressiveness in sequence modeling tasks (Figure 2). The input sequence undergoes a linear projection, followed by a series of alternating RG-LRU (Real Gated Linear Recurrent Unit) layers and local attention layers. RG-LRU layers extend the SSM layer used in MAMBA by adding input and recurrent gating mechanisms, allowing the model to selectively update or discard information based on the current input and historical context. Local attention layers operate on a fixed-size window around each position, capturing contextual information and dependencies within the local context. The output of the last RG-LRU or local attention layer is passed through a nonlinearity before being projected back to the original sequence space. This process is repeated for a specified number of Griffin blocks, enabling the model to capture both long-range dependencies and complex patterns in the data. Compared to SAFE-GPT, Griffin offers a more computationally efficient alternative for modeling long sequences due to the linear complexity of its SSM layers. However, the expressiveness of Griffin's local attention layers may be limited compared to the global self-attention in SAFE-GPT, which can capture dependencies across the entire sequence. The application of Griffin to molecular generation tasks remains an open research question, and challenges may arise in capturing the highly nonlinear and multi-modal nature of molecular data. As Griffin is a relatively new architecture, we will implement it from scratch based on the description provided in the original paper and then train two versions of the model:

- Griffin-Small: 6 RG-LRU layers, 2 local attention layers, hidden state size of 512, 20M parameters.
- Griffin-Large: 12 RG-LRU layers, 4 local attention layers, hidden state size of 1024, 60M parameters.

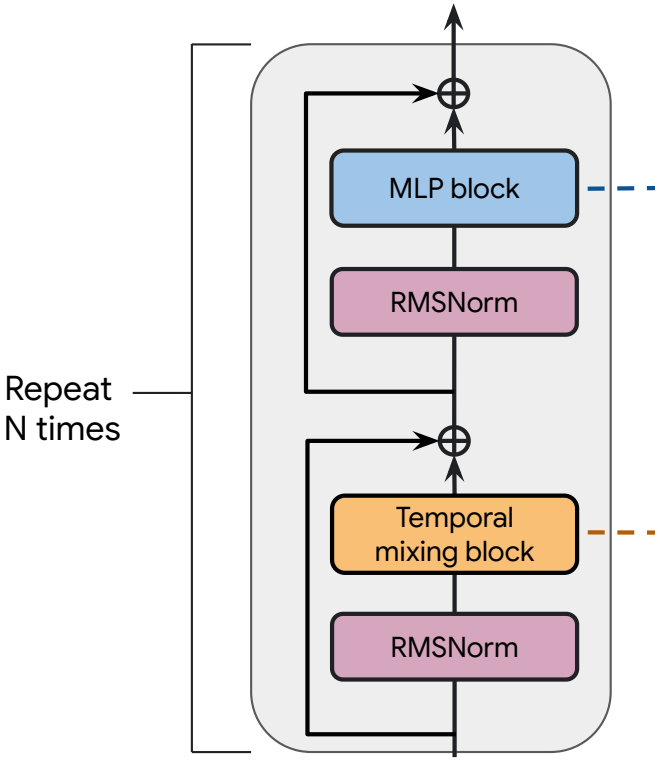

Figure 2: The Griffin architecture, which mixes gated linear recurrences (the RG-LRU layer) with local attention in a layered structure [2].

The smaller versions of each model will allow us to evaluate the performance of each architecture in a resource-limited setting, while the larger versions will assess the potential of each architecture when scaled up to a more powerful model.

4.4 Training Setup

All models will be trained using the UCT High Performance Cluster, with access to Nvidia A100 GPUs. The baseline SAFE-GPT model will be trained on the ZINC dataset for a sufficient number of epochs to ensure convergence, determined by monitoring the validation loss. We will use consistent loss functions, optimizer configurations (AdamW with learning rate 1e-4, batch size 128, weight decay 0.01), and early stopping based on validation loss for the baseline model. The trained baseline model will then be used as the starting point for the tokenization, architecture comparison, and decoder strategy experiments, with further fine-tuning and modifications as required for each sub-project.

If time permits, we intend to further fine-tune the models using reinforcement learning and the reward function specified in the SAFE paper [17]. The reward function is defined as follows:

$$\text{reward}(mol) = \frac{1}{1 + \alpha \cdot |\text{prop}(mol) - \text{target}|} \tag{7}$$

where:

- prop(mol) represents the calculated molecular property value for a given sample.
- target signifies the desired target value for the molecular property.
- α is a scaling factor, typically set to 0.5.

This reward function quantifies the similarity between the calculated molecular property value and the desired target value.

4.5 Decoding Strategies

We will evaluate the performance of the SAFE-GPT model using four decoding strategies: greedy decoding, beam search, nucleus sampling, and temperature sampling. These strategies will be implemented using the Hugging Face Transformers library, and their hyperparameters will be tuned based on validation performance. In addition to testing these strategies on the SAFE-GPT model trained on the SAFE-GPT dataset, we will also evaluate their performance on a SAFE-GPT model trained on the ZINC dataset [10] to assess their effectiveness in a domain-specific setting.

The hyperparameters for each decoder will be optimised and determined through a grid search method, in order to find the parameter values that maximize the generated output against our chosen metrics. Testing to compare the different decoders will then be done with the optimised hyper-parameterized decoders.

4.5.1 *Deterministic Decoders.*

Greedy Algorithm. The greedy algorithm is a simple and intuitive decoding strategy that selects the token with the highest probability at each step of the generation process [9, 26]. Given the previously generated tokens $x_{<i}$, the greedy algorithm chooses the token x_i that maximizes the conditional probability $P(x_i|x_{<i})$,

While the greedy algorithm is computationally efficient, it may lead to suboptimal solutions as it does not consider the long-term impact of its choices.

Beam Search. Beam search is a more sophisticated decoding strategy that maintains a set of W most probable partial sequences, called beams, at each step of the generation process. The algorithm starts with W empty beams and iteratively expands each beam by considering all possible next tokens. The W most probable sequences are then selected as the new set of beams for the next step [9, 26]. This process continues until a termination criterion is met, such as reaching a maximum sequence length or encountering a special end token. Beam search explores a larger portion of the search space compared to the greedy algorithm, but it still only considers a subset of all possible sequences.

4.5.2 *Stochastic Decoders.*

Temperature Sampling. Temperature sampling is a stochastic decoding strategy that introduces randomness into the token selection process by sampling from the next-token probability distribution. The temperature hyperparameter τ controls the flatness of the distribution, allowing for a trade-off between the quality and diversity of the generated molecules [19, 26]. A higher temperature leads to a more uniform distribution, encouraging the generation of diverse molecules, while a lower temperature focuses on high-probability tokens, resulting in molecules that are more likely to be valid and similar to the training data.

The probability of selecting a token x_i given the prefix $x_{<i}$ is given by:

$$P(x_i|x_{<i}) = \frac{e^{z_i/\tau}}{\sum_{j=1}^{V} e^{z_j/\tau}}$$

Nucleus Sampling. Nucleus sampling, also known as top-p sampling, is another stochastic decoding strategy that addresses the limitations of temperature sampling. Instead of sampling from the entire vocabulary, nucleus sampling dynamically selects a subset of the most probable tokens that account for a specified probability mass p [26]. This approach effectively truncates the tail of the probability distribution, eliminating the least likely tokens and reducing the chances of generating chemically invalid or nonsensical molecules.

Given a probability distribution $P(x|x_{1:i-1})$ over the vocabulary V, nucleus sampling first defines the top-p vocabulary $V^{(p)} \subseteq V$ as the smallest set of tokens such that:

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p.$$

$$P'(x|x_{1:i-1}) = \begin{cases} \frac{P(x|x_{1:i-1})}{p'} & \text{if } x \in V^{(p)}, \\ 0 & \text{otherwise.} \end{cases}$$

The next token is then sampled from the re-scaled distribution P'.

4.5.3 *Constraining generation.*

Constraining the decoder output is a promising approach to guide the generation of molecules with desired properties and ensure chemical validity [4, 21]. We will explore various constraining techniques that can be integrated with our selected decoding strategies, such as beam search and nucleus sampling. These methods include property-based constraints to generate molecules with specific physicochemical properties [22], substructure-based constraints to enforce the presence or absence of certain substructures [16], and syntactic constraints to ensure adherence to the rules of the molecular representation format. We will focus on the SAFE representation [17], which encodes molecules as a sequence of interconnected fragments, and compare its performance to other formats like SELFIES [11]. The choice of molecular representation can impact the effectiveness of constraining techniques and the quality of the generated molecules. We will evaluate these techniques using the mentioned evaluation metrics, assessing their impact on the quality, diversity, and relevance of the generated molecules, and refine our methods based on our findings and the characteristics of the decoding strategies.

4.6 Evaluation Metrics

The performance of the trained models will be evaluated using the metrics described in the Related Works section, including validity, uniqueness, novelty, diversity, Quantitative Estimate of Druglikeness (QED), and Synthetic Accessibility Score (SAS). These metrics

will assess the quality, diversity, and chemical feasibility of the generated molecules.

4.7 Computational Resources

All experiments will be conducted using the UCT High Performance Cluster, with access to Nvidia A100 GPUs. The training and evaluation of the models will be distributed across multiple GPUs to accelerate the experiments and enable the processing of large-scale datasets. Techniques like gradient accumulation, mixed-precision training, data parallelism, and gradient checkpointing will be employed to optimize memory usage and training efficiency.

The computational resources required for this study are moderate, given the reduced size of the ZINC dataset compared to the original SAFE-GPT dataset. The training of the baseline SAFE-GPT model on the 200 million molecule subset is estimated to require approximately 3-4 days on 1 Nvidia A100 GPU. The subsequent experiments on tokenization, architecture comparison, and decoder strategies will have varying computational requirements depending on the specific modifications and fine-tuning performed for each sub-project. We will optimize the use of available computational resources to ensure the timely completion of all experiments.

5 ETHICAL, PROFESSIONAL, AND LEGAL ISSUES

Our project will adhere to the ethical guidelines for open-source software development [7]. As our work does not involve human or animal subjects and does not include any privacy-breaching experiments or data collection, we do not anticipate any significant ethical concerns arising directly from the research.

The initial code and datasets used for training our models are open-source and available under permissive licenses. Any derivatives or modifications made to the original open-source components will also be released under compatible open-source licenses. We will ensure that all work is properly attributed, and any third-party resources will be used in accordance with their respective licenses. Upon completion of the project, the code developed by our team will be made publicly available under the Apache-2.0 license.

However, we acknowledge the potential downstream risks associated with the application of generative models for molecular design, such as the creation of harmful or illegal substances. To mitigate these risks, we emphasize that our work is intended solely for research purposes and should not be used for any illegal or unethical activities. We will include clear disclaimers in our code repositories and publications, stating that the generated molecules are hypothetical and have not been tested for safety or efficacy.

6 ANTICIPATED OUTCOMES

6.1 Research

6.1.1 SSM vs Transformer Architectures. We anticipate that SSM-based MAMBA and hybrid Griffin models will outperform the Transformer-based SAFE-GPT in computational efficiency and handling long SAFE sequences while maintaining comparable or better performance on validity, uniqueness, diversity, QED, and SAS metrics. These findings will provide insights into the strengths and limitations of different architectures for molecular generation.

6.1.2 Decoder Strategies. We aim to identify the decoding approach that yields the highest quality and diversity of generated molecules. We expect stochastic methods, particularly nucleus sampling with tuned hyper-parameters, to achieve a better trade-off between novelty, diversity, and chemical validity compared to deterministic methods.

6.1.3 Tokenization Techniques. We expect data-driven subword tokenization methods, such as byte pair encoding or transformer-based tokenizers, to outperform fixed rule-based methods in terms of downstream model performance, sample efficiency, and learning of meaningful molecular representations. These findings will contribute to the development of more expressive and efficient input representations for molecular generation models.

6.2 Impact

The project will provide valuable insights and resources for researchers and practitioners working on molecular generation tasks:

- The SSM vs Transformer architecture comparison will guide the selection of efficient and expressive models for handling long SAFE sequences in molecular generation.
- The evaluation of decoder strategies will inform the choice of decoding methods that balance novelty, diversity, and chemical validity in generated molecules.
- The analysis of tokenization techniques will contribute to the development of more effective input representations for molecular generation models.

Our findings and open-source code will facilitate further research and applications in drug discovery, materials design, and other fields relying on efficient exploration of chemical space.

6.3 Key Success Factors

(1) SSM vs Transformer Architectures: Training and evaluating SAFE-GPT, MAMBA, and Griffin architectures in both large and small model configurations, achieving competitive performance on molecular generation metrics, identifying strengths and weaknesses of each architecture, and open-sourcing trained models and code.
(2) Decoder Strategies: Comprehensive evaluation of maximization-based and stochastic decoder strategies, identifying the approach that achieves the best trade-off between novelty, diversity, and chemical validity of generated molecules.
(3) Tokenization Techniques: Comparative analysis of various tokenization techniques, assessing their impact on downstream model performance, sample efficiency, and learning of meaningful molecular representations, and identifying the most effective and efficient tokenization method.

7 PROJECT PLAN

7.1 Risks and Risk Management Strategies

Our project faces several potential risks, which we have carefully considered and developed appropriate mitigation strategies for. These risks, along with their likelihood, impact, and management approaches, are detailed in Table 1.

7.2 Timeline and Gantt Chart

The project timeline, including key deliverables and milestones, is illustrated in the Gantt chart (Figure 3).

7.3 Resources Required

In addition to those already mentioned, we'll use the following resources as well:

- Open-source software libraries, including PyTorch and RDKit
- Hugging Face Decoders[1]
- Hugging Face Tokenizers[2]

7.4 Deliverables and Milestones

The key deliverables and milestones for this project are outlined in Table 2. This table also includes the expected due dates for each deliverable and its associated subtasks.

7.5 Work Allocation

The project team will contribute equally to the development of the project web page and collaborate on building and training the initial SAFE-GPT model. Following this, the team members will focus on their respective areas of specialization:

- Mahomed Aadil Ally will develop or adapt the code for various tokenization techniques and retrain the SAFE-GPT model accordingly with each tokenization strategy.
- Anri Lombard will code and modify the MAMBA and Griffin models and train them to compare architectures.
- Gabriel Marcus will work on decoder strategies and perform inference using them alongside the trained SAFE-GPT model and a smaller model trained on the Zinc dataset.

REFERENCES

[1] G Richard Bickerton, Gaia V Paolini, J'er'emy Besnard, Sorel Muresan, and Andrew L Hopkins. 2012. Quantifying the chemical beauty of drugs. *Nature chemistry* 4, 2 (2012), 90–98.

[2] Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. 2024. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models. *arXiv preprint arXiv:2402.19427* (2024).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] Daniel C Elton, Zois Boukouvalas, Mark D Fuge, and Peter W Chung. 2019. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design Engineering* 4, 4 (2019), 828–849.

[5] Peter Ertl and Ansgar Schuffenhauer. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics* 1, 1 (2009), 1–11.

[6] Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin H. S. Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. 2020. Molecular representation learning with language models and domain-relevant auxiliary tasks. *CoRR* abs/2011.13230 (2020).

[7] Frances S Grodzinsky, Keith Miller, and Marty J Wolf. 2003. Ethical issues in open source software. *Journal of Information, Communication and Ethics in Society* 1, 4 (2003), 193–205.

[8] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752 [cs.LG]

[9] David Ippolito, Rene Kriz, Masha Kustikova, João Sedoc, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. *arXiv preprint arXiv:1906.06362* (2019).

[10] John Irwin and Brian Shoichet. 2005. ZINC A Free Database of Commercially Available Compounds for Virtual Screening. *Journal of chemical information and modeling* 45 (04 2005), 177–82. https://doi.org/10.1021/ci049714+

[11] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. 2020.

[12] Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *CoRR* abs/1804.10959 (2018).

[13] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *CoRR* abs/1808.06226 (2018).

[14] Greg Landrum et al. 2023. RDKit: Open-source cheminformatics. *Online. http://www. rdkit. org* (2023).

[15] Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. 1997. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews* 23, 1-3 (1997), 3–25.

[16] Łukasz Ma'ziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. 2020. Molecule attention transformer. *arXiv preprint arXiv:2002.08264* (2020).

[17] Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan SC Lim, and Prudencio Tossou. 2023. Gotta be SAFE: A New Framework for Molecular Design. *arXiv preprint arXiv:2310.10773* (2023).

[18] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. 2020. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. arXiv:1811.12823 [cs.LG]

[19] M. Renze and E. Guven. 2024. The Effect of Sampling Temperature on Problem Solving in Large Language Models. (2024). arXiv:2402.05201 [cs.CL]

[20] David Rogers and Mathew Hahn. 2010. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling* 50, 5 (2010), 742–754.

[21] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. 2018. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* 361, 6400 (2018), 360–365.

[22] Petra Schneider, W Patrick Walters, Alleyn T Plowright, Norman Sieroka, Jennifer Listgarten, Robert A Goodnow, Johanna Fisher, J"org M Jansen, Jos'e S Duca, Thomas S Rush, et al. 2020. *Nature Reviews Drug Discovery* 19, 5 (2020), 353–364.

[23] Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5149–5152. https://doi.org/10.1109/ICASSP.2012.6289079

[24] Philippe Schwaller, Teodoro Laino, Th'eophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. 2019. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. In *ACS central science*, Vol. 5. ACS Publications, 1572–1583.

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *CoRR* abs/1508.07909 (2015).

[26] Chuan Shi, Han Yang, Deng Cai, Zhan Zhang, Yan Wang, Yiming Yang, and Wai Lam. 2024. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925* (2024).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[28] David Weininger. 1988. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* 28, 1 (1988), 31–36.

[1] Accessed on April 16, 2024. Available at: https://huggingface.co/docs/transformers/en/main_classes/text_generation

[2] Accessed on April 16, 2024. Available at: https://huggingface.co/docs/tokenizers/index

A RISKS AND MITIGATION STRATEGIES

Risk	Likelihood	Impact	Mitigation Strategies
No access to a device that supports CUDA	High	Medium	- Access to UCT's computer facilities that support CUDA
Run out of HPC time	Medium	Medium	- Train the largest models first to focus on smaller, less compute intensive models as time runs out - Stick to <200M models
Difficulty in implementing and training the models	Medium	High	- Allocate sufficient time for model development and troubleshooting - Use pre-trained models or existing implementations as a starting point if needed
Insufficient or low-quality training data	Low	High	- Use a large, diverse, and well-curated dataset (SAFE-GPT)
Overfitting or poor generalization	Medium	Medium	- Use appropriate regularization techniques (e.g., dropout, weight decay) - Employ cross-validation and hold-out validation sets - Monitor model performance on unseen data during training
Misinterpretation or lack of interpretability	Medium	Medium	- Use explainable AI techniques (e.g., attention visualization, feature importance) - Collaborate with domain experts at InstaDeep to validate interpretations where needed
Failure to meet performance expectations	Low	High	- Set realistic performance goals based on prior research and benchmarks - Regularly monitor and evaluate model performance - Be prepared to iterate and refine models as needed
Delays due to unforeseen technical challenges	Medium	Medium	- Build in buffer time for unexpected issues - Communicate progress and challenges with supervisors and collaborators weekly
Lack of reproducibility or difficulty in sharing code	Low	Low	- Use version control (e.g., Git and GitHub) and document code changes
Research partner becomes unavailable	Low	High	- Establish clear communication channels and expectations among research partners - Document individual contributions and share knowledge regularly with weekly meetings - Have a contingency plan for redistributing work or seeking additional support
Supervisor becomes unavailable due to responsibilities piling up	Low	High	- Schedule regular meetings and updates with the supervisor - Be proactive in seeking guidance and feedback - Establish clear communication channels in case in-person meetings become difficult

Table 1: Risk Assessment

B DELIVERABLES AND DUE DATES

Deliverable	**Due Date**
Literature Review	18 Mar 2024
- Conduct a comprehensive survey of relevant literature	1 Mar 2024
- Synthesize findings and identify research gaps	10 Mar 2024
- Write and revise the literature review	18 Mar 2024
Project Proposal Presentation	22-25 Apr 2024
- Prepare presentation slides	18 Apr 2024
- Rehearse and refine the presentation	20 Apr 2024
- Deliver the project proposal presentation	22-25 Apr 2024
Project Proposal	30 Apr 2024
- Outline the project proposal structure	9 Apr 2024
- Write and revise individual sections	20 Apr 2024
- Integrate sections for first draft	22 Apr 2024
- Finalize the proposal	30 Apr 2024
Project Progress Demonstration	22-26 Jul 2024
- Implement and train models	30 Jun 2024
- Evaluate and document model performance	15 Jul 2024
- Prepare progress demonstration materials	20 Jul 2024
- Deliver the project progress demonstration	22-26 Jul 2024
Complete Draft of Final Paper	23 Aug 2024
- Additional experimentation to address demonstration feedback	6 Aug 2024
- Outline and write individual papers	18 Aug 2024
- Complete the drafts	23 Aug 2024
Final Paper Submission	30 Aug 2024
- Incorporate feedback from the draft	29 Aug 2024
- Submit the final paper	30 Aug 2024
Project Code Final Submission	9 Sept 2024
- Clean and document the project codebases	7 Sept 2024
- Prepare README files and usage instructions	8 Sept 2024
- Submit the final project code	9 Sept 2024
Final Project Demonstration	16-20 Sept 2024
- Prepare demonstration materials and slides	15 Sept 2024
- Deliver the final project demonstration	16-20 Sept 2024
Project Poster	27 Sept 2024
- Design the project poster layout	15 Sept 2024
- Create poster content and visualizations	26 Sept 2024
- Print and submit the project poster	27 Sept 2024
Project Web Page	4 Oct 2024
- Design the project web page layout	15 Sept 2024
- Develop and test the web page	30 Sept 2024
- Finalize web page	3 Oct 2024
- Deploy and submit the project web page	4 Oct 2024

Table 2: Deliverables, Subtasks, and Due Dates

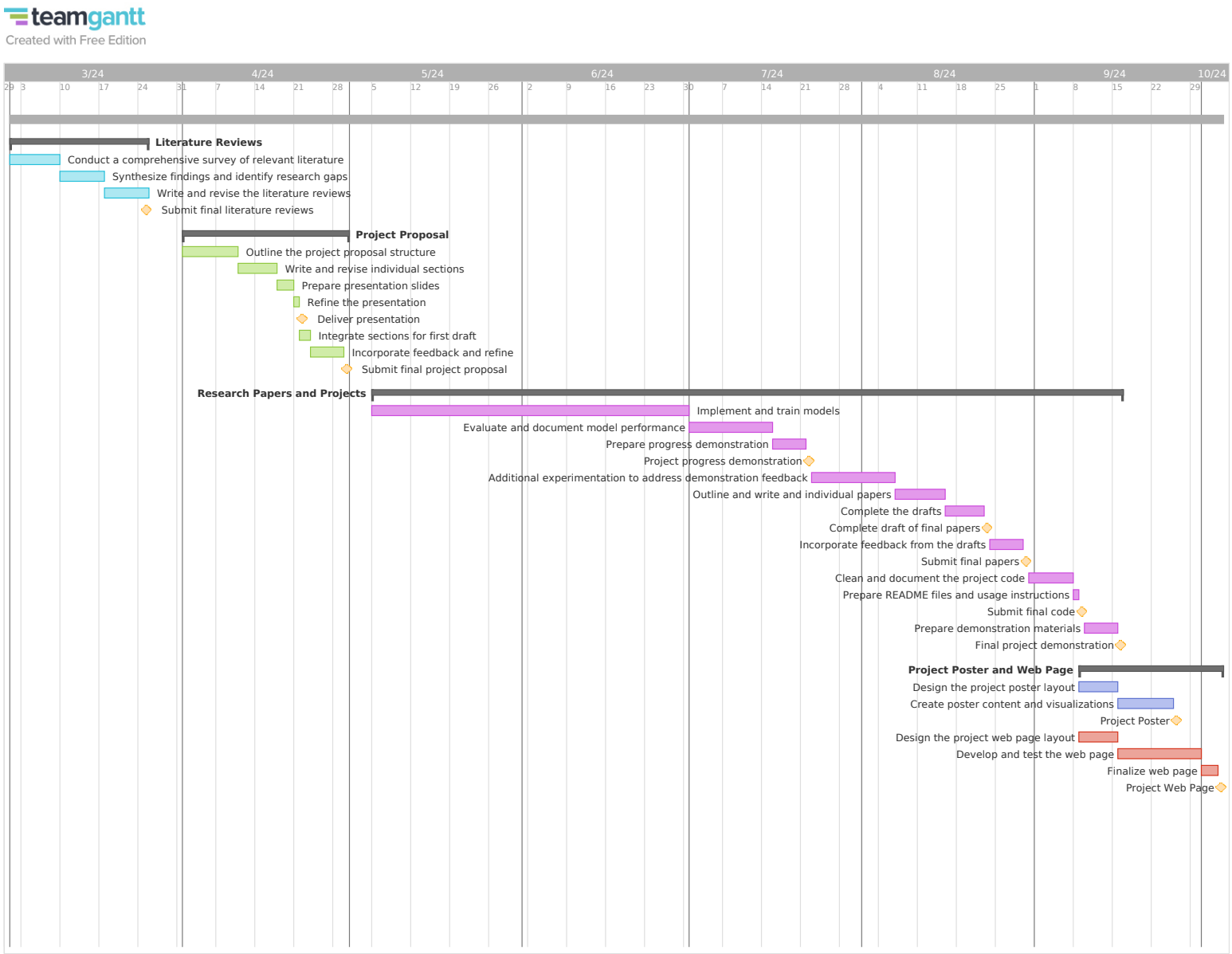

Figure 3: Gantt chart illustrating the project timeline, deliverables, and milestones.