UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SÃO CARLOS

Thiago César Silva Barbieri

PROPOSTA DE UMA GAN PARA GERAÇÃO DE MOLÉCULAS

SÃO CARLOS
2024

THIAGO CÉSAR SILVA BARBIERI

PROPOSTA DE UMA GAN PARA GERAÇÃO DE MOLÉCULAS

Trabalho de Conclusão de Curso submetido à Universidade Federal de São Carlos, como requisito necessário para obtenção do grau de Bacharel em Engenharia de Computação.
Orientador: Prof. Dr. Ricardo Cerri

São Carlos, fevereiro de 2024

UNIVERSIDADE FEDERAL DE SÃO CARLOS

THIAGO CÉSAR SILVA BARBIERI

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Engenharia de Computação, sendo aprovada em sua forma final pela banca examinadora:

Orientador(a): Prof. Dr. Ricardo Cerri
Universidade Federal de São Carlos - UFSCar

Prof. Dr. Alan Demétrius Baria Valejo
Universidade Federal de São Carlos - UFSCar

Prof. Dra. Helena de Medeiros Caseli
Universidade Federal de São Carlos - UFSCar

São Carlos, 30 de janeiro de 2024

Agradecimentos

Gostaria de expressar minha sincera gratidão a diversas pessoas que desempenharam papéis cruciais na jornada de elaboração deste trabalho de conclusão de curso. Em primeiro lugar, um agradecimento especial ao meu amigo Thales, cuja colaboração e apoio foram fundamentais na estruturação e desenvolvimento deste trabalho. Não posso deixar de mencionar o Thiago, do BioMal, que se disponibilizou a me auxiliar na resolução de desafios específicos encontrados durante o desenvolvimento deste projeto. Sua disposição em oferecer assistência e compartilhar seu conhecimento foi essencial para superarmos obstáculos técnicos. Agradeço também aos amigos que, com palavras de incentivo e apoio, motivaram-me a perseverar em momentos desafiadores. Suas encorajadoras mensagens foram uma fonte constante de inspiração e reforçaram meu compromisso em concluir este trabalho com excelência. Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste projeto. A jornada foi desafiadora, mas com o apoio e colaboração de indivíduos incríveis como vocês, foi possível alcançar este marco. Muito obrigado por fazerem parte deste trajeto.

Resumo

No campo da biologia, as moléculas desempenham um papel crucial, agindo como os blocos de construção fundamentais que compõem tudo ao nosso redor, desde elementos como água e ar até as intricadas estruturas que formam nosso próprio corpo. Cada molécula é formada por átomos, que representam os elementos fundamentais da química. A síntese de moléculas é um processo fascinante que implica na criação de novas configurações de átomos para originar moléculas específicas. Essa abordagem é de extrema relevância, uma vez que diferentes moléculas exibem propriedades distintas. Algumas dessas moléculas desempenham um papel essencial no tratamento de doenças, enquanto outras podem ser incorporadas em materiais resistentes ou transformadas em fontes de energia. É evidente a crescente interconectividade entre estudos médicos e a computação para construção de ferramentas eficientes e precisas da inovação farmacológica. Esse trabalho propõe um modelo generativo que utiliza de estratégias de Ensemble Learning aplicadas em uma cGAN (Rede Adversarial Generativa Condicional) para geração de novos compostos químicocandidatos a medicamentos. A abordagem proposta tem várias aplicações potenciais na descoberta de medicamentos, incluindo a identificação de novas terapêuticas e a otimização de compostos principais, e abre margem para pesquisas futuras.

Palavras-chave: Ensemble Learning, Modelo condicional, GAN (Rede Adversarial Generativa), Variational Autoencoders (VAE), Modelos generativos, Geração de moléculas, Características específicas.

Abstract

In the field of biology, molecules play a crucial role, acting as the fundamental building blocks that make up everything around us, from elements like water and air to the intricate structures that make up our own bodies. Each molecule is made up of atoms, which represent the fundamental elements of chemistry. Molecule synthesis is a fascinating process that involves creating new configurations of atoms to create specific molecules. This approach is extremely relevant, since different molecules exhibit different properties. Some of these molecules play an essential role in treating diseases, while others can be incorporated into resistant materials or transformed into energy sources. The growing interconnectivity between medical studies and information technology for the construction of efficient and precise tools for pharmacological innovation is evident. This work proposes a generative model that uses Ensemble Learning strategies applied in a cGAN (Conditional Generative Adversarial Network) to generate new candidate chemical compounds for medicines. The proposed approach has several potential applications in drug discovery, including identifying new therapeutics and optimizing lead compounds, and opens scope for future research.

Keywords: Ensemble Learning, Conditional model, GAN (Generative Adversarial Network), Variational Autoencoders (VAE), Generative models, Molecule generation, Specific characteristics.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Adam: Adaptive Moment Estimation

CHEBI: Chemical Entities of Biological Interest

D: Discriminador

DNNs: Deep Neural Networks

Ensemble: Ensemble Learning

FCD: Fréchet ChemNet Distance

G: Gerador

GANs: Generative Adversarial Networks

KL: Kullback-Leibler

QED11: Estimativa Quantitativa da Semelhança com Drogas

RAM: Random Access Memory

RDKit: Cheminformatics software

ReLu: Rectified Linear Unit

RMSprop: Root Mean Square Propagation

RNNs: Recurrent Neural Networks

RO5: Rule of Five

SGD: Stochastic Gradient Descent

SMILES: Simplified Molecular Input Line Entry System

SN: Spectral Normalization

VAEs: Variational Autoencoders

cVAE: Conditional Variational Autoencoder

logP: Logaritmo do Coeficiente de Partição

Sumário

1 Introdução

No mundo da biologia, as moléculas desempenham um papel fundamental, atuando como as unidades básicas de construção de tudo que nos rodeia, desde elementos como água e ar até as complexas estruturas que constituem nosso próprio corpo. Cada molécula é composta por átomos, que são os blocos fundamentais da química [Alberts et al. 2010].

A síntese de moléculas é um processo que envolve a criação de novas configurações de átomos para formar moléculas específicas. Essa abordagem é de grande importância, pois diferentes moléculas possuem propriedades distintas [Vogel et al. 1996]. Algumas dessas moléculas podem desempenhar um papel vital no tratamento de doenças, enquanto outras podem ser incorporadas em materiais resistentes ou serem convertidas em fontes de energia. 1

O principal papel das estruturas moleculares abordados nesta pesquisa é a sua aplicação no desenvolvimento de medicamentos. Para que uma molécula seja candidata ao desenvolvimento de novas terapêuticas, ela precisa cumprir com padrões estruturais complexos e pouco conhecidos pela ciência. O conceito de “druglikeness”, por exemplo, foi proposto para fornecer diretrizes úteis durante os estágios iniciais da descoberta de medicamentos para aumentar a chance de um produto químico entrar e passar nos ensaios clínicos [Ursu et al. 2011]. Entretanto, tais estratégias não são suficientes para a síntese de novas estruturas moleculares.

Convencionalmente, este processo depende da experiência humana para propor, sintetizar e testar novas moléculas. Embora muitas das descobertas mais importantes da história científica tenham sido feitas desta forma, a descoberta convencional é inerentemente

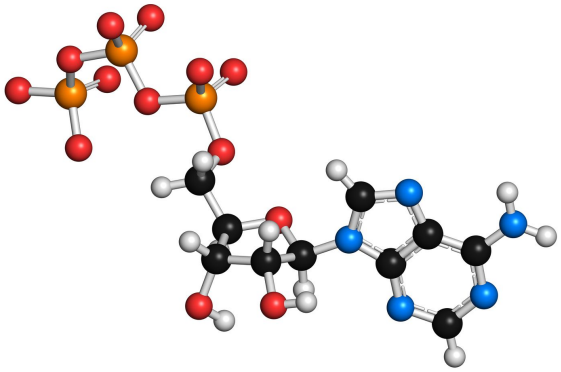

Figura 1 – Figura ilustrativa de uma molécula - "Quebrar as ligações que unem os grupos fosfato ao ATP libera energia."MOLEKUUL/SCIENCE PHOTO LIBRARY / Getty Images

dispendiosa e demorada, limitando o número e a diversidade de moléculas que podem ser razoavelmente exploradas. Foi estimado que o número de compostos já sintetizados gira em torno de 10^8, enquanto o número total de compostos teoricamente viáveis fica entre 10^{23} e 10^{60} [Polishchuk, Madzhidov e Varnek 2013, Reymond e Awale 2012]. Assim, os métodos convencionais de descoberta só são capazes de explorar uma pequena fração do espaço químico.

Nos últimos tempos, a interconectividade entre estudos médicos e a computação vem tendo implicações substanciais para a crescente eficiência e precisão da inovação farmacológica, introduzindo oportunidades para criar novos tratamentos a um custo reduzido [Martinelli 2022]. Além dos conhecidos métodos de regressão, classificação e agrupamento, bastante conhecidos há muito tempo na área de Aprendizado de Máquina, nos últimos tempos há uma crescente popularidade no desenvolvimento de métodos de inteligência artificial generativas, que tentam gerar dados que seguem os padrões das versões reais desses dados. Neste panorama, surge o presente estudo, que propõe o uso de uma Ensemble Generative Adversarial Network condicional, que visa propor novas estruturas moleculares com potencial farmacêutico.

1.1 Organização do Documento

O presente trabalho está organizado da seguinte forma:

Capítulo 2: apresenta os conceitos relacionados à geração de moléculas a partir de modelos de inteligência artificial generativos, bem como estratégias de manipulação dos dados utilizados.

Capítulo 3: descreve os principais recursos e motodologias utilizados, assim como as etapas necessárias no desenvolvimento do modelo, desde o pré-processamento dos dados até o desenvolvimento da arquitetura da GAN desenvolvida.

Capítulo 4: explica a metodologia utilizada na experimentação do modelo e aponta observações importantes extraídas das métricas geradas.

Capítulo 5: aponta as principais dificuldades e resultados dos experimentos até o momento e aponta possibilidades de próximos passos para que maiores avanços sejam alcançados.

Capítulo 6: apresenta métricas geradas a partir de experimentos feitos com o modelo.

2 Fundamentação Teórica

Para melhor entendimento dos principais conceitos abordados no desenvolvimento do estudo em questão, é fornecido um resumo da terminologia aplicada em química computacional e aprendizado de máquina generativo.

2.1 Representação molecular

A força das redes neurais reside na sua capacidade de absorver uma representação de entrada complexa e transformá-la em uma representação latente necessária para resolver uma tarefa específica. Desta forma, a escolha da representação de entrada desempenha um papel fundamental na forma como o modelo aprende informações sobre a molécula. As representações de entrada geralmente se enquadram em uma de três categorias: baseadas em strings, grafos bidimensionais (2D) e grafos tridimensionais (3D) [Yao e Cao 2021].

2.1.1 Representações Baseadas em Strings

O Sistema Simplificado de Entrada em Linha para Moléculas (SMILES), concebido por David Weininger, é uma representação unidimensional que codifica moléculas como strings [Weininger 1988], como mostrado na Figura 2. Inspirado na teoria de grafos moleculares, o SMILES fornece informações sobre átomos, ligações, ramificações e estruturas cíclicas. Uma string SMILES canônica não inclui informações sobre isomerismo e estereoisomeria, diferentemente das strings SMILES isoméricas. Derivativos do sistema SMILES, como o SELFIES, foram desenvolvidos para mitigar problemas de inviabilidade aleatória nas strings SMILES [Krenn et al. 2020]. As strings SELFIES são robustas, tornando-as adequadas para processamento por modelos de aprendizado de máquina, especialmente algoritmos generativos [Krenn et al. 2020].

As impressões digitais moleculares binárias, como as chaves MACCS (Molecular Access System), também representam moléculas de maneira eficaz, sendo strings longas de valores binários [Cereto-Massagué e al. 2015]. As chaves MACCS podem descrever características e estruturas sem uma carga computacional elevada ou complexidade excessiva, sendo relevantes em contextos farmacológicos [Rogers e Hahn 2010, Ward e Beswick 2014].

2.1.2 Grafos Bidimensionais (2D)

Grafos moleculares bidimensionais consistem em conjuntos de nós (V) e arestas (E), representando átomos e ligações, respectivamente [David e al. 2020]. Uma matriz de adjacências indica a posição de um átomo em um composto, enquanto matrizes de

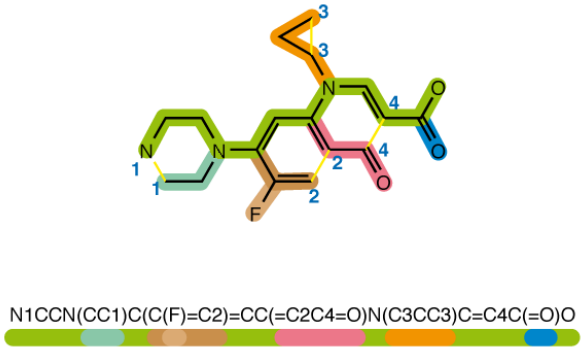

Figura 2 – Ciprofloxacina escrita no formato de SMILES [CreativeCommons]

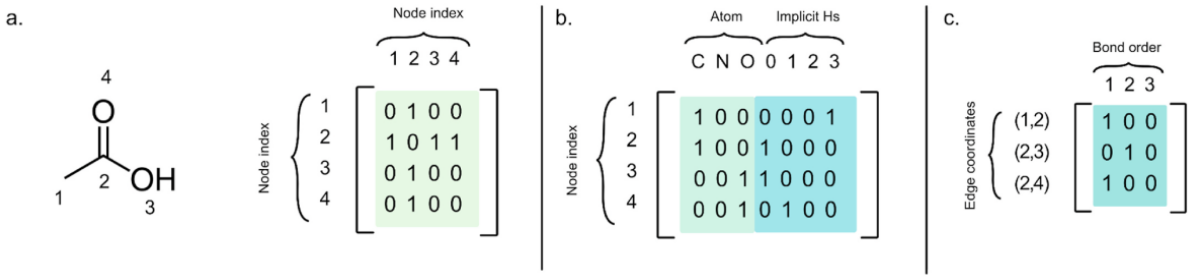

Figura 3 – Ácido acético codificado com (a) uma matriz de adjacências, (b) uma matriz de características de nó com informações sobre os tipos de átomos e o número de átomos de hidrogênio implícitos e (c) uma matriz de características de borda representando as ordens de ligação. [Martinelli 2022]

características de aresta e nó representam ligações e átomos. Essas matrizes podem ser codificadas como "one-hot" ou, para moléculas menos complexas, como matrizes codificadas por inteiros. Esses elementos podem ser visualizados na Figura 3.

2.1.3 Grafos Tridimensionais (3D)

Diversas variações de grafos tridimensionais para moléculas têm sido propostas para uso em algoritmos de aprendizado de máquina, mantendo inspiração na teoria de grafos moleculares. Matrizes diagonais podem descrever ângulos de ligação, comprimentos de ligação e ângulos diédricos [Estrada 2000]. Modelos adicionais incorporam matrizes de características, adjacência e posição relativa, integrando conhecimento existente sobre polaridade de ligação, eletronegatividade e conformação 3D para construir grafos adequados para redes convolucionais de grafos [Cho e Choi 2018]. Alternativamente, as coordenadas explícitas de cada átomo no espaço químico podem ser definidas [Liu e al. 2021].

2.2 Tokenização de SMILES

A tokenização é uma técnica que consiste na divisão de textos em unidades discretas chamadas tokens, que podem ser palavras, caracteres, ou símbolos, dependendo do contexto. A importância da tokenização reside no fato de que ela transforma dados textuais em formatos que podem ser processados e compreendidos por algoritmos de aprendizado de máquina e processamento de linguagem natural. Ao dividir o texto em tokens, torna-se possível realizar análises mais precisas, extrair informações relevantes e alimentar modelos de inteligência artificial, facilitando a interpretação e manipulação eficiente de dados textuais.

De todas as opções de codificação molecular, as strings SMILES canônicas são as mais frequentes. O uso do SMILES enquadra o problema da geração de moléculas como uma tarefa de processamento textual. Como tal, as strings são geralmente codificadas one-hot ou codificadas por inteiro, de modo que cada átomo ou ligação em uma string pudesse ser formatado como uma matriz [Wang et al. 2022].

Além disso, há também abordagens baseadas em fragmentos, que restringem a geração de novos compostos para compreender subestruturas relevantes conhecidas, como as da literatura de química medicinal. Os esquemas de fragmentação desconstroem moléculas usando regras simples (por exemplo, todas as ligações simples acíclicas) ou desconexões inspiradas retrossinteticamente [Lewell et al. 1998]. Uma biblioteca de fragmentos contendo um ou mais átomos pode então ser usada para construir novas moléculas.

Os dois métodos de tokenização citados foram desenvolvidos para aplicação de testes no projeto em questão. Para isso, foi implementada uma classe para codificação e decodificação dos SMILES, que cria um vocabulário específico de acordo com os SMILES que estão sendo carregados ao projeto. Um exemplo de como os elementos dos vocabulários de tokenização são construídos é apresentado na Figura 4.

2.3 Taxonomia CHEBI

A taxonomia CHEBI, ou Chemical Entities of Biological Interest, é um dicionário de entidades moleculares focado em compostos químicos "pequenos". O termo "entidade molecular" refere-se a qualquer átomo, molécula, íon, par de íons, radical, íon radical, complexo, conformero, etc., identificável como uma entidade separadamente distinguível. Essas entidades moleculares podem ser produtos naturais ou produtos sintéticos utilizados para intervir nos processos de organismos vivos.

Uma das características da CHEBI é a sua classificação ontológica, que especifica as relações entre as entidades moleculares ou classes (grupos) de entidades e seus pais e/ou filhos. Além disso, a CHEBI utiliza nomenclatura, simbolismo e terminologia endossados

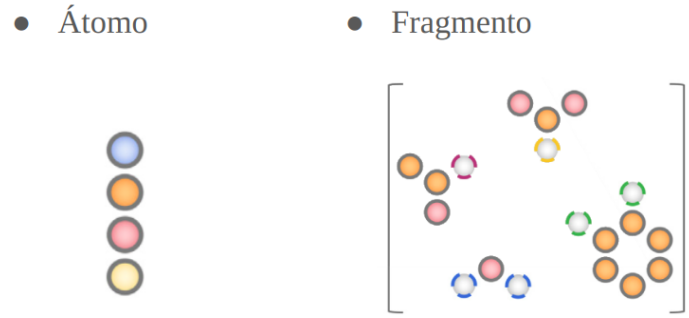

Figura 4 – Ilustração dos vocabulários de representação molecular baseados em átomos e fragmentos. A abordagem baseada em átomos é apoiada por um vocabulário contendo um pequeno número de átomos e ligações. A abordagem baseada em fragmentos é apoiada por um esquema de fragmentação e um conjunto de fragmentos intercambiáveis; átomos cinza indicam pontos de fixação anotados com um tipo de desconexão [Riniker e Landrum 2021].

por organizações científicas internacionais como a União Internacional de Química Pura e Aplicada (IUPAC) e o Comitê de Nomenclatura da União Internacional de Bioquímica e Biologia Molecular (NC-IUBMB) [Hastings et al. 2016].

Os grupos CHEBI são reconhecidos através de seus IDs, que são constituídos de 5 dígitos. Alguns exemplos de grupos CHEBI bastante conhecidos dentre os medicamentos, por exemplo, são:

Tabela 1 – Exemplos de grupos CHEBI

ID CHEBI	**Nome CHEBI**	**Descrição**
35480	Analgésicos	Agente capaz de aliviar a dor sem perda de consciência ou sem produzir anestesia. Além disso, o analgésico é um papel desempenhado por um composto que apresenta a capacidade de causar uma redução dos sintomas de dor.
33281	Antibióticos	Substância que mata ou retarda o crescimento de microorganismos, incluindo bactérias, vírus, fungos e protozoários.
35469	Antidepressivos	Os antidepressivos são drogas estimulantes do humor usadas principalmente no tratamento de transtornos afetivos e condições relacionadas.
35476	Antipsicóticos	Os medicamentos antipsicóticos são agentes que controlam o comportamento psicótico agitado, aliviam estados psicóticos agudos, reduzem os sintomas psicóticos e exercem um efeito calmante.
35472	Medicamentos antiinflamatórios	Uma substância que reduz ou suprime a inflamação.

2.4 Otimizadores

Os otimizadores são algoritmos responsáveis por ajustar os parâmetros de um modelo de forma a minimizar ou maximizar uma função de perda, também conhecida como função objetivo. Essa função quantifica o quão bem o modelo está performando em relação aos dados de treinamento. Os otimizadores são essenciais para atualizar os pesos do modelo durante o treinamento com o objetivo de melhorar gradualmente o desempenho do modelo e torná-lo capaz de tomar decisões mais precisas.

Para os testes de desenvolvimento do modelo proposto neste trabalho, alguns otimizadores foram selecionados por conta de suas características específicas:

RMSprop: O RMSprop [Tieleman e Hinton 2012] é uma escolha sólida de otimizador devido à sua capacidade de se adaptar às diferentes taxas de aprendizado para cada parâmetro da rede. Ele mantém uma média móvel das magnitudes dos gradientes recentes, ajustando as taxas de aprendizado com base nessas informações. Isso é especialmente útil quando lidamos com problemas em que as escalas dos gradientes variam amplamente. O RMSprop ajuda a superar desafios associados a taxas de aprendizado fixas, proporcionando uma convergência mais estável.

Adam: É um dos métodos mais populares para treinamento de redes neurais atualmente, e é considerado um dos otimizadores mais robustos e versáteis. Ele é conhecido por funcionar bem em uma variedade de tarefas e por ser relativamente insensível à escolha de hiperparâmetros. O artigo que propôs o Adam [Kingma e Ba 2014] alcançou um grande sucesso em termos de popularidade. Ele é a combinação de RMSProp e também da estratégia de momentum [Sun 2020].

SGD: Algoritmos de otimização adaptativos, como Adam e RMSprop, apresentaram melhor desempenho de otimização do que o gradiente descendente estocástico (SGD) em alguns cenários. No entanto, estudos recentes mostram que muitas vezes levam a um pior desempenho de generalização do que o SGD, especialmente para o treinamento de redes neurais profundas (DNNs) [Zhang 2018]. A escolha do SGD surge então como um meio de evitar que o modelo vicie em gerar moléculas repetitivas. Além disso, o SGD pode ser eficaz quando combinado com técnicas como momentum. Sua simplicidade o torna eficiente em termos computacionais, sendo útil em cenários onde recursos de hardware são limitados.

Adamax: O Adamax é uma variante do Adam, mas com uma norma infinita (máximo) em vez da norma L^2 para normalizar os gradientes. Essa escolha pode ser benéfica em situações em que a norma L^2 pode não ser apropriada. O Adamax, assim como o Adam, é conhecido por seu desempenho robusto em uma ampla gama de cenários e por sua eficácia em lidar com problemas de otimização não estacionários [Kingma e Ba 2017].

Além desses otimizadores, algumas abordagens já foram feitas a partir de otimizadores que não dependem de gradientes. Métodos metaheurísticos ("livres de gradiente") para design de moléculas usam procedimentos de otimização estocástica baseados em população para navegar no espaço químico [Riniker e Landrum 2021], como Algoritmos Evolutivos [Brown et al. 2004] ou Otimização por Enxame de Partículas [Reutlinger et al. 2014].

2.5 Modelos de inteligência artificial generativos

Recentemente, modelos mais avançados de aprendizado de máquina concebidos para gerar novas informações vêm sendo propostos, entre eles os que usam Recurrent Neural Networks (RNNs) [Rumelhart et al. 1986], Generative Adversarial Networks (GANs) [Goodfellow et al. 2014] e Variatonal Autoencoders (VAEs) [Kingma e Welling 2013]. Essas técnicas têm sido relacionadas a uma ampla gama de problemas, incluindo geração de texto com sentimentos, composição musical, geração e restauração de imagens, predição de trajetória, entre muitos outros [Wang e Wan 2019, Lopez-Rincon et al. 2018, Yu et al. 2018, Ivanovic et al. 2021]. Na literatura especializada em geração de moléculas, destaca-se uma frequência significativa das arquiteturas LSTM-RNN (Redes Neurais Recorrentes de Memória de Longo Prazo), GAN (Redes Generativas Adversariais) e VAE (Variational Autoencoders). Essas estruturas têm sido amplamente exploradas no contexto da descoberta de novas moléculas devido à sua capacidade de capturar padrões complexos em conjuntos de dados químicos e de criar representações latentes que podem ser manipuladas para gerar novas entidades químicas [Martinelli 2022].

2.5.1 LSTM-RNN

As Redes Neurais Recorrentes de Memória de Longo e Curto Prazo (LSTM-RNNs) são particularmente eficientes em tarefas de processamento de linguagem natural e previsão de séries temporais, devido à sua habilidade única de reter informações por extensos períodos, minimizando problemas como o decremento ou explosão de gradientes, um problema comum nas redes neurais recorrentes convencionais. A arquitetura das LSTM-RNNs é composta por células de memória dotadas de portões de esquecimento, entrada e saída, facilitando a gestão inteligente das informações ao longo do tempo. Esta estrutura permite que as LSTMs aprendam padrões complexos em dados sequenciais, permitindo a geração de novas sequências com base nas aprendizagens anteriores [Hochreiter e Schmidhuber 1997]. Sua estrutura é representada na Figura 5. Os modelos LSTM-RNN foram, de longe, as arquiteturas generativas mais difundidas utilizadas no design de novos medicamentos assistido por máquina. Foi também desenvolvida uma RNN utilizando Aprendizagem por Reforço para aprimorar a rede, melhorando significativamente os esforços anteriores para gerar grafos moleculares com uma RNN em [Olivecrona e al. 2017].

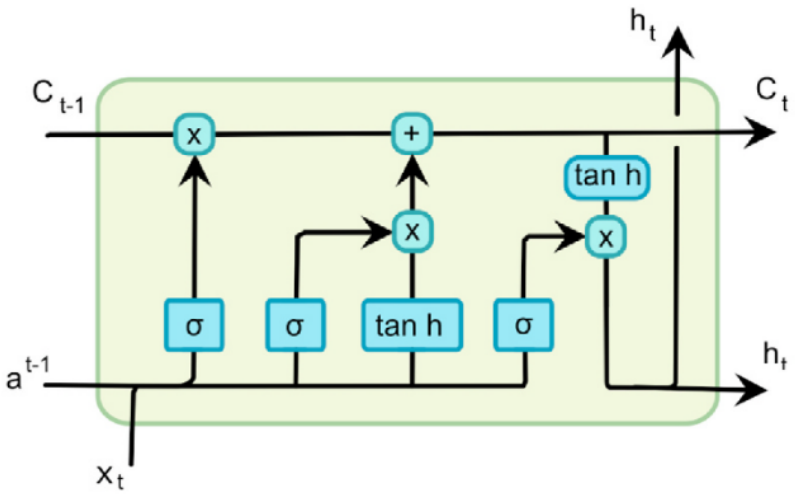

Figura 5 – LSTM-RNN: As entradas são representadas por x_t, a entrada atual; C_{t-1}, o estado da célula; e h_{t-1}, o estado oculto. As saídas são C_t, o próximo estado da célula, e h_t, a nova saída. As não linearidades são representadas por σ e tan h, as camadas sigmóide e tangente hiperbólica, respectivamente. Finalmente, os operadores $\times$ e $+$ indicam multiplicação e adição pontuais [Martinelli 2022].

Como uma das primeiras tentativas de produzir um modelo gerativo para auxiliar na descoberta de medicamentos, seu modelo é frequentemente comparado a outras arquiteturas de RNN [Martinelli 2022].

2.5.2 VAE

As Redes Autoencoder Variacionais (VAEs) são uma técnica de aprendizado de máquina especializada na compactação e geração de dados. Sua arquitetura é bipartida, consistindo de um codificador que transforma dados de entrada em uma representação em um espaço latente, e um decodificador que reconstrói os dados a partir dessa representação. A característica distintiva das VAEs é a implementação de um mecanismo probabilístico no espaço latente, diferentemente dos Autoencoders convencionais. Este mecanismo probabilístico utiliza uma distribuição, tipicamente normal, definida por parâmetros de média e variância, para mapear as entradas. Sua estrutura é representada na Figura 6.

Durante o treinamento, as VAEs aprendem a otimizar esses parâmetros para maximizar a probabilidade dos dados de entrada, mantendo simultaneamente a distribuição latente próxima a uma distribuição normal padrão. Esse processo é mediado por uma função de custo que inclui tanto o erro de reconstrução quanto a divergência Kullback-Leibler, responsável por medir a eficiência da aproximação da distribuição latente à distribuição desejada. O resultado é a capacidade das VAEs de aprender representações eficientes e regulares no espaço latente, tornando-as adequadas para lidar com dados complexos e de alta dimensão, simplificando-os em um espaço latente mais compacto e estruturado, permitindo

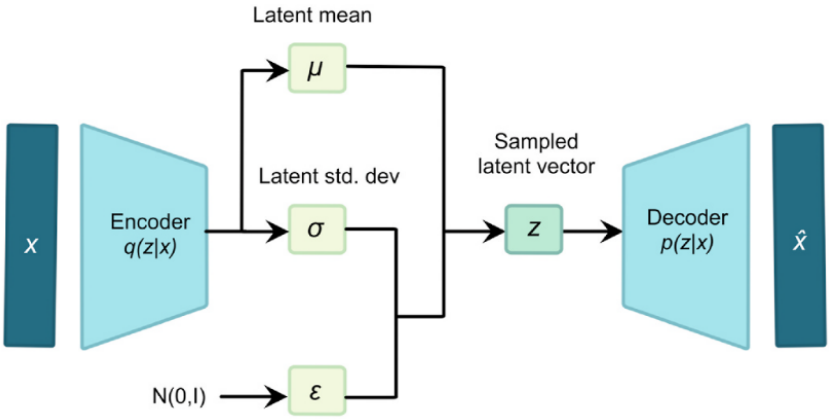

Figura 6 – Um Autoencoder Variacional. Conforme indicado, o codificador e o decodificador são probabilísticos, não determinísticos. O vetor latente amostrado é uma versão compactada da entrada. N(0,1) representa a distribuição normal. O termo de erro é denotado como ε [Martinelli 2022].

a geração de novas amostras nesse espaço e proporcionando controle sobre características específicas [Kingma e Welling 2013]. Um modelo notável de VAE é o GENTRL (Generative Tensorial Reinforcement Learning), desenvolvido por [Zhavoronkov e al. 2019] Este modelo combina aprendizagem por reforço (RL), inferência variacional e decomposições tensoriais para desenvolver moléculas com alta sintetizabilidade, bioatividade e diversidade. Embora os compostos gerados não tenham sido validados experimentalmente, análises computacionais quânticas confirmaram a viabilidade das conformações propostas. Outro exemplo é o PaccMannRL de [Born e al. 2021], no qual dois VAEs são utilizados, permitindo a consideração tanto de informações moleculares quanto de dados transcriptômicos. Mesmo antes de codificar dados experimentais sobre anticarcinógenos, o modelo mostrou uma tendência para a geração de agentes químicos altamente bioativos. Esses exemplos ilustram a capacidade dos VAEs em incorporar informações complexas e diversas na geração de novos candidatos a medicamentos, reforçando o papel da inteligência artificial na descoberta de drogas [Martinelli 2022].

2.5.3 GAN

As Redes Adversárias Generativas (GANs) representam uma abordagem significativa no campo da inteligência artificial, caracterizando-se por sua estrutura única de duas redes neurais em constante interação. A rede geradora tem a função de produzir amostras de dados, enquanto a rede discriminadora avalia essas amostras, diferenciando-as das reais. Este sistema de "adversários" cria um ciclo de feedback contínuo, no qual a rede geradora busca constantemente melhorar sua capacidade de criar dados indistinguíveis dos reais, e a rede discriminadora se aperfeiçoa na identificação de falsificações. Sua estrutura é representada na Figura 7. O aprendizado acontece de maneira não supervisionada, uma

característica distintiva das GANs, permitindo que elas aprendam a replicar a distribuição de dados sem necessidade de rótulos. Essa metodologia proporciona uma forma eficiente de geração de dados, com aplicações que vão desde a síntese de imagens até o aprimoramento de algoritmos de aprendizado de máquina. A natureza competitiva e a capacidade de autoaprimoramento contínuo das GANs as tornam uma ferramenta poderosa para tarefas que exigem adaptação em ambientes de dados complexos [Goodfellow et al. 2014].

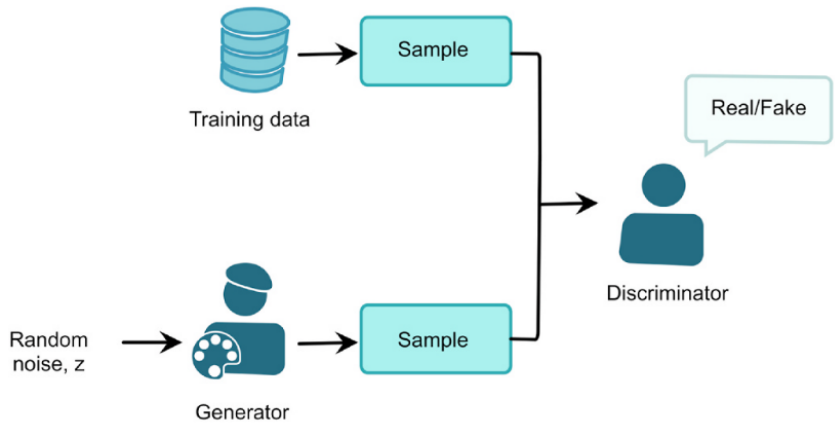

Figura 7 – Uma rede adversária generativa. O gerador monta o ruído aleatório de forma que, idealmente, se assemelhe aos dados de treinamento. O discriminador então categoriza cada amostra [Martinelli 2022].

Inicialmente, o modelo ORGANIC, desenvolvido por [Sanchez-Lengeling e al. 2017], marcou um avanço significativo ao ser um dos primeiros a usar GANs para gerar estruturas moleculares, apesar de uma baixa taxa de geração de moléculas válidas. Posteriormente, o modelo ATNC (Adversarial Threshold Neural Computer) de [Putin e al. 2018] apresentou melhorias notáveis, com uma maior porcentagem de strings SMILES válidas. Além disso, o RANC (Reinforced Adversarial Neural Computer), do mesmo grupo, introduziu em [Putin e al. 2018] um computador neural diferenciável para superar o mode collapse, um desafio comum em GANs. Além disso, o LatentGAN de [Prykhodko e al. 2019] conseguiu gerar compostos químicos diferentes daqueles produzidos por RNNs, evidenciando a potencial complementaridade entre as duas abordagens na diversificação de candidatos a medicamentos. O MolGAN, criado por [Cao e Kipf 2018], se destacou pela alta eficiência na geração de compostos válidos, utilizando uma abordagem implícita e livre de verossimilhança. Por fim, o Mol-CycleGAN de [Maziarka e al. 2020] mostrou ser eficaz na composição de moléculas com propriedades específicas, evidenciando uma melhoria significativa em relação a modelos anteriores. Esses avanços, conforme discutido no artigo de Martinelli, ilustram a importância crescente das GANs na inovação e desenvolvimento de novos medicamentos [Martinelli 2022].

2.6 Fundamentação do modelo Ensemble cGAN

O processo de aprendizado de uma GAN envolve treinar o gerador G e o discriminador D simultaneamente por meio de um processo de aprendizado adversarial. Dados $x_r \sim p_r(x)$, o objetivo de G é aproximar a distribuição real dos dados p_r sobre os dados de treinamento x_r, que é denotada como p_g. G começa com a amostragem da variável de entrada z de uma distribuição uniforme ou normal $p(z)$ e, em seguida, mapeia a variável de entrada $z \sim p(z)$ para novos dados $G(z) \sim p_g(G(z))$. O discriminador D é um classificador que tem como objetivo distinguir os dados reais x_r dos dados gerados por G. Portanto, o objetivo de treinamento das GANs pode ser formulado da seguinte maneira:

$$L_G = -\mathbb{E}_{z\sim p(z)}[\log D(G(z))] \tag{2.1}$$

$$L_D = -\mathbb{E}_{x_r\sim p_r(x)}[\log D(x_r)] - \mathbb{E}_{z\sim p(z)}[1 - \log D(G(z))] \tag{2.2}$$

Por concepção, durante o treinamento do gerador G, a função de erro da Equação 2.1 avalia a perda global em um mini-lote de instâncias geradas como a média das perdas, e o gerador minimiza essa perda por meio de uma técnica de otimização por descida de gradiente. Paralelamente, a função de erro na Equação 2.2 calcula as perdas médias nas instâncias reais e geradas, utilizando uma técnica de otimização por descida de gradiente para minimizar essas perdas no discriminador. Em uma situação ideal, a função de erro na Equação 2.2 constantemente instiga o discriminador a aprimorar sua capacidade de distinguir entre instâncias reais e geradas, fornecendo, assim, um sinal de aprendizado mais eficaz para que o gerador ajuste seus parâmetros. Ao mesmo tempo, a função de erro na Equação 2.1 conduz o gerador a convergir para uma condição específica: as instâncias geradas convergem para a distribuição de grupos com o mesmo pico de posição idêntico aos dados reais. Nesse contexto, o gerador pode ser monitorado para descobrir a função de mapeamento do espaço latente para o espaço de imagem e, assim, gerar instâncias realistas.

Entretanto, ao lidar com um conjunto de dados em grande escala contendo instâncias de alta resolução, a aprendizagem do modelo, orientada pelas funções de erro nas Equações 2.1 e 2.2, revela-se subótima. Pesquisas anteriores apresentam evidências sólidas de que dados reais possuem uma distribuição altamente multimodal [Schönfeld, Schiele e Khoreva 2020, Karnewar e Wang 2020]. Conforme o treinamento progride, o gerador encontra dificuldades em se beneficiar plenamente das instâncias geradas, uma vez que o discriminador negligencia diferenças evidentes entre as instâncias geradas e reais nas partes menos discriminativas (por exemplo, forma), resultando na incapacidade de fornecer gradientes valiosos para otimizar ainda mais o gerador. O problema científico fundamental reside em como o discriminador pode avaliar as instâncias geradas de múltiplas perspec-

tivas, de modo a proporcionar um sinal de aprendizado mais eficiente para o gerador, aprimorando, assim, a qualidade da geração.

A flexibilidade da GAN em incorporar qualquer rede diferenciável como seus componentes constitui uma de suas características distintivas. Essa capacidade inerentemente versátil permite que a GAN tire proveito dos avanços do aprendizado profundo de maneira natural. Ao adotar redes diferenciáveis, a GAN pode ser configurada para se beneficiar de arquiteturas profundas, o que pode facilitar a aprendizagem de representações complexas e hierárquicas nos dados, com inúmeras possibilidades arquiteturais.

2.7 Problemática

Apesar de avanços notáveis, há relativamente poucos exemplos de aplicação de modelos generativos para descobrir moléculas em aplicações concretas. [Yao e Cao 2021] As GANs são consideradas formulações mais complexas [Sun 2020], já que sua otimização depende na verdade de duas redes neurais adversariais que não transmitem explicitamente os padrões aprendidos, e até agora há um número limitado de análises sobre o tema [Saxena e Cao 2020]. No contexto molecular, a maioria dos estudos concentra-se na otimização de moléculas para métricas computacionais, como logP (logaritmo do coeficiente de partição) ou QED11 (estimativa quantitativa da semelhança com drogas) [Yao e Cao 2021].

Além disso, a diversidade molecular é uma importante métrica na geração molecular, que é uma limitação comum às GANs, mesmo quando treinadas em dados multimodais. Por exemplo, ao treinar GANs em dados de dígitos escritos à mão com dez dígitos, o gerador (G) pode ser incapaz de gerar alguns dígitos, resultando no problema conhecido como *mode collapse*. Os químicos humanos têm consciência da diversidade química, mas os algoritmos de aprendizado de máquina precisam ser explicitamente informados sobre a diversidade como uma métrica desejada. Esse é um problema bastante comum, especialmente em GANs [Kadurin e al. 2017, Sanchez-Lengeling e al. 2017, Cao e Kipf 2018]. O problema de *mode collapse* é exemplificado na Figura 8.

Outra dificuldade é a oscilação entre o Gerador e o Discriminador durante o treinamento, em vez de uma convergência para um ponto fixo. Quando um jogador se torna mais poderoso que o outro, o sistema pode não aprender e sofrer de instabilidade. A seleção de hiperparâmetros, como tamanho do lote, momento, weight decay e taxa de aprendizado, é crucial para a convergência do treinamento de GANs.

Além disso, problemas de não convergência e instabilidade podem ocorrer, especialmente quando o gradiente do gerador se perde devido à facilidade de diferenciação entre amostras reais e falsas por parte do discriminador D. Isso pode ser agravado pela escolha inadequada da função objetivo de G, levando a gradientes instáveis e de baixa qualidade.

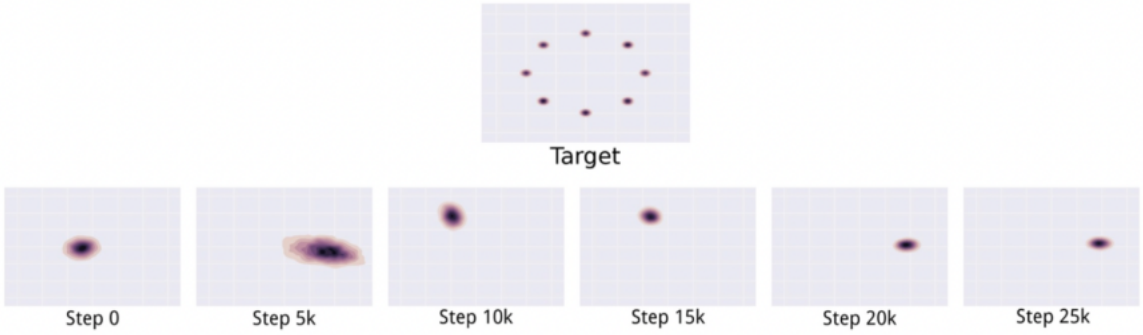

Figura 8 – Exemplo do problema de mode collapse em um conjunto de dados artificiais bidimensionais [Arjovsky, Chintala e Bottou 2017]

A distribuição alvo é uma mistura de gaussianas no espaço 2D. Durante o treinamento das GANs, o Gerador gera apenas elementos de um cluster por vez e continua movendo-se entre diferentes modos à medida que o Discriminador é treinado para descartar modos separados.

Ciente dos conhecidos problemas no treinamento de GANs e após estudos sobre diferentes métodos de solucioná-los ou minimizá-los, nas próximas seções deste capítulo é explicada a arquitetura proposta neste trabalho.

2.8 Discriminador

Os métodos de ensemble learning exploram vários algoritmos de aprendizado de máquina para produzir resultados a partir da junção de resultados com vários mecanismos de votação para obter melhores desempenhos do que aqueles obtidos apenas com qualquer algoritmo constituinte [Zhou 2012]. A partir desse pensamento, [Xie et al. 2022] surgiu com a proposta de um modelo convolucional que é na verdade constituído de múltiplos discriminadores para aproveitar as vantagens de uma estrutura ensemble e ao mesmo tempo reduzir o custo computacional. No conjunto proposto, cada discriminador é forçado a manter uma representação "plausível" em uma escala específica e a concentrar-se nas deficiências de cada um. Ao realizar um treinamento ponta a ponta, cada um dos discriminadores obtidos pode avaliar os SMILES gerados em uma escala específica. Juntos, eles formam um poderoso conjunto de discriminadores que podem atualizar melhor o gerador.

Algumas outras providências foram tomadas para evitar problemas conhecidos em GANs. Dentre elas, a Normalização Espectral (SN) [Miyato et al. 2018] foi aplicada às diferentes camadas convolucionais do Discriminador. SN funciona dividindo as matrizes de pesos do Discriminador pelo seu maior valor singular. A aplicação dessa medida compensa as distribuições espectrais das matrizes de peso para evitar que entrem em colapso [Liu et al. 2019].

A função de perda do Discriminador também foi atualizada de acordo com a proposta de [Xie et al. 2022], que utiliza de fatores moduladores para atribuir pesos mais

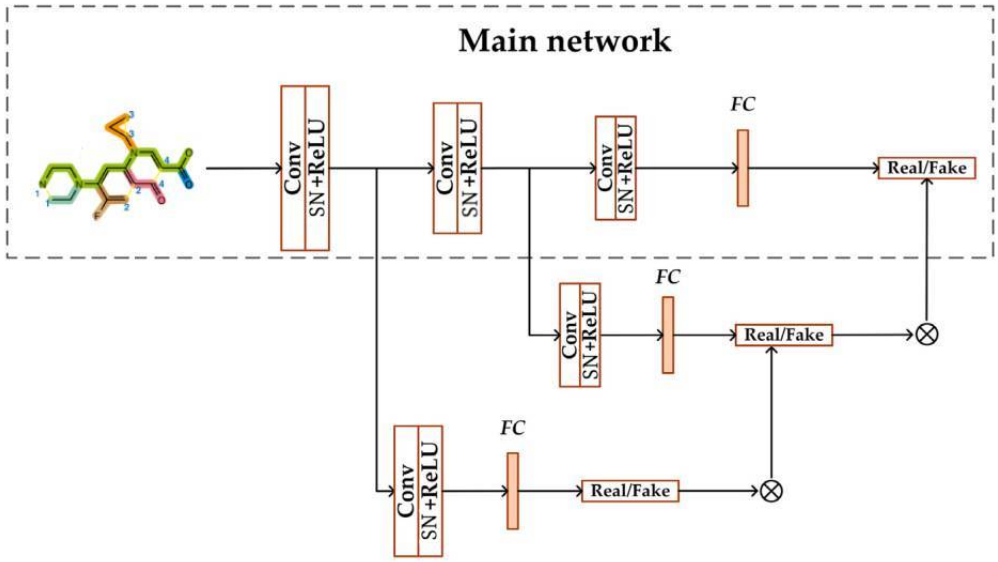

Figura 9 – Arquitetura proposta do Discriminador Ensemble [Xie et al. 2022].

A arquitetura inclui uma rede principal que pode ser qualquer estrutura discriminadora adotada pelos populares GANs, e dois discriminadores auxiliares que são anexados a diferentes camadas da rede principal. As perdas para todos os discriminadores são calculadas de maneira sequencial, e a perda para o discriminador "superficial" é primeiramente calculada e depois usada para calcular o fator de modulação para os discriminadores subsequentes. FC: camada densa; Conv: Camada convolucional; SN: Normalização Spectral; ReLU: função de ativação ReLU.

elevados às instâncias classificadas erroneamente pelos seus membros anteriores do conjunto.

Depois que as perdas de todos os discriminadores nas instâncias reais e geradas forem obtidas, elas podem ser somadas em uma única perda. O Discriminador Ensemble minimiza essa quantidade de perda usando a técnica de otimização de gradiente descendente, para que sua capacidade de distinguir instâncias reais e falsas possa ser melhorada. A arquitetura desenvolvida para o Discriminador pode ser visualizada na Figura 9.

2.9 Gerador

Como mencionado anteriormente, a flexibilidade da GAN em incorporar qualquer rede diferenciável como seus componentes constitui uma de suas características distintivas e, tomando proveito dessa característica, a estrutura principal do Gerador do modelo foi definida a partir da arquitetura de uma Conditional Variational Autoencoder (cVAE).

O cVAE é composto por um codificador $q_\phi(z|x, y)$ e um decodificador $p_\theta(x|z, y)$. O codificador mapeia os dados de entrada x juntamente com a variável condicional y para uma distribuição no espaço latente z. Similarmente ao VAE padrão, essa distribuição é tipicamente modelada como uma distribuição gaussiana multivariada com média μ e desvio padrão σ, ambos dependentes dos dados de entrada x e da variável condicional y.

O codificador é definido como:

$$q_\phi(z|x,y) = \mathcal{N}(z|\mu_\phi(x,y), \sigma_\phi(x,y)^2)$$

em que $\mu_\phi(x,y)$ e $\sigma_\phi(x,y)$ são as médias e desvios padrão parametrizados pela entrada x e a variável condicional y, e ϕ são os parâmetros do codificador.

O decodificador, assim como no VAE padrão, mapeia amostras do espaço latente juntamente com a variável condicional de volta para o espaço de dados. Ele modela a distribuição condicional dos dados x dado um ponto no espaço latente z e a variável condicional y, geralmente modelado como uma distribuição gaussiana:

$$p_\theta(x|z,y) = \mathcal{N}(x|\mu_\theta(z,y), \sigma_\theta(z,y)^2)$$

em que $\mu_\theta(z,y)$ e $\sigma_\theta(z,y)$ são as médias e desvios padrão parametrizados pelo ponto no espaço latente z e a variável condicional y, e θ são os parâmetros do decodificador.

A função objetivo do CVAE é semelhante à do VAE, mas considera a informação da variável condicional y:

$$\mathcal{L}_{\text{CVAE}} = -\mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|z,y)] + \text{KL}(q_\phi(z|x,y)||p(z|y))$$

em que $p(z|y)$ é a distribuição à priori no espaço latente condicionada à variável y.

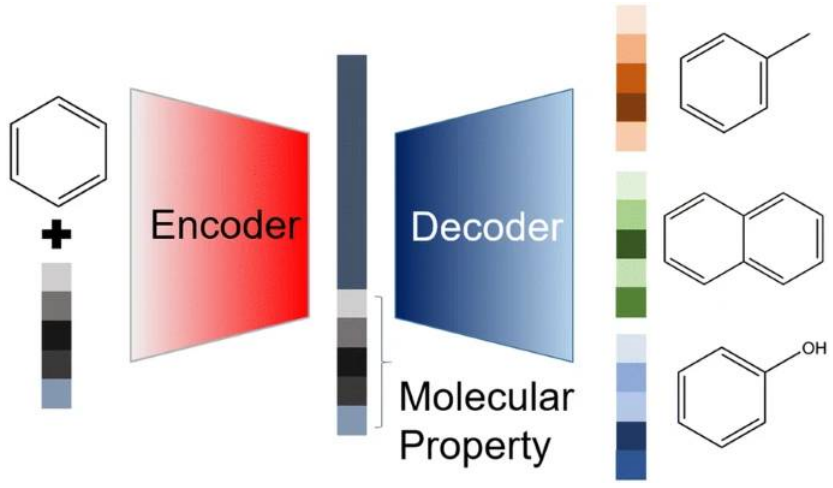

Figura 10 – Arquitetura proposta do Gerador [Lim et al. 2018].

A arquitetura de um Conditional Variational Autoencoder (cVAE) combina elementos de um Variational Autoencoder (VAE) com uma abordagem condicional, permitindo a geração de dados específicos com base em condições fornecidas.

O cVAE é um dos modelos generativos mais populares que gera objetos semelhantes, mas não idênticos, a um determinado conjunto de dados. Em particular, distingue-se do

VAE porque pode impor certas condições nos processos de codificação e descodificação. Tal modelo foi escolhido por apresentar avanços promissores necessitando de um nível de complexidade e custo computacional menores que as RNN. A arquitetura desenvolvida para o Gerador é representada de forma simplificada na Figura 10.

3 Materiais e Métodos

Para conduzir os experimentos no desenvolvimento e teste da Generative Adversarial Network (GAN) deste trabalho, diversos recursos foram empregados. Eles são apresentados nas próximas seções.

3.1 Unidade de Processamento Gráfico (GPU)

A execução intensiva de cálculos durante o treinamento de redes neurais, como no caso da GAN, foi facilitada pela utilização da GPU do BioMal (grupo de pesquisa em Bioinformática e Aprendizado de Máquina) da Universidade Federal de São Carlos, que contém as seguintes especificações: NVIDIA PNY VCNRTXA6000-BLK 1 Quadro RTXA6000 Graphics Card with 48GB GDDR6 PCIe 4.0 - Active Cooling. A sua utilização permitiu acelerar significativamente o processo de treinamento para experimentação do modelo.

3.2 Versionamento de Código com GitHub

A administração e controle da implementação prática do projeto foram conduzidos por meio da plataforma GitHub. A ferramenta facilita o rastreamento de todas as alterações realizadas ao longo do tempo, oferecendo uma abordagem eficiente para o versionamento do código. Um aspecto notável da utilização do GitHub é a estratégia de organização do desenvolvimento por meio de diferentes branches. Esses branches permitem o desenvolvimento paralelo de variações do projeto, proporcionando um ambiente isolado para implementação de novas funcionalidades ou experimentos sem interferir na versão principal e estável do código. Essa abordagem torna o processo de desenvolvimento mais flexível, e foi bastante importante para o desenvolvimento de diferentes versões do projeto simultaneamente. O repositório desenvolvido encontra-se disponível gratuitamente: <https://github.com/barbieriht/Smiles-cGAN>.

3.3 Ambiente de Desenvolvimento Visual Studio Code

A codificação, depuração e visualização dos resultados foram conduzidas no ambiente Visual Studio Code. Este ambiente forneceu uma interface de desenvolvimento integrada (IDE) eficiente, com suporte a extensões que facilitaram o desenvolvimento e a organização do código. Além disso, foram utilizadas de extensões como a [Microsoft ongoing] e [Microsoft ongoing] para que fosse possível a utilização da GPU de forma local.

3.4 Principais Bibliotecas

A implementação foi toda desenvolvida através da linguagem Python, na versão 3.10.12, com auxílio das seguintes principais bibliotecas:

- **PyTorch:** Uma biblioteca essencial para a implementação de Generative Adversarial Networks (GANs), o PyTorch desempenha um papel fundamental ao oferecer estruturas flexíveis para a construção e treinamento de redes neurais. Possibilitando a criação eficiente de arquiteturas complexas de GANs, o PyTorch simplifica o processo de desenvolvimento, permitindo aos pesquisadores e desenvolvedores explorar diversas abordagens na geração de dados realistas. Sua flexibilidade e poder tornam-o uma escolha valiosa para a implementação eficaz de modelos generativos avançados [PyTorch Contributors 2017];

- **Pandas:** O Pandas é amplamente reconhecido por sua capacidade de lidar com estruturas de dados tabulares, oferecendo estruturas de dados flexíveis como DataFrames. Isso é fundamental ao lidar com conjuntos de dados complexos, proporcionando funcionalidades para filtrar, transformar e combinar dados de maneira eficiente [McKinney e Pandas Development Team 2022];

- **NumPy:** É uma biblioteca fundamental para computação numérica em Python, destaca-se por suas operações matriciais eficientes. Isso é crucial para as etapas de treinamento da GAN, onde manipulações matriciais são frequentemente empregadas. NumPy proporciona um conjunto abrangente de funções e métodos para realizar operações matemáticas e estatísticas em larga escala, otimizando o desempenho e a eficiência computacional [Oliphant 2006];

- **RDKit:** É uma ferramenta indispensável na implementação de Generative Adversarial Networks (GANs), desempenhando um papel crucial na manipulação de estruturas químicas. Sua contribuição vital reside na capacidade de facilitar a geração e avaliação de moléculas durante o processo de treinamento da GAN [Landrum 2006].

Além das bibliotecas citadas, foram ocasionalmente utilizados Jupyter Notebooks para experimentação interativa e análise exploratória dos dados.

A base de dados inicial continha 36887 amostras de SMILES de tamanhos variados atribuídas a seus respectivos grupos CHEBI, que eram, no total, 347 grupos. Esses grupos ajudam a organizar e categorizar compostos químicos com base em suas características estruturais e funcionais.

3.5 Etapas de Desenvolvimento

Vale destacar que a estrutura de código desenvolvida está de acordo com as boas práticas de engenharia de software. O código é notavelmente modularizado, o que promove a clareza e adaptabilidade. Além disso, a implementação incorpora funcionalidades de backup e restauração, permitindo a preservação dos estados atuais dos experimentos. Essas abordagens não apenas facilitam a compreensão e manutenção do código, mas também contribuem para uma experiência de desenvolvimento mais eficiente e robusta.

3.5.1 Pré-processamento

Os dados utilizados no desenvolvimento do projeto são disponibilizados por um banco de dados aberto chamado ChEMBL, que contém informações de ligação, funcionais e ADMET para um grande número de compostos bioativos semelhantes a medicamentos [Gaulton et al. 2012].

Os dados utilizados continham strings no formato SMILES seguidas pelos respectivos grupos CHEBI dos quais faziam parte. Todos os SMILES extraídos da base foram validados através da utilização da biblioteca RDKit através do método Chem.MolFromSmiles(), que recebe como entrada uma string SMILES e retorna o objeto Mol associado a essa molécula. Quando o SMILES tem problemas estruturais, o método fica impossibilitado de transformá-lo em um objeto Mol, retornando um objeto vazio em seu lugar.

A base de dados inicial tinha um grande problema com desbalanceamento dos dados, tanto pelo tamanho de cada SMILE (que variava de 2 a 1566 dígitos) e também o número de grupos CHEBI atribuídos a cada SMILE (grupos que eram atribuídos de 1 a 6176 vezes) e também a multiplicidade de grupos por SMILE (de 1 a 14 grupos). O problema de desbalanceamento dos dados pode ser visualizado na Figura 11.

Inicialmente, a ideia era de que o modelo fosse multicondicional, ou seja, que aceitasse múltiplos grupos como entrada, e que deveria gerar estruturas que supostamente teriam as características de tais grupos. Além disso, os SMILES utilizados inicialmente poderiam ter quais quer dimensões. Entretanto, além de ter custo computacional muito maior, o modelo entrava em instabilidade com muita facilidade.

O desbalanceamento nos dados pode ser problemático para o treinamento de uma rede neural por várias razões. Primeiramente, no que diz respeito ao tamanho dos SMILES, se houver uma ampla variação no comprimento, a rede neural pode ter dificuldade em aprender padrões relevantes devido à diferença na escala dos dados. Modelos de redes neurais geralmente funcionam melhor quando os dados de entrada são balanceados.

Em relação ao desbalanceamento no número de grupos CHEBI atribuídos a cada SMILES, se alguns grupos forem significativamente mais comuns do que outros, o modelo

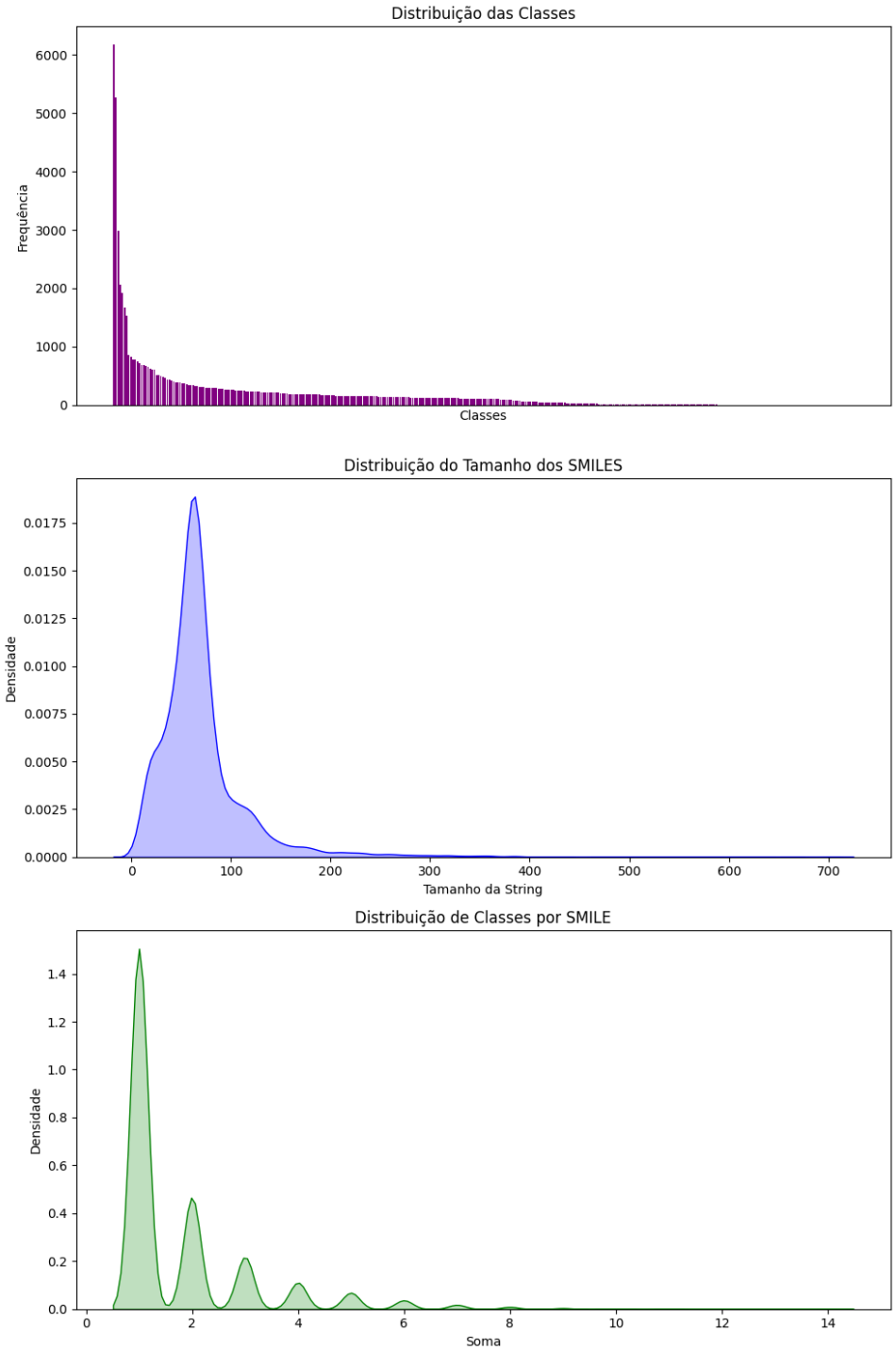

Figura 11 – Distribuição inicial dos dados

pode ficar tendencioso em direção aos grupos majoritários. Isso podse resultar em um desempenho inferior na predição dos grupos minoritários, pois o modelo pode não ter recebido dados suficientes para aprender representações robustas desses grupos menos frequentes.

Além disso, a multiplicidade de grupos por SMILES também pode impactar o treinamento. Se um SMILES estiver associado a vários grupos, isso introduz complexidade adicional ao modelo, que precisará aprender a lidar com essa relação multirrótulo.

Por isso, foi feita uma filtragem em grupos de baixa frequência e também de SMILES cujos tamanhos estão fora de um intervalo específico. Além disso, foi feito um balanceamento desses grupos, assegurando que cada grupo tenha um número semelhante de representantes. A partir dessa seleção, foram geradas duas bases de dados, uma delas contendo SMILES menores que 123 dígitos (com 9163 amostras) enquanto um outro dataset para experimentação também foi gerado com SMILES entre 54 e 84 dígitos (distribuição mais densa), ambas com os SMILES distribuídos entre 7 grupos CHEBI.

Os novos conjuntos de dados gerados foram distribuídos de forma que cada SMILES fosse designado com apenas um grupo, de forma a tentar manter a maior quantidade de SMILES possíveis respeitando a distribuição entre o tamanho das strings e a designação de grupos. Esses conjuntos de dados foram apelidados de Esparso (Figura 12) e Denso (Figura 13) de acordo com a distribuição dos tamanhos de SMILES.

Após o pré-processamento dos dados, infelizmente mais de 75% dos dados acabou sendo inutilizado, já que a utilização desses dados aumentaria a complexidade do modelo sem fornecer uma variedade plausível de dados para que o modelo pudesse aprender uma generalização sobre todos os grupos.

3.5.2 Ingestão de dados

Para amostragem de SMILES ao modelo, foi desenvolvido um DataLoader. Ele funciona como uma interface que facilita o acesso eficiente aos dados durante o treinamento do modelo. Em geral, um DataLoader carrega um conjunto de dados e organiza-o em lotes (batches) que são processados pelo modelo durante cada iteração de treinamento. Isso é particularmente útil quando lidamos com conjuntos de dados extensos que não cabem na memória RAM do sistema.

3.5.3 Parada antecipada

O mecanismo de parada antecipada (ou early stopping) é usado no treinamento de algoritmos de aprendizado de máquina para evitar overfitting. Ele interrompe o treinamento se o desempenho em um conjunto de validação não melhorar por um número definido de épocas consecutivas, ajudando o modelo a generalizar melhor para novos dados.

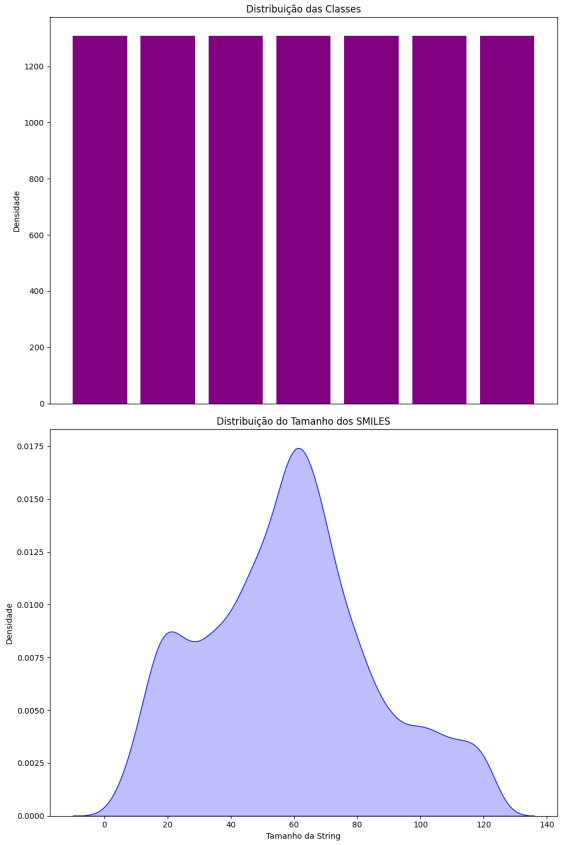

Figura 12 – Conjunto Esparso: O conjunto esparso de dados foi desenvolvido em busca de uma base de dados que pudesse tirar proveito do máximo de SMILES possíveis dentro dos principais grupos CHEBI, com o intuito de alimentar o modelo com dados suficientes para generalização dos dados por grupo. É constituído por aproximadamente 1400 SMILES por grupo, e SMILES de tamanho que variam entre 2 e 123 dígitos.

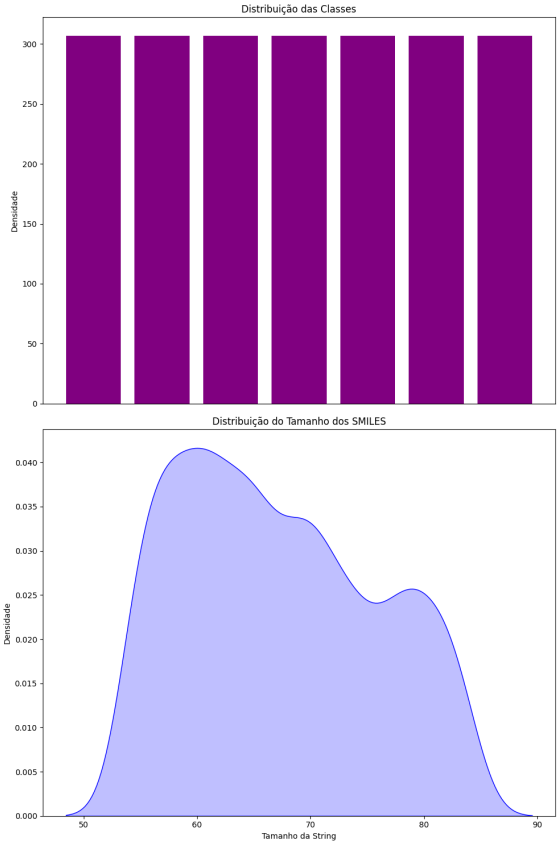

Figura 13 – Conjunto Denso: O conjunto denso de dados foi desenvolvido com o intuito de treinar o modelo com a região em que havia uma distribuição mais densa em relação ao tamanho dos SMILES. A proposta era de que o modelo tivesse o mínimo de dados nulos possíveis para que pudesse aprender melhor sobre a construção de SMILES de tamanhos dentro dos limites definidos. Cada grupo CHEBI era constituído de aproximadamente 300 SMILES, que variam entre 54 e 84 dígitos. Entretanto, a amostragem de SMILES se torna bastante limitada para o aprendizado de um modelo um tanto quanto complexo.

3.5.4 Seleção de Hiperparâmetros

A seleção de hiperparâmetros, como batch size, momentum, weight decay e taxa de aprendizado é um fator extremamente importante para a convergência do treinamento de GANs [Saxena e Cao 2020]. Por isso, foi desenvolvida uma abordagem de forma que diferentes metodologias de treinamento do modelo pudessem ser comparadas entre si. Dentre as combinações de hiperparâmetros, foram inclusos o método de tokenização (baseado em átomos ou em fragmentos), dimensão de ruído, batch size, taxa de aprendizado e multiplicador de taxa de aprendizado para o gerador, dimensões das camadas de neurônios, otimizadores e até mesmo a seleção da base de dados utilizada.

Para auxílio nos testes e otimização dos hiperparâmetros do modelo foi selecionada uma abordagem tradicional que é realizada por meio de uma técnica conhecida como busca em grade (Grid Search). Essa metodologia realiza uma exploração exaustiva em um subconjunto predefinido do espaço de hiperparâmetros associado ao algoritmo de treinamento [Liashchynskyi e Liashchynskyi 2019].

4 Análise Experimental

Durante o desenvolvimento do projeto, diversas arquiteturas foram testadas, assim como múltiplas combinações de hiperparâmetros. Entretanto, os resultados dos últimos experimentos foram os mais satisfatórios, e alguns de seus resultados serão mostrados nesse capítulo.

Apesar de muitas vezes não ser necessária uma função de perda específica para o Gerador, já que ele deve aprender de acordo com o feedback do Discriminador, algumas características desejadas precisavam ser explicitamente declaradas para que o Gerador pudesse seguí-las. Por isso, foram desenvolvidas funções de perda personalizadas que incluíssem essas métricas. Dentre as funções de perda utilizadas, foram utilizadas as seguintes funções: $Untranslatable_{Loss}$, $Repeated_{Loss}$, $Discriminator_{Loss}$ e VAE_{Loss}, que chamaremos respectivamente de U_{Loss}, R_{Loss}, D_{Loss}, VAE_{Loss} para simplificação.

A função U_{Loss} representa a porcentagem de SMILES válidos gerados pelo modelo; a função R_{Loss} representa a porcentagem de SMILES repetidos gerados; D_{Loss} é o feedback do Discriminador em relação às imagens geradas; já a função VAE_{Loss} representa o feedback gerado pela similaridade entre os SMILES de entrada e saída do cVAE.

A seguir, serão demonstrados os resultados dos experimentos de algumas dessas funções de perda no modelo, utilizando dos hiperparâmetros de learning rate do Discriminador com o valor de 1×10^{-5}, o learning rate do Gerador em 5×10^{-5} e tamanho do batch 64. Os testes foram aplicados em diferentes combinações de Método de Tokenização (átomo/fragmento) [Seção 2.2], Otimizador (Adam/Adamax/RMSProp/SGD) [Seção 2.4] e Dataset (esparso/denso) [Seção 3.5.1].

A primeira função de perda foi desenvolvida com o intuito de utilizar do feedback do Discriminador como um multiplicador da perda das outras métricas. Dessa forma, ela seria a função de perda de maior peso, já que interferiria diretamente no valor das outras perdas. A representação matemática dessa primeira função é apresentada na Equação 4.1.

$$F_{Loss(1)} = (1 + D_{Loss}) \times (100 \times (U_{Loss} + R_{Loss}) + VAE_{Loss}) \tag{4.1}$$

Uma segunda função representada pela Equação 4.2 foi desenvolvida afim de tentar priorizar o feedback do cVAE com o intuito de tentar reconstruir as moléculas com o máximo de precisão possível. As outras funções de perda foram utilizadas como multiplicadores de diferentes pesos de interferência na perda final.

$$F_{Loss(2)} = \left(1 + D_{Loss} + U_{Loss} + \frac{R_{Loss}}{2}\right) \times VAE_{Loss} \tag{4.2}$$

A partir de algumas das métricas já coletadas com as funções de perda anteriores, algumas conclusões já puderam ser tiradas em relação a alguns otimizadores e em relação aos datasets usados. A partir de então, os experimentos prosseguiram com a remoção dos hiperparâmetros que se apresentavam ineficientes ao modelo. A partir de múltiplos testes, foi alcançada uma candidata a melhor função de perda, descrita pela Equação 4.3. A ideia seria de que o número de moléculas válidas e o número de moléculas repetidas gerassem métricas correlacionadas, impedindo que o modelo pudesse se beneficiar por gerar várias moléculas válidas, entretanto repetidas, ou moléculas variadas mas sem validade. Essa perda seria então somada ao feedback do Discriminador, que juntos se transformariam em um multiplicador da perda do cVAE.

$$F_{Loss(3)} = \left((1 + D_{Loss}) + \frac{(1 + U_{Loss}) \times (1 + R_{Loss})}{2}\right) \times VAE_{Loss} \tag{4.3}$$

Evidentemente, há inúmeras possibilidades para otimização do modelo. As métricas geradas pelos experimentos citados estão presentes na Seção 4.2. Antes, os resultados obtidos são discutidos na próxima seção.

4.1 Discussão de Resultados

A partir dos primeiros conjuntos de testes, foi possível visualizar uma enorme diferença nos resultados de acordo com o dataset usado. O Dataset Denso (Figura 13), apesar de ter um conjunto de SMILES com tamanhos mais padronizados, consiste em uma amostragem de dados um tanto menor, e isso foi criticamente prejudicial no desempenho do modelo.

Em relação às tokenizações utilizadas, apesar de que a perspectiva seria de que a tokenização por fragmentos minimizaria os problemas com SMILES inválidos já que seriam gerados fragmentos inteiros das moléculas, o objetivo não foi atingido provavelmente por tirar um pouco da liberdade do modelo, já que a amostragem por fragmentos existentes se torna menor por conta do tamanho do dicionário gerado. Ambos os métodos de tokenização parecem promissores e devem ser melhor avaliados, entretanto, a tokenização por átomo teve melhores desempenhos tanto em relação à variedade de SMILES quanto ao número de SMILES válidos gerados.

Dentre os otimizadores, o mais problemático foi o SGD que, apesar de ter sido selecionado como um candidato com o intuito de trazer melhor generalização para o modelo, ele foi o que teve maiores problemas de desempenho em relação às métricas geradas. É recomendado que testes mais extensivos sejam aplicados, já que todos os experimentos foram executados com o mesmo fator de paciência para o mecanismo de Early Stopping de treinamento do modelo. A lenta convergência do SGD por conta de sua simplicidade é

conhecida, e pode ter levado a contagem do mecanismo a acreditar que o modelo estava se desviando da descida do gradiente em momentos em que ele poderia voltar a caminhar em direção a um ótimo melhor. Contudo, apesar de ter falhado nos experimentos até onde esse estudo seguiu, o otimizador não deve ser excluído de testes futuros.

Por fim, os dados gerados através dos experimentos insinuam que os melhores resultados foram alcançados pela medida F_{Loss} (Equação 4.3). Contudo, isso não implica que o modelo prossiga sendo experimentado apenas com essa função de perda. São recomendados testes variados com o modelo, já que o estudo foi desenvolvido ao decorrer de um período curto.

Por fim, um resumo dos resultados indica que o modelo foi capaz de gerar mais da metade das moléculas válidas enquanto a taxa de estruturas geradas repetidas se mantinha bastante baixa. A avaliação de desempenho do modelo é uma tarefa bastante complexa já que, mesmo que as moléculas geradas sejam interpretáveis ou variadas, há grandes chances das moléculas geradas serem estruturas pouco complexas ou sem novidade, como pode se visualizar em alguns exemplos de moléculas geradas na Tabela 2. Alguns fatores devem ser levados em consideração ao analisar as estruturas geradas. Eles são melhor explicados na próxima seção.

Tabela 2 – Exemplos de SMILES gerados

SMILE	**Grupo**
O=C(N[C@@H](CCCCCCCCCCCC)CCCCCCC)CCCCCCCCCCCCCCCCCCCCCC	@CHEBI_76924
O=C(NC(CCCCCCCCCCCC)CCCCCCC)CCCCCCCCCCCCCCC	@CHEBI_16670
O=C(N[C@@H](CCCCCCCCCC)C)CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	@CHEBI_75771
O=C(N[C@@H](CCCCCCCCCC)C)CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	@CHEBI_35610
OCC(NC(CCCCCCCCCCC)CCCCCCC)CCCCCCCCCC	@CHEBI_25212
OCC(NC(CCCCCCCCCCC)CCCCCCC)CCCCCC	@CHEBI_35610
CCC[C@@]1CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	@CHEBI_25212

4.2 Resultados das medidas de avaliação

As tabelas apresentadas a seguir apresentam os valores das perdas resultantes do treinamento do modelo a partir dos parâmetros apresentados em suas legendas, além dos hiperparâmetros citados na seção anterior. As duas primeiras métricas representam os valores de perda geral do Discriminador e do Gerador, enquanto o restante delas são as métricas explícitas geradas para melhor direcionamento do modelo, explicadas na Seção anterior.

É importante indicar que, nas equações citadas no início do capítulo, a métrica $G\ Untranslatable_{Loss}$ tem valores em um intervalo de $[0, 1]$, enquanto nos valores mostrados a seguir, essa escala é transformada para o intervalo de $[0, 100]$

Tabela 3 – FLoss(1) | Tokenização: Átomo - Dataset: Denso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.033
	G Loss	446.872
	G Untranslatable Loss	100.000
	G Disc Loss	0.001
	G Repetition Loss	97.680
	G VAE Loss	248.818

Tabela 4 – FLoss(1) | Tokenização: Átomo - Dataset: Denso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.038
	G Loss	444.867
	G Untranslatable Loss	100.000
	G Disc Loss	0.001
	G Repetition Loss	98.438
	G VAE Loss	245.969

Tabela 5 – FLoss(1) | Tokenização: Átomo - Dataset: Denso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.012
	G Loss	370.099
	G Untranslatable Loss	73.343
	G Disc Loss	0.000
	G Repetition Loss	62.879
	G VAE Loss	233.812

Tabela 6 – FLoss(1) | Tokenização: Átomo - Dataset: Denso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	0.928
	G Loss	631.446
	G Untranslatable Loss	100.000
	G Disc Loss	0.401
	G Repetition Loss	97.964
	G VAE Loss	252.653

Tabela 7 – FLoss(1) | Tokenização: Átomo - Dataset: Esparso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.002
	G Loss	279.698
	G Untranslatable Loss	59.768
	G Disc Loss	0.000
	G Repetition Loss	24.454
	G VAE Loss	195.473

Tabela 8 – FLoss(1) | Tokenização: Átomo - Dataset: Esparso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.001
	G Loss	279.151
	G Untranslatable Loss	57.714
	G Disc Loss	0.000
	G Repetition Loss	25.765
	G VAE Loss	195.670

Tabela 9 – FLoss(1) | Tokenização: Átomo - Dataset: Esparso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.001
	G Loss	264.344
	G Untranslatable Loss	55.223
	G Disc Loss	0.000
	G Repetition Loss	16.827
	G VAE Loss	192.293

Tabela 10 – FLoss(1) | Tokenização: Átomo - Dataset: Esparso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	0.935
	G Loss	602.521
	G Untranslatable Loss	74.508
	G Disc Loss	0.399
	G Repetition Loss	93.400
	G VAE Loss	262.780

Tabela 11 – FLoss(1) | Tokenização: Fragmento - Dataset: Denso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.017
	G Loss	399.780
	G Untranslatable Loss	100.000
	G Disc Loss	0.000
	G Repetition Loss	98.011
	G VAE Loss	201.645

Tabela 12 – FLoss(1) | Tokenização: Fragmento - Dataset: Denso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.007
	G Loss	286.678
	G Untranslatable Loss	70.265
	G Disc Loss	0.000
	G Repetition Loss	42.708
	G VAE Loss	173.678

Tabela 13 – FLoss(1) | Tokenização: Fragmento - Dataset: Denso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	0.954
	G Loss	564.522
	G Untranslatable Loss	80.777
	G Disc Loss	0.425
	G Repetition Loss	96.354
	G VAE Loss	219.007

Tabela 14 – FLoss(1) | Tokenização: Fragmento - Dataset: Esparso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.002
	G Loss	251.654
	G Untranslatable Loss	62.303
	G Disc Loss	0.000
	G Repetition Loss	33.621
	G VAE Loss	155.727

Tabela 15 – FLoss(1) | Tokenização: Fragmento - Dataset: Esparso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.003
	G Loss	265.637
	G Untranslatable Loss	66.270
	G Disc Loss	0.000
	G Repetition Loss	38.582
	G VAE Loss	160.779

Tabela 16 – FLoss(1) | Tokenização: Fragmento - Dataset: Esparso - RMS

Otimizador	**Perda**	**Valor**
RMS	D Loss	0.001
	G Loss	250.076
	G Untranslatable Loss	66.838
	G Disc Loss	0.000
	G Repetition Loss	30.835
	G VAE Loss	152.402

Tabela 17 – FLoss(1) | Tokenização: Fragmento - Dataset: Esparso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	0.819
	G Loss	476.393
	G Untranslatable Loss	62.216
	G Disc Loss	0.326
	G Repetition Loss	96.941
	G VAE Loss	200.188

Tabela 18 – FLoss(2) | Tokenização: Átomo - Dataset: Denso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.017
	G Loss	423.639
	G Untranslatable Loss	73.958
	G Disc Loss	0.000
	G Repetition Loss	0.791
	G VAE Loss	239.985

Tabela 19 – FLoss(2) | Tokenização: Átomo - Dataset: Denso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.075
	G Loss	487.731
	G Untranslatable Loss	100.000
	G Disc Loss	0.003
	G Repetition Loss	0.981
	G VAE Loss	244.608

Tabela 20 – FLoss(2) | Tokenização: Átomo - Dataset: Denso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	1.010
	G Loss	624.679
	G Untranslatable Loss	100.000
	G Disc Loss	0.482
	G Repetition Loss	0.979
	G VAE Loss	252.738

Tabela 21 – FLoss(2) | Tokenização: Átomo - Dataset: Esparso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.002
	G Loss	289.990
	G Untranslatable Loss	68.564
	G Disc Loss	0.000
	G Repetition Loss	0.257
	G VAE Loss	196.979

Tabela 22 – FLoss(2) | Tokenização: Átomo - Dataset: Esparso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.002
	G Loss	289.990
	G Untranslatable Loss	68.564
	G Disc Loss	0.000
	G Repetition Loss	0.257
	G VAE Loss	196.979

Tabela 23 – FLoss(2) | Tokenização: Átomo - Dataset: Esparso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.001
	G Loss	261.210
	G Untranslatable Loss	49.454
	G Disc Loss	0.000
	G Repetition Loss	0.208
	G VAE Loss	193.151

Tabela 24 – FLoss(2) | Tokenização: Fragmento - Dataset: Denso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.009
	G Loss	389.915
	G Untranslatable Loss	100.000
	G Disc Loss	0.000
	G Repetition Loss	0.943
	G VAE Loss	197.758

Tabela 25 – FLoss(2) | Tokenização: Fragmento - Dataset: Denso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.010
	G Loss	362.813
	G Untranslatable Loss	96.828
	G Disc Loss	0.000
	G Repetition Loss	0.795
	G VAE Loss	192.778

Tabela 26 – FLoss(2) | Tokenização: Fragmento - Dataset: Denso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.018
	G Loss	366.612
	G Untranslatable Loss	92.045
	G Disc Loss	0.000
	G Repetition Loss	0.866
	G VAE Loss	193.636

Tabela 27 – FLoss(2) | Tokenização: Fragmento - Dataset: Esparso - Adam

Otimizador	**Perda**	**Valor**
Adam	D Loss	0.001
	G Loss	229.293
	G Untranslatable Loss	60.806
	G Disc Loss	0.000
	G Repetition Loss	0.330
	G VAE Loss	156.093

Tabela 28 – FLoss(2) | Tokenização: Fragmento - Dataset: Esparso - Adamax

Otimizador	**Perda**	**Valor**
Adamax	D Loss	0.001
	G Loss	213.842
	G Untranslatable Loss	46.143
	G Disc Loss	0.000
	G Repetition Loss	0.311
	G VAE Loss	154.188

Tabela 29 – FLoss(2) | Tokenização: Fragmento - Dataset: Esparso - RMSProp

Otimizador	**Perda**	**Valor**
RMSProp	D Loss	0.000
	G Loss	209.966
	G Untranslatable Loss	80.966
	G Disc Loss	0.000
	G Repetition Loss	0.090
	G VAE Loss	144.790

Tabela 30 – FLoss(2) | Tokenização: Fragmento - Dataset: Denso - SGD

Otimizador	**Perda**	**Valor**
SGD	D Loss	0.820
	G Loss	451.167
	G Untranslatable Loss	96.056
	G Disc Loss	0.335
	G Repetition Loss	0.948
	G VAE Loss	197.084

Tabela 31 – FLoss(3) | Tokenização: Átomo - ADAM

Otimizador	**Perda**	**Valor**
ADAM	D Loss	0.001
	G Loss	272.876
	G Untranslatable Loss	65.396
	G Disc Loss	0.000
	G Repetition Loss	0.223
	G VAE Loss	194.775

Tabela 32 – FLoss(3) | Tokenização: Átomo - ADAMAX

Otimizador	**Perda**	**Valor**
ADAMAX	D Loss	0.009
	G Loss	469.959
	G Untranslatable Loss	100.000
	G Disc Loss	0.000
	G Repetition Loss	0.734
	G VAE Loss	251.721

Tabela 33 – FLoss(3) | Tokenização: Átomo - RMS

Otimizador	**Perda**	**Valor**
RMS	D Loss	0.001
	G Loss	250.936
	G Untranslatable Loss	49.268
	G Disc Loss	0.000
	G Repetition Loss	0.211
	G VAE Loss	193.206

Tabela 34 – FLoss(3) | Tokenização: Fragmento - ADAM

Otimizador	**Perda**	**Valor**
ADAM	D Loss	0.002
	G Loss	224.068
	G Untranslatable Loss	60.686
	G Disc Loss	0.000
	G Repetition Loss	0.312
	G VAE Loss	160.208

Tabela 35 – FLoss(3) | Tokenização: Fragmento - ADAMAX

Otimizador	**Perda**	**Valor**
ADAMAX	D Loss	0.001
	G Loss	237.891
	G Untranslatable Loss	77.786
	G Disc Loss	0.000
	G Repetition Loss	0.339
	G VAE Loss	156.335

Tabela 36 – FLoss(3) | Tokenização: Fragmento - RMS

Otimizador	**Perda**	**Valor**
RMS	D Loss	0.001
	G Loss	210.013
	G Untranslatable Loss	58.413
	G Disc Loss	0.000
	G Repetition Loss	0.299
	G VAE Loss	152.329

5 Conclusão e trabalhos futuros

Diante dos resultados obtidos ao longo desta pesquisa, torna-se evidente a complexidade inerente à construção de uma Ensemble cGAN para a geração de moléculas. A utilização da arquitetura proposta, composta por Gerador e Discriminador, demonstrou potencial promissor, como evidenciado pela melhoria na resposta do modelo em diversos testes realizados. Contudo, é crucial reconhecer uma relevante limitação de recursos para uma hiperparametrização abrangente e uma modelagem mais refinada da arquitetura. Ainda há uma grande limitação em relação aos dados utilizados para o treinamento do modelo já que, apesar da grande variedade de SMILES disponíveis, a quantidade de moléculas estudadas e tabuladas de acordo com suas características e grupos CHEBI são de baixa amostragem em proporção à complexidade das estruturas moleculares.

Os desafios enfrentados durante o treinamento do Ensemble cGAN para a geração de moléculas são amplamente reconhecidos na literatura científica. Problemas como a convergência instável, o collapse mode e a dificuldade na geração de dados diversos são conhecidos obstáculos para a efetiva implementação de GANs. Apesar de não se ter alcançado resultados ótimos, o modelo proposto demonstrou uma considerável capacidade na geração de novas estruturas moleculares. Acredita-se fielmente que, com a disponibilidade de maiores recursos computacionais e aplicação de técnicas mais avançadas de fine-tuning, além da aquisição de uma amostragem maior de dados, podem vir a trazer resultados promissores em prol do avanço do estado da arte.

Algumas observações puderam ser feitas a partir dos experimentos feitos. O tamanho das batches aplicadas para o treinamento do modelo teve grande influência em seus resultados. O uso de ambos métodos de tokenização tiveram desempenhos semelhantes no treinamento do modelo. Batches maiores direcionavam o modelo a gerar amostras repetitivas cada vez mais cedo. Aumentar a complexidade dos modelos em relação ao número de neurônios por camada não levou o modelo a visualizar melhor os padrões nos SMILES, gerando um custo computacional desnecessário. O uso do otimizador SGD, apesar de ter sido utilizado com o intuito de fornecer uma maior capacidade de generalização para o modelo, foi o otimizador que mais causou geração de moléculas repetitivas, além de ter uma convergência muito mais lenta. Multiplicadores na taxa de aprendizado para tentar favorecer o gerador não pareceram ter tanto efeito, entretanto, não foram testados valores extravagantes em seu treinamento.

O trabalho desenvolvido está disponível e de fácil contribuição com parâmetros definidos pelo desenvolvedor a partir do repositório <https://github.com/barbieriht/Smiles-cGAN>. Após uma hiperparametrização, com testes de períodos mais longos, é

necessária a avaliação de métricas importantes para análise da qualidade das moléculas geradas, já que a convergência do modelo e a geração de moléculas viáveis ou não repetitivas não necessariamente mostram uma boa eficiência do modelo. Isso acontece pelo simples fato de que o modelo pode aprender a gerar, por exemplo, uma mesma molécula mas com variadas quantidades de ligações de carbono, gerando moléculas que, apesar de válidas e diversas, não contribuem com a descoberta de novas estruturas moleculares. Há algumas formas de tentar se proteger contra esse tipo de trapaça do modelo, como por exemplo aplicando testes com a "Regra de Cinco" (RO5), estimativa quantitativa da semelhança com moléculas drug-like (QED), sintetizabilidade de moléculas, dentre outras métricas. As métricas citadas encontram-se presentes também na biblioteca RDKit.

Além disso, outras técnicas podem ser utilizadas para mais rápida convergência e uma possível melhor generalização do modelo, como a utilização de métricas como a Distância de Fréchet ChemNet (FCD) [Preuer et al. 2018], uma métrica considerada mais robusta que outras métricas de desempenho do modelo, como validade, exclusividade, novidade e divergência KL, mas que necessita de um grande poder computacional para ser incorporado no treinamento de um modelo.

A partir de alguns estudos, outro recurso que parece promissor para o desenvolvimento e avanço do modelo pode ser a abordagem do UnCorrupt SMILES [Schoenmaker et al. 2023], que propõe o treinamento de uma rede neural para corrigir SMILES corrompidos. Esse mesmo modelo poderia ser aplicado às saídas do Gerador em busca de uma maior porcentagem de SMILES válidos, que poderiam estar próximos de algo que o Gerador pretenderia criar.

Referências

ALBERTS, B. et al. *Biologia Molecular da Célula*. 5. ed. [S.l.]: Artmed Editora S/A, 2010. Citado na página 19.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein generative adversarial networks. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2017. p. 214–223. Citado 2 vezes nas páginas 11 e 32.

BORN, J.; AL. et. Paccmannrl: de novo generation of hit-like anticancer molecules from transcriptomic data via reinforcement learning. *iScience*, v. 24, n. 4, p. 102269, 2021. Disponível em: <https://doi.org/10.1016/j.isci.2021.102269>. Citado na página 28.

BROWN, N. et al. A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *J Chem Inf Comput Sci*, v. 44, p. 1079–1087, 2004. Citado na página 26.

CAO, N. D.; KIPF, T. Molgan: An implicit generative model for small molecular graphs. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2018. Citado na página 29.

CAO, N. D.; KIPF, T. Molgan: an implicit generative model for small molecular graphs. 2018. Citado na página 31.

CERETO-MASSAGUÉ, A.; AL. et. Molecular fingerprint similarity search in virtual screening. *Methods*, v. 71, p. 58–63, 2015. Citado na página 21.

CHO, H.; CHOI, I. Three-dimensionally embedded graph convolutional network (3dgcn) for molecule interpretation. *Journal Name*, 2018. Citado na página 22.

CREATIVECOMMONS. Https://creativecommons.org/licenses/by-sa/3.0/. Citado 2 vezes nas páginas 11 e 22.

DAVID, L.; AL. et. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, v. 12, p. 56, 2020. Citado na página 21.

ESTRADA, E. Characterization of 3d molecular structure. *Chemical Physics Letters*, v. 319, n. 5-6, p. 713–718, 2000. Citado na página 22.

GAULTON, A. et al. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Res*, v. 40, n. Database issue, p. D1100–D1107, 2012. Disponível em: <https://doi.org/10.1093/nar/gkr777>. Citado na página 39.

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*. [s.n.], 2014. v. 27. Disponível em: <https://proceedings.neurips.cc/paper/2014/file/5ca3e9b1>. Citado 2 vezes nas páginas 26 e 29.

HASTINGS, J. et al. Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic Acids Research*, Oxford University Press, v. 44, n. D1, p. D1214–D1219, 2016. Citado na página 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 26.

IVANOVIC, B. et al. Multimodal deep generative models for trajectory prediction: a conditional variational autoencoder approach. *IEEE Robotics and Automation*, v. 6, n. 2, p. 295–302, 2021. Citado na página 26.

KADURIN, A.; AL. et. Drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular Pharmaceutics*, v. 14, 2017. Citado na página 31.

KARNEWAR, A.; WANG, O. Msg-gan: Multi-scale gradients for generative adversarial networks. In: *Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. Citado na página 30.

KINGMA, D.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. Citado na página 26.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 25.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Citado na página 25.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. Citado na página 28.

KRENN, M. et al. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, IOP Publishing, v. 1, n. 3, p. 035024, 2020. Citado na página 21.

LANDRUM, G. Rdkit: Open-source cheminformatics. *SourceForge*, 2006. Citado na página 38.

LEWELL, X. et al. Recap–retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *Journal of Chemical Information and Computer Sciences*, American Chemical Society, v. 38, n. 4, p. 511–522, 1998. Citado na página 23.

LIASHCHYNSKYI, P.; LIASHCHYNSKYI, P. *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. 2019. Citado na página 43.

LIM, J. et al. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics*, v. 10, n. 1, p. 31, 2018. Disponível em: <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-018-0286-7>. Citado 2 vezes nas páginas 11 e 34.

LIU, K. et al. Spectral regularization for combating mode collapse in gans. In: *International Conference on Computer Vision*. [S.l.: s.n.], 2019. Citado na página 32.

LIU, S.; AL. et. Pre-training molecular graph representation with 3d geometry. *Journal Name*, 2021. Citado na página 22.

LOPEZ-RINCON, O. et al. Algorithmic music composition based on artificial intelligence: a survey. In: *IEEE International Conference on Electronics, Computing and Communication Technologies*. [S.l.: s.n.], 2018. p. 187–193. Citado na página 26.

MARTINELLI, D. D. Generative machine learning for de novo drug discovery: A systematic review. *Computers in Biology and Medicine*, v. 145, p. 105403, 2022. ISSN 0010-4825. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010482522001950>. Citado 7 vezes nas páginas 11, 20, 22, 26, 27, 28 e 29.

MAZIARKA, s.; AL. et. Mol-cyclegan: a generative model for molecular optimization. *J. Cheminf.*, v. 12, p. 2, 2020. Disponível em: <https://doi.org/10.1186/s13321-019-0404-1>. Citado na página 29.

McKinney, W.; Pandas Development Team. pandas-dev/pandas: Pandas. *Zenodo*, 2022. Citado na página 38.

Microsoft. *Dev Containers*. [S.l.]: Microsoft VS Marketplace, ongoing. <https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers>. Version 0.327.0. Citado na página 37.

Microsoft. *Remote - SSH*. [S.l.]: Microsoft VS Marketplace, ongoing. <https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh>. Version 0.108.0. Citado na página 37.

MIYATO, T. et al. Spectral normalization for generative adversarial networks. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2018. Citado na página 32.

OLIPHANT, T. E. Travis e. oliphant et al., guide to numpy, usa, trelgol publishing, (2006). USA, Trelgol Publishing, 2006. Citado na página 38.

OLIVECRONA, M.; AL. et. Molecular de-novo design through deep reinforcement learning. *J. Cheminf.*, v. 9, n. 2017, p. 48, 2017. Disponível em: <https://doi.org/10.1186/s13321-017-0235-x>. Citado na página 26.

POLISHCHUK, P. G.; MADZHIDOV, T. I.; VARNEK, A. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of Computer-Aided Molecular Design*, v. 27, n. 8, p. 675–679, 2013. Citado na página 20.

PREUER, K. et al. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, v. 58, n. 9, p. 1736–1741, 2018. Disponível em: <https://doi.org/10.1021/acs.jcim.8b00234>. Citado na página 56.

PRYKHODKO, O.; AL. et. A de novo molecular generation method using latent vector based generative adversarial network. *J. Cheminf.*, v. 11, p. 74, 2019. Disponível em: <https://doi.org/10.1186/s13321-019-0397-9>. Citado na página 29.

PUTIN, E.; AL. et. Adversarial threshold neural computer for molecular de novo design. *Mol. Pharm.*, v. 15, 2018. Disponível em: <https://doi.org/10.1021/acs.molpharmaceut.7b01137>. Citado na página 29.

PUTIN, E.; AL. et. Reinforced adversarial neural computer for de novo molecular design. *J. Chem. Inf. Model.*, v. 58, n. 6, p. 1194–1204, 2018. Disponível em: <https://doi.org/10.1021/acs.jcim.7b00690>. Citado na página 29.

PyTorch Contributors. *PyTorch*. [S.l.]: GitHub, 2017. <https://github.com/pytorch/pytorch>. Citado na página 38.

REUTLINGER, M. et al. Multi-objective molecular de novo design by adaptive fragment prioritization. *Angew Chem Int Ed Engl*, v. 53, p. 4244–4248, 2014. Citado na página 26.

REYMOND, J.-L.; AWALE, M. Exploring chemical space for drug discovery using the chemical universe database. *ACS Chemical Neuroscience*, v. 3, p. 649–657, 2012. Citado na página 20.

RINIKER, S.; LANDRUM, G. A. De novo molecular design and generative models. *Drug Discovery Today*, Elsevier, v. 26, n. 10, p. 2218–2225, 2021. Citado 3 vezes nas páginas 11, 24 e 26.

ROGERS, D.; HAHN, M. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, v. 50, n. 5, p. 742–754, 2010. Citado na página 21.

RUMELHART, D. et al. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Citado na página 26.

SANCHEZ-LENGELING, B.; AL. et. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *ChemRxiv*, 2017. Preprint. Disponível em: <https://doi.org/10.26434/chemrxiv.5309668.v3>. Citado na página 29.

SANCHEZ-LENGELING, B.; AL. et. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). 2017. ChemRxiv. Citado na página 31.

SAXENA, D.; CAO, J. Generative adversarial networks (gans): Challenges, solutions, and future directions. *CoRR*, abs/2005.00065, 2020. Disponível em: <https://arxiv.org/abs/2005.00065>. Citado 2 vezes nas páginas 31 e 43.

SCHOENMAKER, L. et al. Uncorrupt smiles: a novel approach to de novo design. *Journal of Cheminformatics*, v. 15, n. 22, p. 1–12, 2023. Disponível em: <https://doi.org/10.1186/s13321-023-00696-x>. Citado na página 56.

SCHöNFELD, E.; SCHIELE, B.; KHOREVA, A. A u-net based discriminator for generative adversarial networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Citado na página 30.

SUN, R.-Y. Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 2020. Citado na página 25.

SUN, R.-Y. Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 2020. Citado na página 31.

TIELEMAN, T.; HINTON, G. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.*, v. 4, n. 2, p. 26–31, 2012. Citado na página 25.

URSU, O. et al. Understanding drug-likeness. *WIREs Comput Mol Sci*, v. 1, p. 760–781, 2011. Citado na página 19.

VOGEL, A. et al. *Vogel's Textbook of Practical Organic Chemistry*. 5th. ed. [S.l.]: Prentice Hall, 1996. ISBN 0-582-46236-3. Citado na página 19.

WANG, K.; WAN, X. Automatic generation of sentimental texts via mixture adversarial networks. *Artificial Intelligence*, v. 275, p. 540–558, 2019. Citado na página 26.

WANG, W. et al. Generative machine learning for de novo drug discovery: A systematic review. *Computers in Biology and Medicine*, Elsevier, v. 140, p. 105403, 2022. Citado na página 23.

WARD, S.; BESWICK, P. What does the aromatic ring number mean for drug design? *Expert Opinion on Drug Discovery*, v. 9, n. 9, p. 995–1003, 2014. Citado na página 21.

WEININGER, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, v. 28, n. 1, p. 31–36, 1988. Citado na página 21.

XIE, Y. et al. A lightweight ensemble discriminator for generative adversarial networks. *Knowledge-Based Systems*, v. 250, p. 108975, 2022. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705122004725>. Citado 3 vezes nas páginas 11, 32 e 33.

YAO, J.; CAO, Z. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, Wiley Online Library, v. 11, n. 6, p. e1608, 2021. Citado 2 vezes nas páginas 21 e 31.

YU, J. et al. Generative image inpainting with contextual attention. In: *CPVR*. [S.l.: s.n.], 2018. p. 5505–5514. Citado na página 26.

ZHANG, Z. Improved adam optimizer for deep neural networks. In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. [S.l.: s.n.], 2018. p. 1–2. Citado na página 25.

ZHAVORONKOV, A.; AL. et. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nat. Biotechnol.*, v. 37, n. 9, p. 1038–1040, 2019. Disponível em: <https://doi.org/10.1038/s41587-019-0224-x>. Citado na página 28.

ZHOU, Z.-H. *Ensemble Methods: Foundations and Algorithms*. [S.l.]: Chapman and Hall/CRC, 2012. Citado na página 32.