UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Redes Adversárias Generativas: uma alternativa para modelagem de dados de entrada em projetos de simulação

Afonso Teberga Campos

Itajubá, agosto de 2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Afonso Teberga Campos

Redes Adversárias Generativas: uma alternativa para modelagem de dados de entrada em projetos de simulação

Tese submetida ao Programa de Pós-graduação em Engenharia de Produção como parte dos requisitos para obtenção do Título de Doutor em Ciências em Engenharia de Produção.

Área de concentração: Engenharia de produção

Orientador: Prof. Dr. José Arnaldo Barra Montevechi

Agosto de 2022

Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Afonso Teberga Campos

Redes Adversárias Generativas: uma alternativa para modelagem de dados de entrada em projetos de simulação

Tese aprovada por banca examinadora em 31 de agosto de 2022, conferindo ao autor o título de **Doutor em Ciências em Engenharia de Produção.**

Banca examinadora:

Prof. Dr. José Arnaldo Barra Montevechi (Orientador)

Prof. Dr. Aneirson Francisco da Silva (UNESP)

Prof. Dr. Anibal Tavares de Azevedo (UNICAMP)

Prof. Dr. Alexandre Ferreira de Pinho (UNIFEI)

Prof. Dr. Fabiano Leal (UNIFEI)

Itajubá

2022

DEDICATÓRIA

À minha família, pelo grande Amor, formação e apoio. E a todos os meus professores, pela generosidade em cada ensinamento.

AGRADECIMENTOS

Ao receber o *Emmy Lifetime Achievement Award* em 1997, Fred Rogers, pedagogo, apresentador e animador norte-americano, fez o inesquecível convite: “você pararia comigo por dez segundos para pensar nas pessoas que te ajudaram a se tonar quem você é? Aqueles que se importaram com você e quiseram o melhor para você na vida. Dez segundos de silêncio. Eu checo o tempo”. Nesse momento, não poderia deixar de atender a esse mesmo convite.

Agradeço primeiramente a Deus e aos meus amados pais, Beatris e Júlio, e avós, Tiana, Romão, Lúcia e Toninho, por todo Amor, apoio, incentivo e pelo nunca medido esforço. Agradeço também a meus amados irmãos, Luísa, José Marcelo e Marcela, pelo afeto e companheirismo. E a Luiz Guilherme, pela amizade e carinho.

Agradeço ao professor José Arnaldo por toda orientação, acadêmica e pessoal, confiança e amizade. Por ter acreditado, investido, incentivado e valorizado meu trabalho. Foi um privilégio contar com sua orientação.

Não poderia deixar de agradecer a todos os meus professores e professoras que me acompanharam ao longo da vida, os quais foram determinantes para minha caminhada. O incentivo, a generosidade de compartilhar seus conhecimentos e sabedoria e a alegria pelo crescimento de seus alunos não têm preço. Agradeço à Universidade Federal de Itajubá (UNIFEI) pela acolhida e pela excelente formação durante minha graduação, mestrado e doutorado.

Agradeço também aos meus amigos do Núcleo de Estudos Avançados para Auxílio à Decisão (NEAAD), que sempre estiveram dispostos a contribuir, discutir, sonhar e construir juntos novas ideias. Em especial, aos professores José Antonio de Queiroz, que me coorientou durante o mestrado, Fabiano Leal, Alexandre Pinho e Rafael Miranda e a meus companheiros de mestrado e doutorado Alexandre, Tábata, Gustavo, Paula, Carlos, Wilson e João Victor.

Agradeço à FlexSim pelo grande apoio oferecido para que eu pudesse dar continuidade aos meus estudos e para o desenvolvimento deste trabalho, em especial aos líderes Michael, Flávio e Paula.

Meus agradecimentos também à CAPES, CNPq e FAPEMIG pelo suporte financeiro para este e outros trabalhos.

A todos que de forma direta ou indireta contribuíram para que chegasse até aqui, o meu muito obrigado!

EPÍGRAFE

"Dados! Dados! Dados! Não posso fazer tijolos sem barro!" (Sherlock Holmes em *The Adventure of the Copper Beeches*)

RESUMO

De forma geral, a simulação estocástica consiste em dados de entrada e lógicas, sendo os primeiros as fontes básicas de incerteza em um modelo de simulação. Por essa razão, a modelagem de dados é uma etapa essencial no desenvolvimento de projetos na área. Muitos avanços foram observados nos últimos anos nos programas de simulação e em ferramentas para coleta. Porém, os métodos para modelagem de dados permanecem praticamente inalterados há mais de 30 anos. Em seu dia a dia, praticantes de simulação enfrentam dificuldades relacionadas à escolha de Modelos de Dados de Entrada (MDEs), principalmente devido ao desafio da modelagem de dados não Independentes e Identicamente Distribuídos (IID), o que requer ferramentas específicas e não oferecidas por programas de simulação e seus pacotes de estatísticos. Por essa razão, poucos estudos consideram elementos de complexidade como heterogeneidades, dependências e autocorrelações, subestimando a incerteza do sistema estocástico. Diante dos novos desenvolvimentos na área de Inteligência Artificial, é possível buscar sinergias para resolução desse problema. O presente estudo tem como objetivo avaliar os resultados da aplicação de Redes Adversárias Generativas, ou *Generative Adversarial Networks* (GANs) para obtenção de MDEs. Tais redes constituem uma das mais recentes arquiteturas de redes neurais artificiais, sendo capazes de aprender distribuições complexas e, com isso, gerar amostras sintéticas com o mesmo comportamento dos dados reais. Para tanto, esta tese propõe um método para Modelagem de Dados de Entrada baseado em GANs (MDE-GANs) e o implementa por meio da linguagem Python. Considerando uma série de objetos de estudo teóricos e reais, são avaliados os resultados em termos de qualidade de representação dos MDEs e realizadas comparações com métodos tradicionais. Como principal conclusão, foi possível identificar que a aplicação de MDE-GANs permite obter MDEs com forte acurácia, superando os resultados dos métodos tradicionais nos casos de dados não IID. Com isso, a presente tese contribui ao oferecer uma nova alternativa para a área, capaz de contornar alguns dos desafios enfrentados por modeladores.

Palavras-chave: Simulação a Eventos Discretos, Modelagem de Dados de Entrada, *Generative Adversarial Networks*

ABSTRACT

In general, stochastic simulation consists of input data and logic, the former being the basic source of uncertainty in a simulation model. For this reason, data modeling is an essential step in the development of stochastic simulation projects. Many advances have been observed in recent years in simulation software and in data collection tools. However, the methods for input data modeling have remained largely unchanged for over 30 years. In their daily lives, modelers face difficulties related to the choice of input data models, mainly due to the challenge of modeling non Independent and Identically Distributed Data (IID) data, which requires specific tools not offered by simulation software and their data modeling packages. For this reason, few studies consider elements of complexity such as heterogeneities, dependencies, and autocorrelations, underestimating the uncertainty of the stochastic system. Given the new developments in Artificial Intelligence, it is possible to seek synergies to solve this problem. The present study aims to evaluate the results of the application of Generative Adversarial Networks (GANs) for input data modeling. Such networks constitute one of the most recent architectures of artificial neural networks, being able to learn complex distributions and, therefore, generate synthetic samples with the same behavior as real data. Therefore, this thesis proposes a method for Input Data Modeling based on GANs (MDE-GANs) and implements it through the Python language. Considering a series of theoretical and real study objects, the results are evaluated in terms of representation quality of the input models and comparisons are made with traditional modeling methods. As a main conclusion, it was possible to identify that the application of MDE-GANs allows obtaining input data models with strong accuracy, surpassing the results of traditional methods in cases of non-IID data. Thus, the present thesis contributes by offering a new alternative for input data modeling, capable of overcoming some of the challenges faced by modelers.

Keywords: *Discrete-Event Simulation, Input Data Modeling, Generative Adversarial Networks*

LISTA DE ABREVIATURAS

AM	Aprendizado de Máquina
AP	Aprendizado Profundo
ARMA	*Autoregressive Moving Average* (Modelo Autoregressivo de Médias Móveis)
C2ST	*Classifier 2-Sample Test* (Teste Classificador para Duas Amostras)
DEP	Distribuição Estatística Paramétrica
GAN	*Generative Adversarial Network* (Rede Adversária Generativa)
IA	Inteligência Artificial
IID	Independentes e Identicamente Distribuídos
k-NN	*k-Nearest Neighbors* (k Vizinhos Próximos)
MDE	Modelo de Dados de Entrada
MDE-GANs	Modelagem de Dados de Entrada baseada em GANs
NHPP	*Non-Homogeneous Poisson Processes* (Processos Poisson Não Homogêneos)
OE	Objeto de Estudo
PCA	*Principal Component Analysis* (Análise de Componentes Principais)
Q-Q	Quantil-Quantil
ReLU	*Retified Linear Unit*
RNA	Rede Neural Artificial
SED	Simulação a Eventos Discretos
TT	Tempo Total para execução de um MDE

SUMÁRIO

1. INTRODUÇÃO

Neste capítulo, é apresentada a contextualização sobre a pesquisa e são discutidas suas justificativas, contribuições, questões, hipóteses e objetivos. Por fim, é descrito o método de pesquisa adotado.

1.1. Contextualização

Técnicas de simulação computacional encontram-se entre as ferramentas mais maduras e aplicadas na área de Pesquisa Operacional (GREASLEY; EDWARDS, 2021; LAW, 2014), figurando também entre os métodos de pesquisa utilizados em Engenharia de Produção (MIGUEL, 2007). Seu uso não se restringe ao ambiente manufatureiro, sendo encontrados estudos em problemas militares, serviços de saúde, logística, transportes e construção civil (GABRIEL *et al.*, 2020). Em especial, tem se destacado o crescimento do tema na área de serviços de saúde (VÁZQUEZ-SERRANO; PEIMBERT-GARCÍA; CÁRDENAS-BARRÓN, 2021).

Dentre as modalidades de simulação computacional, a Simulação a Eventos Discretos (SED) é identificada como a predominante ao longo dos anos (GABRIEL *et al.*, 2020). Trata-se da vertente focada, mas não restrita, à modelagem de processos discretos, cujas variáveis mudam instantaneamente e em momentos específicos (LAW, 2014).

A SED é uma ferramenta de análise que possibilita o planejamento e a otimização da tomada de decisão, do desenho e das operações de sistemas produtivos complexos e inteligentes (DE PAULA FERREIRA; ARMELLINI; DE SANTA-EULALIA, 2020). Tem como objetivo final estimar quantitativamente o impacto do desenho do processo em sua própria performance, podendo auxiliar no entendimento e análise para a gestão estratégica e melhoria de processos (GARCIA-GARCIA *et al.*, 2020). Segundo os autores, a SED pode ajudar a avaliar riscos, custos, barreiras de implementação e impactos operacionais. Em sentido semelhante, Ahmed, Page e Olsen (2020) afirmam ser possível estudar cenários positivos ou negativos, simulando ações para reduzir custos e aumentar qualidade, produtividade e competitividade (GARCIA-GARCIA *et al.*, 2020).

Além disso, a SED possui papel relevante no contexto da Indústria 4.0, especialmente por ser a principal modalidade de simulação utilizada no desenvolvimento de gêmeos digitais (DOS SANTOS *et al.*, 2022), cópias virtuais capazes de se conectar a sistemas físicos, espelhando seu comportamento e apoiando a tomada de decisão (WRIGHT; DAVIDSON, 2020). Nesse tipo de aplicação, têm sido reportados benefícios como a integração com outras ferramentas de

análise, utilização facilitada por tomadores de decisão, obtenção de modelos cada vez mais acurados e disseminação de inteligência em produtos, procedimentos e processos (DOS SANTOS *et al.*, 2022; EROL *et al.*, 2016; LONGO; NICOLETTI; PADOVANO, 2017; MOTYL *et al.*, 2017).

Entretanto, ainda são reportados na literatura desafios para a utilização de SED nas organizações. Dentre eles, destaca-se para o presente trabalho a dificuldade com a etapa de modelagem de dados de entrada. Conforme explica Law (2014), quase todos os processos reais contêm uma ou mais fontes de aleatoriedade, as quais frequentemente são representadas em modelos de SED por meio de Modelos de Dados de Entrada (MDE). Nesse tópico, Greasley e Edwards (2021) afirmam que a qualidade e a disponibilidade de dados são desafios, mas que poderão ser superados com avanços recentes como *Big Data*. Assim, espera-se que o gargalo na etapa de modelagem de dados de entrada migre da coleta para a análise e processamento (VOLOVOI, 2016).

Como, por definição, um modelo é uma aproximação da realidade, quando presente a aleatoriedade, os dados gerados por um MDE nunca serão idênticos aos dados reais. Por essa razão, é importante que o modelador reconheça que não existe um MDE "correto", sendo o objetivo obter uma aproximação que capture as principais características do processo (BILLER; GUNES, 2010). Em outras palavras, por mais que o modelador busque que os dados gerados por MDEs sejam os mais próximos possíveis da realidade, estes ainda não serão verdadeiros. E é nesse ponto que a modelagem de dados de entrada encontra paralelo com um fenômeno cada vez mais presente no dia a dia, o de conteúdos "*deep fake*".

Muita atenção tem sido dada a esses conteúdos, mídias que, apesar de falsas, são realistas e convencem ou até mesmo enganam os expectadores. Como exemplo, é possível citar vídeos ou imagens em que uma voz ou rosto é substituído pelo de outra pessoa. Usualmente conhecidos como "*fake*", estes conteúdos podem ser definidos tecnicamente como "sintéticos", emprestando o sentido utilizado na área química para denotar substâncias artificiais e que, em geral, visam se aproximar de algo natural (OXFORD UNIVERSITY PRESS, 2022). Realizando o paralelo com a modelagem de dados de entrada, nesse sentido, MDEs também geram dados sintéticos. Além disso, simulação, de uma forma ampla, envolve a geração de histórias definidas como artificiais (BANKS, 1998).

Antes dos últimos avanços na área de Inteligência Artificial (IA), técnicas para geração de imagens e vídeos falsos tinham a habilidade de criar texturas simplórias e os resultados estavam longe de enganar os olhos humanos. Entretanto, a situação mudou com avanços

significativos na área, especialmente de arquiteturas de Redes Neurais Artificiais (RNAs) para Aprendizado Profundo (AP) (LI *et al.*, 2020). Destaca-se, nesse contexto, o trabalho desenvolvido por Goodfellow *et al.* (2014), introduzindo as Redes Adversárias Generativas, mais conhecidas, do inglês, como *Generative Adversarial Networks* (GANs), foco do presente estudo. Após o desenvolvimento das GANs, a qualidade e o realismo de imagens sintéticas evoluíram significativamente, como demonstrado na Figura 1.1.

Figura 1.1 – Evolução de imagens sintéticas geradas por GANs
Fonte: adaptado de Zhang *et al.* (2021)

Outros exemplos de imagens sintéticas geradas por GANs podem ser encontrados em diversos trabalhos e *websites* tais como "*This Person Does Not Exist*" (2022), "*This Artwork Does Not Exist*" (2022) e "*This Cat Does Not Exist*" (2022), que reúnem, respectivamente, imagens falsas de pessoas, obras de arte e gatos.

Especificamente em AP, conteúdos sintéticos são popularmente chamados de "*deep fake*", termo que se originou em 2017 após um usuário da plataforma *Reddit*, chamado "*deepfakes*", alegar ter desenvolvido um algoritmo de aprendizado de máquina que permitiria transpor rostos de celebridades a vídeos de conteúdo adulto. Nos anos seguintes, foram notados usos de mídias "*deep fake*" para outros fins considerados danosos, tais como notícias falsas e fraudes financeiras, o que tem gerado considerável preocupação por parte de autoridades em todo o mundo (TOLOSANA *et al.*, 2020).

Entretanto, também têm sido registradas aplicações para finalidades positivas, muitas na área de saúde. Sorin *et al.* (2020) reúnem diversos exemplos de aplicação para geração de resultados de exames de imagem significativamente realistas, mas que, por serem sintéticos, não ferem requisitos de confidencialidade e proteção de dados dos pacientes. Essas imagens

podem ser utilizadas para fins acadêmicos e para treinamento de outros modelos de aprendizado de máquina. Os autores também destacam o uso de "*deep fake*" para, ao invés de gerar imagens totalmente sintéticas, completar ou refinar imagens existentes, permitindo melhorar a qualidade de resultados de exames radiológicos sem aumentar as doses de radiação empregadas, o que pode habilitar o uso de tais exames com maior frequência, para prevenção e triagem, sem comprometer a saúde dos pacientes. Ali *et al.* (2022) reúnem estudos que aplicam GANs na área de neuroimagem, também identificado aplicações de geração de resultados sintéticos, para alimentação de outros algoritmos e propostas; segmentação de exames de imagem, identificando áreas de interesse no cérebro e diagnosticando doenças; e remoção de ruído e aumento de resolução, para melhoria da qualidade dos exames sem demandar doses adicionais de radiação. Jeong *et al.* (2022) identificam estudos semelhantes em áreas como cardiologia, pneumologia e oftalmologia.

Já na área de manufatura, têm sido reportadas aplicações de GANs pra projetos de engenharia, com a geração de novos *designs* e produtos; previsão de consumo energético considerando variabilidades; e, de forma semelhante aos estudos na área de saúde, a geração de dados sintéticos para melhorar o treinamento de outros algoritmos, tais como sistemas de detecção de falhas (KUSIAK, 2020).

Considerando os impactos positivos observados nessas áreas, poderiam tais técnicas dar suporte à atividade de modelagem de dados de entrada, ajudando a enfrentar seus desafios? O presente trabalho considera que isso merece ser investigado, o que será discutido na próxima seção, e propõe a utilização de GANs para obtenção de MDEs.

1.2. Justificativa e contribuições

1.2.1. Impacto prático

Segundo Robertson e Perera (2002), o tempo alocado para coleta e modelagem de dados varia entre projetos e organizações. Entretanto, tradicionalmente essa fase tem se apresentado como uma das mais longas, estando presente em grande parte do projeto, como ilustrado na Figura 1.2.

Ainda segundo os autores, as atividades relacionadas aos dados de entrada ocupam de 20% a 50% do tempo total utilizado em projetos de simulação. De forma semelhante, Skoogh e Johansson (2009) apontam que cerca de 30% do tempo total de projetos na área são gastos com o gerenciamento dos dados. Segundo os autores, as atividades que consomem mais tempo

para modelagem de dados de entrada são a coleta, preparação, análise e mapeamento de dados disponíveis, representando 70% do tempo consumido.

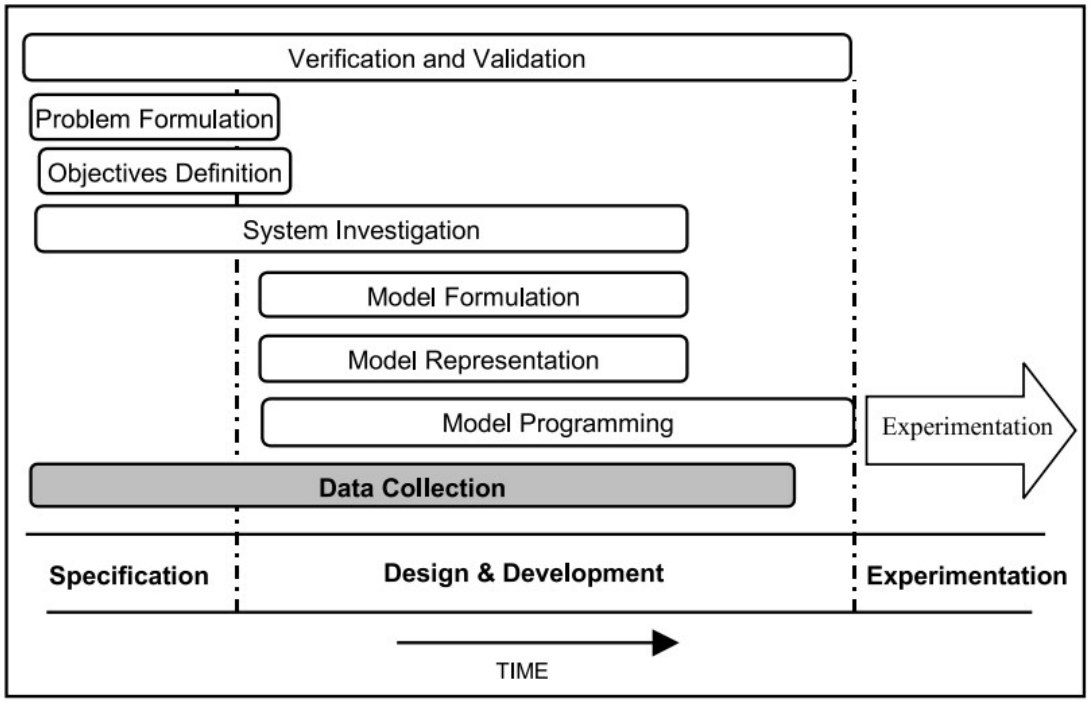


Figura 1.2 – Principais estágios em projetos de simulação
Fonte: Robertson e Perera (2002)

Após os trabalhos dos autores, muitos estudos surgiram e ainda têm surgido com o objetivo de reduzir o esforço para coleta de dados, especialmente propondo novas tecnologias para automação e integração de sistemas. Isso deve se intensificar nos próximos anos com o renascimento dos dados na fabricação, principalmente por causa da implantação de sensores, pela tecnologia sem fio e pelo progresso na análise de dados (GREASLEY; EDWARDS, 2021; KUSIAK, 2018). Projetos de simulação podem ser impactados positivamente pelo aumento na disponibilidade de dados, com o uso crescente de sensores, *internet* das coisas e processamento de texto, áudio e vídeo por algoritmos de aprendizado de máquina (CEN; HERBERT; HAAS, 2019), o que já tem sido observado em diversas aplicações (DOS SANTOS *et al.*, 2022).

Mas, se, por um lado, muitos avanços têm sido alcançados para coleta de dados, por outro, poucas melhorias foram sugeridas nas últimas décadas para modelagem dos dados obtidos. Entre as décadas de 1980 e de 2010, todos os tutoriais anuais sobre o tema publicados no *Winter Simulation Conference*, principal congresso sobre simulação a nível mundial, apresentaram conteúdos similares, com poucas mudanças ano a ano (CHENG, 2017), seguindo o mesmo comportamento nas edições posteriores.

Modelar os dados de entrada continua sendo uma das atividades mais desafiadoras para não *experts* e muitos erros são observados nessa fase. Segundo Cen, Herbert e Haas (2019), um dos principais deles é considerar que os dados são variáveis aleatórias Independentes e Identicamente Distribuídas (IID). Variáveis desse tipo apresentam dados originados em sua totalidade de uma mesma distribuição, não exibindo tendências ou flutuações. Além disso, suas observações são independentes entre si, o que significa que uma ocorrência não impacta outra. Segundo os autores, o problema é que, para fins de modelagem estatística, tal pressuposto quase nunca é verdadeiro, sendo frequentemente encontrados dados de estrutura mais complexa. Por exemplo, tempos entre chegadas, importantes para a avaliação de filas, apresentam comumente sazonalidades e horários de pico. Os autores afirmam que é muito comum que modeladores escolham distribuições que desconsideram essas dependências, assumindo que os dados são IID.

Em seu trabalho, Law (2014) aborda classes de dados de entrada que podem ser encontrados em projetos de SED (Figura 1.3) e que estão relacionadas à complexidade da atividade de modelagem de dados de entrada.

Figura 1.3 – Classes de dados que podem ser encontrados em sistemas a serem modelados
Fonte: adaptado de Law (2014)

Segundo o autor, os dados podem ser homogêneos, se provenientes de uma única distribuição, ou heterogêneos, se oriundos de misturas de distribuições, como casos multimodais. Podem ser classificados como independentes (univariados) ou dependentes (multivariados). Por fim, podem apresentar ou não autocorrelação, ou seja, dependência em relação às próprias observações anteriores dos dados, o que é conhecido como processo estocástico. Dados IID se enquadram, por definição, nas classes “homogêneos”, por serem identicamente distribuídos e “independentes” e “não autocorrelacionados” por não poderem apresentar qualquer dependência.

Distribuições de dados não IID são tipicamente difíceis de capturar e modelar (CEN; HERBERT; HAAS, 2019). Várias técnicas foram propostas para esses casos, tais como cópulas

para dados multivariados (BILLER, 2009), variações de modelos autorregressivos de médias móveis (ARMA) para dados de entrada autocorrelacionados (UHLIG; ROSE; RANK, 2016) e processos Poisson não homogêneos (NHPP) geralmente utilizados para taxas de chegadas (DE SANTIS *et al.*, 2021; MORGAN *et al.*, 2019). Do ponto de vista do modelador, são técnicas que requerem expertise adicional, o que é considerado uma barreira para projetos de simulação (MOURTZIS, 2020), podendo bloquear seu progresso (KUHL *et al.*, 2008). Além disso, pacotes comerciais de simulação e suas ferramentas de modelagem de dados de entrada não suportam estas técnicas (CEN; HERBERT; HAAS, 2020; UHLIG; ROSE; RANK, 2016), o que pode ser confirmado nas documentações de programas como ExpertFit® e Stat::Fit®, utilizados ou compatíveis com os principais pacotes comerciais de simulação, tais como AnyLogic®, Arena®, ProModel®, Simio®, SIMUL8® e FlexSim® (AVERILL M. LAW & ASSOCIATES, 2022; GEER MOUNTAIN SOFTWARE, 2022).

Além disso, conforme apontado por Kuhl *et al.* (2008), outras dificuldades são rotineiramente encontradas nessa fase de projetos de simulação, destacando-se as situações em que Distribuições Estatísticas Paramétricas (DEPs), como as distribuições normal, exponencial e lognormal, não conseguem representar adequadamente o comportamento probabilístico de processos do mundo real.

Com isso, nesse momento vale definir o conceito de dados complexos, adotado para o presente trabalho: são dados não IID ou com comportamentos probabilísticos que não podem ser representados adequadamente por DEPs, em especial as univariadas. Por consequência, geram dificuldades para os praticantes de simulação no momento de sua modelagem, demandam tempo e esforço adicionais e podem até mesmo distorcer significativamente os resultados do modelo (BILLER; GUNES, 2012), pois a fidelidade das respostas obtidas depende claramente da fidelidade dos MDEs utilizados (NELSON *et al.*, 2021). Por essa razão, frequentemente dados e MDEs com problemas são apontados como desafios e razões pelas quais um modelo não é validado (GREASLEY; EDWARDS, 2021; SARGENT, 2013). Os autores reforçam que, à medida que modelos de simulação se tornam ferramentas cada vez mais usadas para projeto e análise de sistemas complexos, é importante desenvolver MDEs suficientemente flexíveis para capturar as propriedades probabilísticas dos dados obtidos.

Enquanto em projetos pontuais de SED o impacto desses problemas no tempo necessário para análise pode ser eventualmente tolerado, isso pode não se repetir em projetos em que o modelo se tornará uma ferramenta de tomada de decisão recorrente, como gêmeos digitais.

Nesses casos, é necessária a atualização frequente dos dados de entrada, garantindo a qualidade destes e de seus MDEs, o que é um desafio (DOS SANTOS *et al.*, 2022).

Desenvolver e avaliar novas formas de modelagem de dados de entrada, que permitam reduzir o tempo, esforço e expertise necessários por parte do modelador, pode trazer impactos práticos relevantes. E é nesse contexto que o presente estudo se encontra e contribui com o desenvolvimento e avaliação do desempenho de GANs para modelagem de dados de entrada. É importante ressaltar que o uso de GANs também demanda conhecimentos técnicos, mas, em se tratando de uma ferramenta capaz de lidar com dados complexos e de diferentes características, espera-se que ferramentas de modelagem de dados de entrada baseadas em GANs ou técnicas similares sejam capazes de responder conjuntamente a diversos desafios encontrados por modeladores. Por fim, o presente trabalho também contribuirá ao desenvolver algoritmos com a implementação de passos para utilização de GANs no contexto de modelagem de dados de entrada.

1.2.2. Impacto acadêmico

O ineditismo desta tese se encontra na elaboração de uma proposta de modelagem de dados de entrada em projetos de SED com a utilização de GANs. O estudo sobre essas áreas de forma conjunta é limitado, não havendo bases teórica ou empírica até o presente momento.

Para demonstrar o interesse pelas áreas envolvidas, no dia 07 de agosto de 2022, foram conduzidas buscas de estudos na literatura utilizando os termos relacionados a SED, GAN e MDE. A busca foi realizada na base de dados Scopus® para artigos e artigos de congressos, considerando os campos de título, resumo e palavras-chave. Para SED, a expressão buscada foi "*Discrete-Event Simulation*". Para GANs, "*Generative Adversarial Net**", com o asterisco permitindo encontrar variações de termos. E, para MDE, "*Input Model**" ou "*Input Data Model**". Os resultados podem ser observados na Figura 1.4.

É possível notar que as áreas de DES e MDE são mais antigas, com artigos datados desde a década de 1960. Já os estudos sobre GANs são mais recentes, visto que seu artigo seminal foi publicado em 2014. Nota-se ainda que ao longo dos anos o interesse nas áreas de DES e MDE cresceu, estabilizando-se a partir da década de 2010. Já a área de GANs apresentou um desenvolvimento acentuado, superando em cerca de cinco vezes o número de publicações sobre DES em 2021 e 2022 e 20 vezes o número de publicações sobre MDE nos mesmos anos, o que demonstra o grande interesse pela área e a oportunidade de trazer este tópico relevante para as discussões nas demais áreas relacionadas à tese. Ressalta-se que as quedas observadas

nos últimos dois anos do gráfico se referem a 2022 (ainda em curso) e a 2023 (incluindo poucos artigos já programados para publicação).

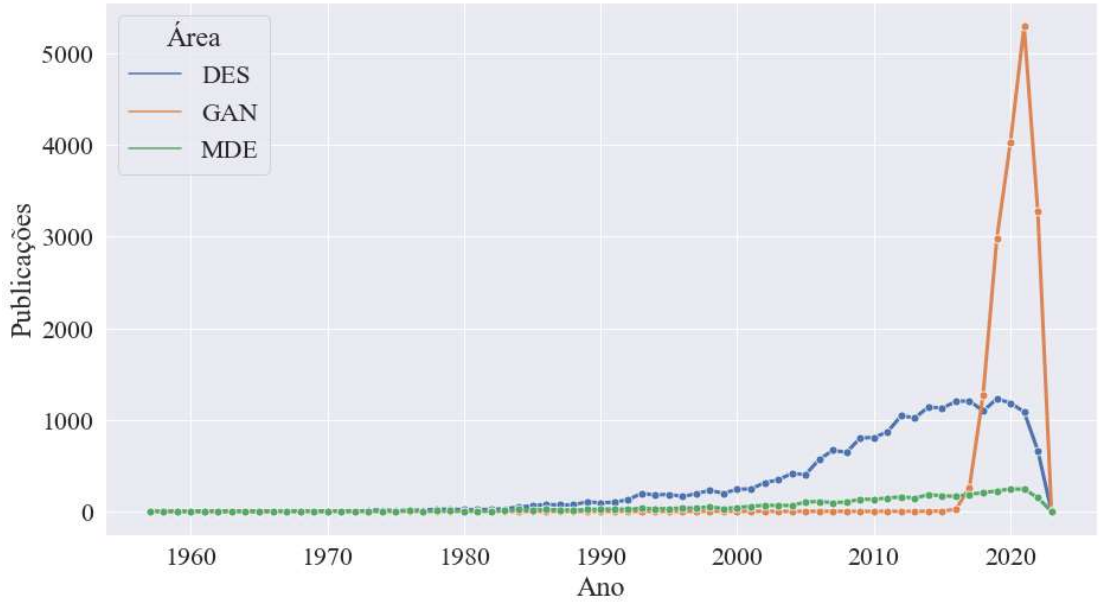


Figura 1.4 – Evolução das publicações sobre SED, MDE e GANs ao longo dos anos

Para confirmar o ineditismo e ilustrar o campo de publicações conjuntas nas áreas, na mesma data foram conduzidas buscas de estudos utilizando combinações dos termos citados anteriormente. Os resultados estão apresentados na Tabela 1.1.

Tabela 1.1 – Resultados das buscas de publicações nas áreas relacionadas à tese

Abrangência	**Expressão de busca**	**Intervalo de publicações**	**Quantidade**
SED	*"Discrete Event Simulation"*	1966 a 2023	20.587
GAN	*"Generative Adversarial Net*"*	2014 a 2023	17.161
MDE	*"Input Model*" OR "Input Data Model*"*	1957 a 2022	3.636
SED + GAN	*"Discrete Event Simulation" AND "Generative Adversarial Net*"*	2018	1
SED + MDE	*"Discrete Event Simulation" AND ("Input Model*" OR "Input Data Model*")*	1982 a 2022	41
GAN + MDE	*"Generative Adversarial Net*" AND ("Input Model*" OR "Input Data Model*")*	2020 a 2022	4
Todos	*"Discrete Event Simulation" AND "Generative Adversarial Net*" AND ("Input Model*" OR "Input Data Model*")*	-	0

Nesse caso, é possível visualizar que, individualmente, as áreas de SED, GAN e MDE apresentam vasta literatura. Porém, poucos estudos foram realizados de forma a integrar os tópicos e a buscar sinergias, com exceção dos trabalhos envolvendo SED e MDE, já que a modelagem de dados de entrada é uma etapa tradicional dos projetos de simulação.

Quanto aos estudos de SED e GANs, apenas o de Nitanda e Suzuki (2018) foi encontrado. Entretanto, o estudo não está relacionado à SED, tendo sido erroneamente indexado de forma automática pela plataforma Scopus com essa palavra-chave.

Já para a busca conjunta de GANs e MDE, Mehra *et al.* (2021) e Mehra *et al.* (2020) utilizam o termo "*input model*" para se referirem a modelos de teclado para entrada em dispositivos móveis ("*keyboard input models*"). Nie *et al.* (2022) denotam o conjunto de camadas de neurônios de um dos componentes das GANs como "*input model*". E Montevechi *et al.* (2021) apresentam resultados originados desta tese.

Por fim, nenhum estudo foi encontrado unindo as três áreas. Nota-se, portanto, a existência de uma lacuna na literatura para proposta e avaliação da utilização de GANs para modelagem de dados de entrada em projetos de SED, tornando este trabalho uma contribuição especialmente para a própria área de SED e para a área de MDE. Adicionalmente, o estudo contribui para a área de GANs ao descrever sua aplicação em dados tabulares, já que as publicações sobre o tema são majoritariamente relacionadas a imagens, vídeos e sons (ALQAHTANI; KAVAKLI-THORNE; KUMAR, 2021; FARAJZADEH-ZANJANI *et al.*, 2022). Para demonstrar essa tendência, os resultados da busca de artigos relacionados a GANs foram analisados bibliometricamente utilizando o *software* VOSviewer®. A Figura 1.5 identifica padrões de relacionamentos entre os termos presentes nos títulos dos trabalhos, subdividindo-os em três agrupamentos.

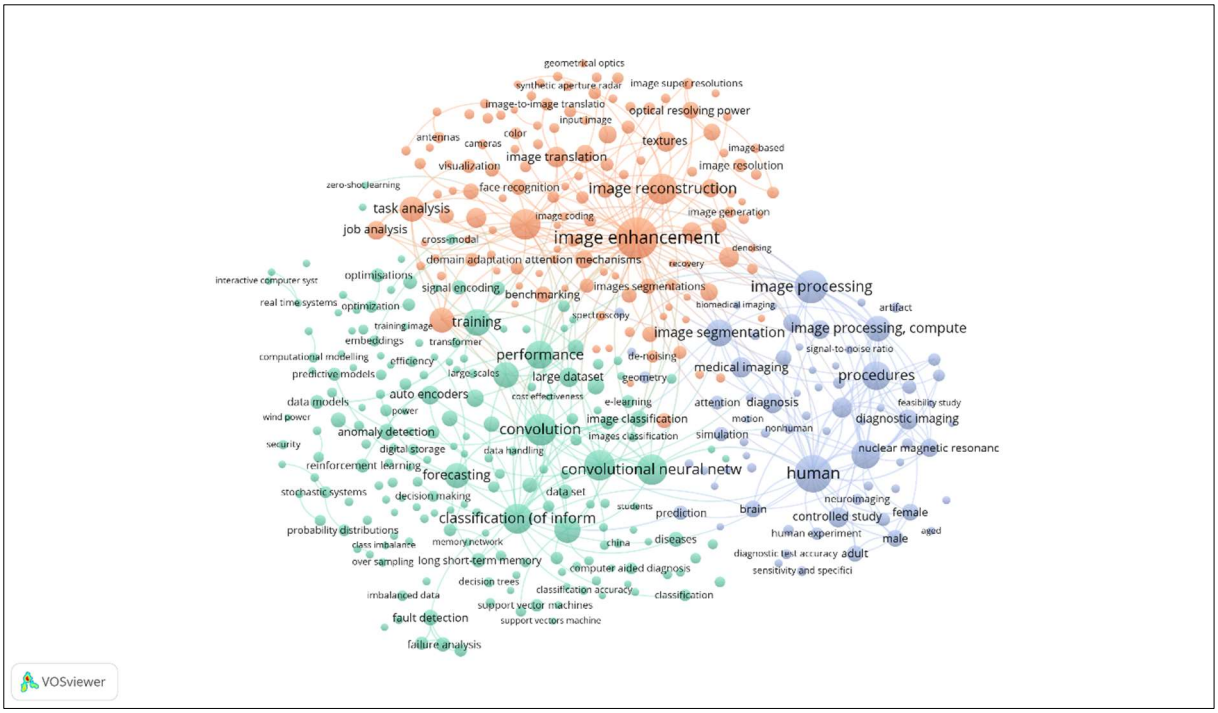


Figura 1.5 – Nuvem de palavras e seus relacionamentos para estudos na área de GANs

O 1º agrupamento (Figura 1.6), em laranja, indica o tipo majoritário de dados em estudos de GANs: imagens. Contém termos como “visão computacional”, “detecção de objetos” e expressões relacionadas à qualidade de imagens, tais como “melhoria”, “reconstrução”, “resolução” e “super resolução”. O presente estudo se diferencia por aplicar GANs para processamento de dados tabulares.

Já o 2º agrupamento (Figura 1.7), em verde, contém termos relacionados à arquitetura e treinamento de GANs, também com forte presença de expressões relacionadas a arquiteturas de RNAs para processamento de imagens, como “convolucional”, referente às Redes Neurais Convolucionais (RNCs), amplamente utilizada na área (ALQAHTANI; KAVAKLI-THORNE; KUMAR, 2021). Por focar no processamento de dados tabulares, esta tese se diferencia ao demandar um tipo de arquitetura de RNAs que, apesar de mais simples, é distinto do amplamente utilizado pela comunidade atuante em GANs. Trata-se de um potencial desafio, já que parte das boas práticas e recomendações na literatura da área podem estar vinculadas e restritas à utilização de RNCs.

O 3º e último agrupamento (Figura 1.8), em roxo, reúne termos relacionados à área predominante de aplicação de GANs: saúde. São encontrados termos relacionados a procedimentos de diagnóstico, como “humano”, “diagnóstico”, “exame de imagem”, “ressonância magnética”, “raio-X”, “tomografia” e “neuroimagem”. Naturalmente, o presente

estudo se diferencia por não se restringir à área de saúde, englobando, de forma geral, dados de entrada utilizados em projetos de SED.

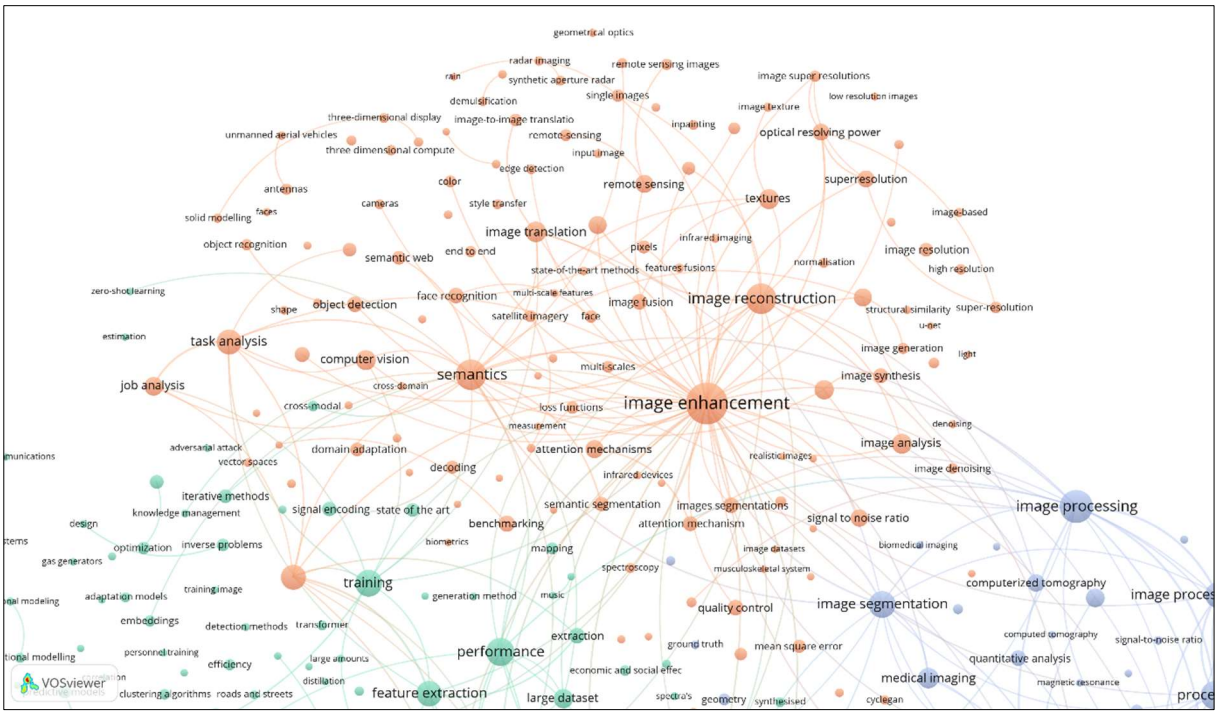


Figura 1.6 – 1º agrupamento de termos presentes em títulos de estudos sobre GANs

Por fim, vale ressaltar que, de forma geral, o potencial de impacto positivo de GANs na área de Pesquisa Operacional foi reconhecido por Kraus, Feuerrigel e Oztekin (2020) em seu estudo avaliando aplicações de AP e suas implicações à área. Os autores apontam como roteiro para pesquisas futuras a adoção de GANs, afirmando que seu benefício ainda precisa ser demonstrado.

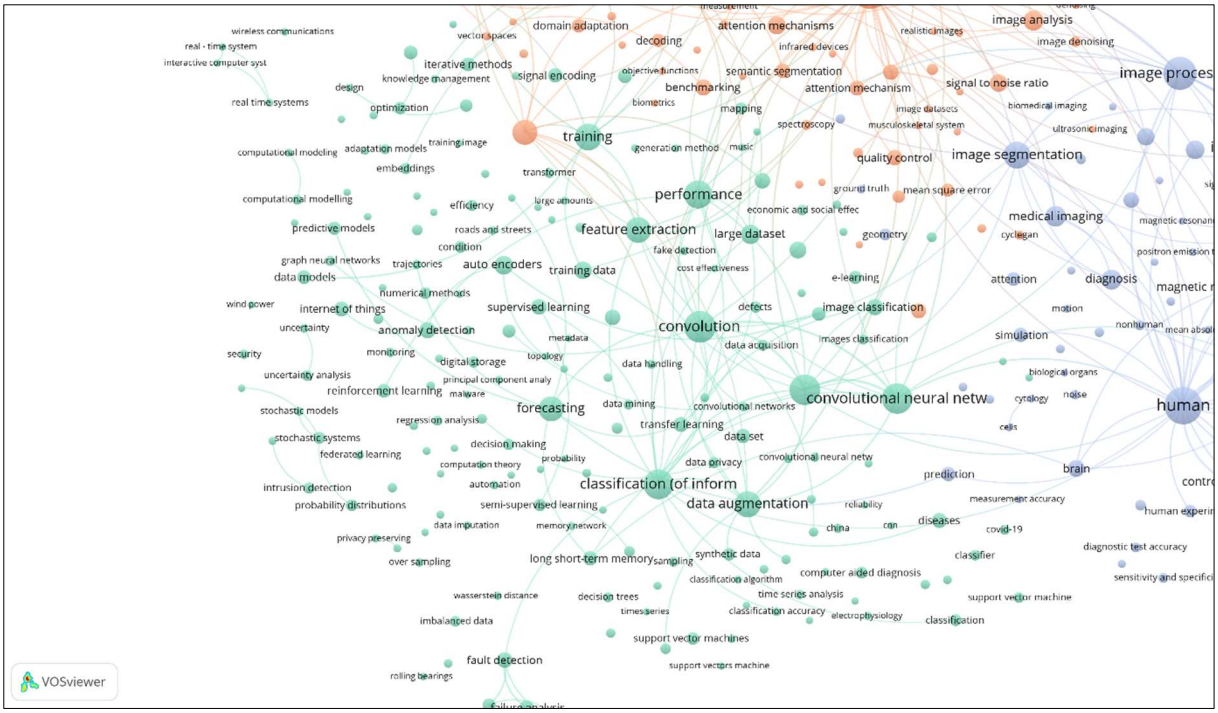


Figura 1.7 – 2º agrupamento de termos presentes em títulos de estudos sobre GANs

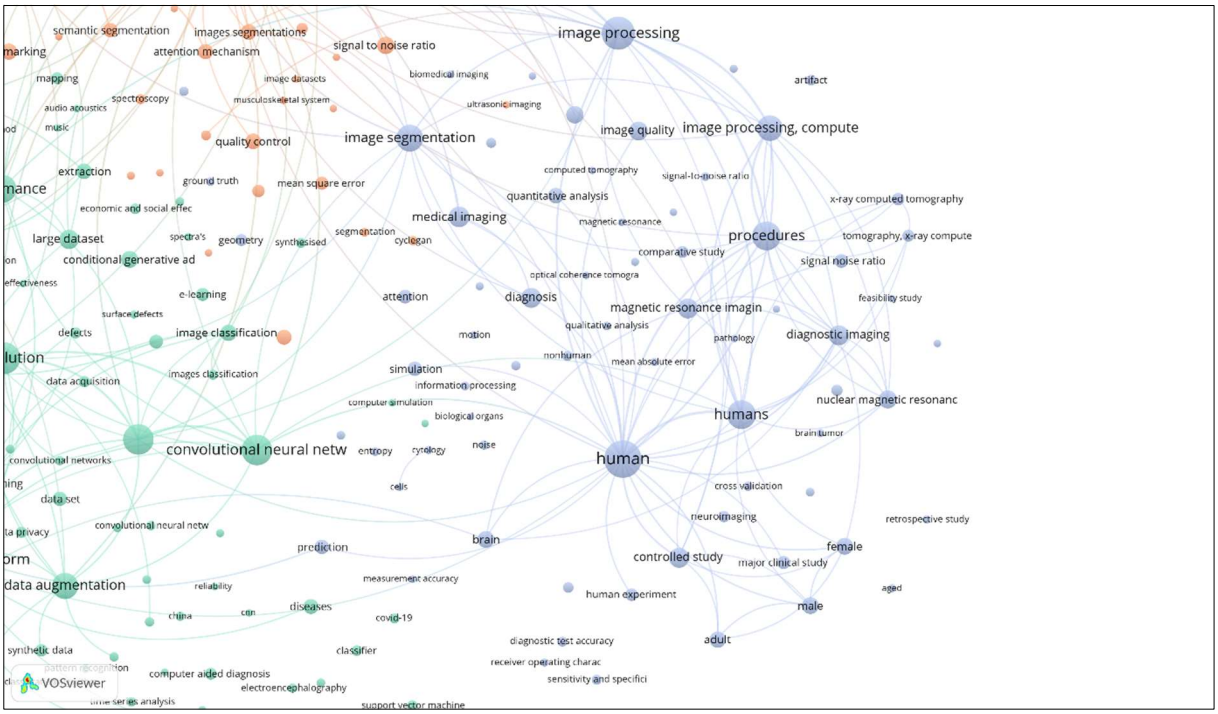


Figura 1.8 – 3º agrupamento de termos presentes em títulos de estudos sobre GANs

1.3. Questões de pesquisa e hipóteses

Considerando o desafio da modelagem de dados de entrada em projetos de SED e os possíveis impactos discutidos na seção anterior, esta pesquisa pode ser resumida nas seguintes questões.

- **Questão 01:** GANs são adequadas para modelagem de dados de entrada em projetos de SED?
- **Questão 02:** GANs representam uma alternativa competitiva para modelagem de dados de entrada em projetos de SED?
- **Questão 03:** Qual o impacto da quantidade de dados coletados na qualidade dos MDEs baseados em GANs?

Em relação à primeira questão, um desempenho "adequado" está relacionado, nesta tese, ao grau de acurácia do MDE, uma medida de qualidade de representação que será mais bem discutida na seção 1.5.3.4.

De forma semelhante, a acurácia do MDE será levada em consideração para responder a segunda e a terceira questão. O termo "competitiva" estará relacionado à comparação dos resultados obtidos por GANs com os resultados obtidos por MDEs tradicionalmente utilizados em projetos de SED, as DEPs.

Como direcionadores, foram estabelecidas quatro hipóteses relacionadas às questões de pesquisa, as quais estão apresentadas no Quadro 1.1.

Quadro 1.1 – Hipóteses de pesquisa

Questões de pesquisa	Hipóteses
Questão 01: GANs são adequadas para modelagem de dados de entrada em projetos de SED?	**H1:** GANs geram MDEs com forte qualidade de representação.
Questão 02: GANs representam uma alternativa competitiva para modelagem de dados de entrada em projetos de SED?	**H2:** Para dados de maior complexidade, GANs geram MDEs com maior qualidade de representação do que DEPs.
	H3: Para dados de baixa complexidade, GANs geram MDEs com menor qualidade de representação do que DEPs.
Questão 03: Qual o impacto da quantidade de dados coletados na qualidade dos MDEs baseados em GANs?	**H4:** Assim como para modelagem baseada em DEPs, a quantidade de dados disponíveis é significativa para a qualidade de representação do MDE.

Retomando a definição realizada na seção 1.2.1, são considerados dados complexos os dados não IID ou com comportamentos probabilísticos que não podem ser representados adequadamente por DEPs univariadas.

Em relação às hipóteses H2 e H3, nota-se a diferenciação de expectativa em relação à complexidade dos dados a serem modelados. Para dados complexos, espera-se que GANs sejam capazes de capturar mais características das distribuições reais, oferecendo maior qualidade de representação do que DEPs. Por outro lado, para dados simples, é esperado que DEPs performem melhor. Por exemplo, se a distribuição real segue um comportamento normal, a própria DEP normal será uma boa candidata e procedimentos tradicionais de ajuste de distribuições precisarão calcular apenas dois parâmetros, média e desvio, para obter uma boa aproximação. Já um MDE baseado em GANs precisará passar por um processo de treinamento mais difícil, com o aprendizado de uma quantidade significativamente maior de parâmetros (pesos e vieses das RNAs, o que será discutido na seção 2.4).

Já em relação à hipótese H4, será necessário avaliar o desempenho de MDEs baseados em GANs para diferentes tamanhos amostrais a serem fornecidos para treinamento das RNAs.

1.4. Objetivos

O principal objetivo desta tese é avaliar o desempenho de GANs para modelagem de dados de entrada, respondendo as questões e testando as hipóteses definidas na seção anterior.

Os seguintes objetivos específicos também foram estabelecidos:

- Propor um método para modelagem de dados de entrada baseada em GANs (MDE-GANs);
- Desenvolver programas que realizem as etapas definidas para o método proposto;
- Avaliar os resultados do método por meio de casos teóricos e reais.

Em relação aos tipos de casos a serem considerados, esta tese tem como objetivo avaliar a MDE-GANs em parte das classes de dados apresentadas por Law (2014) e discutidas na seção 1.2.1. São elas: dados homogêneos e heterogêneos e dados dependentes e independentes. Nesse sentido, serão avaliados os resultados para dados IID, mas também para distribuições multivariadas e multimodais.

Casos teóricos referentes a distribuições paramétricas serão utilizados para responder as questões de pesquisa e testar as hipóteses formuladas, pois permitem uma análise mais controlada (sabe-se de antemão a distribuição verdadeira e suas características) e por não apresentarem restrição de tamanho amostral para os testes. Os demais casos, inclusive os reais, descritos no Capítulo 5, serão utilizados principalmente com a finalidade de demonstração do uso de MDE-GANs.

1.5. Materiais e métodos

Esta seção descreve os materiais necessários para condução do estudo e a base metodológica escolhida. São discutidos a classificação da pesquisa, o método e as etapas para sua condução.

1.5.1. Materiais

Dentre os materiais necessários, destaca-se o interpretador da linguagem de programação Python® (versão 3.10), a qual será utilizada para implementação de todos os algoritmos da proposta. A comunidade atuante em Python é considerada ativa e disponibiliza uma grande quantidade de bibliotecas para os mais diferentes propósitos. Por exemplo, o repositório *PyPi* reúne aproximadamente 400 mil projetos compartilhados (PYTHON SOFTWARE FOUNDATION, 2022). O Quadro 1.2 lista as principais bibliotecas Python utilizadas no presente estudo e suas aplicações.

Quadro 1.2 – Bibliotecas Python utilizadas para implementação da proposta

Biblioteca Python	**Versão**	**Aplicação**
TensorFlow® (TENSORFLOW, 2022)	2.4.1	Criar modelos de aprendizado de máquina
Scikit-Learn® (SCIKIT-LEARN, 2022)	0.23.2	
Pandas® (PANDAS, 2022)	1.1.3	Manipulação e análise de dados
NumPy® (NUMPY, 2022)	1.19.2	Computação científica
SciPy® (SCIPY, 2022)	1.5.2	
Seaborn® (SEABORN, 2022)	0.11.0	Visualização de dados
Matplotlib (MATPLOTLIB, 2022)	3.3.2	

Além disso, foram utilizados os programas FlexSim (versão 22.0), para avaliação das formas de importação dos MDEs, Excel®, para armazenagem de dados e Minitab®, para análises estatísticas.

1.5.2. Classificação da pesquisa

O presente estudo aplica a forma clássica de classificação de pesquisas científicas em Engenharia de Produção, definida por Turrioni e Mello (2012) e que segmenta os estudos conforme sua natureza, objetivos, abordagem e método, como apresentado na Figura 1.9.

Segundos os autores, quanto à natureza, pesquisas básicas buscam o progresso científico sem a preocupação de utilizá-lo na prática, criando generalizações, princípios e leis. Já as de natureza aplicada possuem interesse prático, avaliando os resultados em problemas que ocorrem na realidade. O presente estudo se classifica como aplicado, pois há o interesse de

avaliar os resultados da proposta de MDE-GANs em situações que ocorrem no cotidiano da área de Modelagem e Simulação.

Já em relação aos objetivos, as pesquisas podem ser classificadas em exploratórias, visando maior familiaridade com o problema; descritivas, quando descrevem as característica de uma população ou fenômeno; explicativas, identificado os fatores que determinam ou contribuem para a ocorrência de fenômenos; e normativas, interessadas no desenvolvimento de políticas, estratégias e ações que permitam aperfeiçoar os resultados existentes na literatura ou comparar estratégias para resolução de um problema. Devido ao interesse comparativo, de avaliar a qualidade de representação de MDEs baseados em GANs e em DEPs, o presente trabalho se classifica como normativo.

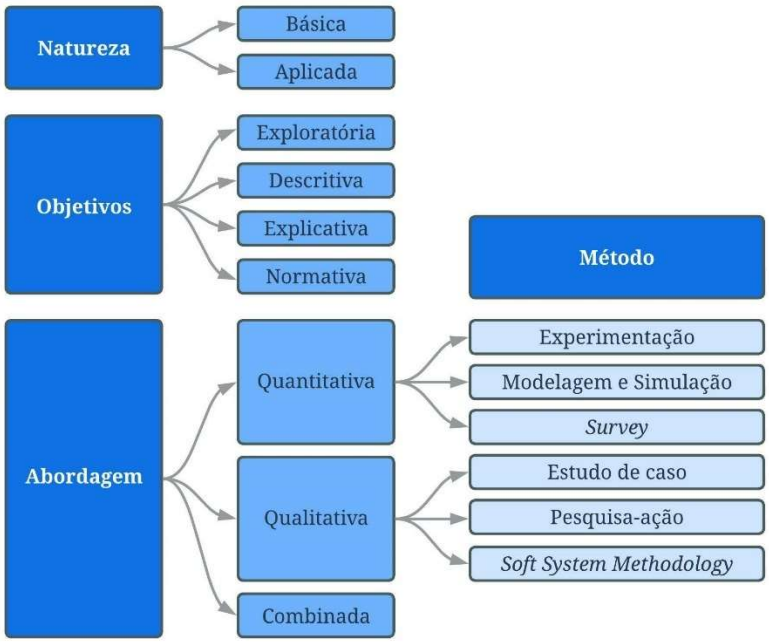


Figura 1.9 – Classificação de pesquisas em Engenharia de Produção
Fonte: Adaptado de Turrioni e Mello (2012)

Considerando a dimensão de abordagem, uma pesquisa pode ser considerada como quantitativa, quando traduz opiniões e informações em números para analisá-las; qualitativa, buscando avaliar aspectos subjetivos e de forma intuitiva; ou combinada, quando combina as duas modalidades anteriores. A pesquisa empírica quantitativa se preocupa em testar a validade de modelos científicos desenvolvidos em pesquisa teórica quantitativa ou avaliar a usabilidade e a performance destas soluções quando aplicadas a processos reais. As observações empíricas são direcionadas por hipóteses baseadas em teorias desenvolvidas previamente. Tais projetos não devem ser confundidos com projetos de melhoria, que escolhem teorias e frequentemente

assumem que estas serão aplicáveis e irão performar bem, não sendo desenhados para, de fato, testar cientificamente sua validade (BERTRAND; FRANSOO, 2002).

A principal avaliação a ser realizada por esta tese se refere à qualidade de representação de MDEs, considerando-a de forma quantitativa por meio de uma medida de acurácia, diretamente relacionada às hipóteses do estudo. Por essa razão, trata-se de uma pesquisa de abordagem quantitativa.

Dentre os métodos de pesquisa que se enquadram nessa abordagem, este trabalho adota a experimentação, empregada quando se determina um objeto de estudo, escolhem-se variáveis que seriam capazes de influenciá-lo, definem-se formas de controle e de observação dos efeitos gerados. Nesse sentido, os objetos de estudos serão diferentes amostras de dados de entrada que passarão por modelagem utilizando a proposta MDE-GANs. O controle será realizado por meio da comparação com os resultados obtidos pela modelagem clássica com DEPs. Por fim, o próprio comportamento das diferentes distribuições e a quantidade de dados de entrada disponíveis para treinamento das GANs são considerados, na perspectiva do algoritmo de aprendizado, fatores incontroláveis para os quais se deseja avaliar a robustez dos resultados.

É importante ressaltar que a área de Modelagem e Simulação está diretamente relacionada à tese, mas não constitui o método de pesquisa, pois as questões formuladas não serão respondidas por meio de resultados de modelos de simulação ou por cenários realizados nestes, mas, sim, pela medição da qualidade de representação diretamente nos MDEs obtidos. Como dados de entrada são apenas um dos fatores que influenciam os resultados e a acurácia de modelos de simulação (SARGENT, 2013), melhorias em sua qualidade de representação podem até mesmo aumentar a diferença entre indicadores reais e simulados (para validação), tornando evidentes, na realidade, outros problemas que o modelo possuía, mas que estavam encobertos e se "anulavam" com a qualidade insuficiente dos MDEs. Assim, avaliar empiricamente ganhos na acurácia de modelos de simulação devido à proposta de MDE-GANs torna-se difícil, visto que em aplicações reais não será possível conhecer previamente outros erros que possam impactar os resultados do modelo, especialmente se estes estiverem encobertos por problemas nos MDEs originalmente utilizados.

1.5.3. Método

Bertrand e Fransoo (2002) sugerem passos para condução da pesquisa empírica quantitativa, os quais serão adotados para este trabalho e estão apresentados na Figura 1.10. As seções a seguir descrevem cada um desses passos.

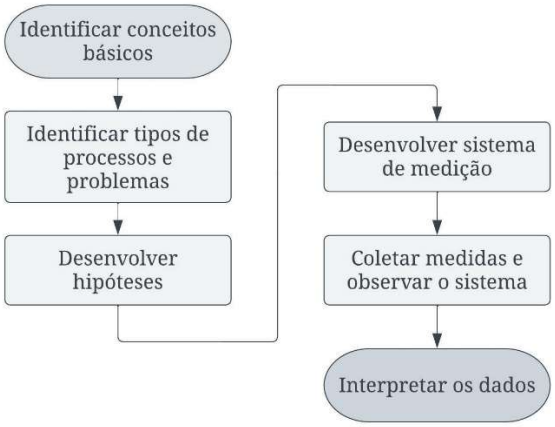


Figura 1.10 – Passos para a condução de pesquisas empíricas quantitativas
Fonte: Adaptado de Bertrand e Fransoo (2002)

1.5.3.1. Identificar conceitos básicos

Segundo os autores, o primeiro passo é a identificação dos pressupostos, reunindo conceitos teóricos. Essa etapa foi realizada por meio de revisão da literatura e está registrada nas seções anteriores e no capítulo 2, com a discussão de aspectos teóricos sobre SED, modelagem de dados de entrada e GANs, dentre outros temas. Com isso, foi possível compreender desafios para modelagem de dados de entrada, o potencial da utilização de GANs e os mecanismos essenciais para construção da proposta.

1.5.3.2. Identificar tipos de processos e problemas

O segundo passo se refere à identificação dos tipos de processos e problemas que deverão ser considerados como objeto de estudo e critérios objetivos para o enquadramento nesses tipos. Os objetos de estudo serão conjuntos de dados teóricos e reais que passarão pelo processo de modelagem de dados de entrada. Considerando as classes de dados escolhidas na definição dos objetivos da tese, os objetos de estudo deverão se enquadrar nas categorias homogêneo ou heterogêneo e independente ou dependente. Quanto à presença de autocorrelação, como limitação, o presente estudo incluirá apenas casos não autocorrelacionados. Os objetos de estudo estão descritos no Capítulo 5.

1.5.3.3. Desenvolver hipóteses

O desenvolvimento de hipóteses compreende o terceiro passo. Segundo os autores, as hipóteses devem se referir a variáveis ou fenômenos que possam ser medidos ou observados de

forma objetiva. Quanto mais hipóteses forem formuladas, mais forte será a pesquisa. As hipóteses do presente estudo foram definidas na seção 1.3.

1.5.3.4. Desenvolver sistema de medição

Todas as hipóteses estão relacionadas à qualidade de representação, sendo necessário defini-la de forma objetiva, o que será feito no próximo passo, o de definição de um sistema de medição. Para Bertrand e Fransoo (2002), trata-se de uma atividade crucial e que deve ser documentada, pois, frequentemente, não há consenso quanto à definição das variáveis e à sua forma de medição. O pesquisador deve desenvolver seu próprio sistema de medição, identificando as características relevantes do processo e como medi-las.

Law (2014) subdivide as técnicas para avaliação da representatividade dos MDEs em procedimentos gráficos e testes estatísticos de qualidade de ajuste.

Dentre os procedimentos gráficos, o autor cita histogramas e gráficos Quantil-Quantil (Q-Q). Os primeiros servem para comparação das frequências observadas nos dados reais e nos dados gerados pelo MDE, como ilustrado na Figura 1.11. Caso o MDE seja uma boa representação, as frequências calculadas para as mesmas faixas de valores serão semelhantes.

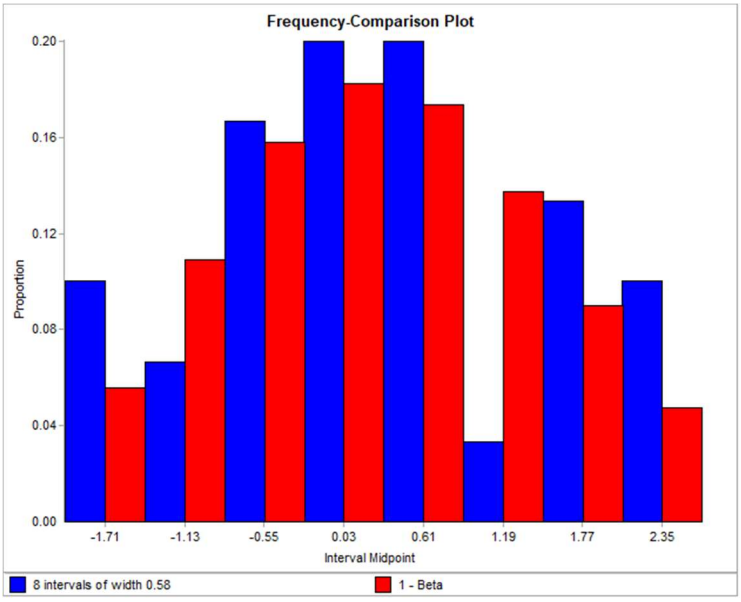


Figura 1.11 – Exemplo de histograma comparativo

Já os gráficos Q-Q demonstram a relação entre os quantis dos dados reais e os quantis calculados por meio da função distribuição da DEP hipotética, como ilustrado na Figura 1.12. Caso o MDE não seja do tipo DEP e a função distribuição for desconhecida, os quantis do MDE podem ser calculados a partir de uma amostra de dados gerados pelo modelo. Se o MDE for

uma boa representação dos dados reais, os pontos presentes no gráfico (pares de quantis) estarão aproximadamente distribuídos ao longo da diagonal com intercepto na origem e inclinação igual a 1.

Os procedimentos gráficos contribuem para a visualização da qualidade de representação de um MDE e, por essa razão, comparações por histogramas e gráficos Q-Q serão incluídos nas avaliações dos resultados dos MDEs obtidos tanto por GANs quanto por DEPs. Entretanto, em se tratando de uma pesquisa quantitativa, também se faz necessária a definição de um sistema de medição com essa característica.

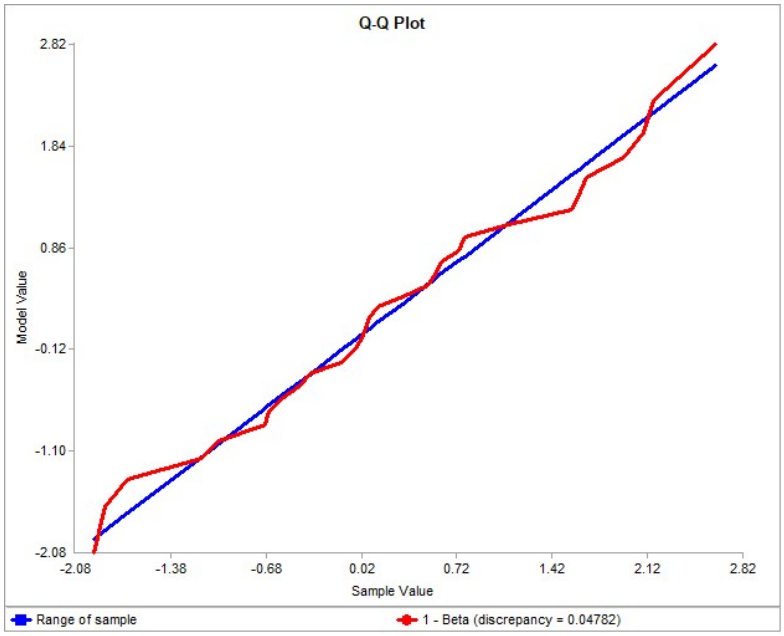


Figura 1.12 – Exemplo de gráfico Q-Q

Nesse sentido, Law (2014) descreve os testes estatísticos de qualidade de ajuste, citando os testes Anderson-Darling, Kolmogorov-Smirnov e Qui-Quadrado. Os dois primeiros requerem que a distribuição hipotética para o MDE seja paramétrica, o que não é o caso de MDEs baseados em GANs. Já o teste Qui-Quadrado se aproxima da comparação entre histogramas. Entretanto, segundo o autor, tem a desvantagem de requisitar a definição do número e tamanho de subdivisões (*bins*) dos dados, não existindo uma recomendação definitiva, problema que se acentua quando considerados MDEs multivariados, os quais fazem parte do rol de objetos de estudo desta tese. Por essas razões, optou-se pela avaliação da qualidade de representação utilizando uma das técnicas adotadas para avaliação de GANs, o teste classificador para duas amostras, mais conhecido por seu nome em inglês, *Classifier 2-Sample Test* (C2ST) (BORJI, 2019). Trata-se de um teste que, segundo os autores, é capaz de

identificar divergências entre distribuições e que apresenta um resultado de fácil entendimento: uma medida com os limites bem definidos [0, 1].

Em sua forma original, o objetivo do C2ST é avaliar se duas amostras são provenientes da mesma distribuição. Mais precisamente, o teste avalia se há evidências suficientes para se rejeitar a hipótese nula de que as duas distribuições de probabilidade, V (sintética) e W (real), são iguais, constituindo, assim, as hipóteses apresentadas na Equação 1.1. Caso H_0 seja verdadeira, 100% dos dados gerados pelo MDE poderiam ser considerados como oriundos de W, o que denotaremos como uma acurácia do MDE (A_{MDE}) igual a 100%.

$$\begin{aligned} H_0&: V = W \\ H_1&: V \neq W \end{aligned} \tag{1.1}$$

Como por definição um modelo de dados de entrada é uma aproximação e, assim, não será idêntico à distribuição real, sabe-se de antemão que H_0 nunca será verdadeira. Por essa razão, o presente trabalho considera a noção de tolerância. Assim, é necessário redefinir as hipóteses conforme a Equação 1.2.

$$\begin{aligned} H_0&: A_{MDE} > 1 - \delta \\ H_1&: A_{MDE} < 1 - \delta \end{aligned} \tag{1.2}$$

Sendo δ a tolerância escolhida pelo modelador equivalente ao percentual aceitável de observações de V que não sustentem a hipótese nula original $V = W$. Naturalmente, surge a questão sobre quais níveis de δ são adequados, o que dependerá em muito da sensibilidade do modelo de simulação em relação aos dados de entrada sendo modelados. Ou seja, caso os resultados do modelo sejam significativamente impactados quando a variável de entrada for alterada, será mais interessante atuar com uma A_{MDE} mais alta. Porém, como orientação geral para a escolha dessa tolerância, o presente estudo sugere faixas de interpretação comumente utilizadas para o Kappa de Cohen, uma forma de medição de correlação amplamente aplicada para a análise da concordância e que tem contribuído também para estudos experimentais em áreas como Processamento de Linguagem Natural e Mineração de Texto (KOLESNYK; KHAIROVA, 2022; MCHUGH, 2012). A Tabela 1.2 apresenta os valores sugeridos para interpretação e planejamento da A_{MDE} seguindo os percentuais de dados confiáveis esperados para cada classe de concordância do Kappa de Cohen.

Tabela 1.2 – Valores de referência para interpretação da acurácia do MDE

Nível de acurácia do MDE (interpretação)	A_{MDE}
Nenhuma	0 – 4%
Mínima	4 – 15%
Fraca	15 – 35%
Moderada	35 – 64%
Forte	64 – 82%
Quase perfeita	82 – 100%

O C2ST é realizado a partir do acesso a duas amostras $S_V = \{\boldsymbol{x_1}, \dots, \boldsymbol{x_n}\} \sim V^n$ e $S_W = \{\boldsymbol{y_1}, \dots, \boldsymbol{y_n}\} \sim W^n$, sendo $\boldsymbol{x_i}$ e $\boldsymbol{y_i}$ vetores compostos por m variáveis do conjunto de dados de entrada e pertencentes ao mesmo espaço $\mathcal{X}$; e n o número de observações reais. O resultado é alcançado após as seguintes etapas.

Primeiramente, é necessário construir o conjunto de dados $\mathcal{D}$, formado por meio da união dos dados sintéticos e dos dados reais, rotulados com o atributo $l_i = 0$ se sintéticos e $l_i = 1$ se reais, conforme a Equação 1.3.

$$\mathcal{D} = \{(\boldsymbol{x_i}, 0)\}_{i=1}^{n} \cup \{(\boldsymbol{y_i}, 1)\}_{i=1}^{n} =: \{(\boldsymbol{z_i}, l_i)\}_{i=1}^{2n} \tag{1.3}$$

Em seguida, treinar um classificador binário $f : \mathcal{X} \to \{0, 1\}$ utilizando $\mathcal{D}$ e visando separar corretamente S_V e S_W.

O terceiro passo consiste em calcular a acurácia de classificação $\hat{t}$ conforme a Equação 1.4, sendo $\mathbb{I}$ a função indicador. A intuição por trás do teste é de que, se $V = W$, a acurácia da classificação deve permanecer próxima do acaso (0.5). Em outras palavras, sendo V e W tão próximos, o classificador não lograria êxito em separar as amostras S_V e S_W, classificando-as aleatoriamente como reais ou sintéticas. Por outro lado, caso o classificador performe melhor que o acaso, haveria indicação de que $V \neq W$.

$$\hat{t} = \frac{1}{n} \sum_{(\boldsymbol{z_i}, l_i) \in \mathcal{D}} \mathbb{I}\left(f(\boldsymbol{z_i}) = l_i\right) \tag{1.4}$$

Segundo Borji (2019), em princípio, qualquer classificador binário pode ser utilizado como f para computar o C2ST, sendo o algoritmo k-Vizinhos Próximos ou, em inglês, *k-Nearest Neighbors* (k-NN) uma boa escolha por demandar treinamento simples e pouca otimização de hiperparâmetros. Além disso, é considerado um algoritmo intuitivo, não paramétrico e que tem se provado altamente eficiente e efetivo para resolver vários tipos de problemas de classificação (ABU ALFEILAT *et al.*, 2019; NADKARNI, 2016). Por essas razões, o algoritmo k-NN foi escolhido para o presente estudo.

Conforme definido por Abu Alfeilat *et al.* (2019), contextualizado às definições realizadas até o momento, as etapas básicas realizadas pelo k-NN estão descritas no Algoritmo 1.

Algoritmo 1: Etapas básicas do k-NN

Input: Conjunto de dados $\mathcal{D}$
Output: Rótulo (previsto) para cada $\boldsymbol{z_i}$
1: Computar a distância entre $\boldsymbol{z_i}$ e todas as demais observações em $\mathcal{D}$
2: Escolher as k observações de $\mathcal{D}$ mais próximas a $\boldsymbol{z_i}$, denotando o grupo como P ($\in \mathcal{D}$)
3: Atribuir a $\boldsymbol{z_i}$ o rótulo l que for mais frequente em P

Os dois principais hiperparâmetros do classificador são a métrica de distância e o número de vizinhos (k). Em relação à primeira, foi utilizada a distância Euclidiana (ABU ALFEILAT *et al.*, 2019), por ser o padrão da biblioteca *Scikit-Learn*. Já a definição de k foi implementada para ocorrer de forma dinâmica por meio de uma estratégia de afinação (*tuning*) conhecida como busca em grade (*grid-search*) com validação cruzada. Nesse procedimento, é definida uma grande de possíveis valores para o hiperparâmetro e os dados a serem classificados são subdivididos em q grupos exclusivos. Para cada combinação de q – 1 grupos, o algoritmo encontra o melhor hiperparâmetro dentre as opções da grade e obtém a acurácia de teste no grupo remanescente (SEYEDZADEH *et al.*, 2019). Assim, a acurácia do k-NN será a média das q acurácias de teste obtidas. A grade de possíveis valores foi definida em múltiplos de $\sqrt{n}$, valor comumente recomendado como ponto de partida, sendo n o número de observações a serem classificadas (NADKARNI, 2016).

O quarto passo para execução do C2ST é constituir as hipóteses a serem testadas. É importante notar que t e A_{MDE} estão relacionados, mas apresentam valores diferentes. Caso $V = W$, como afirmado anteriormente, $E(t) = 0.5$, enquanto $E(A_{MDE}) = 1.0$. Por outro lado, se $V \neq W$, $E(t) = 1.0$ e $E(A_{MDE}) = 0.0$. Com isso, é possível converter A_{MDE} em t e vice-versa utilizando a Equação 1.5.

$$t = 0.5 + \frac{(1 - A_{MDE})}{2} \tag{1.5}$$

Como o algoritmo k-NN retornará diretamente t, para o enquadramento do resultado aos níveis de acurácias apresentados anteriormente, é necessário reformular as hipóteses definidas na Equação 1.2, obtendo suas versões equivalentes com a aplicação da Equação 1.5. Com isso, são estabelecidas as hipóteses apresentadas na Equação 1.6. Por fim, para se testar as hipóteses, é necessário computar o *p-value* utilizando o teste unilateral para uma proporção, visto que t é uma razão entre classificações corretas e o total de observações.

$$H_0: t < 0.5 + {}^{\delta}/_{2}$$
$$H_1: t > 0.5 + {}^{\delta}/_{2} \quad (1.6)$$

1.5.3.5. Coletar medidas e observar o sistema

O procedimento para coleta de medidas encontra-se sumarizado na Figura 1.13, emprestando a notação utilizada pelo IDEF-SIM (MONTEVECHI *et al.*, 2010), por facilitar a indicação de responsabilidades, regras de controle e de fluxo. Como afirmado anteriormente, a coleta de dados para análise das hipóteses formuladas nesta tese será focada nos objetos de estudo teóricos paramétricos. Visando obter evidências estatisticamente válidas, serão realizadas réplicas dos experimentos (modelagens de dados) para cada objeto de estudo. Isso pode ser alcançado pela amostragem de tantos conjuntos de dados quantas forem as réplicas. Tais conjuntos representam as amostras para treinamento das GANs ou para escolha de DEPs. Como ponto de partida, serão realizadas 10 réplicas.

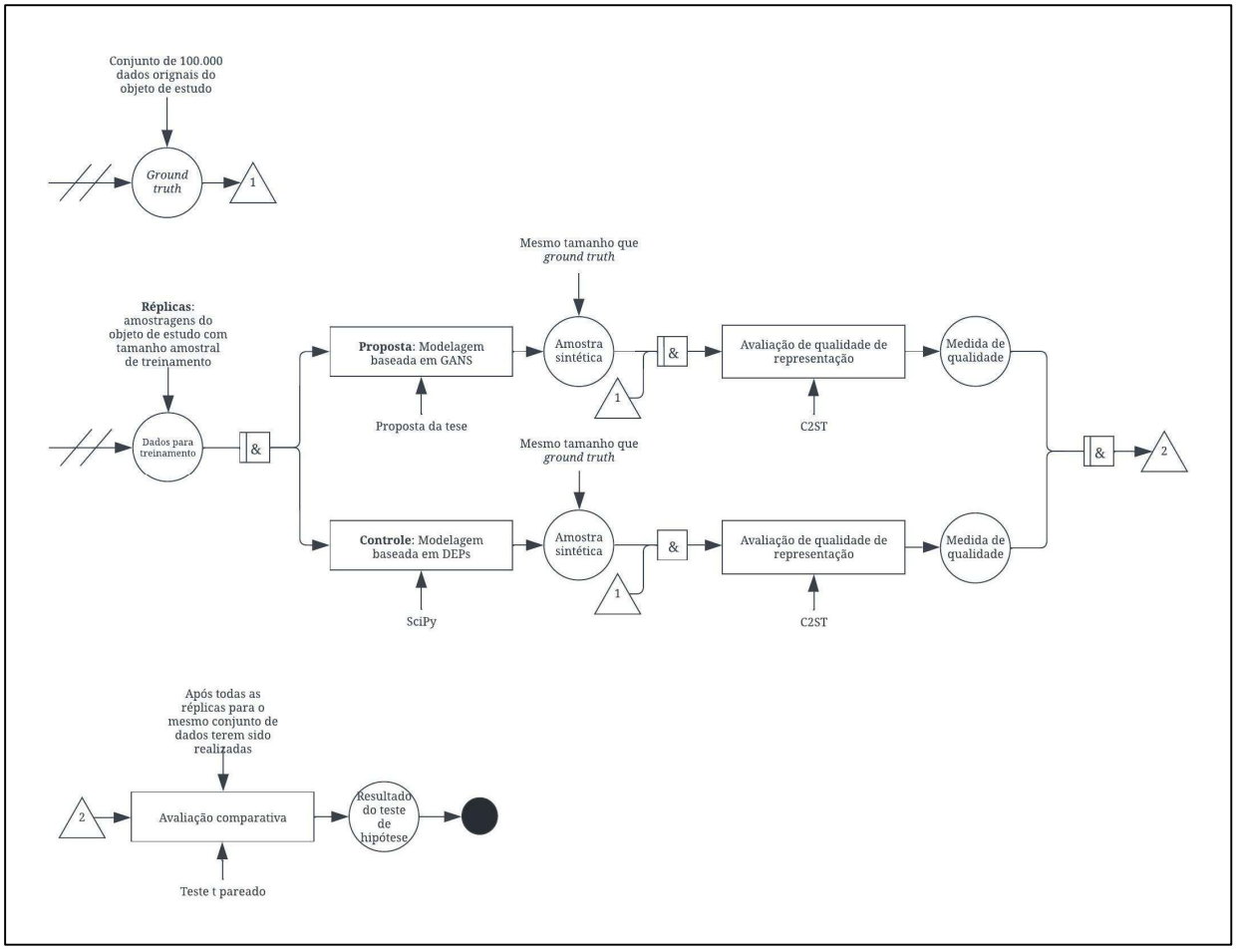


Figura 1.13 – Procedimento para coleta de medidas

Após a definição da amostra de dados de um objeto de estudo, os mesmos dados serão fornecidos para o algoritmo desenvolvido nesta tese (MDE-GANs) e para o controle, ou seja, para o processo de modelagem de dados de entrada por meio de DEPs utilizando a biblioteca SciPy. Obtidos os MDEs baseados em GANs e em DEPs, uma amostra sintética de tamanho elevado (100.000) é gerada utilizando cada MDE, sendo transmitida juntamente com o conjunto de dados sabidamente verdadeiros (*ground truth*) para avaliação via C2ST. Por sua vez, o C2ST retornará a A_{MDE} para cada um dos modelos.

Por fim, como os resultados obtidos pela proposta MDE-GANs e por DEPs sempre partirão da mesma amostra utilizada para treinamento, os valores podem ser avaliados de forma pareada. Assim, após a conclusão de todas as réplicas, os conjuntos de resultados de A_{MDE} para cada um dos tipos de modelo são fornecidos para comparação por meio do teste t pareado, que retornará a avaliação estatística sobre a existência ou não de evidência de diferença entre as performances dos métodos.

1.5.3.6. Interpretar os dados

A última etapa do método se refere à interpretação dos dados coletados, o que deve ser feito para investigar as hipóteses formuladas para a tese. Nesse sentido, dentre as atividades de interpretação, destacam-se as indicadas no Quadro 1.3.

Quadro 1.3 – Atividades de interpretação dos dados coletados

Hipóteses	**Atividades de interpretação**
H1: GANs geram MDEs com forte qualidade de representação.	Avaliar para o conjunto de réplicas dos objetos de estudo os níveis obtidos de A_{MDE} conforme a Tabela 1.2, especialmente para as faixas "forte" e "quase perfeita".
H2: Para dados de maior complexidade, GANs geram MDEs com maior qualidade de representação do que DEPs.	Comparar as acurácias de MDE-GANs e DEPs por meio do teste t pareado.
H3: Para dados de baixa complexidade, GANs geram MDEs com menor qualidade de representação do que DEPs.	
H4: Assim como para modelagem baseada em DEPs, a quantidade de dados disponíveis é significativa para a qualidade de representação do MDE.	Comparar os resultados de A_{MDE} em face dos diferentes tamanhos amostrais testados.

1.6. Estrutura da tese

O restante da tese está estruturado da seguinte forma. O Capítulo 2 apresenta os principais conceitos teóricos sobre as áreas relacionadas. Já os Capítulos 3 e 0 descrevem, respectivamente, a proposta de MDE-GANs e sua implementação em Python. Os objetos de

estudo e os resultados das aplicações são apresentados respectivamente nos Capítulos 5 e 6. Por fim, o Capítulo 7 traz as conclusões da tese.

2. FUNDAMENTAÇÃO TEÓRICA

O presente capítulo reúne os principais conceitos sobre SED, MDE e GANs.

2.1. Simulação a Eventos Discretos

Estudar cientificamente os sistemas, muitas vezes, requer realizar suposições de suas relações matemáticas e/ou lógicas, constituindo um modelo, que representa o comportamento do sistema. A simplicidade de um modelo pode permitir que sua análise seja realizada por meio de métodos matemáticos, como álgebra e cálculo, resultando em soluções analíticas. Entretanto, muitos sistemas são demasiadamente complexos para esse tipo de avaliação e precisam ser estudados por meio de simulação (LAW, 2014).

Banks (1998) define a simulação como a imitação do funcionamento de sistemas do mundo real, sejam estes existentes ou não, gerando histórias que, apesar de artificiais, podem ser utilizadas para a realização de inferências sobre as características operacionais do sistema. Assim, é possível responder a questões do tipo “e se?” e utilizar suas respostas para compreender o comportamento do sistema e, se desejado, alterar o desenho do sistema real.

Law (2014) classifica os modelos de simulação como uma classe dos modelos matemáticos, que se contrapõem aos modelos físicos, de natureza tangível. Os modelos de simulação podem ser determinísticos ou estocásticos, estáticos ou dinâmicos e discretos ou contínuos. Modelos determinísticos não apresentam variáveis aleatórias, ao contrário dos modelos estocásticos. Modelos estáticos avaliam o sistema em um determinado instante do tempo, enquanto os dinâmicos consideram a evolução do sistema no tempo, de forma discreta ou contínua. Nos modelos discretos, a evolução no tempo ocorre por meio de eventos finitos, ao passo que, nos modelos contínuos, as variáveis mudam continuamente, envolvendo taxas de variação ao longo do tempo. O presente trabalho abordará somente um tipo de simulação, com modelos estocásticos, dinâmicos e discretos: a SED. A Figura 2.1 demonstra a classificação elaborada por Law (2014).

Ainda segundo o autor, ao longo dos anos, a área de simulação despertou interesse crescente, o que foi confirmado pela análise da quantidade de publicações na área, apresentada na seção 1.2.2. Esse interesse pode ser explicado pelo fato de que a simulação:

- Pode ser utilizada para análise de sistemas reais com elementos estocásticos que dificilmente poderiam ser avaliados de forma acurada por outros tipos de modelos matemáticos;

- Permite estimar a performance de um sistema segundo um conjunto de projeções de condições operacionais;
- É capaz de comparar alternativas de *design* de sistemas e de políticas operacionais, de modo a encontrar aquela que melhor atenda aos requisitos definidos;
- Permite o controle das condições experimentais em estudos;
- E pode comprimir ou expandir o tempo, atendendo a diferentes objetivos de análise.

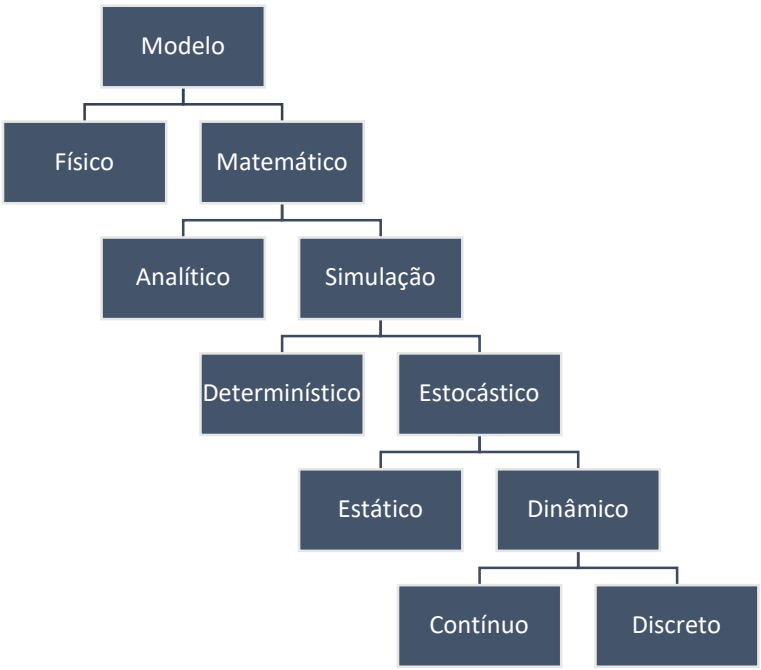


Figura 2.1 – Classificação de modelos
Fonte: adaptado de Law (2014)

Porém, o autor também cita desvantagens da utilização de simulação, sendo elas:

- Cada execução de um modelo de simulação estocástica produz apenas estimativas das características do sistema restritas ao conjunto de dados de entrada considerado e sorteado, sendo necessária a execução de réplicas para compreensão dos resultados do sistema;
- Modelos de simulação são usualmente caros e demandam tempo considerável de desenvolvimento, o que em parte está relacionado ao desafio da coleta de dados de entrada, como discutido na seção 1.2.1;
- A grande quantidade de número produzidos por um modelo e os recursos persuasivos de animação geralmente criam uma tendência de confiança excessiva nos resultados do modelo.

Diversas abordagens de simulação podem ser encontradas na literatura, muitas relacionadas com as ramificações apresentadas na Figura 2.1. Como exemplos, é possível citar Sistemas Dinâmicos, Simulação Baseada em Agentes, Redes de Petri, Simulação Monte Carlo e SED, sem deixar de considerar aplicações combinadas desses tipos, conhecidas como Simulação Híbrida (GARCIA-GARCIA *et al.*, 2020). Entretanto, historicamente, aplicações de SED têm sido as mais prevalentes na literatura (GARCIA-GARCIA *et al.*, 2020; ROY; VENKATESAN; GOH, 2020).

Além disso, a área de SED não se limita à manufatura, sendo aplicada há décadas em serviços de saúde (DOUDAREVA; CARTER, 2022; VÁZQUEZ-SERRANO; PEIMBERT-GARCÍA; CÁRDENAS-BARRÓN, 2021). Os autores identificam uma expansão significativa de publicações na área, especialmente após a década de 2010. Segundo Zhang (2018), isso reflete indiretamente a crescente popularidade da SED entre os gestores da área. Esse fato é corroborado por Salmon *et al.* (2018), que identificam uma tendência positiva de publicações de SED na área de saúde, especialmente após 2008. Segundo os autores, a maioria dos estudos parece ter origem acadêmica, focando-se, ao nível operacional, em questões como capacidade, e dimensionamento de mão de obra.

Outras áreas têm utilizado técnicas de simulação, tais como a área acadêmica em construção civil (ABDELMEGID *et al.*, 2020), transportes, logística e outros serviços (GABRIEL *et al.*, 2020).

Dentre os propósitos da simulação, em especial no contexto da Indústria 4.0, destacam-se as análises prescritivas, que incluem recomendações para melhorar a efetividade de operações, e preditivas, analisando a relação entre variáveis e testando hipóteses (DE PAULA FERREIRA; ARMELLINI; DE SANTA-EULALIA, 2020). Em ambos os casos, segundo os autores, é possível integrar conceitos e tecnologias de outras áreas, incluindo princípios como flexibilidade e otimização. Nesse sentido, ganham espaço estudos que visam integrar a simulação a outras ferramentas de análise, tais como Manufatura Enxuta (GOIENETXEA URIARTE; NG; URENDA MORIS, 2020), *Big Data* (GREASLEY; EDWARDS, 2021) e Seis Sigma (AHMED; PAGE; OLSEN, 2020).

2.2. Modelagem de dados de entrada

De forma geral, a simulação estocástica consiste de dados de entrada e lógicas, sendo os primeiros as fontes básicas de incerteza em um modelo de simulação (NELSON *et al.*, 2021). Por essa razão, a modelagem de dados de entrada é uma etapa essencial no desenvolvimento de

projetos na área, contribuindo para o entendimento dos comportamentos presentes no sistema em estudo (CORLU; AKCAY; XIE, 2020; ZHANG, J. *et al.*, 2021). Segundo os autores, essa atividade se torna ainda mais relevante no contexto de tomada de decisão em sistemas sociotécnicos complexos: entregar serviços críticos como saneamento, energia e transporte para uma população dinâmica tem se tornado cada vez mais desafiador e interconectado.

Ao mesmo tempo, Law (2016) avalia que a modelagem de dados de entrada é frequentemente negligenciada em projetos de simulação. Isso é crítico, pois a fidelidade das respostas obtidas por meio de modelos depende da fidelidade dos MDEs utilizados e essa relação geralmente não é trivial (NELSON *et al.*, 2021). Consequentemente, muitas vezes MDEs são causas de divergência nos resultados do modelo, sendo apontados como razões para estes não serem validados (LAW, 2016; SARGENT, 2013).

A modelagem de dados de entrada frequentemente se inicia a partir de amostras finitas dos dados do sistema em estudo (LIU; LIN; ZHOU, 2021), passando, então, por técnicas de análise. Apesar dos avanços em *software* de simulação, os métodos para modelagem de dados de entrada avançaram pouco nos últimos 30 anos (CHENG *et al.*, 2017; NELSON *et al.*, 2021), consistindo na prática de avaliar uma série de DEPs candidatas para representar os processos aleatórios pertencentes ao sistema simulado (LAW, 2014).

Tradicionalmente, o processo de ajuste a distribuições de probabilidade compreende três passos principais: (i) selecionar uma ou mais DEPs candidatas; (ii) determinar seus parâmetros; e (iii) checar a qualidade do ajuste via testes e análises gráficas (JAQUES *et al.*, 2016; NELSON *et al.*, 2021).

Dentre os métodos para estimar os parâmetros das distribuições, é possível identificar comumente o da máxima verossimilhança, o dos momentos, o dos percentis e dos mínimos quadrados (LAW, 2014). A escolha do método geralmente não impacta de forma significativa os resultados, desde que os algoritmos desenvolvidos sejam estáveis (BILLER; GUNES, 2010).

Já a checagem da qualidade do ajuste é feita usualmente por meio de testes de falta de ajuste (*lack-of-fit*), comumente chamados de testes de qualidade de ajuste (*goodness-of-fit*) (BILLER; GUNES, 2010). Segundo os autores, tais testes visam provar a hipótese alternativa de que os dados não seguem a distribuição candidata, o que, na maioria das vezes é verdade. Isso ocorre pois, em conjuntos de dados reais, desvios e ruídos, mesmo que mínimos, impossibilitam um ajuste perfeito. Nesse sentido, geralmente, quando tais testes não rejeitam a hipótese nula de que os dados seguem a distribuição candidata, isso ocorre devido ao tamanho amostral insuficiente para detectar a diferença existente. Por outro lado, ainda segundo os

autores, os testes podem rejeitar muitas ou todas as distribuições quando o tamanho amostral é demasiado, tornando-se sensíveis a pequenas diferenças, mesmo que na prática sejam insignificantes. Com isso, Biller e Gunes (2010) defendem que, apesar de sua popularidade, os testes de falta de ajuste devem ser utilizados com cuidado e em conjunto com análises gráficas como histogramas.

Quando nenhuma distribuição candidata oferece um bom ajuste, é recomendada a utilização de distribuições empíricas para representar a incerteza. Tais distribuições se assemelham às distribuições multinomiais geradas quando da utilização de histogramas, mas podem incluir técnicas de suavização e determinação de comportamentos de caudas. À medida que o tamanho amostral cresce, a distribuição empírica converge para a distribuição verdadeira (BILLER; GUNES, 2010).

Tais abordagens consideram que os dados de entrada são IID, como já discutido anteriormente, o que frequentemente não é verdade. Biller e Gunes (2012) e Lei, Hu e Zhu (2022) afirmam ser comum, por exemplo, a existência de dependências, citando casos na literatura como tempos de processamento de peças ao longo de estações de trabalho, características médicas de doadores e receptores de órgãos, tempos entre chegadas de ligações em um *call center* e demandas por produtos e transações em redes globais de *supply chain*.

Nesse momento, o modelador tem duas opções. Na primeira, considera mesmo assim que os dados de entrada são IID, optando por uma dentre as diversas opções de DEPs disponíveis em programas de simulação (BILLER; GUNES, 2010). Nesse caso, estará subestimando a incerteza do sistema estocástico, ignorando a natureza diversa e heterogênea dos dados (ZHANG, J. *et al.*, 2021). Já a segunda opção segue na direção de buscar ferramentas específicas para modelagem de cada conjunto de dados conforme suas características, o que, segundo os autores, levará a um processo de análise mais complexo e demorado. Isso se deve ao fato de que, para cada tipo de dado não IID, existir uma ou mais ferramentas especializadas para a modelagem, tais como cópulas para dados multivariados (BILLER, 2009), modelos autogressivos como ARMA e ARIMA para dados de entrada autocorrelacionados (UHLIG; ROSE; RANK, 2016) e NHPP para taxas de chegadas (DE SANTIS *et al.*, 2021; MORGAN *et al.*, 2019). Devido à dificuldade no uso dessas ferramentas, poucos estudos de simulação as utilizam (LEI; HU; ZHU, 2022).

E se insere como um complicador o fato de que programas de simulação e seus complementos estatísticos para modelagem de dados de entrada em sua maioria não fornecem ferramentas para modelagem ou até mesmo uso de MDEs desses tipos, o que vem sendo

reportado por estudos nas últimas décadas. Leemis (1997), por exemplo, identificou essa deficiência há mais de 20 anos, indicando que essa restrição fazia com que modeladores se limitassem a utilizar modelos simples, univariados e estacionários. Nos anos 2000, Biller e Nelson (2002) apontaram que, apesar de todos as plataformas de simulação incluírem, à época, a possibilidade de modelar e/ou utilizar modelos de dados de entrada para variáveis IID, poucas ofereciam recursos para modelar variáveis que não seguem esse pressuposto. Esse fato foi reafirmado recentemente por Cen, Herbert e Haas (2019). Além disso, em consulta às documentações disponibilizadas por dois dos principais suplementos utilizados por plataformas de simulação para modelagem de dados de entrada, Stat::Fit® e ExpertFit®, é possível notar que estes ainda não oferecem ferramentas para análise de dados complexos (AVERILL M. LAW & ASSOCIATES, 2022; GEER MOUNTAIN SOFTWARE, 2022).

2.3. Inteligência Artificial

Segundo Duan, Edwards e Dwivedi (2019), o termo Inteligência Artificial (IA) foi introduzido na década de 1950 e, apesar de não haver consenso em sua definição, normalmente é definido como a habilidade de uma máquina aprender por meio da experiência, ajustar-se e realizar atividades comumente desempenhadas por humanos. Tendo enfrentando altos e baixos ao longo das últimas décadas, a área de IA vive, hoje, um momento de revitalização e desenvolvimento, especialmente devido a avanços acelerados em tecnologias como as de armazenamento e processamento de dados (FRANKISH; RAMSEY, 2014; LEMLEY; BAZRAFKAN; CORCORAN, 2017).

Frequentemente, os termos IA, Aprendizado de Máquina (AM), Ciência de Dados e mesmo AP são utilizados como sinônimos, especialmente na mídia popular e de negócios (KOTU; DESHPANDE, 2019). Entretanto, segundo os autores, trata-se de campos distintos, o que está ilustrado na Figura 2.2. O mesmo entendimento é defendido por Alzubaidi *et al.* (2021), afirmando ser AP uma subárea de AM, que, por sua vez, também é inserida como uma subárea de IA (Figura 2.3).

Para Alzubaidi *et al.* (2021), IA pode ser definida como um programa capaz de sentir, raciocinar, agir e se adaptar. Já para Kotu e Deshpande (2019), significa dar a máquinas a capacidade de reproduzir o comportamento humano, especialmente funções cognitivas. Exemplos seriam o reconhecimento facial, direção autônoma e triagem de correspondência. Em alguns casos, segundo os autores, máquinas têm superado a capacidade humana. Porém, em outros, ainda estão longe de atingir o desempenho obtido por pessoas. Muitas técnicas se

encaixam na área de IA, tais como linguística, processamento de linguagem natural, ciência de decisão, visão computacional e robótica.

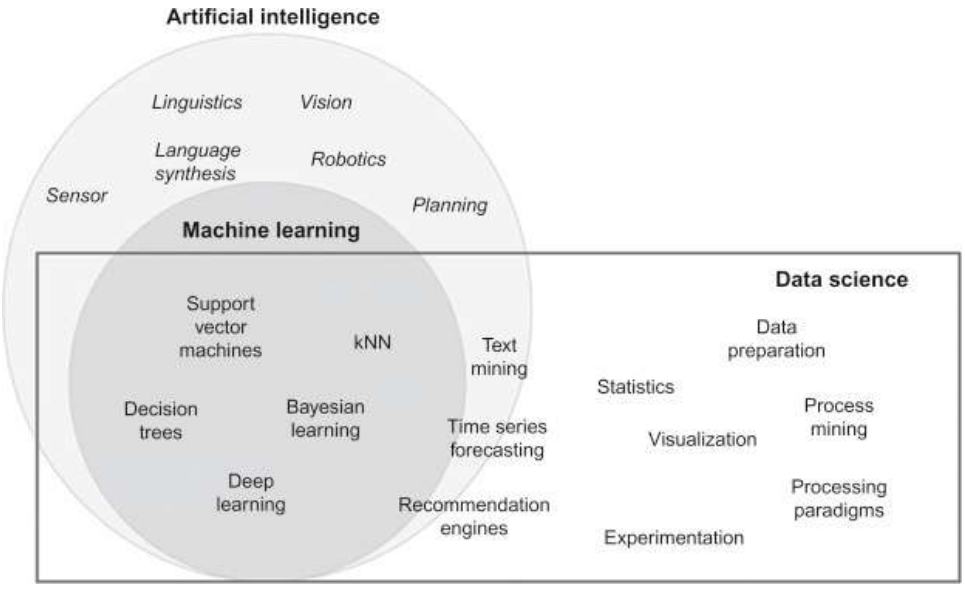


Figura 2.2 - Relação entre Inteligência Artificial, Aprendizado de Máquina e Ciência de Dados
Fonte: Kotu e Deshpande (2019)

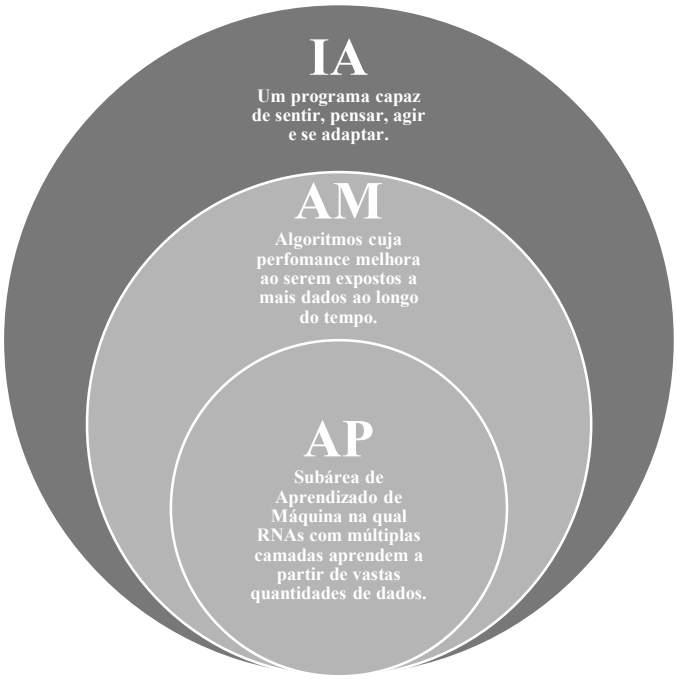


Figura 2.3 – Relação entre Aprendizado Profundo, Inteligência Artificial e Aprendizado de Máquina
Adaptado de Alzubaidi *et al.* (2021)

Já a área de AM engloba, na visão de Alzubaidi *et al.* (2021), algoritmos cuja performance melhora ao serem expostos a mais dados ao longo do tempo. Para Kotu e

Deshpande (2019), em sentido semelhante, trata-se da área com o objetivo de dotar máquinas da capacidade de aprender da experiência (obtida por meio de dados), detectando padrões ou regularidades estatísticas. Por exemplo, muitas organizações, especialmente as proprietárias de plataformas sociais virtuais, revisam postagens de seus usuários, buscando remover conteúdo abusivo. Máquinas podem aprender a moderar esse tipo de ambiente, sendo expostas a conteúdos claramente classificados como abusivos ou não e treinadas para distingui-los. Uma vez que o aprendizado for realizado e validado, isto é, o modelo de AM for concluído, a máquina pode iniciar a categorizar novas postagens (KOTU e DESHPANDE, 2019). Entretanto, relativamente à cognição humana, tais algoritmos são ainda indiferentes ao significado das variáveis de entrada e desprovidos de criatividade própria. Essas são algumas das razões pelas quais parte dos especialistas avalia, filosoficamente, que AM ainda não é um aprendizado verdadeiro, mas "apenas" uma complexa sequência de operações que inevitavelmente demanda alguma participação humana para especificação e controle. Quem obtém o real aprendizado é o humano que controla e valida o modelo (FRANKISH e RAMSEY, 2014).

Ainda sobre o tema, é preciso ressaltar que muitas das técnicas de AM não são recentes e já eram conhecidas como técnicas estatísticas. Algumas delas foram renomeadas, reinterpretadas ou reclassificadas, geralmente por pequenos ajustes, o que é apontado por parte dos especialistas como simples "modismos" (DUAN, EDWARDS e DWIVEDI, 2019).

Já Ciência de Dados é a aplicação de IA, AM e outros campos quantitativos como estatística, visualização de dados e matemática, interessada em extrair valor dos dados para favorecer negócios. Exemplos de aplicações compreendem sistemas de recomendação de filmes, alerta de fraude para transações de cartão de crédito e sistemas de previsão de vendas (KOTU e DESHPANDE, 2019).

Dentre as mais promissoras técnicas de AM, destacam-se atualmente as RNAs, cujo objetivo é imitar o funcionamento do cérebro humano. Mesmo tendo suas origens nos anos 1940, as RNAs encontram-se na fronteira da expansão da IA. De maneira geral, RNAs são constituídas de camadas de neurônios artificiais conectados.

Por fim, para Alzubaidi *et al.* (2021), AP pode ser definido como a subárea de AM na qual RNAs de múltiplas camadas aprendem a partir de uma vasta quantidade de dados. O termo está diretamente relacionado ao fato de apresentarem duas ou mais camadas de neurônios, sendo consideradas "profundas" (DUAN, EDWARDS e DWIVEDI, 2019). Esse tipo de técnica será mais bem explorado na seção 2.4.

O presente trabalho se insere nesse contexto por utilizar GANs, que consistem em uma das técnicas de AP, fazendo com que este estudo esteja circunscrito nas áreas de IA, AM, RNAs e AP (da mais ampla para a mais específica). Conceitos mais detalhados sobre GANs estão apresentados na seção 2.5.

2.4. Redes Neurais Artificiais e Aprendizado Profundo

RNAs são inspiradas pelo sistema biológico, particularmente pela pesquisa sobre o cérebro humano. Originaram-se pelo trabalho de McCullogh e Pitts (1943), que apresentava um modelo simples do funcionamento de um neurônio, capaz de realizar qualquer operação booleana e, logo, qualquer computação (FRANKISH e RAMSEY, 2014).

As RNAs aprendem por meio de experiência (obtida por dados), podendo capturar padrões estatísticos sutis, os quais são desconhecidos ou difíceis de descrever (ZHANG; PATUWO; HU, 1998). Além disso, são capazes de generalizar, produzindo saídas sensatas para novas entradas, isto é, que não foram apresentadas à rede durante seu treinamento (HAYKIN, 1998).

Uma RNA é uma coleção de unidades conectadas, chamados neurônios artificiais. Cada neurônio recebe entradas externas à rede ou oriundas (geralmente) de outros neurônios. Então, computa a soma ponderada dessas entradas, conforme vetor de pesos ($\boldsymbol{w}$), e mapeia (transforma) essa soma a uma saída, aplicando uma função de ativação (σ). Por fim, a saída é passada para os próximos neurônios, como suas entradas (KRAUS; FEUERRIEGEL; OZTEKIN, 2020). Esse é o exemplo de uma RNA livre de ciclos ou *feedback*, com todos os neurônios passando informações apenas para frente, o que está representado na Figura 2.4.

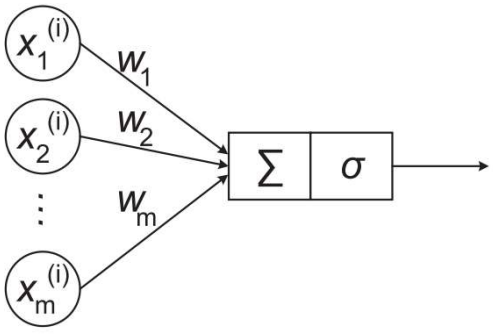


Figura 2.4 – Neurônio ilustrativo em uma RNA
Fonte: Kraus, Feuerrigel e Oztekin (2020)

Segundo os autores, o conceito apresentado na figura se refere à mais simples RNA, com apenas uma camada de neurônios, sendo que o resultado de sua saída (f) é computado pela

aplicação de uma função de ativação (σ) à combinação linear das entradas do neurônio, o que resulta na formulação matemática apresentada na Equação 2.1:

$$f_1(\boldsymbol{x}, \boldsymbol{w}, b) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b) \quad (2.1)$$

Sendo $\boldsymbol{x}$ as entradas, $\boldsymbol{w}$ o vetor de pesos e b o intercepto (também chamado de viés). Pesos e interceptos existentes em uma RNA compreendem o conjunto θ de parâmetros da rede. A escolha da função de ativação permite flexibilizar os resultados, sendo escolhas comuns indicadas por Kraus, Feuerrigel e Oztekin (2020) as apresentadas no Quadro 2.1.

Quadro 2.1 - Funções de ativação comumente utilizadas em RNAs

Função de ativação	Equacionamento matemático
Sigmoide	$\frac{1}{1 + e^{-z}} \in (0, 1)$
Tangente Hiperbólica	$\frac{e^{2z} - 1}{e^{2z} + 1} \in [-1, 1]$
Unidade Linear Retificada (ReLU)	$\max(0, z) \in [0, \infty)$

Fonte: adaptado de Kraus, Feuerrigel e Oztekin (2020)

Em comum, as funções de ativação demandam que um certo limite seja excedido para que ocorra a ativação. Esse comportamento é biologicamente inspirando nos neurônios, que também requerem um estímulo mínimo para serem ativados (STACHENFELD; BOTVINICK; GERSHMAN, 2018). Cada escolha apresenta suas vantagens e desvantagens. Por exemplo, a função ReLU pode levar a neurônios que resultem em saídas nulas para todas as entradas possíveis, perdendo flexibilidade (KRAUS, FEUERRIGEL e OZTEKIN, 2020). Já as funções Sigmoide e Tangente Hiperbólica podem levar a problemas de dissipação ou explosão de gradientes durante os ciclos de otimização para treinamento (PEDAMONTI, 2018).

O processo de aprendizado tem como objetivo minimizar uma função de custo (L), sendo que, em aplicações de aprendizado supervisionado, L geralmente leva em consideração a diferença entre o resultado da saída do neurônio ($\hat{y}$) e o valor esperado (y), conhecido previamente. Considerando a Equação 2.1, essa definição resulta na Equação 2.2.

$$L(\hat{y}, y) = L(f(\boldsymbol{x}, \boldsymbol{w}, b), y) \quad (2.2)$$

Segundo Kraus, Feuerrigel e Oztekin (2020), graças às funções de ativação, RNAs podem modelar funções não lineares e não convexas. Entretanto, como efeito colateral, elas fazem com que o problema de otimização para treinamento seja de difícil resolução. Isto é, não

é possível resolver analiticamente o problema para obter o mínimo global, sendo necessário utilizar métodos iterativos, como o gradiente descendente, para encontrar um ótimo local (KRAUS, FEUERRIGEL e OZTEKIN, 2020). Essa estratégia tem permitido encontrar soluções próximas ao ótimo global (CHOROMANSKA; HENAFF; MATHIEU, 2015).

RNAs com apenas uma camada de neurônio apresentam várias limitações. Por exemplo, a depender da escolha da função de ativação, a RNA não poderá aprender funções como a operação lógica "ou exclusivo" (XOR) (KRAUS, FEUERRIGEL e OZTEKIN, 2020). Em contrapartida, segundo os autores, RNAs com múltiplas camadas (AP) podem transformar (por meio das camadas intermediárias) suas entradas em representações dimensionalmente superiores, mais complexas e, então, mapeá-las às saídas. Conforme o teorema da aproximação universal (CYBENKO, 1989), RNAs com três camadas (entrada, intermediária e saída) são suficientes para representar qualquer função. Entretanto, a prática no treinamento de RNAs sugere que modelos mais profundos podem reduzir o erro de generalização, ou seja, o erro do modelo ao prever saídas para dados ainda não observados (GOODFELLOW, 2016). Com isso, a maior diferença entre os primeiros experimentos com RNAs e modelos de AP é a dimensão das redes, que pode conter até centenas de camadas, milhões de neurônios e complexas estruturas de conexão entre eles, como pode ser observado em He *et al.* (2016).

Essa habilidade introduziu uma inédita flexibilidade para modelar relações, entre preditores e saídas, altamente complexas e não lineares, o que tem permitido que modelos de AP superem a capacidade preditiva e a performance operacional de outros tipos de modelos de AM em diversas atividades (KRAUS, FEUERRIGEL e OZTEKIN, 2020). Com isso, modelos de AP têm recebido atenção crescente (LECUN; BENGIO; HINTON, 2015). Entretanto, na área de Pesquisa Operacional, existe ainda significativo potencial inexplorado de aplicação de modelos de AP (KRAUS, FEUERRIGEL e OZTEKIN, 2020).

2.5. Redes Adversárias Generativas

GANs são arquiteturas de RNAs profundas atualmente estudadas e utilizadas com grande foco na geração de imagens sintéticas, mas, significativamente realistas (SORIN *et al.*, 2020). Para compreendê-las, além de entender o funcionamento de RNAs (o que foi explorado na seção 2.4), é preciso entender o conceito de modelos generativos e o *framework* de redes adversárias.

Segundo Foster (2019), muitas aplicações de AM utilizam modelos discriminativos. Trata-se de modelos que, considerando as características ($\boldsymbol{x}$) de uma observação, buscam

estimar sua probabilidade (p) de fazer parte de uma categoria (y), ou seja, $p(y|\boldsymbol{x})$. Isso é feito por meio do mapeamento das características das observações às categorias de interesse, o que demanda definição e classificação prévias de tais categorias para que possa ser aplicado o aprendizado supervisionado. O autor cita o exemplo hipotético de classificar pinturas visando identificar se foram concebidas por Van Gogh ou por outro artista. Com dados suficientes e previamente categorizados é possível treinar um modelo discriminativo para que, considerando características como cores, formas e texturas, indique a probabilidade de a pintura ter sido elaborada ou não por tal pintor.

Modelos generativos, por sua vez, não tem como objetivo classificar observações com base em suas características ($\boldsymbol{x}$), mas estimar as probabilidades (p) de ocorrem por si só, ou seja, $p(\boldsymbol{x})$. Nesse caso, o modelo deve imitar a distribuição das características observadas o mais próximo possível para, então, acoplado a um elemento estocástico, gerar novas observações, sintéticas, que apresentem características como se fizessem parte do conjunto de dados real. Ou seja, visam descrever, em termos de um modelo probabilístico, como um conjunto de dados é gerado. Retomando-se o exemplo do parágrafo anterior, um modelo generativo com aprendizado baseado apenas nas pinturas elaboradas por Van Gogh seria capaz de gerar novas pinturas com as características empregadas pelo artista (FOSTER, 2019).

Apesar de modelos discriminativos representarem até hoje a maior parte do impulso por trás dos avanços em AM, nos últimos anos muitos dos desenvolvimentos vieram de novas aplicações de modelos generativos. A área tem sido considerada chave para habilitar formas sofisticadas de IA, comparáveis à humana, que vão muito além do que modelos discriminativos podem oferecer (FOSTER, 2019).

Diversos são os exemplos desse tipo de comportamento em seres humanos, como a habilidade de imaginar um animal sob diversos ângulos, finais alternativos para um filme ou planejar a próxima semana considerando diversos cenários. Neurocientistas sugerem que a percepção humana não é baseada em um modelo discriminativo altamente complexo operando com base em entradas sensoriais, mas em um modelo generativo treinado desde o nascimento e que é capaz de gerar simulações do nosso ambiente que visam corresponder ao futuro. Os novos desenvolvimentos têm buscado entender como máquinas podem adquirir tal habilidade e a presente tese visa, nesse sentido, usar modelos generativos em favor da simulação computacional (FOSTER, 2019).

Além de generativos, modelos de GANs adotam o *framework* de redes adversárias. Segundo Goodfellow *et al.* (2014) e Agnese (2020), nesse *framework*, um modelo generativo é

colocado contra um adversário: um modelo discriminativo que aprende a determinar se uma amostra faz parte da distribuição real ou não. Conforme exemplificam Goodfellow *et al.* (2014), o modelo generativo pode ser visto como um time de falsificadores buscando produzir notas falsas e usá-las sem serem detectados. Por outro lado, o modelo discriminador é análogo a uma autoridade, tentando detectar notas falsas. A competição nesse jogo faz com que ambos os times melhorem seus métodos até que os falsificadores atinjam um nível de excelência que impossibilite a terceiros distinguir suas notas das genuínas. Ou seja, essa arquitetura pode ser utilizada para gerar amostras sintéticas, novas, que se enquadram nas características da respectiva população verdadeira (AGNESE, 2020).

A rede geradora aprende a gerar uma distribuição sintética p_g mapeando uma variável de entrada latente $\boldsymbol{z}$ (um ruído, sem significado prático) ao espaço dos dados: $G(\boldsymbol{z}, \theta_g)$, sendo G uma função diferenciável representada por uma RNA de parâmetros θ_g. Além disso, é necessário definir uma segunda RNA, o discriminador $D(\boldsymbol{x}, \theta_d)$ que resulta em um único escalar, representando a probabilidade de uma observação $\boldsymbol{x}$ ser originada dos dados reais e não de p_g (GOODFELLOW *et al.*, 2014).

Ainda segundo os autores, a rede D é treinada tendo acesso tanto aos dados reais quando às observações sintéticas geradas por G. O objetivo do treinamento é maximizar a probabilidade de classificar corretamente tanto as observações sintéticas quanto as reais. Por outro lado, o treinamento de G é feito de modo a minimizar a probabilidade de D classificar dados sintéticos como falsos, minimizando $\log\left(1 - D(G(\boldsymbol{z}, \theta_g), \theta_d\right)$. É possível notar que, quanto maior for a probabilidade resultante de D, o que indica um julgamento de que a observação é genuína, mais próximo de zero será o valor para o qual se calculará o logaritmo. E, com isso, o logaritmo retornará valores mais negativos, o que está alinhado com a direção de otimização. A Figura 2.5 ilustra a arquitetura típica observada em GANs. Destacam-se as duas RNAs, G e D, os conjuntos de dados reais e sintéticos, respetivamente X e $G(z)$ e o espaço latente Z.

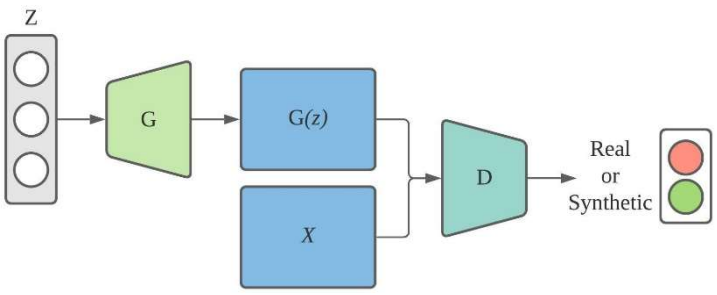


Figura 2.5 – Arquitetura típica de GANs

As GANs têm obtido sucesso em diversas áreas, destacando-se a área médica. Yi, Walia e Babyn (2019) afirmam que o uso de GANs nesse contexto tem crescido rapidamente, sendo utilizadas para sintetizar imagens médicas e aumentar tamanhos amostrais (mitigando os problemas de faltas de dados e *overfitting*). De modo geral, as GANs têm impactado positivamente a área de visão computacional (SORIN *et al.*, 2020).

Independente da aplicação, de Goodfellow *et al.* (2014) defendem que GANs são capazes de aprender distribuições complexas que representam o comportamento de uma dada população e, com isso, gerar novas amostras coerentes com o mesmo comportamento. Essa habilidade constitui-se como conceito-chave para o presente trabalho.

Isso se deve ao fato de que o processo de modelagem de dados de entrada tem justamente como objetivo a representação do comportamento de uma população, geralmente abrangendo seus tempos de ciclo, padrões de chegada e probabilidades, de modo a gerar novas amostras que também sigam o comportamento identificado. Ou seja, ao modelar dados de entrada, praticantes de SED estão criando MDEs que também gerarão dados sintéticos.

3. PROPOSTA

O presente capítulo descreve a proposta de utilização de GANs para a modelagem de dados de entrada em projetos de simulação. O fluxo geral do método consiste nas etapas apresentadas na Figura 3.1, que serão descritas nas seções a seguir.

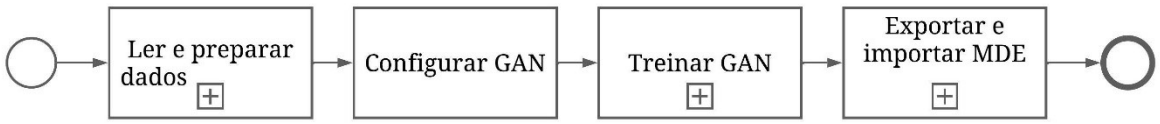


Figura 3.1 - Proposta para utilização de GANs

3.1. Ler e preparar dados

A etapa de leitura e preparação dos dados reais está subdividida nas atividades de leitura dos dados, tratamento de dados faltantes, conversão de variáveis categóricas e normalização dos dados, conforme apresentado na Figura 3.2.

Figura 3.2 – Atividades para leitura e preparação dos dados

O processo se inicia com a leitura dos dados, que devem estar organizados de forma tabular, com as variáveis de entrada dispostas em colunas e as observações em linhas, como exemplificado na Tabela 3.1, que registra tempos de atividades realizadas para peças em um processo produtivo ilustrativo. Todos os dados presentes em uma linha devem se referir ao mesmo item observado e, em caso de algum dado faltante, não se deve deslocar os valores da coluna para cima, mas, sim, manter a célula vazia, como o tempo da atividade B para a peça 4. Porém, nesses casos, será necessário tratar os dados faltantes, o que é realizado na próxima etapa.

O processo de treinamento de GANs não é compatível com dados faltantes. Como delimitação, o presente trabalho considera que, quando existentes, esses dados ocorrem de forma completamente aleatória. Isso significa que não existe relação entre qualquer variável

(presente ou não no conjunto de dados) e o fato de que uma entrada está faltando (SCHOSSER, 2022).

Tabela 3.1 – Exemplo de dados tabulares para treinamento

Peça	Tempo atividade A (min)	Tempo atividade B (min)	Tempo atividade C (min)
Peça 1	9.8	8.8	1.4
Peça 2	2.9	1.8	5.4
Peça 3	5.1	9.2	6.7
Peça 4	8.4	-	3.7
Peça 5	8.8	2.0	2.0

Como exemplo, é possível citar o caso de um cronoanalista que está acompanhando o processamento de várias peças e, para cada uma delas, medindo os tempos das etapas de fabricação. Se, em alguns momentos aleatórios, o cronoanalista não conseguir medir alguma etapa de qualquer peça, o tempo faltante ocorrerá também de forma aleatória. Mas, se não for possível coletá-lo pois o tipo da peça não requer uma parte do processo de fabricação, existirá uma dependência entre tipo de peça e ocorrência de dado faltante. A estratégia de tratamento de dados faltantes adotada na presente proposta é a de análise de casos completos. Nesse caso, observações que apresentam dados faltantes são eliminadas (MCMAHON, ZHANG e DWIGHT, 2020).

Em seguida, parte-se para a etapa de conversão de variáveis categóricas. GANs e muitos algoritmos de AM requerem como *input* matrizes numéricas. Quando variáveis categóricas estão presentes nos dados, é necessário codificá-las em vetores numéricos adequados. A proposta do presente trabalho utiliza para tal finalidade a técnica de "*one-hot encoding*", considerada uma técnica simples e amplamente utilizada (CERDA, VAROQUAUX e KÉGL, 2018; MUMTAZ e GIESE, 2021). Segundo os autores, ela consiste na construção de vetores de tamanho igual ao número de categorias presentes nos dados e em que cada categoria é ortogonal e equidistante às demais, o que está de acordo com o conceito de variáveis categóricas. Por exemplo, uma variável contendo as categorias "iniciação científica", "mestrado" e "doutorado" pode ter seus valores codificados, respectivamente, pelos vetores [1, 0, 0], [0, 1, 0] e [0, 0, 1].

Para finalizar a fase de leitura e preparação dos dados, é necessário normalizá-los. A normalização consiste no processo de transformar os dados para que tenham escalas similares, fazendo com que as variáveis tenham pesos uniformes durante o aprendizado (SINGH; SINGH, 2020). É uma etapa comumente realizada em projetos de AM, ajudando a acelerar os cálculos dos algoritmos (THARA; PREMA; XIONG, 2019). Os autores citam diversas técnicas

utilizadas para a normalização, muitas vezes apresentando resultados semelhantes. A proposta deste trabalho considera a possibilidade de utilização de duas destas técnicas: escalonamento mínimo-máximo (Equação 3.1) e escalonamento robusto (Equação 3.2).

$$t_i = \frac{x_i - x_{min}}{x_{máx} - x_{min}} \tag{3.1}$$

Sendo t_i o valor normalizado de x_i, a i-ésima observação da variável x. E x_{min} e $x_{máx}$ os valores mínimo e máximo da variável x, respectivamente.

$$t_i = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{3.2}$$

Sendo, novamente, t_i o valor normalizado de x_i. E $Q_1(x)$ e $Q_3(x)$ o primeiro e o terceiro quartis da variável x, respectivamente.

Enquanto o escalonamento mínimo-máximo retorna valores entre 0 e 1, o escalonamento robusto pode resultar em valores em todo $\mathbb{R}$. Além disso, o escalonador robusto gera menos distorções em casos de *outliers*, sendo recomendado para situações em que tais observações estejam presentes e o modelador desejar mantê-las. Algumas decisões sobre a arquitetura da GAN estão relacionadas à escolha da técnica de normalização e são discutidas na seção 0.

3.2. Configurar GAN

Na etapa de configuração das GANs, é necessário configurar seus hiperparâmetros. Em AM, hiperparâmetros representam valores que devem ser fixados antes do processo de aprendizagem, podendo estar relacionados ao treinamento do modelo e à sua arquitetura. Já os demais parâmetros são resultado do próprio aprendizado (DONG et al, 2021). Considerando a área de AP, os autores citam como exemplos de parâmetros os coeficientes da rede neural, que serão modificados ao longo do treinamento. Por outro lado, características como taxa de aprendizado e tamanho de lote são considerados, nesse caso, como hiperparâmetros.

Yang e Shami (2020) listam diversos hiperparâmetros que devem ser definidos para modelos de AP. GANs requerem, em linhas gerais, a definição das mesmas variáveis (ALARSAN; YOUNES, 2021; COURTENAY; GONZÁLEZ-AGUILERA, 2020). Porém, apresentam duas particularidades. Primeiro, como uma GAN é formada por duas redes neurais, cada uma pode ter seus próprios hiperparâmetros. Segundo, GANs demandam a definição de um espaço latente, sendo necessário definir o seu tamanho e a distribuição de ruído da qual as

variáveis latentes serão amostradas. Como o espaço latente serve como *input* para o Gerador, a definição de seu tamanho também impacta a arquitetura dessa rede. Os hiperparâmetros propostos para configuração das GANs estão subdivididos em três tipos: proporcionais à dimensionalidade dos dados, dependentes do contradomínio do MDE e outros.

3.2.1. Hiperparâmetros proporcionais à dimensionalidade dos dados

Dois hiperparâmetros de modelos de AP devem ser definidos conforme a complexidade dos dados, para que os modelos tenham capacidade suficiente de atingirem seus objetivos: o número de camadas intermediárias de neurônios e sua densidade, isto é, a quantidade de neurônios por camada (YANG e SHAMI, 2020). Considerando o contexto das GANs, é preciso considerar ainda o tamanho do espaço latente, ou seja, o número de variáveis latentes a serem fornecidas como *input* para o Gerador. Visando permitir uma arquitetura flexível, a proposta prevê que duas das três variáveis citadas (tamanho do espaço latente e densidade das camadas intermediárias de neurônios) sejam definidas de forma proporcional à dimensionalidade dos dados, uma das características de complexidade que podem estar presentes. O conceito de dimensionalidade adotado é o de número de componentes (N_v) que expliquem um percentual predeterminado (v) da variância observada nos dados reais. Os componentes podem ser obtidos por meio de Análise de Componentes Principais (PCA) (BRO; SMILDE, 2014). Por fim, é necessário definir o fator de proporcionalidade (α) a ser multiplicado N_v, podendo este ser diferente para cada um dos hiperparâmetros citados. Para o presente trabalho, estes valores foram definidos em testes exploratórios e estão apresentados na Tabela 3.2.

Tabela 3.2 – Hiperparâmetros proporcionais à dimensionalidade dos dados

Hiperparâmetro	**Valor definido**
Densidade das camadas intermediárias	$8N_v$
Tamanho do espaço latente	$3N_v$

3.2.2. Hiperparâmetros dependentes do contradomínio do MDE

O segundo tipo de hiperparâmetros é dependente do contradomínio desejado para o MDE, ou seja, à faixa de valores aceitáveis para o modelo. Usualmente, MDEs podem ser divididos em limitados e ilimitados. Como exemplos de DEPs limitadas é possível citar as distribuições beta, triangular e uniforme. Já a distribuição normal é um exemplo de DEP ilimitada (LAW, 2014). O autor explica que algumas distribuições ilimitadas podem gerar, mesmo que raramente, valores arbitrariamente grandes e que podem causar problemas em modelos de simulação. Daí a utilidade de se truncar o MDE em certas situações.

O tipo de contradomínio a ser obtido pela GAN também pode ser ajustado para obter MDEs totalmente limitados ou ilimitados, sendo necessário escolher uma função adequada de ativação da camada de saída do Gerador. Além disso, é importante escolher, na fase de tratamento dos dados, uma técnica de normalização compatível com a função de ativação escolhida. Ou seja, o contradomínio do Gerador deve coincidir com a faixa de valores normalizados para treinamento. Caso contrário, por melhor que seja o aprendizado do Gerador, sua função de ativação não permitirá ou terá mais dificuldade de gerar valores semelhantes aos reais normalizados. Por exemplo, se a função de ativação do Gerador for a de tangente hiperbólico, cujo contradomínio se encontra em [-1, 1], idealmente os dados devem ser normalizados utilizando uma função que apresente o mesmo contradomínio. Não fará sentido, portanto, tratar os dados reais utilizando o escalonador robusto, pois este resultará em dados ilimitados. O Gerador terá capacidade de gerar apenas na faixa [-1, 1], não tendo qualquer habilidade de gerar observações fora dessa faixa.

Considerando a proposta deste trabalho, caso o modelador deseje um MDE limitado, recomenda-se a utilização da função de ativação sigmoide para a camada de saída do Gerador, visto que esta tem seu contradomínio estritamente limitado em [0, 1]. Como ao Gerador deve ser dada a possibilidade de gerar dados semelhantes aos reais, estes últimos devem ser tratados para também estarem nessa faixa, sendo então adequada a utilização do escalonador mínimo-máximo.

Por outro lado, caso o modelador deseje um MDE ilimitado, recomenda-se a utilização da função de ativação linear. Por consequência, os dados reais não precisam estar em uma faixa rígida e podem ser normalizados utilizando-se o escalonador robusto, especialmente se apresentarem *outliers* e o modelador não desejar eliminá-los.

Por fim, destaca-se que após o treinamento do Gerador, será necessário acoplar uma função de reversão do escalonamento, para que os dados fornecidos ao modelo de simulação apresentem o contradomínio originalmente observado.

3.2.3. Outros hiperparâmetros

Parte dos demais hiperparâmetros, não relacionados à capacidade de aprendizado e ao contradomínio do MDE, foi definida a partir de valores considerados como bons pela literatura (ALARSAN; YOUNES, 2021; COURTENAY; GONZÁLEZ-AGUILERA, 2020; GOODFELLOW, 2016; GOODFELLOW *et al.*, 2014) e estão apresentados no Quadro 3.1.

Por fim, é necessário definir a duração total da etapa de aprendizagem, ou seja, o número de épocas. Nesta proposta, considera-se que este hiperparâmetro é altamente dependente do tempo disponível para treinamento, devendo ser definido caso a caso.

Quadro 3.1 – Hiperparâmetros adotados na presente proposta

Hiperparâmetro	Valor adotado
Função de custo	Entropia cruzada binária
Função de ativação do Discriminador	Sigmoide
Função de ativação de camadas intermediárias	*Leaky* ReLU (*slope* = 0,2)
Distribuição do espaço latente	Normal padrão
Suavização de atributos	Unilateral
Otimizador	Adam
Taxa de aprendizado para o Gerador	0,00002
Taxa de aprendizado para o Discriminador	0,0002
Taxa de decaimento (1º momento)	0,5
Taxa de decaimento (2º momento)	0,999
Tamanho de lote para treinamento	64

3.3. Treinar GAN

Após a configuração da GAN, é possível iniciar o processo de aprendizado. Esta etapa abrange os passos de treinamento, pontos de checagem e avaliação do Gerador, conforme apresentado na Figura 3.3.

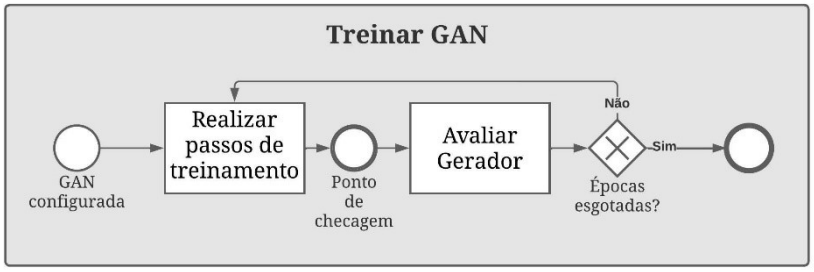


Figura 3.3 – Atividades para treinamento da GAN

O treinamento é um problema de otimização iterativo que visa minimizar uma função de custo, medindo, em aplicações de aprendizado supervisionado, a dissimilaridade entre valores reais e previstos (GOODFELLOW, BENGIO e COURVILLE, 2016), conforme discutido na seção 2.5.

Segundo os autores, o treinamento de GANs naturalmente apresenta oscilações e nem sempre os resultados atingem um equilíbrio ao longo das épocas. Por essa razão, não são adequadas as estratégias de interrupção antecipada da aprendizagem, conhecidas como *early stopping* (YANG; SHAMI, 2020), que visam principalmente evitar a degradação dos resultados obtidos por RNAs com treinamentos demasiadamente longos. Como consequência, o estado do

Gerador após a último passo de atualização de seus pesos pode não ser o melhor que fora observado ao longo do treinamento. Nesse sentido, esta proposta sugere o registro periódico de pontos de checagem. Trata-se de uma técnica em que o valor exato de todos os parâmetros usados por um modelo são capturados e salvos (TENSORFLOW, 2022). Mantendo-se o registro desses momentos e das respectivas medidas de qualidade de representação, será possível, na etapa de tradução do Gerador, carregar os parâmetros do melhor ponto de checagem registrado.

Em cada ponto de checagem, é necessário, então, avaliar o Gerador em termos de qualidade de representação da distribuição real. Essa avaliação visa oferecer ao modelador evidências de que o modelo de dados de entrada representa de forma suficientemente adequada o processo em estudo. Esta etapa encontra paralelo na avaliação da qualidade do ajuste em testes *goodness-of-fit*.

A literatura propõe diversas técnicas para avaliação do desempenho de GANs. Borji (2019) e Borji (2022) reúnem 54 medidas de avaliação, dividindo-as entre quantitativas e qualitativas. Enquanto medidas qualitativas são geralmente baseadas na avaliação humana subjetiva, medidas quantitativas são baseadas em pontuações numéricas. Ainda segundo o autor, uma boa medida de avaliação de GANs deve:

- Favorecer modelos que gerem amostras sintéticas fidedignas;
- Favorecer modelos que gerem amostras com diversidade;
- Resultar em pontuações em faixas limitadas e bem definidas, facilitando a interpretação;
- Concordar com julgamentos humanos subjetivos;
- Demandar amostras pequenas;
- Ter baixa complexidade computacional.

Considerando tais critérios, o presente trabalho sugere a utilização de avaliações gráficas e do C2ST, que já foram descritos como parte do método do estudo, na seção 1.5.3.4. Enquanto as avaliações gráficas permitem ao modelador identificar visualmente a qualidade de representação, o teste C2ST identifica quantitativamente divergências entre as distribuições e apresenta um resultado de fácil entendimento: a medida A_{MDE}, com os limites bem definidos [0, 1].

3.4. Traduzir e exportar o Gerador

Como o propósito é a construção de MDEs a serem utilizados em modelos de simulação, é necessário que seja desenvolvida a capacidade de integração entre ambos. Isso é um desafio, pois tais interfaces entre programas usualmente não são simples de serem estabelecidas por modeladores sem conhecimento da arquitetura do *software* de simulação. A proposta deste trabalho inclui um algoritmo para tradução do Gerador para uma linguagem compatível com o *software* de simulação adotado e foi implementado para o FlexSim. As atividades dessa etapa estão ilustradas na Figura 3.4.

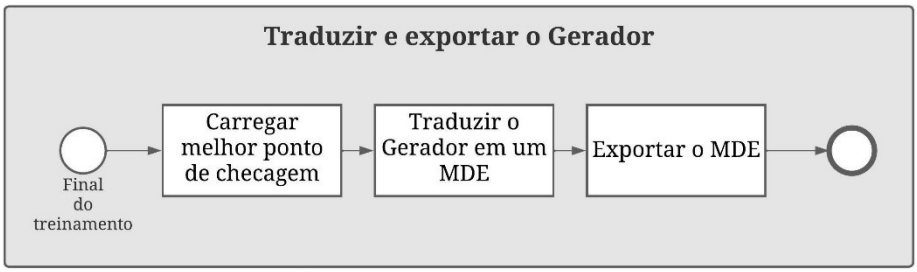


Figura 3.4 – Atividades para tradução e exportação do Gerador

Traduzir o MDE (Gerador) para a linguagem do *software* de simulação adotado requer construir uma função matemática com sintaxe compatível e que inclua as operações descritas a seguir.

Inicialmente, é preciso amostrar o vetor de variáveis do espaço latente ($\boldsymbol{z}$) para que atue como entrada da equação do Gerador. Para tanto, deve-se criar e amostrar de uma distribuição normal padrão tantas variáveis quantas estiverem previstas no tamanho de $\boldsymbol{z}$.

Em seguida, a partir de $\boldsymbol{z}$, é preciso estabelecer operações para cálculo do valor de saída de cada neurônio de cada camada (l) da RNA do Gerador, conforme o processo de *forward propagation* definido na Equação 3.3. Sendo $a_j^{[l]}$ o j-ésimo neurônio da camada l da RNA; $g^{[l]}$ a função de ativação utilizada na camada l; k o número de neurônios da camada anterior ($l-1$); $w_{jk}^{[l]}$ o peso dado ao valor de entrada obtido do k-ésimo neurônio da camada anterior para $a_j^{[l]}$; e $b_j^{[l]}$ o valor do intercepto (viés) para $a_j^{[l]}$.

$$a_j^{[l]} = g^{[l]}\left(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}\right) \tag{3.3}$$

Enquanto w e b são parâmetros aprendidos durante o treinamento do Gerador e que podem ser consultados por meio de métodos como *get_weights()* para camadas de RNAs implementadas em TensorFlow (TENSORFLOW, 2022), a função de ativação g é um hiperparâmetro, discutido na seção 3.2. Para camadas intermediárias, a função utilizada é a *Leaky* ReLU. E, para a camada de saída do Gerador, esta proposta recomenda a função linear, em caso de contradomínio ilimitado, ou a função sigmoide, caso o contradomínio seja totalmente limitado.

Por fim, após a definição das equações para os neurônios da camada de saída do Gerador, deve-se tratar seus resultados para que apresentem o contradomínio observado nos dados reais. Como a GAN é treinada tendo acesso a dados normalizados, é necessário acoplar uma função de reversão do escalonamento. Se a técnica de escalonamento escolhida for a de mínimo-máximo, a reversão deve ser realizada por meio da Equação 3.4.

$$x = t(x_{máx} - x_{min}) + x_{min} \tag{3.4}$$

Sendo x o valor sintético não normalizado (e de interesse para o modelo de simulação), t o valor sintético escalonado (saída do Gerador) e x_{min} e $x_{máx}$ os valores mínimo e máximo da variável x, respectivamente, observados nos dados reais utilizados pela GAN para treinamento, conforme a etapa de leitura e preparação dos dados, descrita na seção 3.1.

Caso a técnica de normalização escolhida previamente tenha sido a de escalonamento robusto, a reversão deve ser realizada por meio da Equação 3.5.

$$x = t\big(Q_3(x) - Q_1(x)\big) + Q_1(x) \tag{3.5}$$

Sendo, novamente, x o valor sintético não normalizado, t o valor sintético escalonado e $Q_1(x)$ e $Q_3(x)$ o primeiro e terceiro quartis da variável x, respectivamente, observados nos dados reais utilizados pela GAN para treinamento.

Caso uma ou mais variáveis dos dados reais tratados sejam inteiras, é necessário incluir o cálculo para arredondamento. E, por fim, se os dados reais apresentarem variáveis categóricas e que, portanto, passaram pelo processo de *one-hot encoding*, deve-se recodificar os resultados para as categorias originais.

O Algoritmo 2 reúne os passos descritos de forma sumarizada. O resultado do algoritmo de tradução do Gerador deve ser um texto que possa ser importado ao modelo de simulação, seja qual for o formato do arquivo, e que contenha instruções (código) compatíveis e executáveis no *software* de simulação escolhido.

Algoritmo 2 Tradução do Gerador

Entrada: tamanho do espaço latente $\boldsymbol{z}$; parâmetros $\boldsymbol{w}, \boldsymbol{b}$; hiperparâmetro $\boldsymbol{g}$; informações para reversão do escalonamento e recodificação
Saída: código com operações para amostragem do MDE

1: Definir $\boldsymbol{z}$ ← distribuição normal padrão (tamanho do espaço latente)
2: Definir a equação dos neurônios de camadas intermediárias ($\boldsymbol{w}$, $\boldsymbol{b}$ e $\boldsymbol{g}$)
3: Definir $\boldsymbol{x_1}$ ← vetor de dados sintéticos brutos
4: Definir $\boldsymbol{x_2}$ ← vetor de dados sintéticos sem escalonamento
5: Definir $\boldsymbol{x_3}$ ← vetor de dados sintéticos com arredondamentos, se aplicável
6: Definir $\boldsymbol{x}$ ← vetor de dados sintéticos recodificados, se aplicável
7: Retorna o código com as operações definidas

3.5. Importar Gerador no modelo computacional

Em geral, programas de simulação permitem a importação de arquivos, como planilhas eletrônicas. O FlexSim®, em sua versão 22.2, pode importar arquivos em Excel e, caso em suas células estejam presentes textos compatíveis, é capaz de executá-los utilizando métodos de tabela como o *executeCell()* ou a função *executestring()* (FLEXSIM, 2022).

Uma alternativa à estratégia de importação é a conexão do modelo a um código externo. Isso significa que, durante a simulação, funções específicas definidas e salvas externamente (ou seja, que não foram importadas ao modelo) podem ser chamadas, retornando valores de resposta (FLEXSIM, 2022). Novamente, considerando o FlexSim, a partir de sua versão 22.1, é possível se conectar a códigos externos definidos em Python. Nesse caso, como o TensorFlow permite salvar modelos de AP (suas arquiteturas e parâmetros), utilizando arquivos do tipo HDF5 (para armazenagem de dados heterogêneos), e carregá-los novamente para aplicação (TENSORFLOW, 2022), é possível criar códigos em Python responsáveis especificamente por carregar sob demanda os MDEs salvos pelo TensorFlow, executá-los e retornar os dados de entrada sorteados para o modelo.

Durante a implementação dos códigos, apenas a modalidade de importação do MDE será considerada, pois a funcionalidade de conexão a códigos externos em Python foi disponibilizada pela FlexSim apenas em 04 de abril de 2022.

3.6. Considerações finais

3.6.1. Sobre o tamanho amostral

Se, por um lado, tamanhos amostrais maiores favorecem a geração de dados sintéticos mais próximos da realidade, por outro, o uso de menos dados reduz requisitos de memória e processamento computacional (TOUTOUH; HEMBERG; O'REILLY, 2019). Entretanto, a

definição de um tamanho amostral adequado para o treinamento de GANs permanece um tópico em aberto na literatura. Nesse sentido, estudos têm utilizado os mais diversos tamanhos amostrais. Por exemplo, o número de dados para treinamento observado por Hernandez *et al.*, (2022) em 29 aplicações de GANs em registros de saúde, varia entre aproximadamente 500 e três milhões, com uma mediana de 15 mil. Nesses casos, os altos valores também podem ser explicados pela maior maturidade de bancos de dados em parte das organizações da área de saúde, possibilitando uma oferta maior de dados. Já na área de consumo energético de edifícios, Ye *et al.* (2022) testam amostras de aprendizado de tamanhos 10 a 100, avaliando os resultados das GANs como bons especialmente para tamanhos amostrais maiores que 30 e acompanhados de treinamentos mais longos (1.000 a 2.000 épocas).

O presente estudo medirá a acurácia de MDEs obtidos por meio de GANs, avaliando os resultados para tamanhos amostrais de 64 a 1.024, visando oferecer mais evidências sobre o tópico.

3.6.2. Sobre o tempo computacional para amostragem

Se, por um lado, a utilização de GANs para modelagem de dados de entrada pode representar uma alternativa competitiva em termos de qualidade de representação dos dados reais, o que este trabalho visa avaliar, por outro, também são esperados impactos no tempo computacional para amostragem de um MDE durante a simulação. Esse tempo necessariamente será maior com a utilização de GANs.

O tempo computacional total para execução de um MDE univariado (TT) é uma função do número de sorteios do MDE ($n_{sorteios}$) e do tempo computacional esperado para execução de um único sorteio (t_{MDE}), assim como descrito na Equação 3.6, e partindo-se da premissa de que t_{MDE} é constante.

$$TT = n_{sorteios}.\, t_{MDE} \tag{3.6}$$

Como exemplo, tomemos a modelagem de uma distribuição univariada qualquer. Além disso, suponhamos que o tempo computacional necessário para amostragem de uma DEP, independente de sua família, seja constante e igual a t_{DEP}. Caso opte-se por um MDE do tipo DEP, *softwares* como ExpertFit® retornariam uma distribuição candidata, a qual seria o único componente a ser utilizado para simular o MDE de interesse. Nesse caso, o TT para o MDE do tipo DEP (TT_{DEP}) poderia ser calculado pela Equação 3.7.

$$TT_{DEP} = n_{sorteios}.\, t_{DEP} \tag{3.7}$$

Mas, ao optarmos por um MDE baseado em GAN, a equação matemática para amostragem é a do Gerador, a qual, como definido na seção 3.2.1, inclui o sorteio do espaço latente, normalmente distribuído e de dimensão proposta igual a três vezes o número de variáveis do conjunto de dados. Ou seja, nesse caso, o MDE demandará como ponto de partida três sorteios de distribuições normais para o espaço latente, e, portanto, exigirá no mínimo três vezes TT_{DEP}. Além disso, como o espaço latente passa por camadas de cálculos (as camadas do Gerador) até a obtenção da amostra sintética, por mais que tais operações matemáticas sejam majoritariamente simples, os cálculos da RNA também terão um custo computacional a cada sorteio (t_{RNA}). Assim, o tempo total para execução de um MDE baseado em GAN durante uma simulação (TT_{GAN}) pode ser definido conforme a Equação 3.8.

$$TT_{GAN} = n_{sorteios}(3\ t_{DEP} + \ t_{RNA}) \gg TT_{DEP} \tag{3.8}$$

Entretanto, tal impacto deve ser interpretado relativamente à capacidade de processamento do computador em uso e o tempo de resposta aceitável. Apenas para exemplificar a primeira dimensão citada, em um computador equipado com o processador Intel® Core™ i7-10750H 2,60GHz, o FlexSim em sua versão 22.2 é capaz de realizar dezenas de milhões de sorteios de DEPs ou centenas de milhões de multiplicações ou somas (de dois valores) por segundo. Já em relação ao tempo de resposta aceitável para o modelo de simulação, aplicações mais próximas do tempo real e integradas a outros sistemas da organização demandarão modelos mais ágeis e, assim, o tempo computacional para amostragem dos MDEs pode se tornar crítico.

4. IMPLEMENTAÇÃO

Neste capítulo é descrita a implementação de algoritmos em Python para realização das etapas descritas nas seções 3.1 a 3.4. A última etapa, de importação do Gerador no modelo computacional, não será implementada por meio de um programa, pois o *software* FlexSim já é dotado das funcionalidades necessárias para tal.

Seguindo a estrutura discutida, o programa foi organizado em módulos temáticos. Módulos são objetos (arquivos) que servem como unidade de organização de códigos em Python, podendo ser importados por outros códigos quando necessário (PYTHON SOFTWARE FOUNDATION, 2022). Os Apêndices A a F reúnem os códigos desenvolvidos para cada uma das etapas da proposta e para atividades auxiliares, conforme apresentado nos Quadros 4.1 e 4.2.

Quadro 4.1 – Códigos para as etapas do método MDE-GANs

Etapa da proposta	**Código de implementação**
Ler e preparar dados	Apêndice A – Módulo *gandata*
Configurar GAN	Apêndice B – Módulo *gan*
Treinar GAN	
Avaliar Gerador	Apêndice C – Módulo *c2st*
Traduzir o gerador	Apêndice D – Módulo *savemodel*

Quadro 4.2 – Códigos auxiliares

Atividades auxiliares	**Código de implementação**
Utilidades em geral	Apêndice E – Módulo *utilities*
Visualização dos dados	Apêndice F – Módulo *ganplot*

As lógicas foram definidas por meio de classes e funções, que, segundo a *Python Software Foundation* (2022), podem ser definidos como:

- **Classes:** *templates* que proporcionam uma forma de organizar de forma conjunta dados (atributos) e funcionalidades (métodos) aplicáveis a algum objeto;
- **Funções:** séries de comandos que podem retornar algum valor de resposta.

Duas classes estão presentes na implementação, *Data* e GAN. Seus principais atributos e métodos estão apresentados no diagrama UML (Figura 4.1). Para entendimento, a notação UML para classes subdivide as informações em ao menos três células sequenciais: nome da classe, atributos e métodos. Além disso, utiliza o sinal "+" para indicar elementos que podem ser acessados pelo usuário (públicos). Por fim, nesta tese, apenas para organização, elementos com nome iniciado por "_" são destinados ao uso interno nas lógicas, sem interface com o usuário.

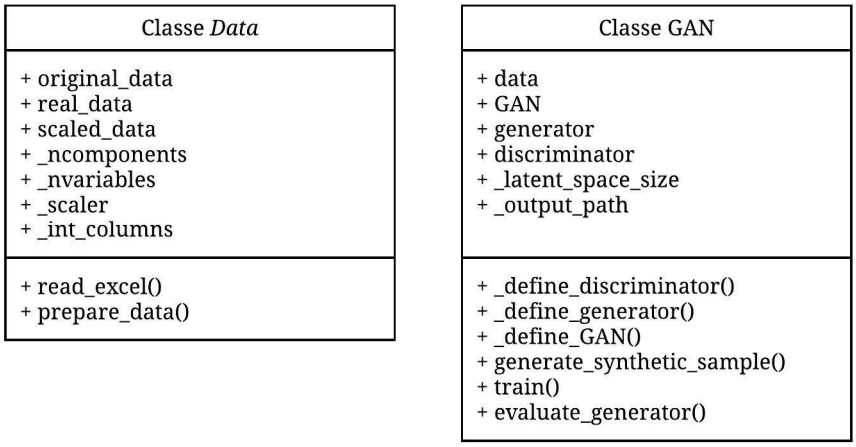


Figura 4.1 – Classes e seus principais atributos e métodos

A classe *Data* está definida no módulo *gandata* e reúne atributos e métodos relacionados aos dados reais fornecidos para treinamento das GANs, descritos no Quadro 4.3 e utilizados na etapa de leitura e preparação dos dados.

Quadro 4.3 – Principais atributos e métodos da classe Data

Tipo	Nome	Descrição
Atributo	*original_data*	*DataFrame* Pandas com os dados originais fornecidos para treinamento.
	real_data	*DataFrame* Pandas com os dados tratados. Ou seja, sem dados faltantes e com variáveis categóricas codificadas.
	scaled_data	*DataFrame* Pandas com os dados normalizados.
	_ncomponents	Número de componentes que explicam a variação dos dados reais, resultado do PCA. Será utilizado para definição de hiperparâmetros das GANs dependentes da dimensionalidade dos dados.
	_nvariables	Número de variáveis presentes nos dados. Será utilizado para definir a densidade da camada de saída do Gerador e da de entrada do Discriminador.
	_scaler	Escalonador utilizado para normalização dos dados, já treinado. Será utilizado para reverter a normalização dos dados sintéticos.
	_int_columns	Vetor de nomes das variáveis inteiras. Será utilizado para construção da equação do MDE, incluindo etapas de arredondamento.
Método	*read_excel()*	Facilitar a leitura de planilhas em Excel.
	Prepara_data()	Executar atividades para converter os dados originais (*original_data*) até chegarem ao estado normalizado (*scaled_data*).

Já classe GAN está definida no módulo *gan* e reúne atributos e métodos relacionados às GANs, descritos no Quadro 4.4 e utilizados na etapa de configuração e treinamento das GANs.

Por fim, o Quadro 4.5 lista as principais funções presentes nos demais módulos.

Quadro 4.4 – Principais atributos e métodos da classe GAN

Tipo	Nome	Descrição
Atributo	*data*	Objeto da classe *Data*. Será utilizado para configuração e treinamento das GANs.
	GAN	Modelo de RNA em TensorFlow que combina Gerador e Discriminador.
	Generator	Modelo de RNA em TensorFlow para o Gerador. É o elemento que será traduzido, após o treinamento, para utilização como MDE.
	Discriminator	Modelo de RNA em TensorFlow para o Discriminador.
	_latent_space_size	Tamanho do espaço latente, igual a três vezes o número de componentes dos dados reais. Será utilizado para definir a densidade da camada
	_output_path	Caminho utilizado para salvar os resultados do treinamento das GANs.
Método	*define_discriminator()*	Definir a RNA para o Discriminador.
	define_generator()	Definir a RNA para o Gerador.
	define_GAN()	Combinar as RNAs criadas para o Discriminador e para o Gerador.
	generate_synthetic_sample()	Gerar dados sintéticos ao longo do treinamento, para o próprio treinamento e para avaliação da qualidade do gerador.
	train()	Treinar as GANs.
	evaluate_generator()	Avaliar a qualidade de representação do Gerador, executando as funções do teste C2ST.

Quadro 4.5 – Principais funções desenvolvidas

Módulo	Função	Descrição
c2st	*c2st_knn()*	Realizar o teste C2ST baseado no classificador k-NN
savemodel	*save_input_model()*	Traduzir a RNA do Gerador para a linguagem utilizada pelo FlexSim, adicionando etapas para reversão do escalonamento, arredondamento e decodificação de variáveis categóricas, caso aplicável.
Ganplot	*plot_data()*	Comparar dados reais e sintéticos visualmente por meio de histograma (dados univariados) ou gráficos pareados (dados multivariados).
	qqplot()	Comparar dados reais e sintéticos visualmente por meio de gráficos Q-Q.
	plot_losses()	Acompanhar visualmente a evolução das funções de perda do Gerador e do Discriminador ao longo do treinamento.
Utilities	*C2ST_to_model_accuracy()*	Converter a acurácia do teste C2ST em acurácia do MDE, conforme a Equação 1.5.
	Model_to_C2ST_accuracy()	Converter a acurácia do MDE em acurácia do C2ST, conforme a Equação 1.5.

5. OBJETOS DE ESTUDO

Os objetos de estudo considerados para o presente trabalho são distribuições de probabilidade. Destas distribuições, serão obtidas amostras e, a partir delas, realizada a atividade de modelagem de dados de entrada. A tese avalia 14 objetos de estudo (OE), os quais se agrupam em distribuições teóricas paramétricas, teóricas não paramétricas e reais.

5.1. Distribuições teóricas paramétricas

As distribuições teóricas paramétricas foram escolhidas para avaliação do desempenho de GANs para modelagem de dados de entrada com diferentes características, de acordo com os critérios de inclusão definidos na seção 1.5.3.2. Foram considerados oito objetos de estudo: normal (OE01), lognormal (OE02), exponencial (OE03), bimodal (OE04), Poisson (OE05), normal bivariada (OE06), normal multivariada (OE07) e multinomial (OE08). O Quadro 5.1 apresenta possíveis aplicações destas distribuições, conforme exemplificado por Law (2014) e Montgomery e Runger (2013).

Quadro 5.1 – Possíveis aplicações das distribuições objeto de estudo

Distribuição	Possíveis aplicações
Normal	Erros de diversos tipos, quantidades que são somas de um grande número de outras quantidades.
Lognormal	Tempo para realizar uma atividade, quantidades que são produto de um grande número de outras quantidades.
Exponencial	Tempos entre chegadas, tempos entre falhas.
Multimodal	Tempos para reparo, quando há tipos de falhas distintos.
Poisson	Número de eventos em um intervalo de tempo (se ocorrem a uma taxa constante), número de itens de um lote de tamanho aleatório.
Normal multivariada	Possui uso limitado como modelo de dados de entrada, mas pode servir como ponto de partida para outras distribuições multivariadas mais úteis.
Multinomial	Número de ocorrências de um conjunto de categorias.

Adaptado de Law (2014) e Montgomery e Runger (2013)

Já o Quadro 5.2 apresenta a classificação desses objetos de estudo conforme os critérios de inclusão discutidos em 1.5.3.2.

Quadro 5.2 – Classificações dos objetos de estudo

Distribuição	Homogeneidade	Interdependência
Normal	Homogênea	Independente
Lognormal		
Exponencial		
Bimodal	Univariada	
Poisson	Homogênea	
Normal bivariada		Dependente
Normal multivariada		
Multinomial		

Os parâmetros considerados para cada uma das distribuições encontram-se descritos no Quadro 5.2, enquanto as Tabelas 5.1 e 5.2 complementam as informações para a normal bivariada e a normal multivariada, respectivamente, apresentando suas matrizes variância-covariância.

Quadro 5.3 – Parâmetros dos objetos de estudo

Distribuição	Parâmetros
Normal	Média = 100; Desvio = 3
Lognormal	Média = 0; Variância = 0,6 (referentes à normal subjacente)
Exponencial	Escala = 1
Bimodal	Mistura de duas normais (50%/50%) Normal "A": Média = 100; Desvio = 3 Normal "B": Média = 110; Desvio = 1
Poisson	Lambda = 10
Normal bivariada	Médias = [100; 100]
Normal multivariada (5 variáveis)	Médias = [0; 0; 0; 0; 0]
Multinomial (6 variáveis)	Probabilidades = [0,17; 0,17; 0,17; 0,17; 0,17; 0,17]

Tabela 5.1 – Matriz variância-covariância para normal bivariada

3,0	2,4
2,4	3,0

Tabela 5.2 – Matriz variância-covariância para normal multivariada

0,60	0,63	-1,21	-2,25	0,46
0,63	2,39	-0,13	-2,36	2,24
-1,21	-0,13	3,99	4,99	1,33
-2,25	-2,36	4,99	10,39	-0,93
0,46	2,24	1,33	-0,93	4,82

Por fim, serão demonstrados visualmente os dados de cada objeto de estudo, construídos a partir de amostras de tamanho 10 mil (tamanho escolhido apenas para demonstrar claramente as características das distribuições). A Figura 5.1 apresenta os histogramas para os casos univariados (OE01 a OE05), enquanto as Figuras 5.2 a 5.4 apresentam gráficos pareados para os casos multivariados (OE06 a OE08). É possível notar que os objetos de estudo apresentam características diversas. Ressalta-se que isso também é observado nas dependências presentes nos dados multivariados: para o caso normal bivariado, existe uma correlação positiva; para o normal multivariado, correlações positivas e negativas de intensidades variadas; e para o caso multinomial, uma dependência mais complexa, pois exige que $\sum_{i=1}^{6} x_i$ seja constante, com x_i sendo cada uma das seis variáveis do conjunto de dados.

5.2. Distribuições teóricas não paramétricas

As distribuições teóricas não paramétricas foram selecionadas a partir do estudo *Datasaurus* (CAIRO, 2016), o qual, com o objetivo de conscientizar sobre a importância da visualização de dados, reúne conjuntos de dados bivariados que, apesar de apresentarem

algumas estatísticas idênticas (com precisão de duas casas decimais), são significativamente distintos quando visualizados graficamente. Isso ocorre devido a estruturas de dependência complexas. Essas distribuições foram escolhidas para se avaliar a capacidade de GANs em capturar tais dependências.

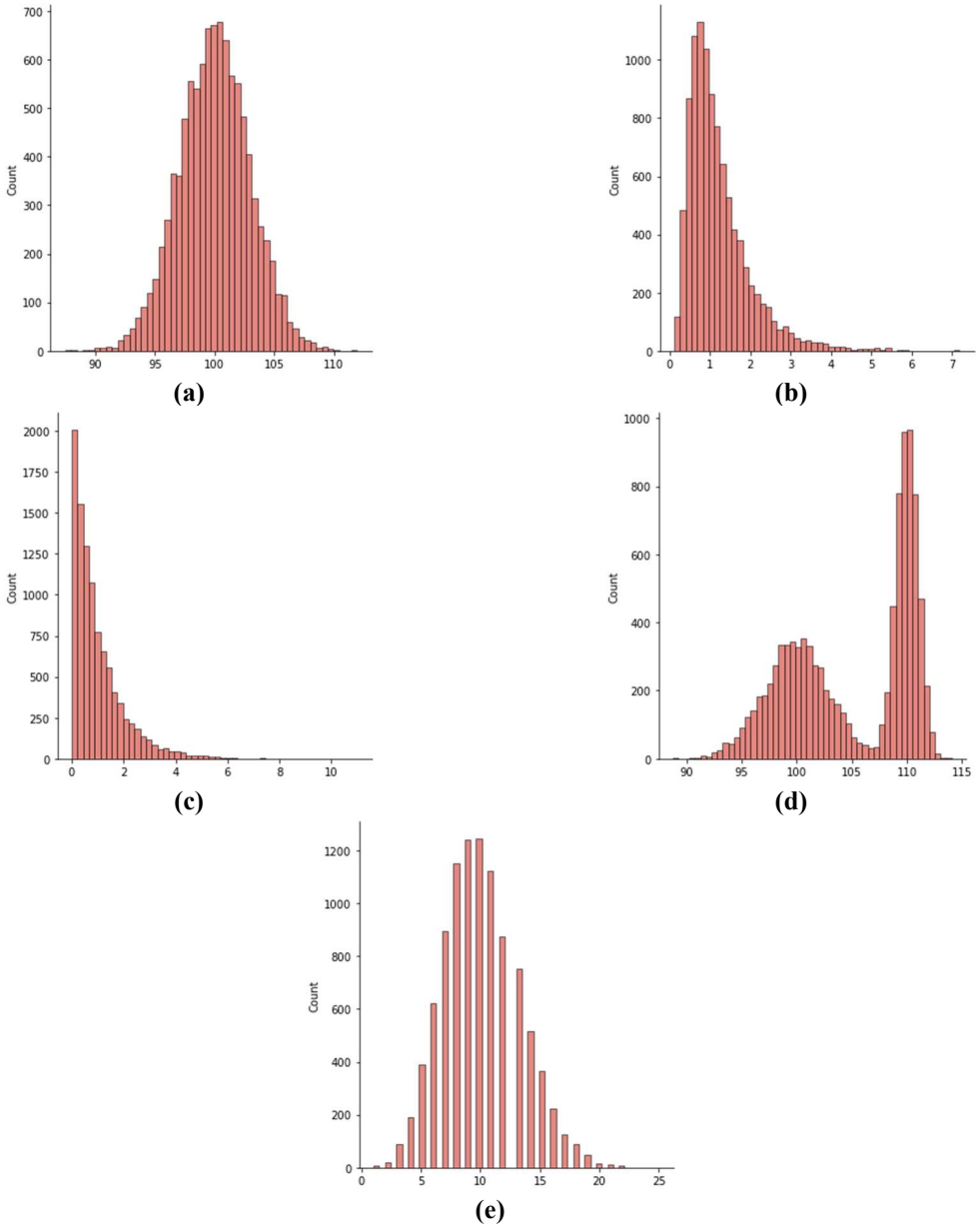


Figura 5.1 – Visualização dos objetos de estudo OE01 a OE05. (a) normal. (b) lognormal. (c) exponencial. (d) bimodal. (e) Poisson.

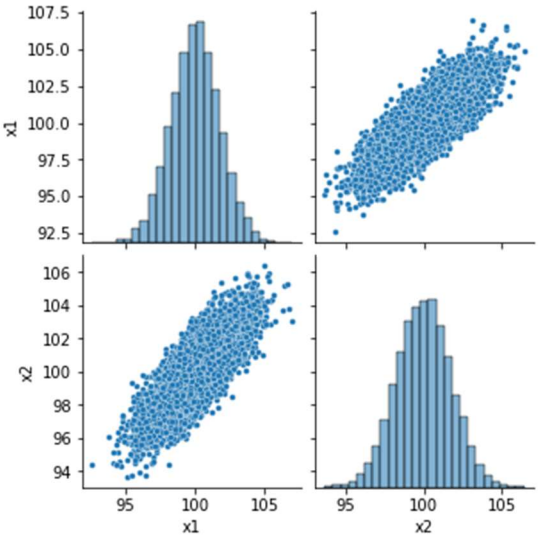


Figura 5.2 – Visualização do OE6 (normal bivariada)

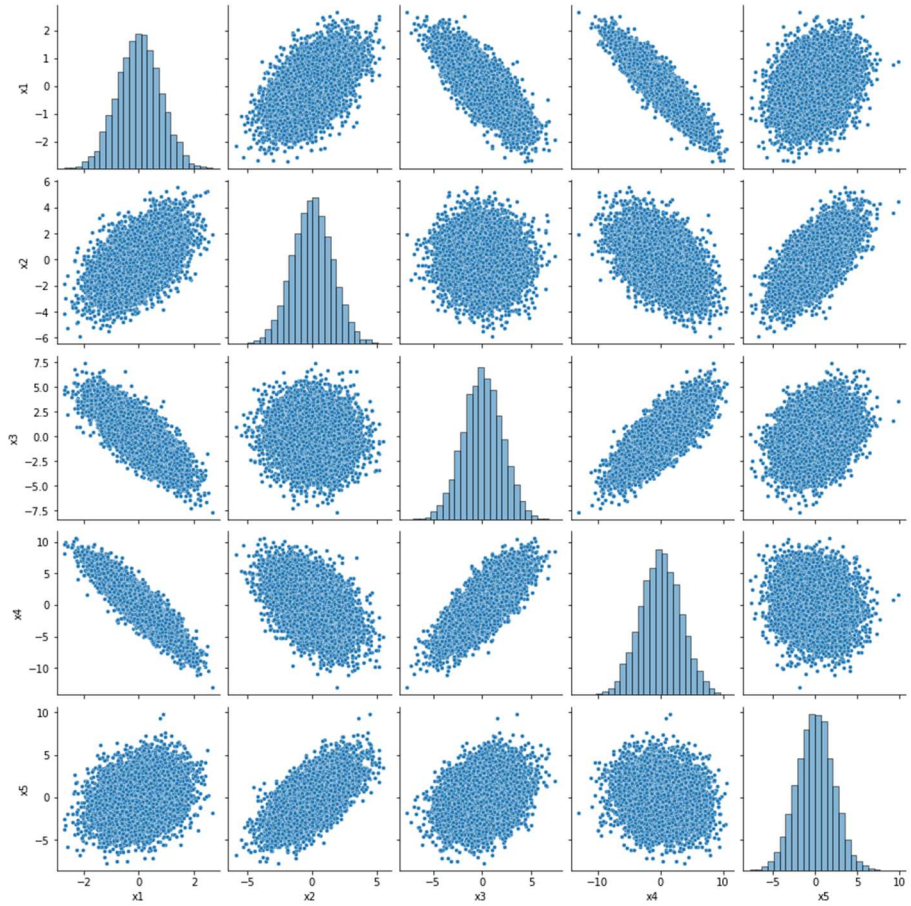


Figura 5.3 – Visualização do OE07 (normal multivariada)

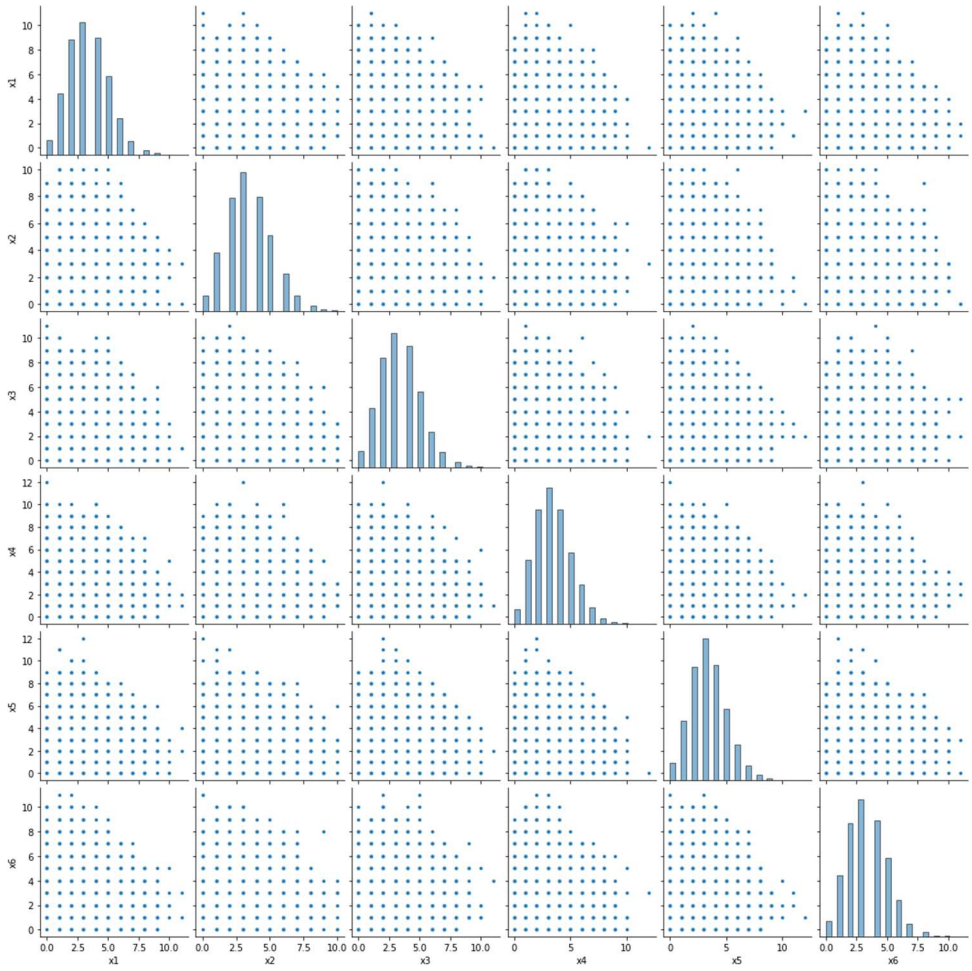


Figura 5.4 – Visualização do OE08 (multinomial)

Foram considerados quatros objetos de estudos, os conjuntos denominados “xis” (OE09), “círculo” (OE10), “estrela” (OE11) e “linhas paralelas” (OE12), que podem ser visualizados por meio dos gráficos pareados na Figura 5.5. Nos quatro casos, as variáveis x e y possuem, respectivamente, médias 54,26 e 47,83, desvios-padrão 16,76 e 26,93 e correlação -0,06. Além disso, os autores disponibilizam apenas 142 observações de cada conjunto e, por essa razão, não será possível avaliar o treinamento das GANs para a grade planejada de tamanhos amostrais. Nesse caso, será avaliado apenas o desempenho obtido por meio do treinamento utilizando todas as 142 observações disponíveis.

5.3. Distribuições reais

Por fim, os objetos de estudo reais abrangem dois conjuntos de dados utilizados em projetos de simulação. Além de servirem igualmente para avaliação da qualidade de

representação obtida por GANs, essas distribuições servirão como demonstração de aplicação em situações reais.

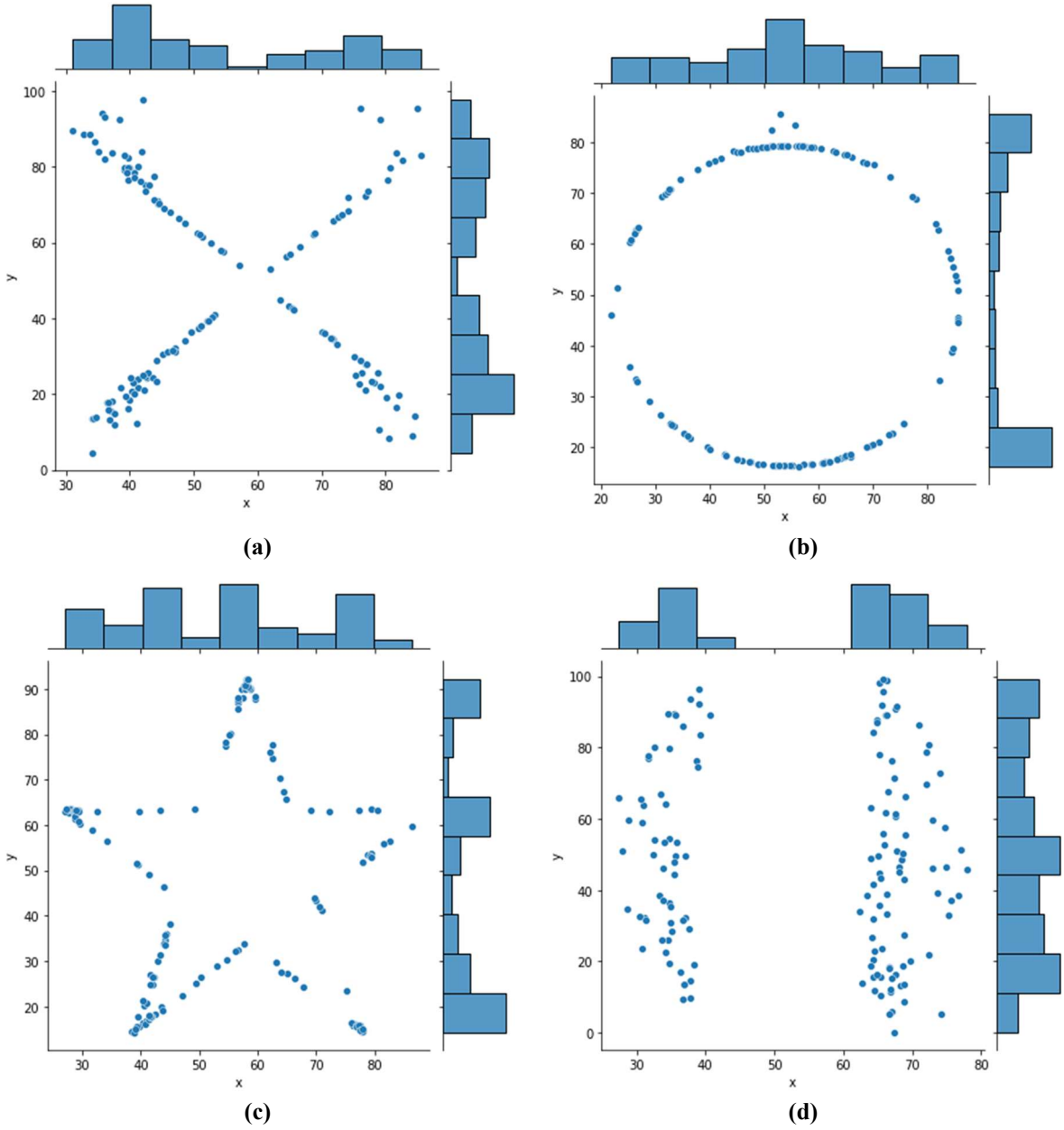


Figura 5.5 – Visualização dos objetos de estudo OE09 a OE12. (a) “xis”. (b) “círculo”. (c) “estrela”. (d) “linhas paralelas.

O primeiro deles (OE13) é oriundo de uma empresa do setor de transformação. O projeto de simulação foi desenvolvido e concluído antes da proposta dessa tese, abrangendo atividades de logística interna em uma das plantas da empresa. Apesar de ter sido construído o modelo de SED, este não será disponibilizado, por razões de confidencialidade. A empresa disponibilizou um conjunto de dados históricos que abrange características dimensionais (diâmetro e largura,

em centímetros, e peso, em toneladas) de mais de 40 mil unidades de seus produtos. Essas dimensões são relevantes para a modelagem do sistema por estarem diretamente relacionadas aos tempos dos processos. Os dados podem ser visualizados por meio da Figura 5.6. Trata-se de um conjunto de dados desafiador, com a presença de dependências entre variáveis e irregulares nas distribuições.

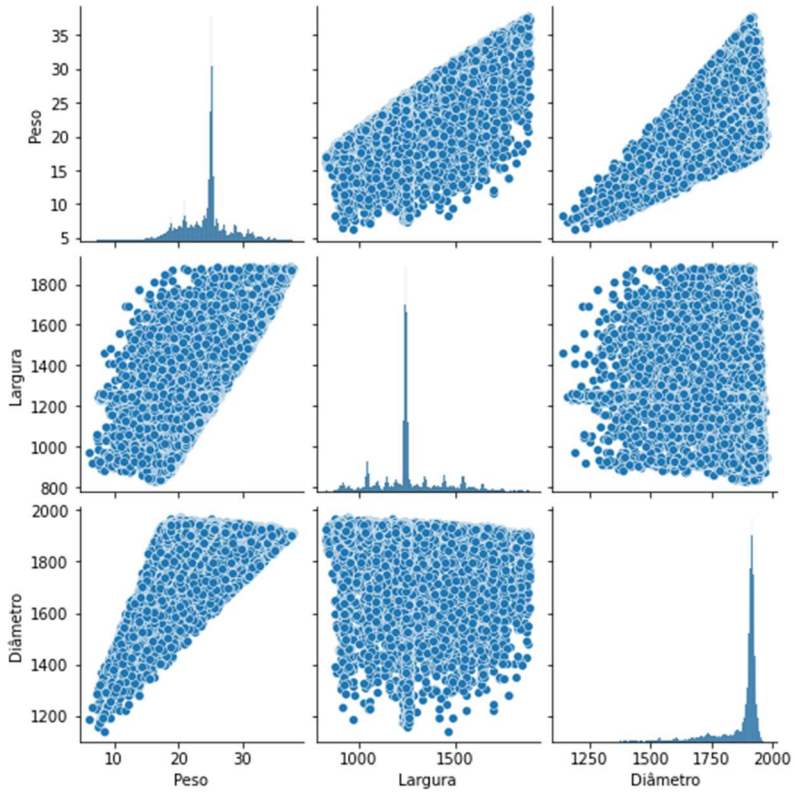


Figura 5.6 – Visualização do OE13 (dimensões de produtos)

O segundo objeto de estudo real (OE14) abrange dados utilizados no projeto de SED do departamento de pronto atendimento de um dos maiores hospitais brasileiros. O modelo de simulação (Figura 5.7) foi desenvolvido e concluído antes da proposta dessa tese, abrangendo atividades de diagnóstico e tratamento de pacientes. Nesse caso, foram disponibilizados históricos de dados relacionados à performance do atendimento, sem qualquer identificação dos pacientes, garantindo a confidencialidade. O conjunto de dados escolhido para o OE14 contém cerca de 100 mil registros e abrange tempos de triagem e de consulta, ambos em minutos, além da classificação de risco do paciente, uma nota variando de 1 a 5, sendo 1 o valor mais crítico. Tal classificação é relevante para o modelo, pois determina os fluxos pelos quais o paciente passará. Os dados podem ser visualizados pelos gráficos na Figura 5.8.

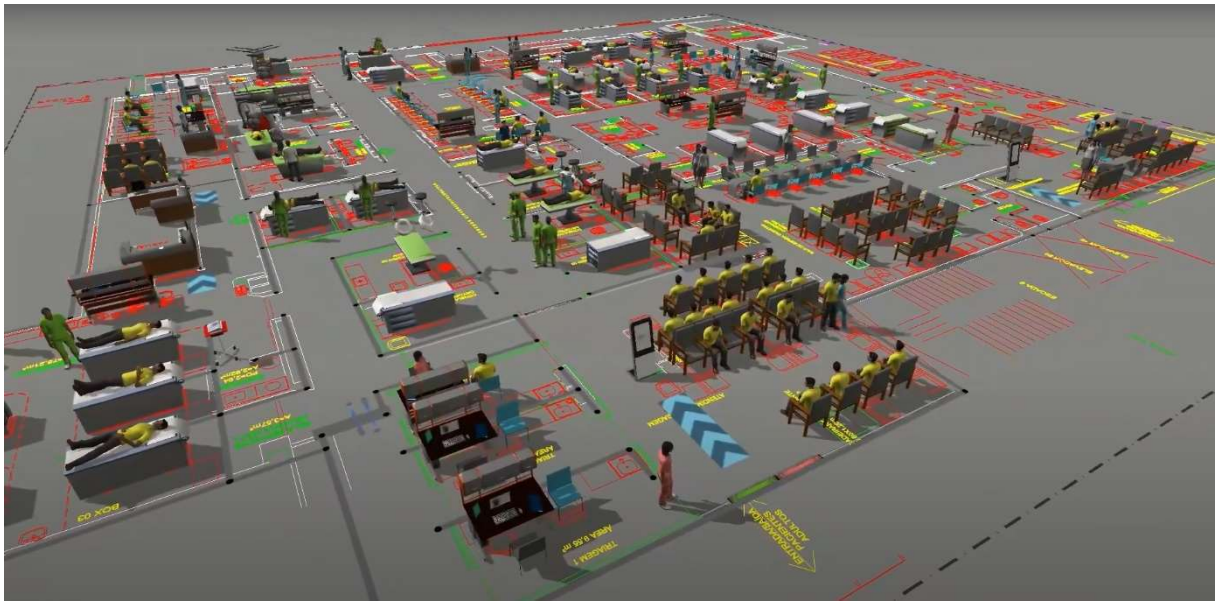


Figura 5.7 – Modelo de simulação do pronto atendimento

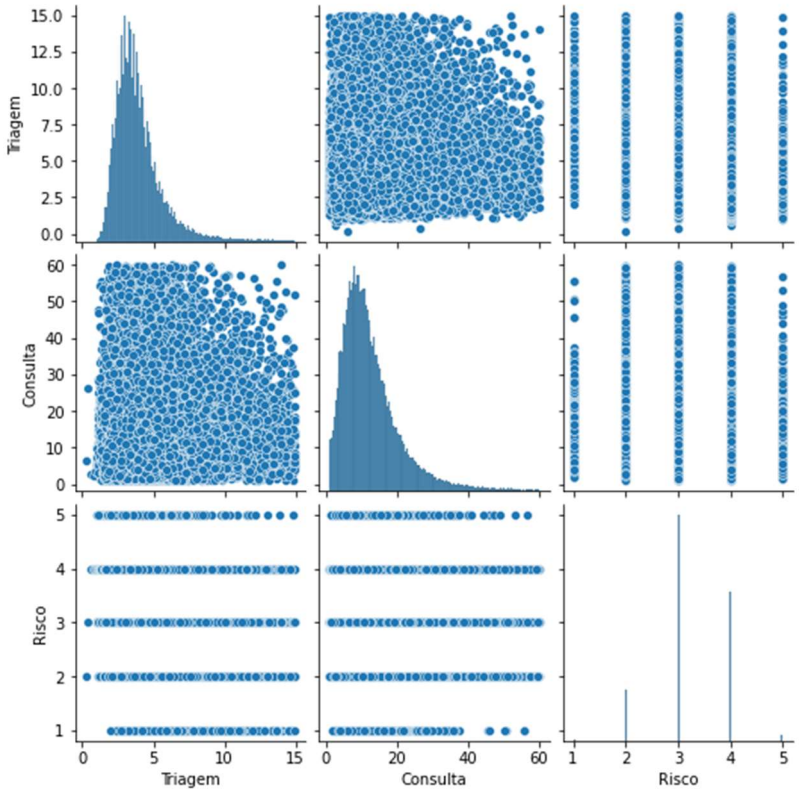


Figura 5.8 – Visualização do OE14 (pronto atendimento)

6. APLICAÇÃO DA PROPOSTA E RESULTADOS

Este capítulo descreve os resultados obtidos por meio da aplicação da proposta. Para todos os objetos de estudo, as GANs foram treinadas para um número fixo de 16.384 passos de aprendizado. Considerando o tamanho de lote de 64, o número de épocas variou de 1.024 a 16.384, dependendo do tamanho amostral para treinamento. Além disso, foram definidos cinco pontos de checagem durante o treinamento.

O Apêndice G demonstra a utilização da proposta, incluindo a execução de cada uma das grandes etapas descritas no Capítulo 3. Já o Apêndice H exemplifica o formato do MDE exportado para o objeto de estudo "círculo" do conjunto *Datasaurus*. Neste capítulo, a seção 6.1 explora a evolução do treinamento ao longo das épocas, enquanto as seções 6.2 a 6.5 avaliam os resultados obtidos. Por fim, considerações finais são realizadas na seção 6.6.

6.1. Evolução do treinamento

Como afirmado anteriormente, o processo de treinamento de GANs ocorre de forma iterativa. O conjunto de dados para treinamento é subdividido em lotes de tamanho fixo que são fornecidos às GANs para aprendizagem. A cada passo do treinamento, os parâmetros das RNAs são atualizados por meio da minimização da função de custo (entropia cruzada binária), diretamente relacionada à taxa de acertos obtida pelo Discriminador ao classificar as observações como reais ou sintéticas e ao objetivo de cada componente da GAN. A função custo do Gerador é inversamente proporcional à acurácia do Discriminador quando este recebe apenas dados sintéticos. Já a função custo do Discriminador é diretamente proporcional à acurácia quando este recebe dados verdadeiros ou sintéticos. Ao longo dos passos (*steps*) de treinamento, o algoritmo armazena os resultados da função de custo (*loss*) para o Discriminador (D) e Gerador (G).

A Figura 6.1 demonstra a evolução das funções de custo ao longo do treinamento da GAN para o OE07 (normal multivariada) e tamanho amostral de 1.024. É possível notar que o comportamento se estabiliza após uma certa quantidade de passos. Isso não significa que Gerador e Discriminador deixaram de aprender, mas que estão competindo de forma equilibrada. Cada melhoria no desempenho do Gerador é acompanhada por uma melhoria no desempenho do Discriminador, fazendo com o que os resultados sejam melhorados continuamente. Já a Figura 6.2 apresenta a mesma evolução, mas para o OE08 (normal multinomial). Aqui, é possível notar que a partir do passo 7.500 se inicia um processo de degradação da estabilidade, o que pode levar ao fenômeno conhecido como colapso. Por essa

razão, o critério de pontos de checagem incluso na proposta é importante, permitindo que sejam registrados os resultados das GANs antes de um eventual colapso. Com isso, ao final do treinamento, o MDE assumirá os parâmetros do melhor registro realizado.

Considerando as 400 execuções do método para os casos teóricos paramétricos, mais da metade utilizou para gerar o MDE um ponto de checagem anterior à última época de treinamento, como demonstra a Figura 6.3, devido à identificação de que pontos anteriores apresentavam melhor qualidade de representação.

Nas Figuras 6.1 e 6.2, nota-se também que a função de custo do Gerador (verde) é sempre superior à do Discriminador quanto este recebe dados sintéticos (laranja). Tal fato é importante e significa que, por mais que o Gerador melhore, este não é capaz de enganar totalmente o Discriminador, tendo sempre um gradiente que o leva a melhorar o desempenho. Caso contrário, ou seja, se o Gerador tivesse uma função de custo inferior ao Discriminador, haveria indicação de que os dados sintéticos facilmente confundem o Discriminador, levantando a um incentivo baixo ou mesmo nulo para aprendizado do Gerador.

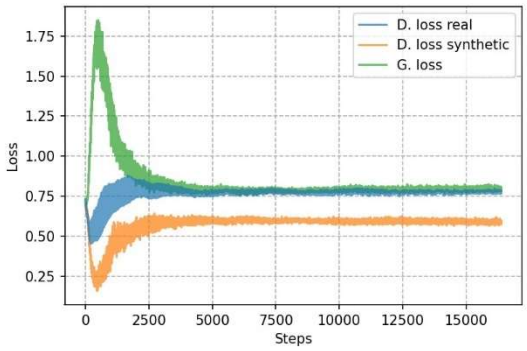


Figura 6.1 – Funções de custo para OE07 (normal multivariada)

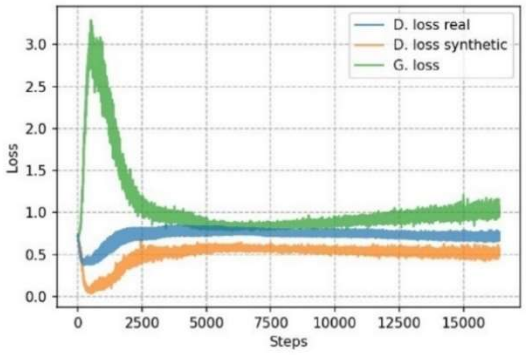


Figura 6.2 – Funções de custo para OE08 (multinomial)

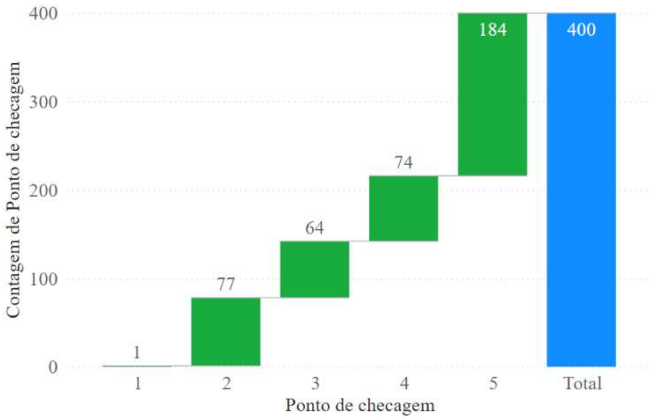


Figura 6.3 – Pontos de checagem utilizados para exportação do MDE

6.2. Casos teóricos paramétricos

A primeira questão de pesquisa busca entender se GANs são adequadas para modelagem de dados de entrada, considerando-as adequadas se proporcionarem uma forte qualidade de representação, conforme descrito na hipótese H1. Como discutido anteriormente, o conceito de forte qualidade de representação adotado está ligado aos níveis de A_{MDE}, significando uma acurácia de ao menos 64%. Para cada uma das réplicas realizadas para o processo de MDE-GANs nos casos teóricos paramétricos, foi obtida por meio do teste C2ST a classe de A_{MDE} para a qual houvesse significância estatística de pertencimento, variando de mínima a quase perfeita. Para essa avaliação, independentemente do tamanho amostral para treinamento, o teste C2ST recebeu 100.000 dados sintéticos, oriundos do MDE, e 100.000 dados verdadeiros. A Figura 6.4 apresenta a frequência observada de cada classe de acurácia para as distribuições teóricas paramétricas univariadas e considerando os diferentes tamanhos amostrais para treinamento avaliados.

É possível notar que para estes objetos de estudo, mesmo o menor tamanho amostral permite atingir ao menos a classe de acurácia forte, enquanto o tamanho amostral de 256 foi, na maioria dos casos, o menor patamar necessário para obtenção de acurácias quase perfeitas.

Já a Figura 6.5 apresenta a frequência observada de cada classe de acurácia para as distribuições teóricas paramétricas multivariadas, também considerando os diferentes tamanhos amostrais para treinamento avaliados. Mais uma vez, em todos os casos foi identificado ao menos um tamanho amostral capaz de entregar níveis de acurácia fortes.

Porém, nota-se que o número de variáveis presentes nos conjuntos de dados apresenta uma relação com a classe de A_{MDE} atingida por GANs. Enquanto no caso normal bivariado o tamanho amostral de 64 continua sendo suficiente para obtenção de fortes acurácias, as amostras mínimas sobem para 265 no caso multivariado (cinco variáveis) e 512 no caso multinomial (seis variáveis). Isso pode indicar que quanto maior a dimensionalidade dos dados, maior o tamanho amostral necessário para capturar as dependências entre as variáveis existentes.

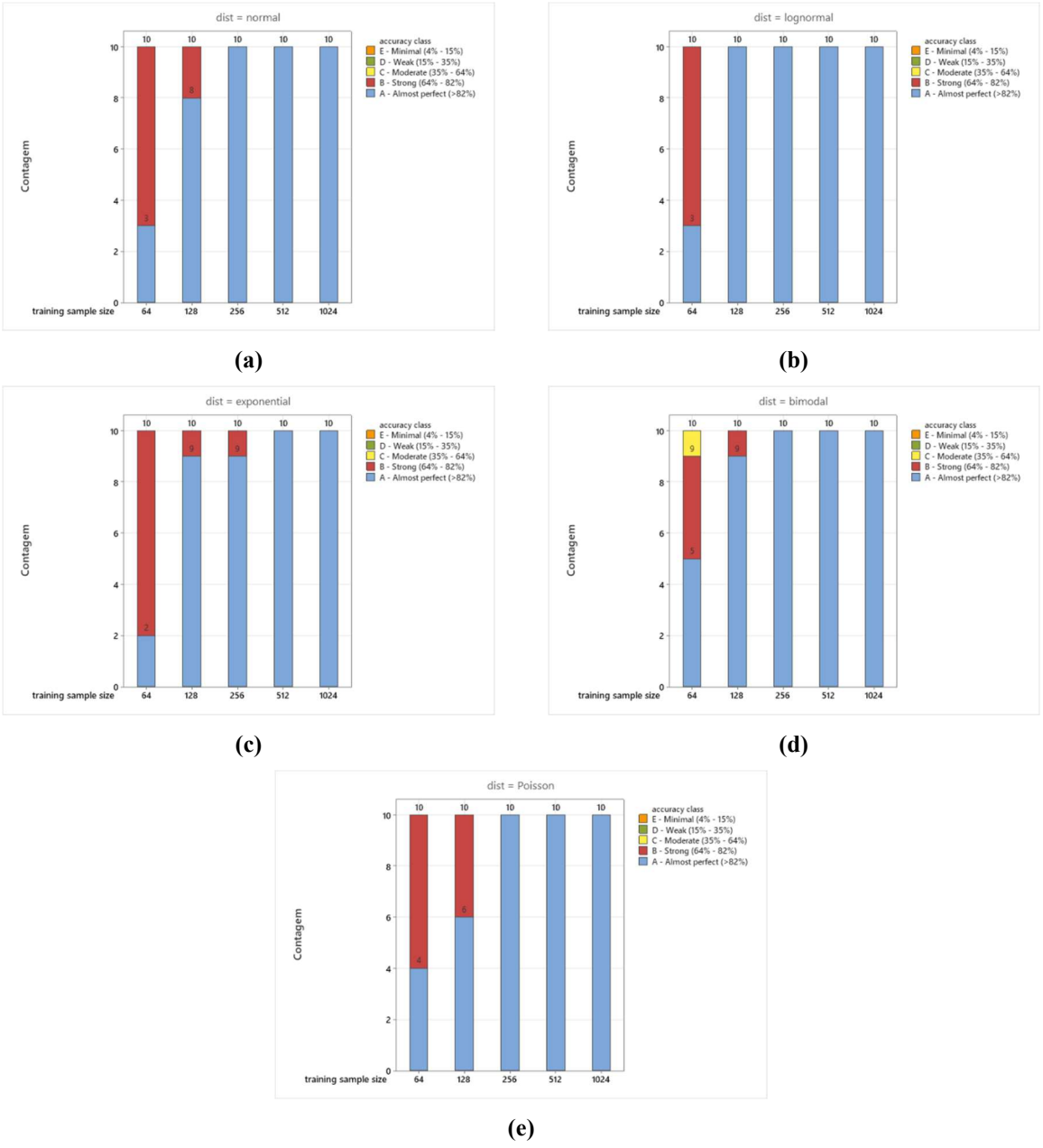


Figura 6.4 – Classes de A_{MDE} obtidas nos OE01 a OE05 (univariados). (a) normal. (b) lognormal. (c) exponencial. (d) bimodal. (e) Poisson.

Para avaliar graficamente a aderência dos dados sintéticos gerados pelo MDE baseado em GANs, a Figura 6.6 apresenta histogramas para os casos teóricos univariados, sobrepondo os gráficos obtidos para as amostras de dados reais utilizados no treinamento (verde) e amostras de dados sintéticos (vermelho) de mesmo tamanho (1.024). Aqui, todas as variáveis são denotadas como x_1. Quanto mais sobrepostos estiverem os dados reais e sintéticos, melhor a capacidade da GAN em gerar dados próximos da realidade.

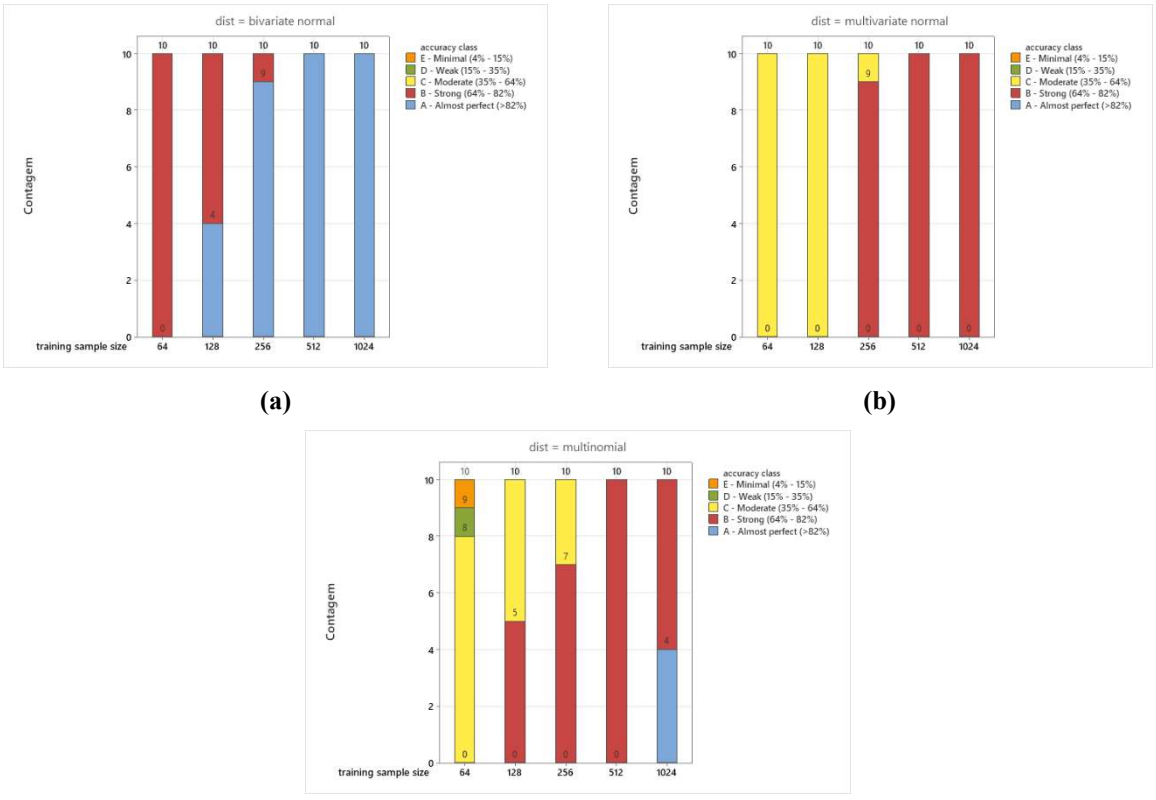


Figura 6.5 – Classes de A_{MDE} obtidas nos OE06 a OE08 (multivariados). (a) normal bivariada. (b) normal multivariada. (c) multinomial.

Os resultados confirmam a capacidade de GANs em capturar características como assimetria e bimodalidade, ressaltando que em todos os casos não houve a necessidade de configuração manual de quaisquer parâmetros de funcionamento. Além disso, nota-se que mesmo com faixas de valores diferentes, o algoritmo permite o aprendizado, o que se deve em muito ao processo de normalização dos dados para treinamento. Por fim, apesar de serem notadas algumas divergências no histograma do caso Poisson as distribuições ainda se assemelham.

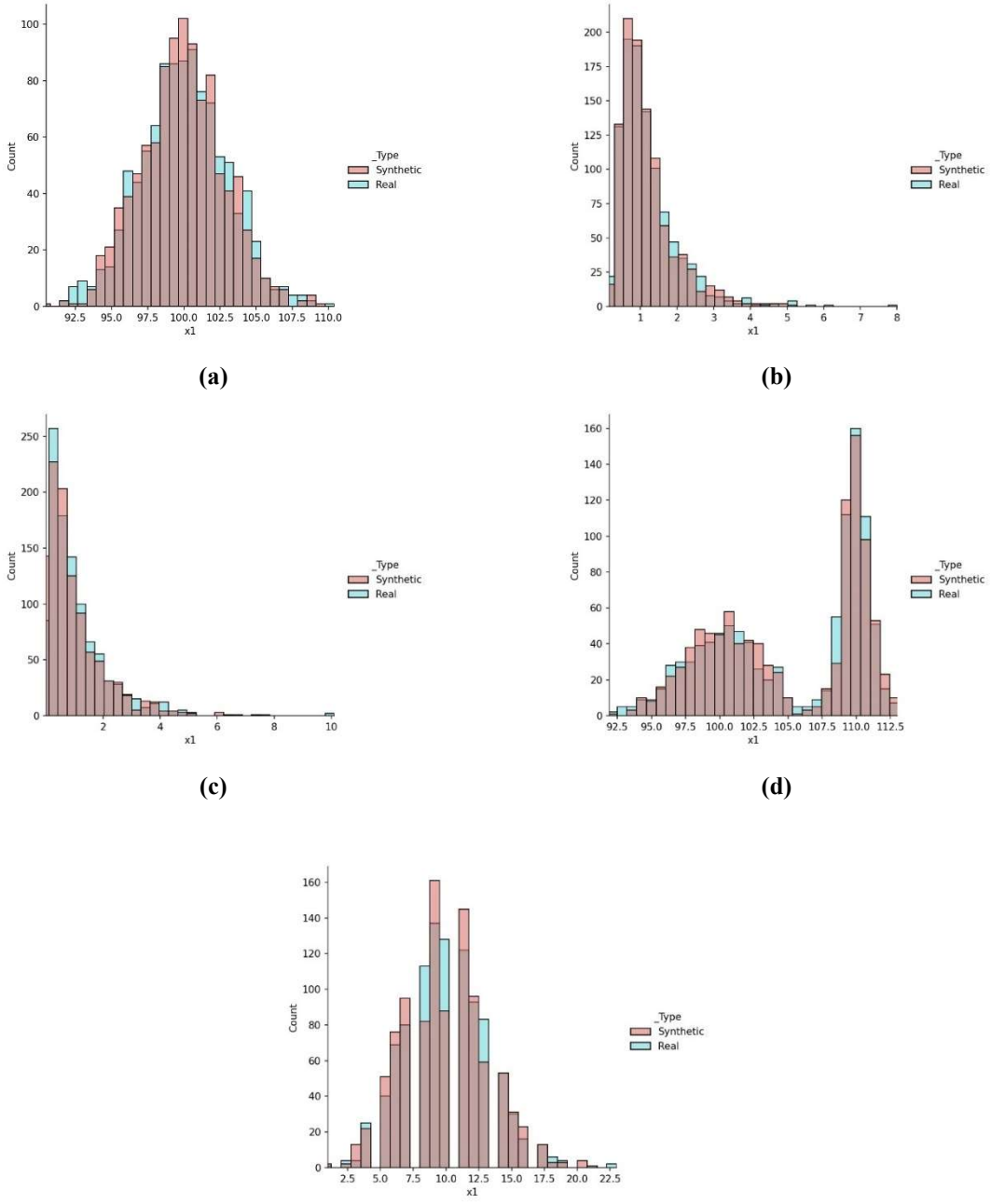


Figura 6.6 – Histograma para os dados sintéticos e reais dos OE01 a OE05 (univariados). (a) normal. (b) lognormal. (c) exponencial. (d) bimodal. (e) Poisson.

Complementarmente, a Figura 6.7 apresenta gráficos Q-Q comparando os quantis da amostra oriunda da distribuição verdadeira (real) e dos dados sintéticos obtidos pelo gerador. Aqui, quanto mais próximos estiverem os dados da diagonal em vermelho, mais próximas as distribuições.

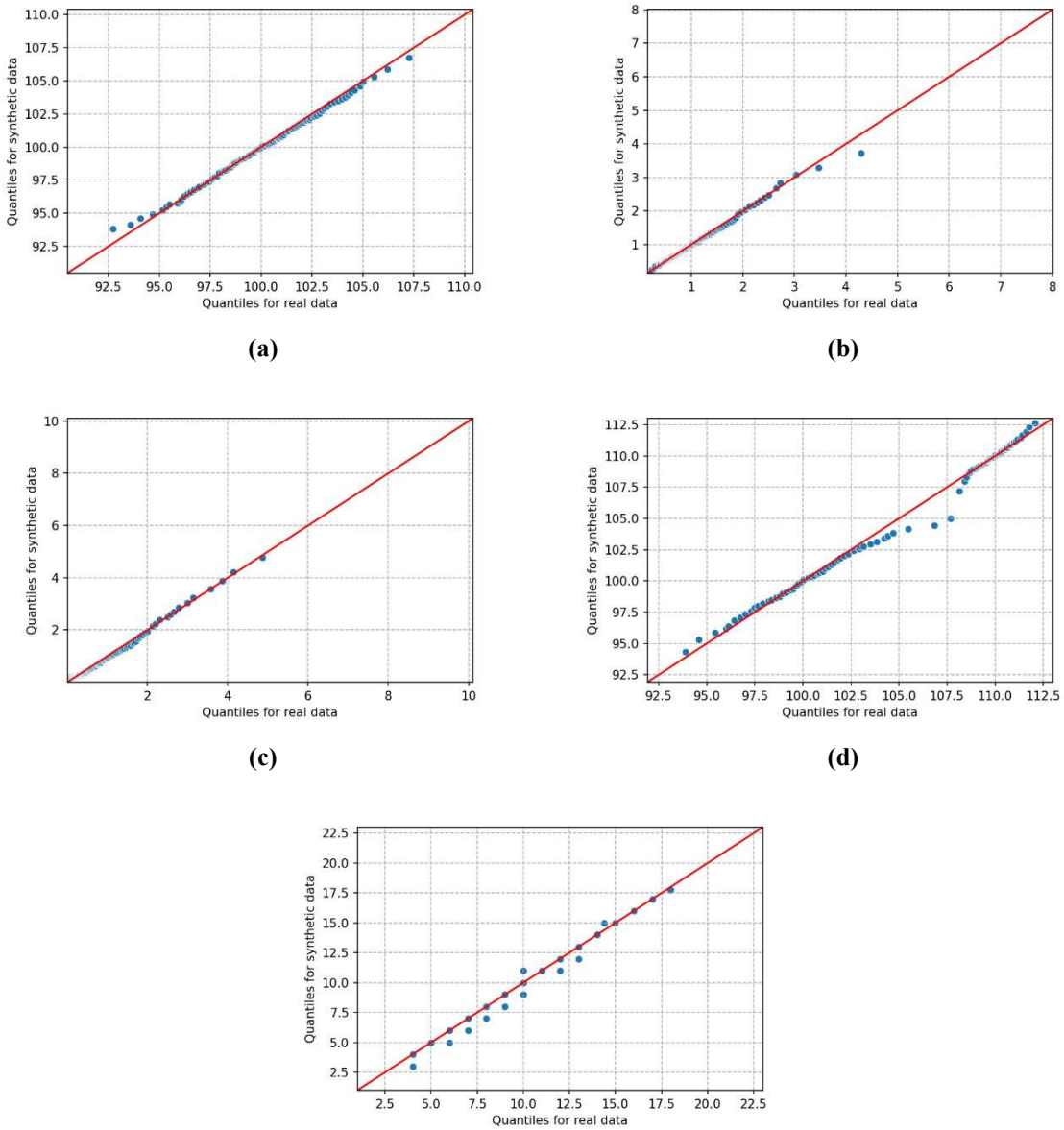


Figura 6.7 – Gráfico Q-Q para os dados sintéticos e reais dos OE01 a OE05 (univariados). (a) normal. (b) lognormal. (c) exponencial. (d) bimodal. (e) Poisson.

Já para os casos multivariados, os resultados podem ser visualizados por meio dos gráficos pareados. A Figura 6.8 apresenta os resultados para o OE06 (normal bivariada), enquanto a Figura 6.9 para o OE07 (normal multivariada). Todos eles se referem aos treinamentos com amostras de tamanho 1.024. As variáveis x_i representam cada uma das i variáveis do caso, conforme a dimensionalidade de cada um. É possível notar que MDEs baseados em GANs foram capazes de capturar dependências de diferentes modalidades e intensidades entre variáveis em todos os casos.

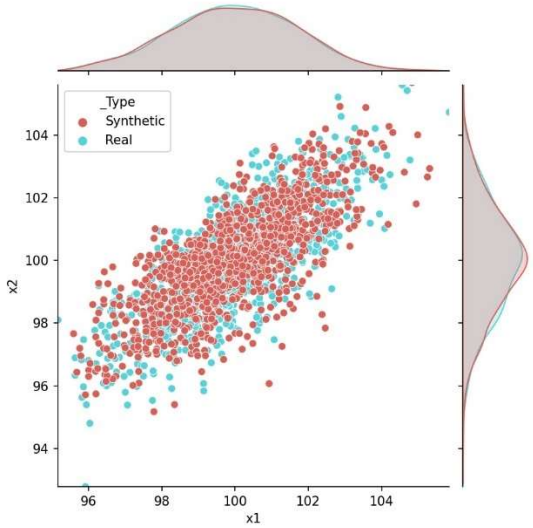


Figura 6.8 – Gráfico pareado para o OE06 (normal bivariada)

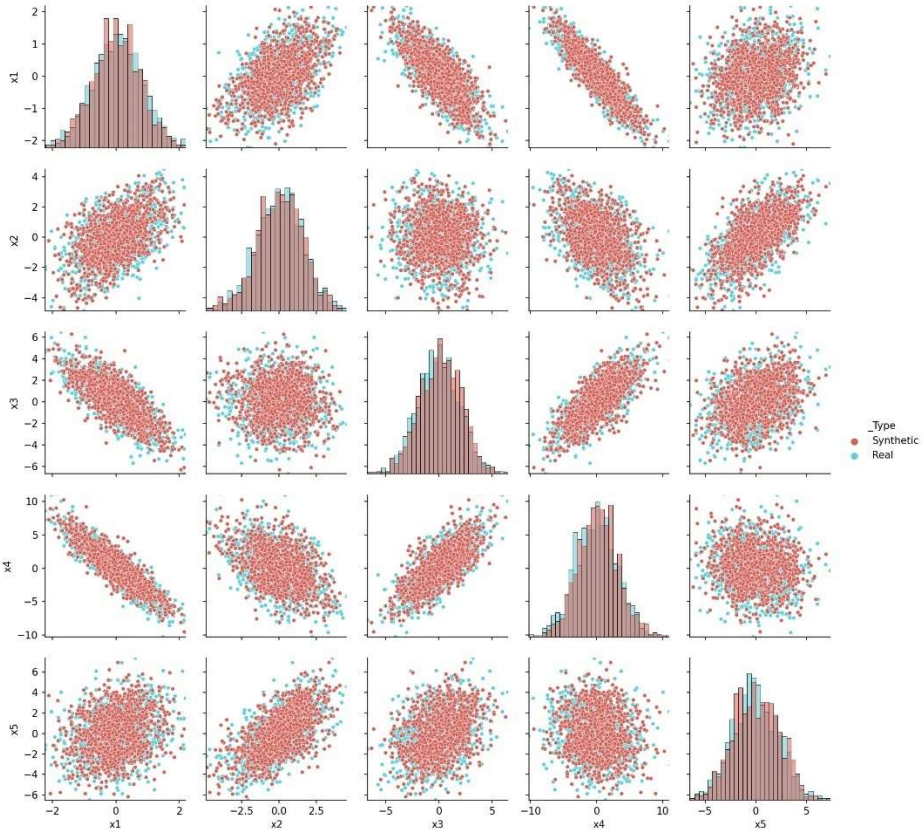


Figura 6.9 – Gráfico pareado para OE07 (normal multivariada)

Os resultados também são positivos para o caso multinomial (Figura 6.10) que conta com complexidade adicional devido aos dados serem inteiros e com um tipo de dependência específico, de que $\sum_{i=1}^{6} x_i$ seja constante.

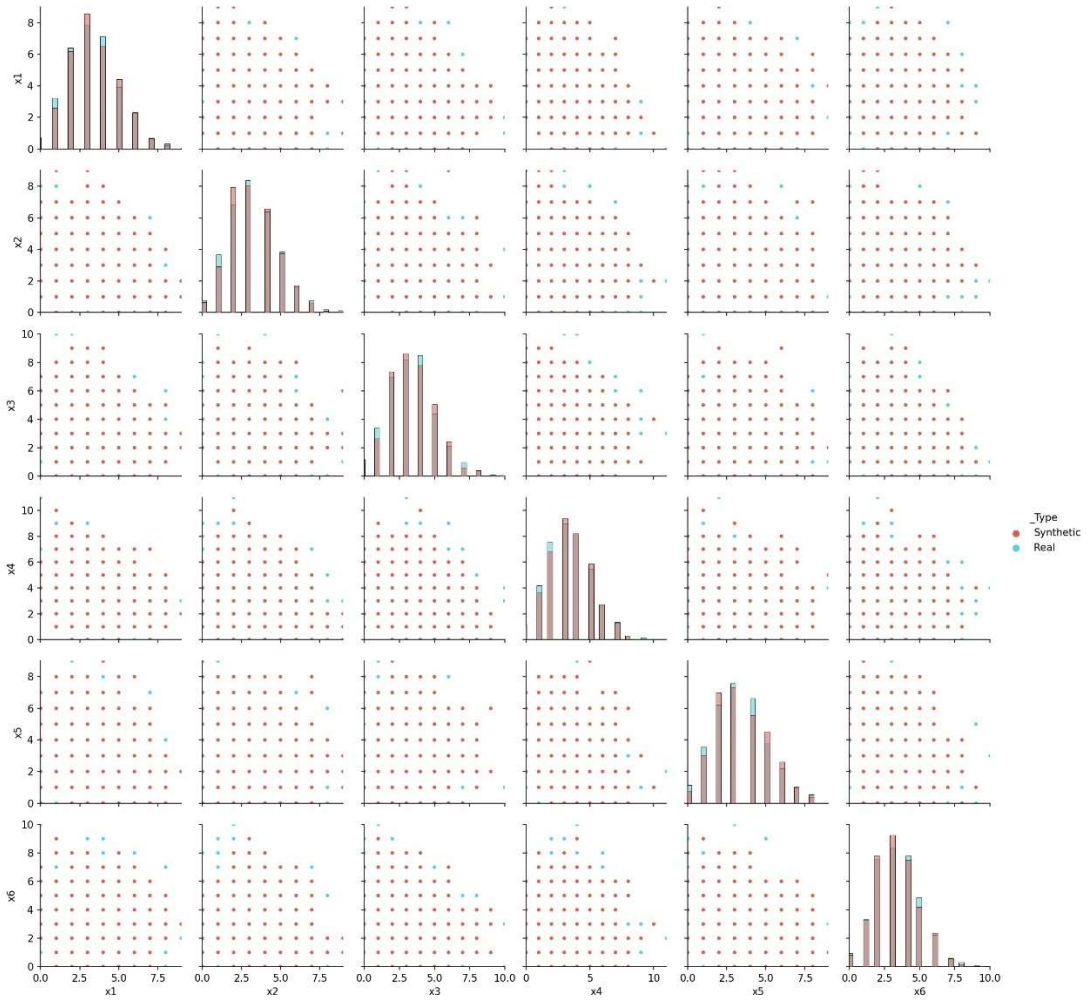
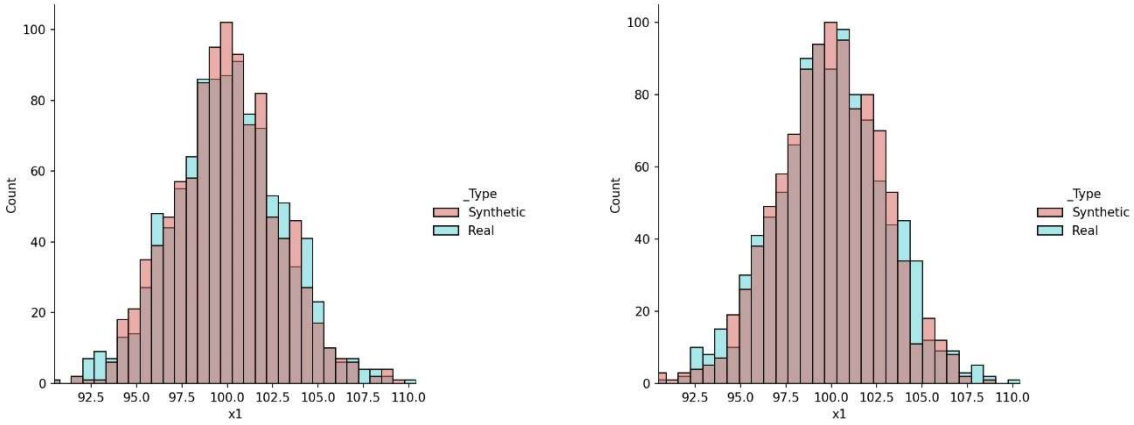


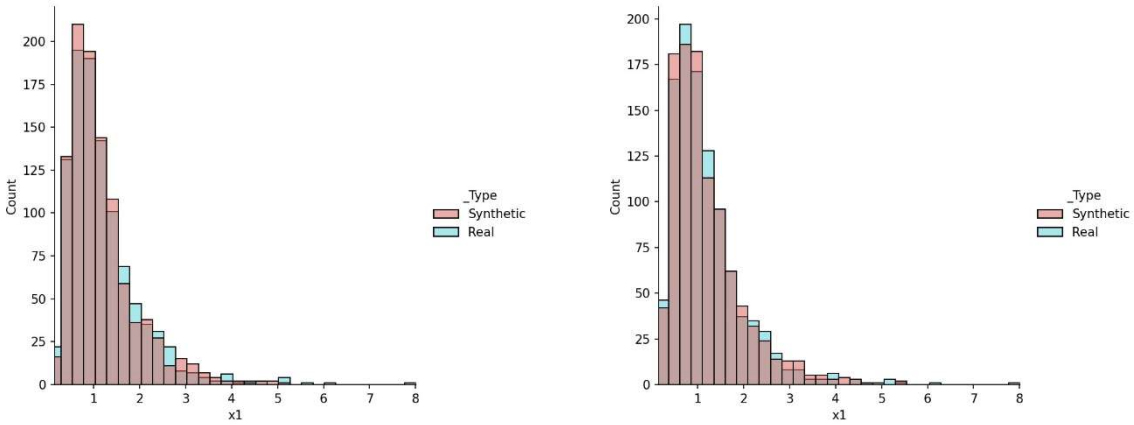
Figura 6.10 – Gráfico pareado para OE08 (multinomial)

Considerando o tamanho amostral de 1.024, as Figuras 6.11 a 6.15 permitem visualizar lado a lado, por meio de histogramas, os resultados obtidos por cada uma das estratégias de modelagem. Os resultados obtidos por MDE-GANs são semelhantes aos obtidos por DEPs nos casos normal, lognormal, exponencial e Poisson. Já para o caso bimodal, é evidente que os resultados obtidos por MDE-GANs são significativamente superiores aos obtidos por DEPs, que não foram capazes de capturar os diferentes modos da distribuição.



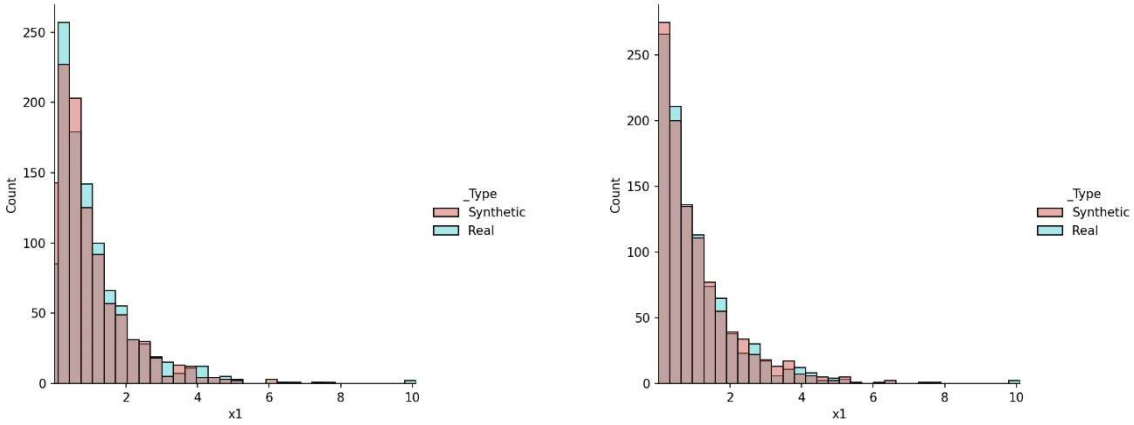
MDE-GANs **DEPs**

Figura 6.11 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso normal



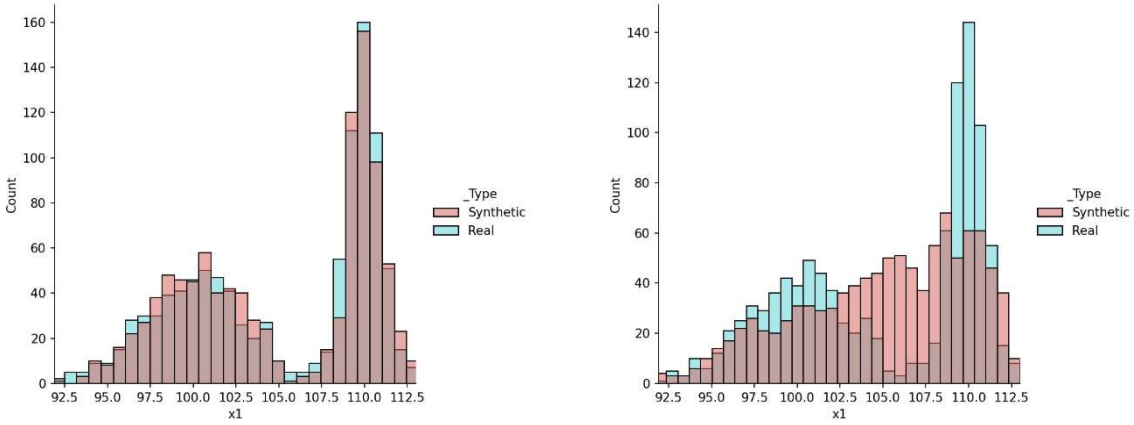
MDE-GANs **DEPs**

Figura 6.12 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso lognormal



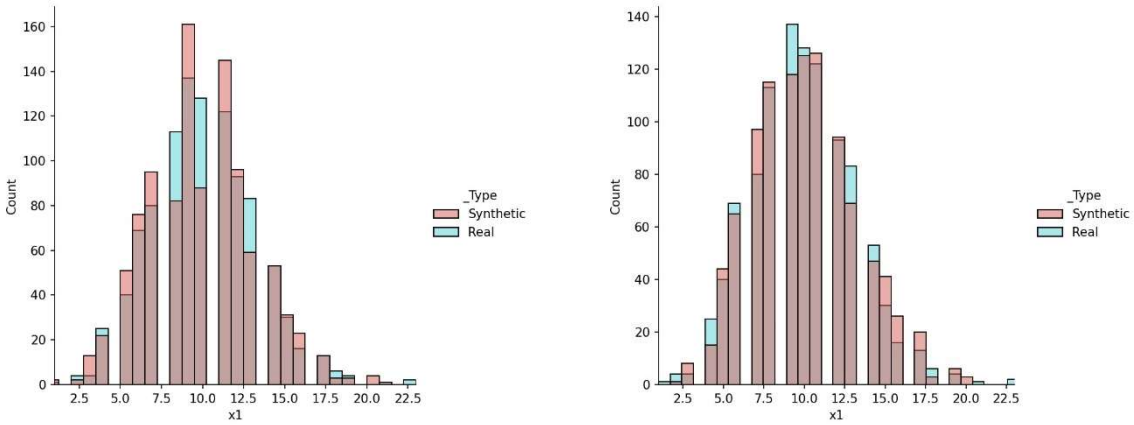
MDE-GANs **DEPs**

Figura 6.13 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso exponencial



MDE-GANs **DEPs**

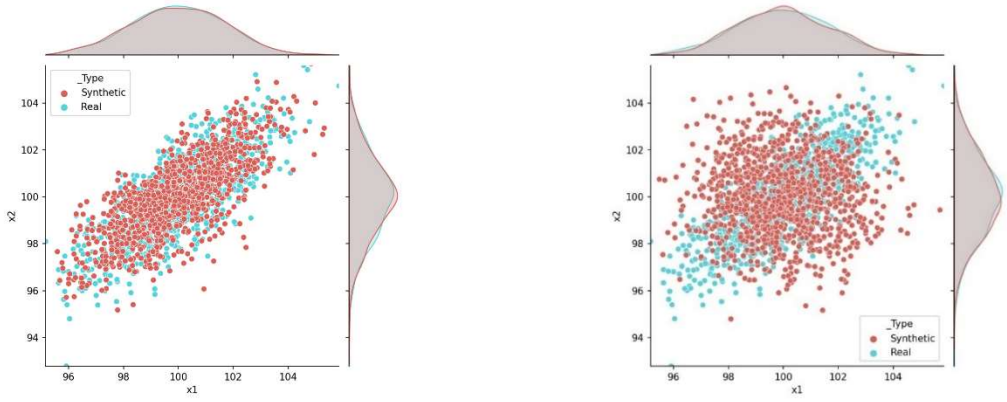
Figura 6.14 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso bimodal



MDE-GANs **DEPs**

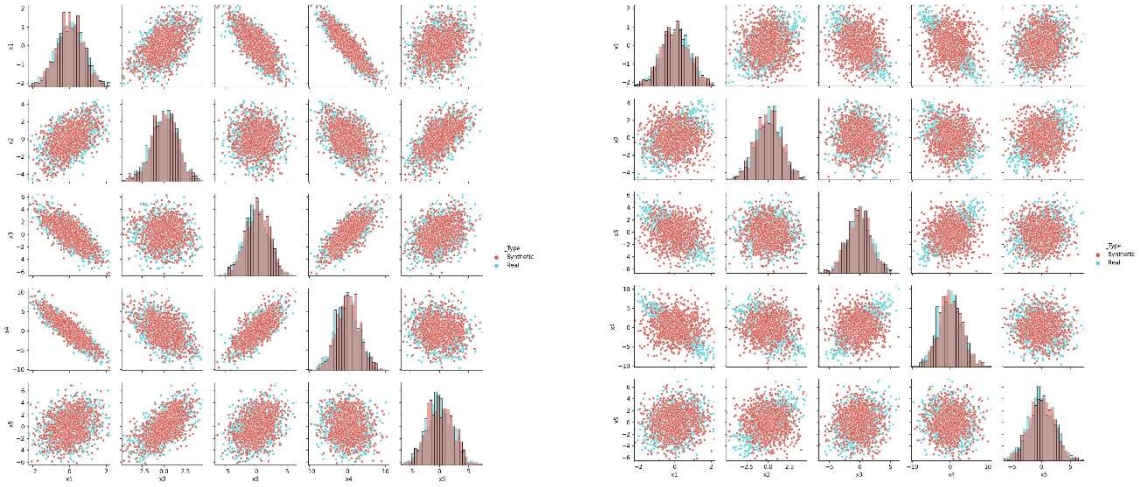
Figura 6.15 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso Poisson

De forma semelhante, as Figuras 6.16 a 6.18 comparam lado a lado os resultados obtidos para os casos teóricos paramétricos multivariados. Aqui, em todos os casos os resultados obtidos por MDE-GANs são claramente superiores aos obtidos por DEPs.



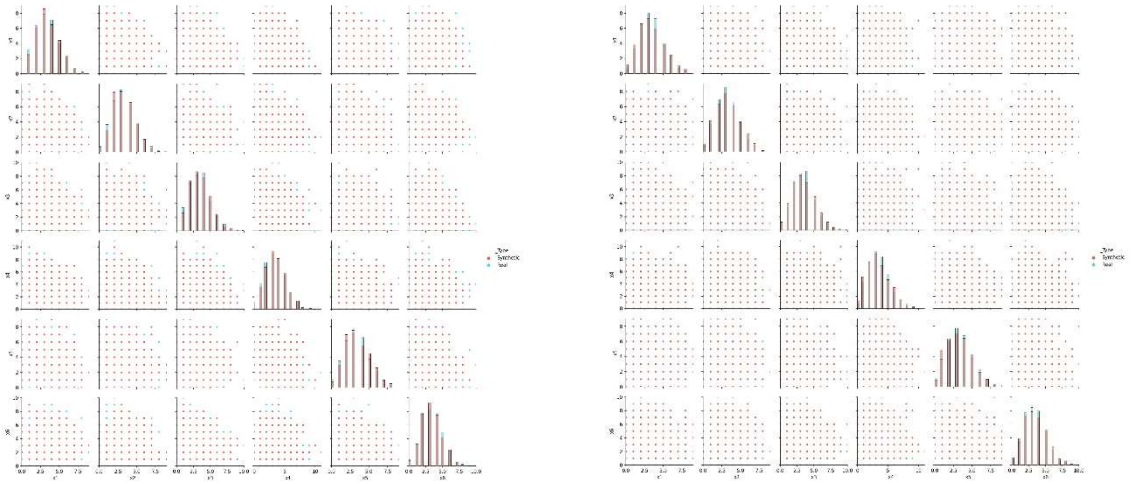
MDE-GANs **DEPs**

Figura 6.16 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso normal bivariado



MDE-GANs **DEPs**

Figura 6.17 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso normal multivariado



MDE-GANs **DEPs**

Figura 6.18 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso multinomial

Para os OE01 a OE05, as Tabelas 6.1 a 6.5 apresentam os resultados quantitativos para comparação entre GANs e DEPs: a A_{MDE} média, e seu intervalo de confiança 95% para cada tamanho amostral testado, e o percentual de réplicas que atingiram ao menos a classe de acurácia considerada como forte (% $A_{MDE} \geq$ forte). Além disso, indicam a estratégia de MDE que obteve a melhor A_{MDE}, informação obtida por meio de testes t pareados entre as réplicas de mesmo tamanho amostral e distribuição (com nível de significância de 5%).

MDEs baseados em DEPs apresentaram melhor qualidade de representação para os casos normal, lognormal, exponencial e Poisson, em linha com a hipótese H3, de que distribuições de menor complexidade seriam mais facilmente modeladas por DEPs. Porém,

ressalta-se que em todos eles, a classe de acurácia obtida por GANs foi forte ou superior. Já para o caso bimodal, MDEs baseados em GANs apresentaram resultados estatisticamente superiores, em consonância com a hipótese H2, de que GANs permitiriam melhores representações para distribuições com mais elementos de complexidade.

Já as Tabelas 6.6 a 6.8 apresentam os resultados obtidos para os objetos de estudo teóricos paramétricos multivariados.

MDEs baseados em GANs apresentaram estatisticamente melhor qualidade de representação para todos os casos, novamente em consonância com a hipótese H2. Nota-se que ao menos nos tamanhos amostrais 512 e 1.024, GANs permitiram atingir acurácias fortes em todas as réplicas. Por outro lado, DEPs praticamente não foram capazes de entregar resultados nessa classe de acurácia, mesmo com tamanhos amostrais elevados.

Tabela 6.1 – Resultados comparativos de A_{MDE} para os casos de distribuição normal

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	A_{MDE} DEP	Melhor resultado
64	100%	78,9 (74,7 - 83,1)	91,8 (88,8 - 94,8)	DEP[1]
128	100%	86,8 (82,6 - 91,1)	95,6 (93,7 - 97,6)	
256	100%	92,0 (90,6 - 93,4)	97,4 (95,9 - 98,8)	
512	100%	94,3 (93,0 - 95,6)	98,8 (98,3 - 99,3)	
1024	100%	95,4 (94,5 - 96,4)	99,3 (98,9 - 99,7)	

[1] *p-value* < 0.05

Tabela 6.2 – Resultados comparativos de A_{MDE} para os casos de distribuição lognormal

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	A_{MDE} DEP	Melhor resultado
64	100%	82,0 (78,6 - 85,4)	90,2 (87,9 - 92,5)	DEP[1]
128	100%	86,9 (84,0 - 89,9)	95,1 (92,7 - 97,5)	
256	100%	91,8 (90,1 - 93,5)	96,6 (95,4 - 97,9)	
512	100%	93,6 (92,2 - 95,0)	98,7 (98,2 - 99,2)	
1024	100%	95,2 (94,0 - 96,4)	98,9 (98,0 - 99,9)	

[1] *p-value* < 0.05

Tabela 6.3 – Resultados comparativos de A_{MDE} para os casos de distribuição exponencial

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	A_{MDE} DEP	Melhor resultado
64	100%	79,5 (76,6 - 82,4)	91,5 (88,2 - 94,8)	DEP[1]
128	100%	87,8 (85,3 - 90,3)	95,7 (93,8 - 97,5)	
256	100%	88,6 (85,8 - 91,4)	96,7 (95,1 - 98,2)	
512	100%	91,3 (88,9 - 93,7)	98,8 (97,7 - 99,8)	
1024	100%	93,9 (93,1 - 94,8)	99,2 (98,6 - 99,9)	

[1] *p-value* < 0.05

Tabela 6.4 – Resultados comparativos de A_{MDE} para os casos de distribuição bimodal

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	% A_{MDE} DEP ≥ forte	A_{MDE} DEP	Melhor resultado
64	90%	78,9 (69,2 - 88,5)	100%	70,4 (69,5 - 71,2)	GAN[1]
128	100%	85,4 (83,6 - 87,3)	100%	70,4 (69,7 - 71,1)	
256	100%	88,6 (86,7 - 90,4)	100%	70,2 (69,6 - 70,7)	
512	100%	91,7 (89,7 - 93,7)	100%	70,3 (69,9 - 70,7)	
1024	100%	93,0 (90,6 - 95,4)	100%	70,1 (69,6 - 70,7)	

[1] *p-value* < 0.05

Tabela 6.5 – Resultados comparativos de A_{MDE} para os casos de distribuição Poisson

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	% A_{MDE} DEP ≥ forte	A_{MDE} DEP	Melhor resultado
64	100%	83,5 (78,1 - 88,9)	100%	98,2 (95,8 - 100,6)	DEP[1]
128	100%	84,6 (81,1 - 88,2)	100%	97,9 (96,6 - 99,3)	
256	100%	89,8 (87,5 - 92,1)	100%	99,3 (98,6 - 99,9)	
512	100%	91,6 (89,4 - 93,8)	100%	99,3 (98,8 - 99,8)	
1024	100%	93,5 (90,8 - 96,2)	100%	99,5 (99,1 - 100,0)	

[1] *p-value* < 0.05

Tabela 6.6 – Resultados comparativos de A_{MDE} para os casos de distribuição normal bivariada

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	% A_{MDE} DEP ≥ forte	A_{MDE} DEP	Melhor resultado
64	100%	73,6 (72,0 - 75,2)	10%	60,1 (58,1 - 62,1)	GAN[1]
128	100%	81,1 (78,2 - 84,0)	0%	60,5 (59,6 - 61,3)	
256	100%	86,1 (84,4 - 87,7)	0%	61,0 (60,3 - 61,7)	
512	100%	89,5 (88,0 - 91,0)	0%	61,2 (60,7 - 61,6)	
1024	100%	91,6 (90,4 - 92,9)	0%	61,1 (60,8 - 61,4)	

[1] *p-value* < 0.05

Tabela 6.7 – Resultados comparativos de A_{MDE} para os casos de distribuição normal multivariada

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	% A_{MDE} DEP ≥ forte	A_{MDE} DEP	Melhor resultado
64	0%	49,8 (45,0 - 54,7)	0%	26,3 (24,9 - 27,6)	GAN[1]
128	0%	58,9 (55,7 - 62,1)	0%	26,1 (25,6 - 26,7)	
256	90%	68,7 (66,8 - 70,6)	0%	25,9 (25,5 - 26,3)	
512	100%	74,7 (73,1 - 76,2)	0%	26,2 (26,0 - 26,5)	
1024	100%	78,3 (76,7 - 79,9)	0%	26,4 (26,2 - 26,5)	

[1] *p-value* < 0.05

Tabela 6.8 – Resultados comparativos de A_{MDE} para os casos de distribuição normal multinomial

N treinamento	% A_{MDE} GAN ≥ forte	A_{MDE} GAN	% A_{MDE} DEP ≥ forte	A_{MDE} DEP	Melhor resultado
64	0%	41,7 (31,0 - 52,4)	0%	46,7 (45,9 - 47,4)	GAN[1]
128	50%	61,7 (57,3 - 66,2)	0%	47,7 (47,3 - 48,2)	
256	70%	66,3 (64,4 - 68,3)	0%	48,2 (47,9 - 48,5)	
512	100%	76,0 (75,1 - 76,9)	0%	48,5 (48,2 - 48,7)	
1024	100%	81,0 (79,1 - 82,8)	0%	48,6 (48,3 - 48,8)	

[1] *p-value* < 0.05

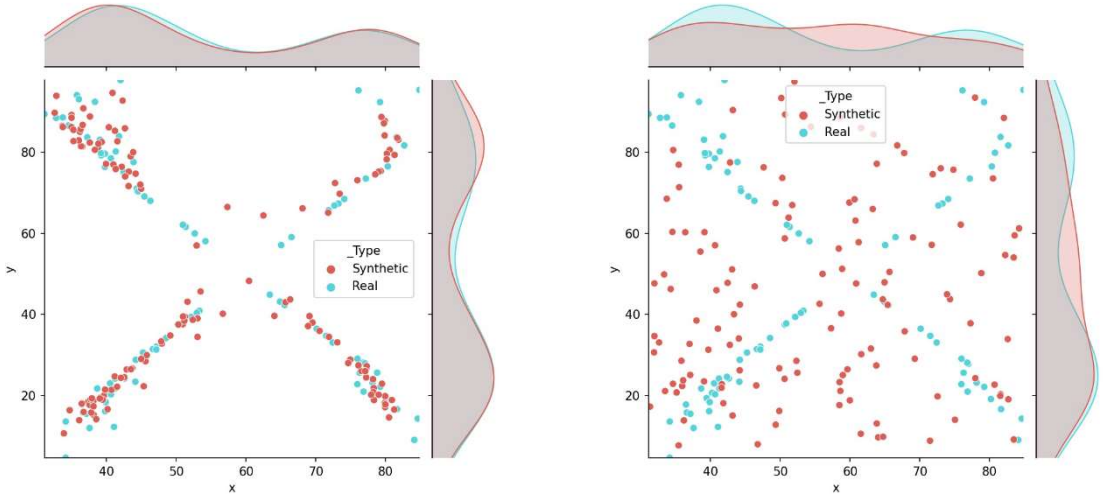
6.3. Casos teóricos não paramétricos

Como mencionado anteriormente, os casos teóricos não paramétricos selecionados a partir do estudo *Datasaurus* objetivam avaliar a capacidade de GANs em capturar dependências complexas. Para esses objetos de estudo, o treinamento foi realizado utilizando todos os dados disponíveis (142). Como pode ser observado na Tabela 6.9, os resultados proporcionados pela MDE-GANs foram todos de acurácia forte ou quase perfeita. Além disso, superaram a performance alcançada por DEPs.

Tabela 6.9 – Resultados comparativos de A_{MDE} para os casos do conjunto *Datasaurus*

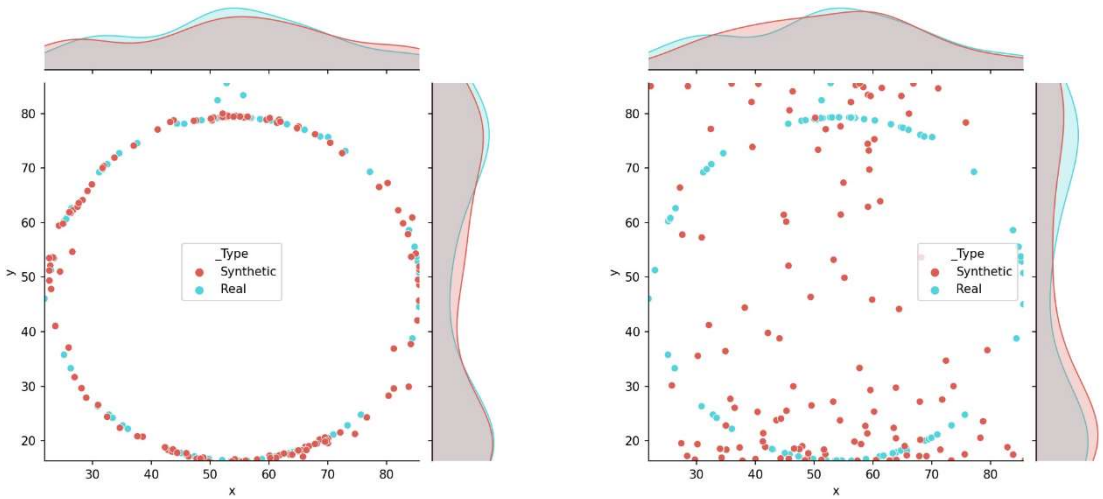
Objeto de estudo	GANs		DEPs	
	A_{MDE}	Classe de acurácia	A_{MDE}	Classe de acurácia
"xis"	90,9%	Forte	64,1%	Moderada
"círculo"	100,0%	Quase perfeita	73,3%	Forte
"estrela"	100,0%	Quase perfeita	69,8%	Moderada
"linhas paralelas"	93,7%	Quase perfeita	58,5%	Moderada

As Figuras 6.19 a 6.22 comparam lado a lado os resultados obtidos por MDE-GANs e DEPs. Novamente, por serem conjuntos bivariados, os resultados apresentados por GANs são visualmente melhores. Por fim, o Apêndice H demonstra o formato do MDE exportado para o objeto de estudo "círculo". Sua equação pode ser copiada e colada em campos do tipo *custom code* no FlexSim para teste.



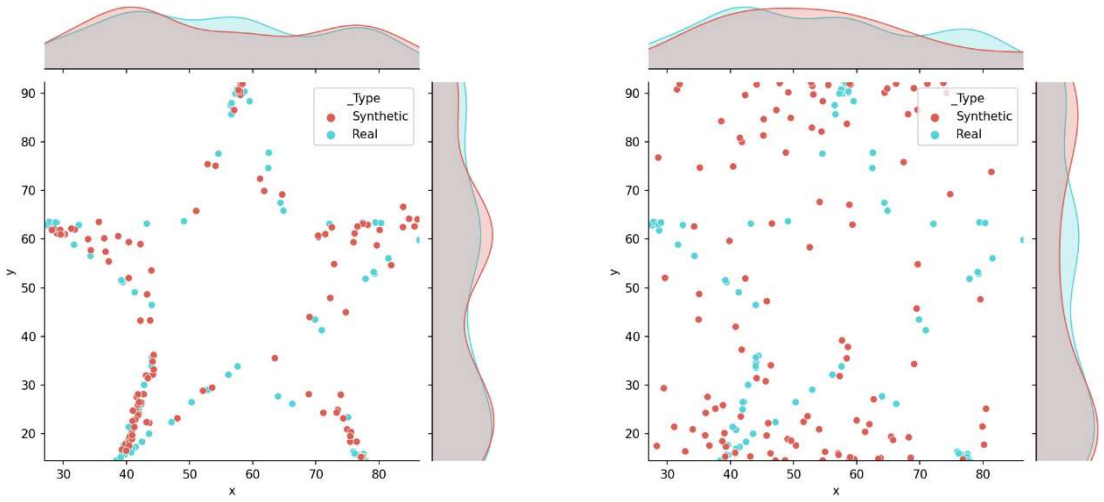
MDE-GANs DEPs

Figura 6.19 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso "xis"



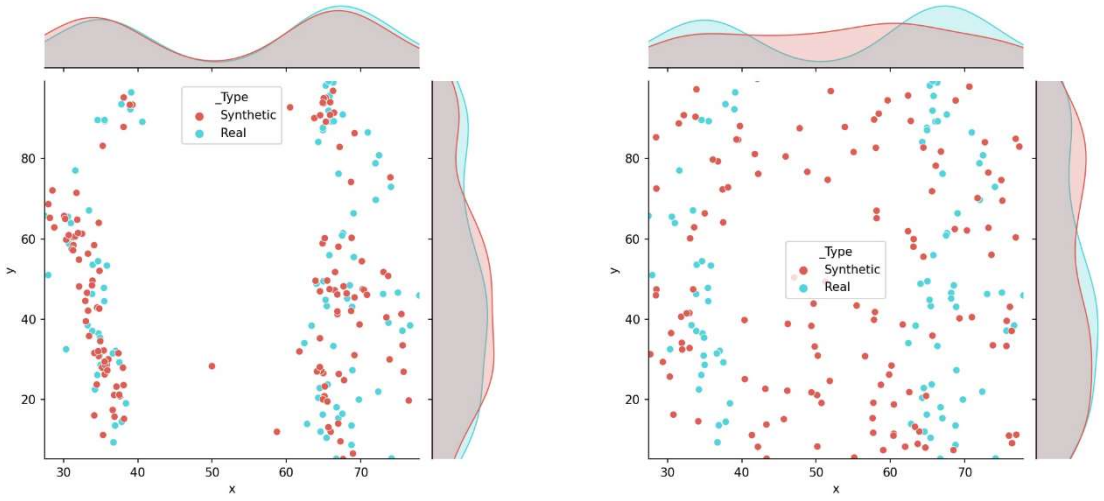
MDE-GANs **DEPs**

Figura 6.20 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso "círculo"



MDE-GANs **DEPs**

Figura 6.21 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso “estrela”



MDE-GANs **DEPs**

Figura 6.22 – Comparação gráfica entre resultados obtidos por MDE-GANs e DEPs para o caso “linhas paralelas”

6.4. Casos reais

Por fim, completando os resultados para os objetos de estudo, a Tabela 6.10 apresenta o comparativo dos resultados obtidos pela MDE-GANs e por DEPs. O treinamento foi realizado utilizando todos os dados reais disponíveis. Como pode ser observado os resultados de MDE-GANs foram superiores. Enquanto para o OE14 (pronto atendimento), o resultado de MDE-GANs foi expressivamente melhor, para o OE13 (dimensões das peças), devido à complexidade dos dados, as GANs não conseguiram atingir uma acurácia forte.

Tabela 6.10 – Resultados comparativos de A_{MDE} para os casos reais

Objeto de estudo	GANs		DEPs	
	A_{MDE}	Classe de acurácia	A_{MDE}	Classe de acurácia
OE13	58,6%	Moderada	45,5%	Moderada
OE14	84,6%	Quase perfeita	23,3%	Fraca

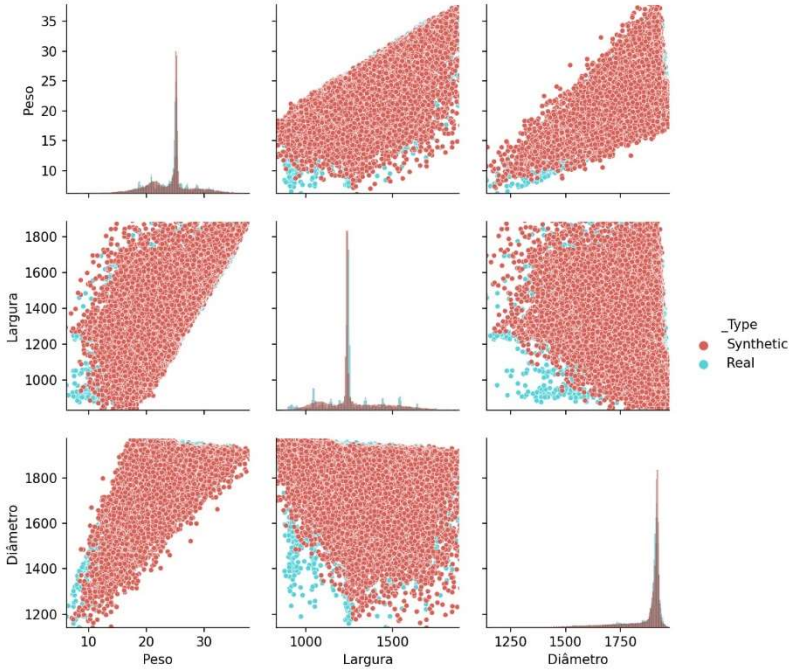
Por meio das análises gráficas apresentadas nas Figuras 6.23 e 6.24, nota-se que a MDE-GANs permitiu capturar de forma mais fidedigna as dependências entre as variáveis.

6.5. Impacto do tamanho amostral

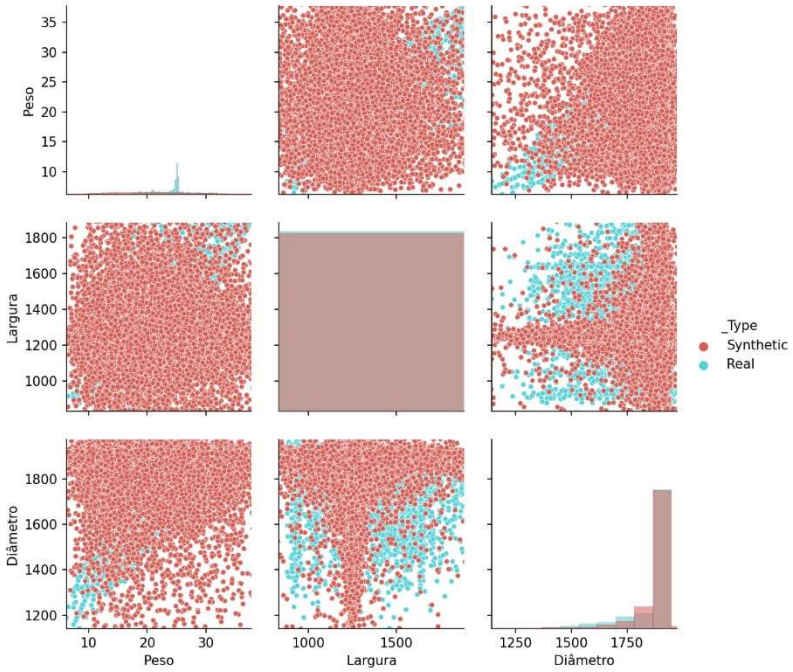
Os resultados apresentados nas seções anteriores já demonstram o impacto do tamanho amostral na qualidade de representação obtida em MDE-GANs, o que sustenta a hipótese H4.

Adicionalmente, para avaliar de forma gráfica o impacto do tamanho amostral nos MDEs obtidos por meio de GANs, as Figuras 6.25 a 6.29 apresentam comparativos dos histogramas para os casos teóricos univariados, com tamanhos amostrais de treinamento (N) de 64 a 1.024, novamente sobrepondo as amostras de dados reais utilizados no treinamento (verde) e amostras de dados sintéticos (vermelho). Quanto mais sobrepostos estiverem os dados reais e sintéticos, melhor a capacidade da GAN em gerar dados próximos da realidade.

Os resultados indicam que, com exceção de alguns casos, os dados sintéticos se aproximam da amostra verdadeira de forma razoável, independentemente do tamanho do conjunto utilizado para treinamento. Isso indica que mesmo com tamanhos amostrais pequenos, GANs podem manter sua capacidade de replicar estritamente os dados fornecidos para treinamento. E, consequentemente, traz evidências de que GANs dependem de uma quantidade maior de dados verdadeiros de modo que tenham informação estatística suficiente para não replicar apenas o comportamento da amostra, mas, sim, o da própria distribuição real.

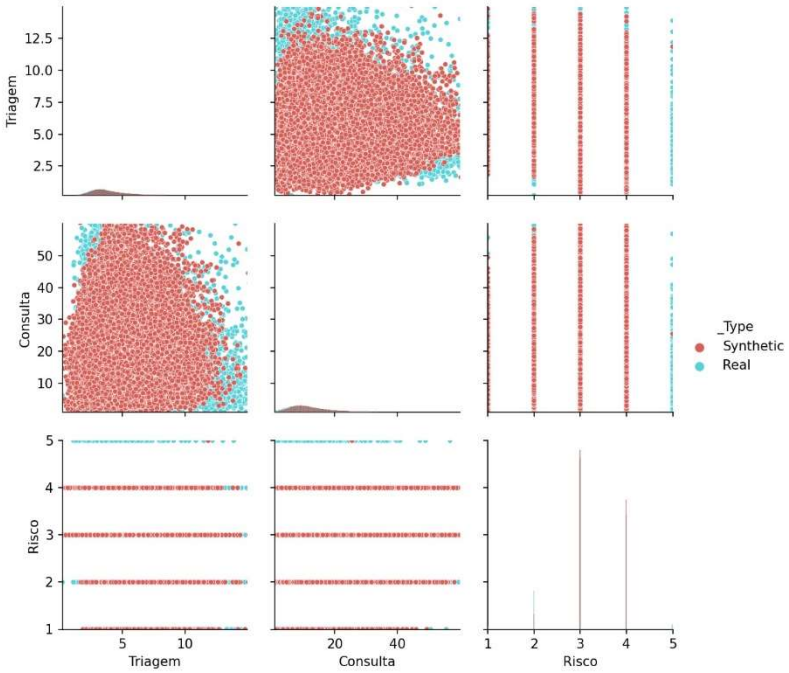


(a)

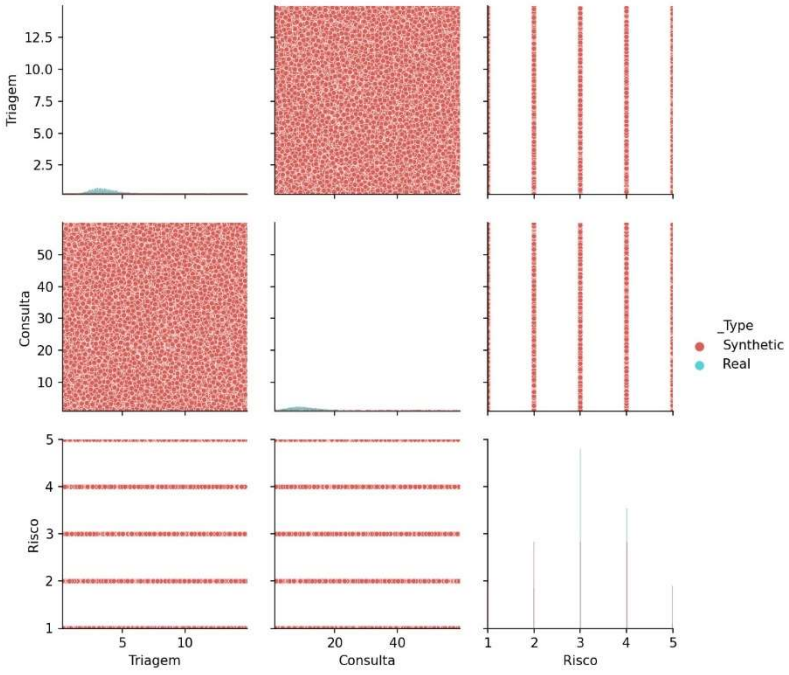


(b)

Figura 6.23 – Comparativo de resultados para OE13 (dimensões de peças). (a) MDE-GANs. (b) DEPs.

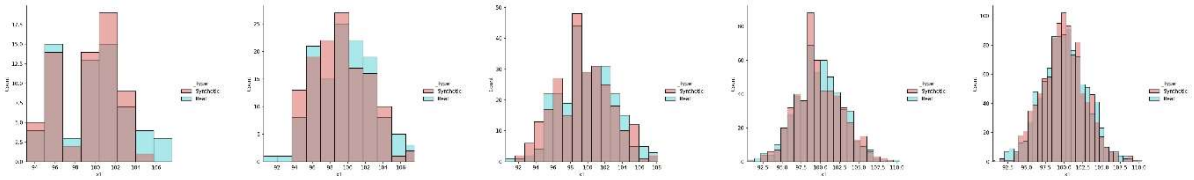


(a)



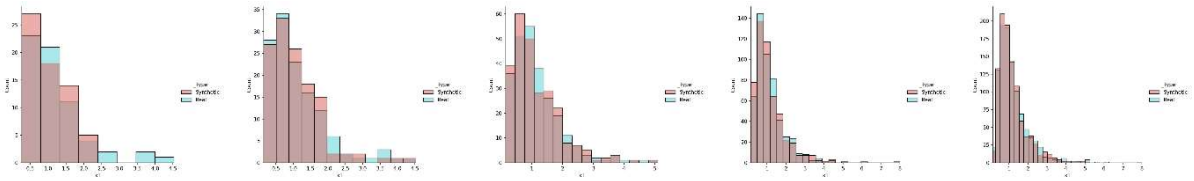
(b)

Figura 6.24 – Comparativo de resultados para OE14 (pronto atendimento). (a) MDE-GANs. (b) DEPs.



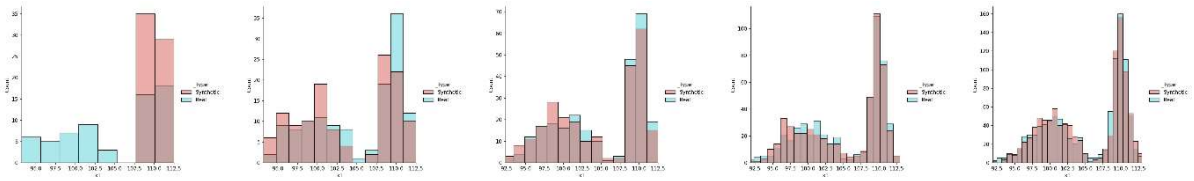
N = 64 **N = 128** **N = 256** **N = 512** **N = 1.024**

Figura 6.25 – Histogramas para o caso normal, com diferentes tamanhos amostrais de treinamento



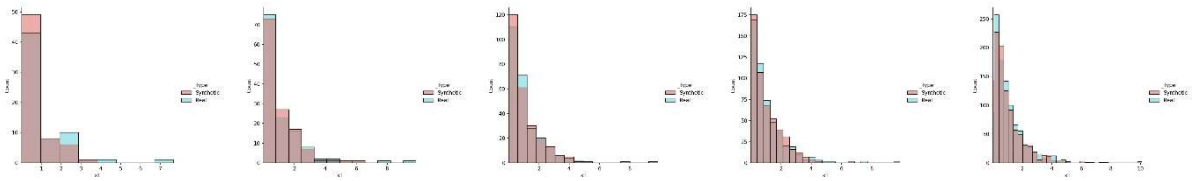
N = 64 **N = 128** **N = 256** **N = 512** **N = 1.024**

Figura 6.26 – Histogramas para o caso lognormal, com diferentes tamanhos amostrais de treinamento



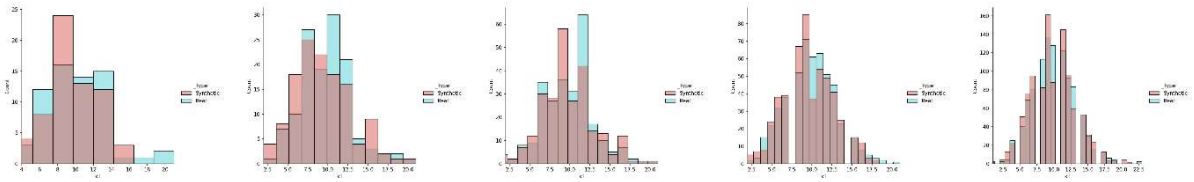
N = 64 **N = 128** **N = 256** **N = 512** **N = 1.024**

Figura 6.27 – Histogramas para o caso bimodal, com diferentes tamanhos amostrais de treinamento



N = 64 **N = 128** **N = 256** **N = 512** **N = 1.024**

Figura 6.28 – Histogramas para o caso exponencial, com diferentes tamanhos amostrais de treinamento



N = 64 **N = 128** **N = 256** **N = 512** **N = 1.024**

Figura 6.29 – Histogramas para o caso Poisson, com diferentes tamanhos amostrais de treinamento

As Figuras 6.30 a 6.32 apresentam o mesmo comparativo para os casos multivariados utilizando gráficos pareados. É possível notar que, mesmo com tamanhos amostrais pequenos, as GANs conseguiram capturar dependências entre as variáveis. Entretanto, tamanhos amostrais iguais ou maiores que 128 apresentaram melhores resultados, especialmente para o caso multinomial, mais uma vez trazendo indícios de que amostras maiores podem ser necessárias para o aprendizado de distribuições multivariadas.

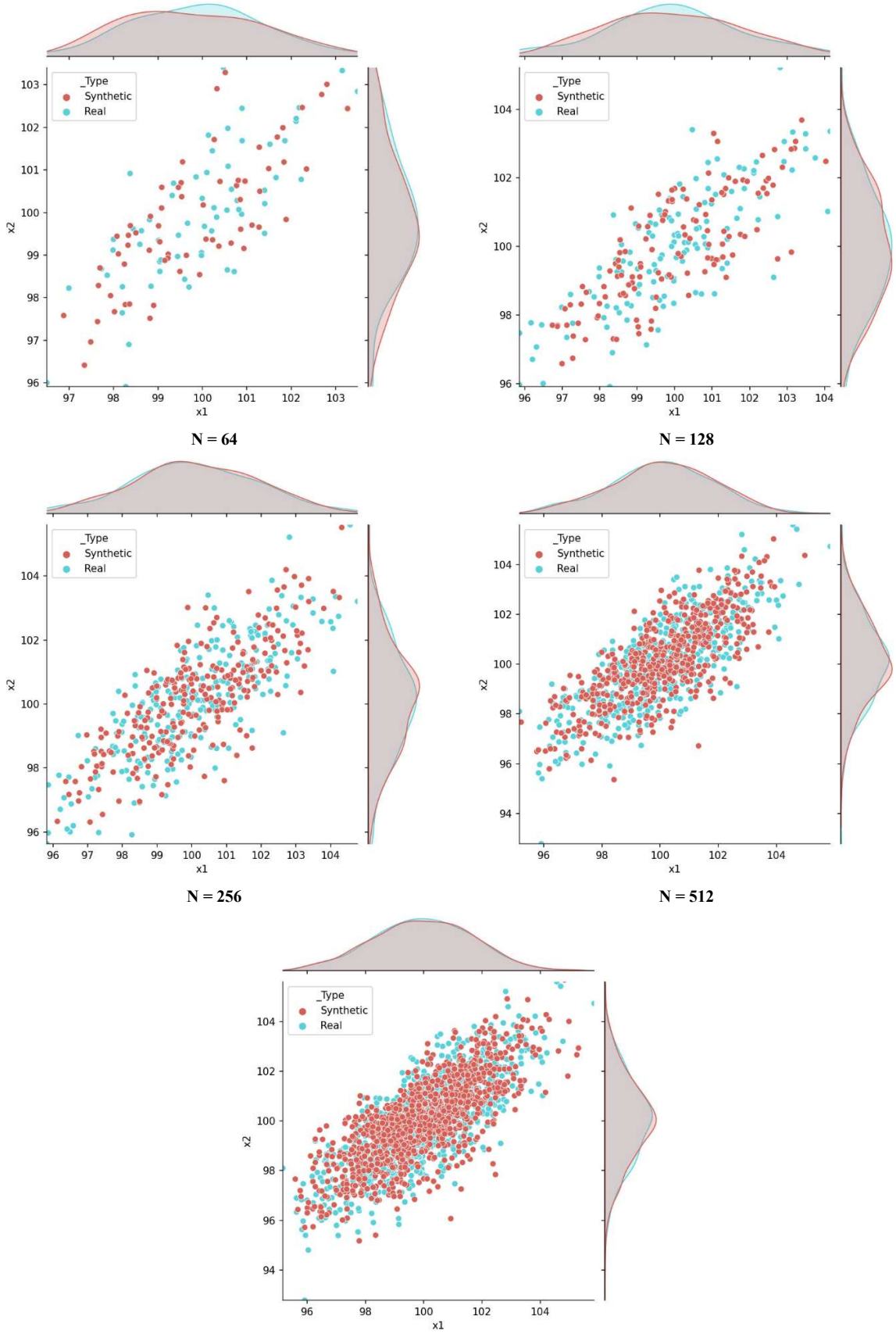


Figura 6.30 – Gráficos pareados para o caso multivariado de duas variáveis, com diferentes tamanhos amostrais de treinamento

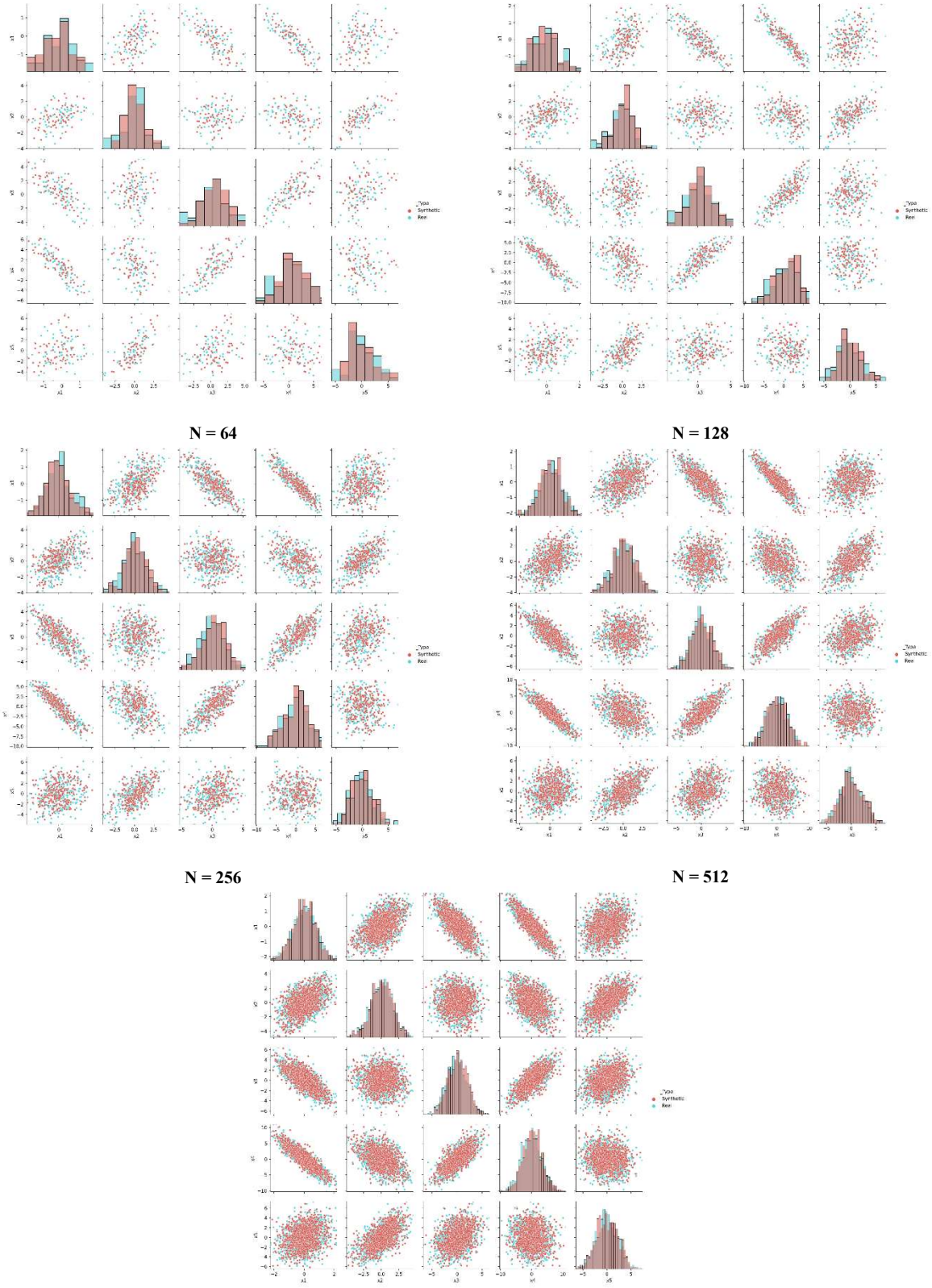


Figura 6.31 – Gráficos pareados para o caso multivariado de cinco variáveis, com diferentes tamanhos amostrais de treinamento

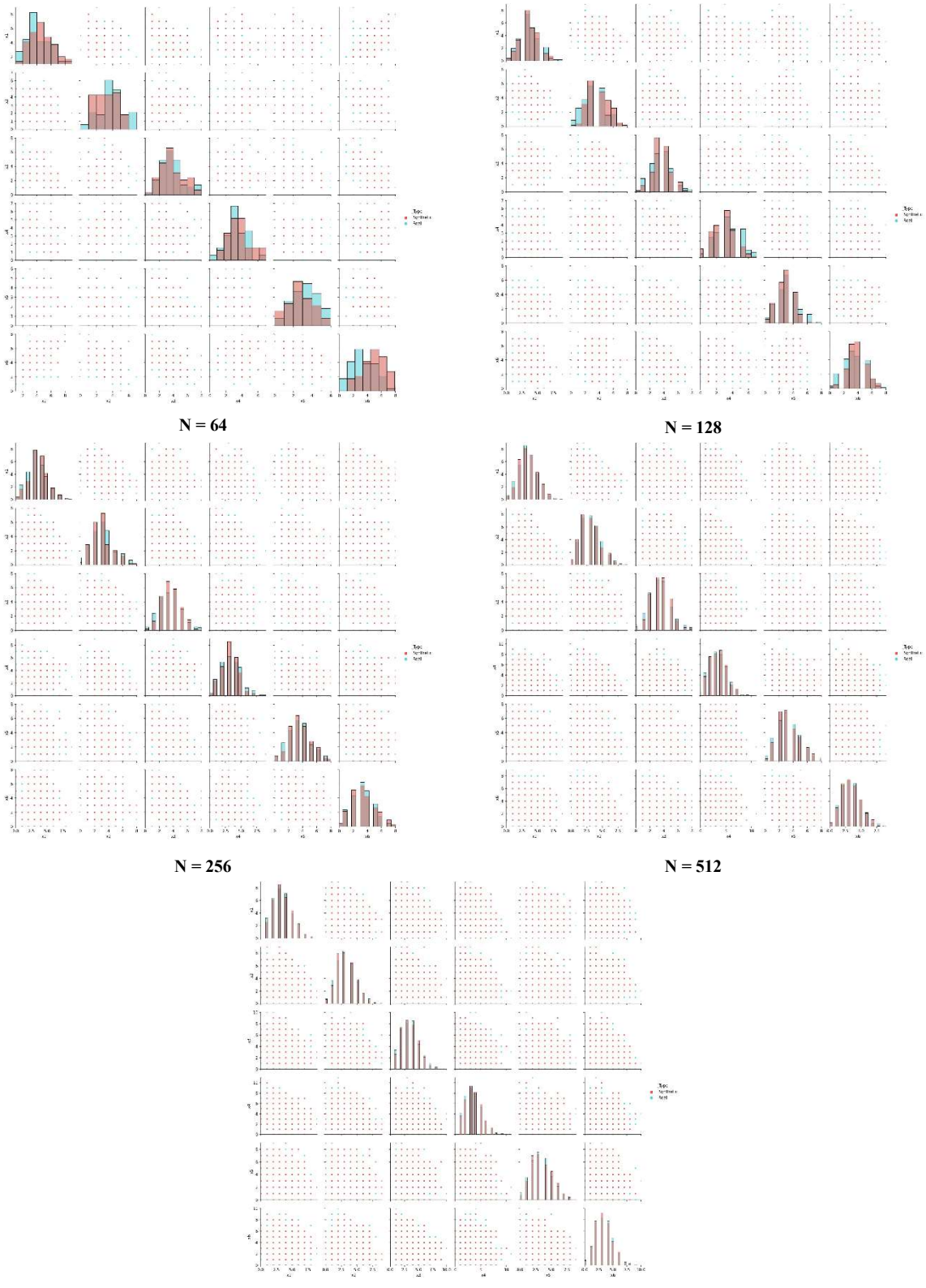


Figura 6.32 – Gráficos pareados para o caso multinomial, com diferentes tamanhos amostrais de treinamento

6.6. Considerações finais

Analisar os resultados a partir de todas as perspectivas demonstradas oferece evidências detalhadas para teste das hipóteses formuladas neste trabalho. Adicionalmente, nesse momento, serão realizados alguns testes estatísticos para obtenção de evidências resumidas que auxiliem a responder as questões da pesquisa de forma objetiva.

Em relação à Questão 01, é importante avaliar se GANs permitiram atingir sistematicamente níveis de acurácia maiores do que 64% (forte). Considerando os resultados agregados para todos os objetos de estudo teóricos paramétricos, a A_{MDE} atingiu o valor médio de 82,4% e desvio-padrão de 13,2%. Por meio de um teste t para uma amostra, com hipótese alternativa de que A_{MDE} é maior do que 64%, é possível confirmar estatisticamente que as acurácias alcançadas são, em média e considerando os objetos de estudo avaliados, fortes (*p-value* < 0,001), como demonstra a Figura 6.33.

Já em relação à Questão 02, é necessário comparar os resultados obtidos entre GANs e DEPs. Novamente avaliando os resultados de forma agregada e geral, é possível calcular a diferença média pareada entre as acurácias obtidas por GANs e DEPs. Esse valor atingiu 8,5%, em favor das GANs, com desvio-padrão de 19,3%. Realizando o teste t pareado, é possível confirmar a hipótese alternativa de que os resultados das GANs são, em média e considerando os objetos de estudo avaliados, superiores (*p-value* < 0,001), como demonstra a Figura 6.34. Entretanto, ficam claros três grupos de resultados (três modos da distribuição de diferenças pareadas). É possível explorar esse comportamento por meio de árvores de decisão, que buscam a melhor forma de segmentar os dados em relação a características preditoras.

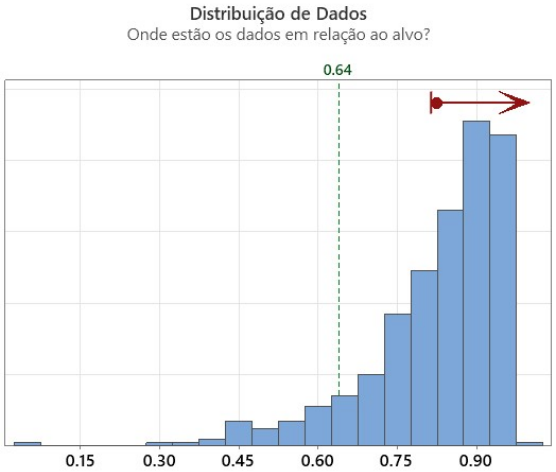


Figura 6.33 – Distribuição geral dos resultados em relação ao alvo

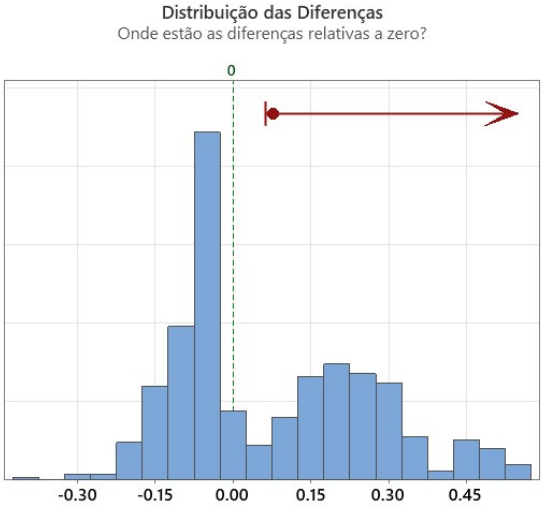


Figura 6.34 – Distribuição geral das diferenças pareadas

Considerando como variáveis potencialmente explicativas das diferenças pareadas a distribuição do objeto de estudo e o tamanho amostral utilizado, é possível criar uma árvore de decisão que explica 87% da variação observada nos resultados, o que está apresentado na Figura 6.35. Notam-se três grupos de nós terminais. O primeiro (nó terminal 1) inclui todos os resultados para os objetos de estudo exponencial, lognormal, normal e Poisson, sendo o único que apresenta, em média, uma diferença em favor das DEPs. Trata-se dos casos já discutidos anteriormente, de distribuições IID, mais facilmente modeladas por DEPs. O segundo grupo (nós terminais 2 e 3) inclui todos os testes para as distribuições bimodal, multinomial e normal bivariada. Tais casos apresentam uma menor diferença pareada, pois ainda não representam o máximo de dificuldade para as DEPs. Já o último grupo (nós terminais 4 e 5) abrange apenas os testes para a distribuição normal multivariada. Esse foi o caso de maior dificuldade de modelagem para DEPs, por apresentar correlações de diferentes intensidades e sentidos.

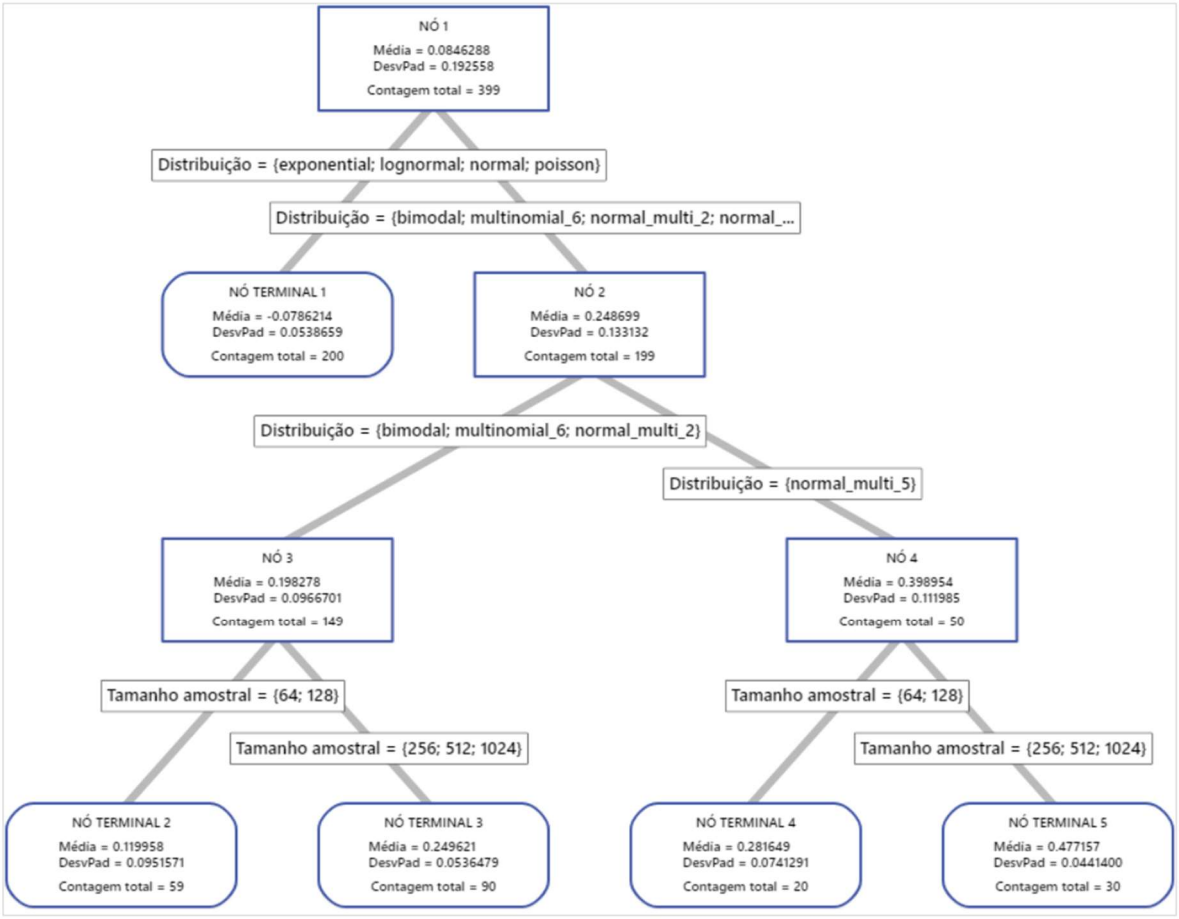


Figura 6.35 – Árvore de decisão para variáveis explicativas da diferença de resultados entre GANs e DEPs

Ainda em relação à árvore de decisão, nota-se que o segundo e terceiro grupos são influenciados significativamente pelo tamanho amostral, sendo valores iguais ou superiores a 256 determinantes para resultados em média mais consistentes para GANs. Esse tópico está relacionado à terceira e última questão, que se dedica ao impacto do tamanho amostral nos resultados. Isso é confirmado por meio de uma Análise de Variância (ANOVA) para um fator, buscando identificar diferenças na acurácia de MDEs baseados em GANs em relação aos tamanhos amostrais testados (*p-value* < 0,05), como demonstra o gráfico de intervalos para comparação das médias (Figura 6.36). O gráfico indica, ainda, que quanto maior o tamanho amostral, menor a variabilidade dos resultados obtidos por MDE-GANs.

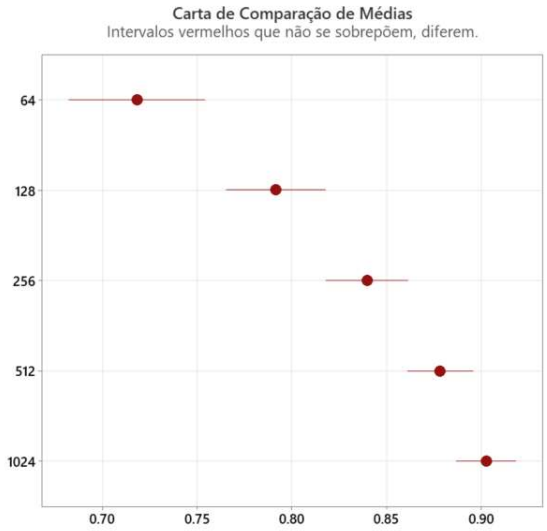


Figura 6.36 – Carta de comparação de médias das acurácias obtidas por MDE-GANs em relação aos tamanhos amostrais

7. CONCLUSÕES

No presente Capítulo, serão estabelecidas as conclusões da tese em face das questões de pesquisa e hipóteses formuladas no Capítulo 1, discutindo-se as considerações finais e sugestões para trabalhos futuros. Adicionalmente, será avaliado o atendimento aos objetivos definidos.

7.1. Questões de pesquisa e hipóteses

Três questões de pesquisa foram formuladas para este trabalho, as quais foram desdobradas em quatro hipóteses.

A Questão 01 diz respeito à capacidade das GANs em modelar dados de entrada para projetos de SED. Em primeiro lugar, destaca-se que foi possível implementar o método proposto, chegando até a etapa de tradução do Gerador para a linguagem utilização pelo *software* de simulação FlexSim. Também foi obtido sucesso em importar o MDE gerado, em formato Excel, para utilização em modelos de simulação.

Quanto à qualidade de representação obtida, para todos os objetos de estudo teóricos paramétricos, os quais apresentavam diferentes características, foi possível chegar a MDEs baseados em GANs com níveis de acurácia considerados como fortes ou quase perfeitos. A hipótese H1, de que GANs podem gerar MDEs com forte qualidade de representação foi, portanto, validada. Avaliações gráficas por meio de histogramas e gráficos Q-Q confirmaram os resultados, demonstrando a capacidade de MDEs baseados em GANs em capturar elementos de complexidade como bimodalidade e interdependência. Para dados univariados e bivariados,

estes resultados foram obtidos mesmo com o menor tamanho amostral testado para treinamento, de 64. Já para os casos multivariados, com cinco e seis variáveis, acurácias fortes só foram obtidas a partir de tamanhos amostrais de 256 ou 512. Trata-se de um resultado importante, demonstrando o potencial de utilização de GANs para modelagem de dados de entrada em projetos de SED. Os resultados para os casos teóricos não paramétricos reforçam as conclusões acima, tendo apresentado qualidades de representação fortes ou quase perfeitas. Já para os casos reais, os resultados foram mistos, tendo a MDE-GANs apresentado quantitativamente uma acurácia moderada para o OE13. Porém, as avaliações gráficas demonstraram bons resultados.

A Questão 02 se refere à competitividade da proposta MDE-GANs em relação à modelagem tradicional realizada por meio de DEPs. Confirmando as hipóteses H2 e H3, os resultados comparativos apontaram melhores qualidades de representação para DEPs em casos de menor complexidade, os dados IID. Ressalta-se, entretanto, que o desempenho obtido por MDE-GANs, apesar de inferior, não se distanciou das classes de acurácia forte e quase perfeita. Já para dados não IID, em especial os casos com a presença de interdependências, GANs apresentaram qualidades de representação significativamente superiores às DEPs. Nestes objetos de estudo, DEPs praticamente não foram capazes de entregar resultados com níveis de acurácia fortes, mesmo com tamanhos amostrais elevados. Esses resultados servem de resposta à Questão 02, demonstrando a competitividade da proposta MDE-GANs, capaz de atingir resultados com níveis de acurácia equivalentes ou superiores em relação aos obtidos por DEPs. Nesse sentido, o tema pode impactar positivamente projetos de SED, oferecendo uma alternativa competitiva e capaz de modelar dados com diferentes características sem demandar a definição de parâmetros por parte do modelador.

A confirmação das hipóteses H2 e H3 também permite traçar a conclusão de que, pelo menos nesta proposta, GANs e DEPs possuem áreas específicas onde se destacam. Nesse sentido, pode-se aproveitar a complementariedade das estratégias, escolhendo, caso a caso, e de preferência de forma automática, a melhor modalidade de modelagem. Ou, ainda, utilizar métodos *ensemble* para combiná-las.

Por fim, a Questão 03 levanta a discussão sobre a relação entre a quantidade de dados disponibilizados para o treinamento e qualidade de representação obtida por GANs. Nesse tópico, os resultados confirmam a hipótese H4, indicando que tamanhos amostrais maiores tendem a favorecer melhores acurácias dos MDEs. Em especial, esse comportamento foi notado para os casos multivariados, com acurácias fortes sendo observadas apenas para tamanhos

amostrais de 256 ou 512. Isso significa que, para capturar interdependências em conjuntos de maior dimensionalidade, GANs requerem maior quantidade de dados para treinamento.

7.2. Objetivos específicos

Três objetivos específicos foram estabelecidos para esta tese. O primeiro diz respeito à proposta do método MDE-GANs e foi cumprido por meio das discussões e definições realizadas no Capítulo 3.

Já o segundo objetivo demanda o desenvolvimento de programas para realização das etapas definidas para o método MDE-GANs. Isso foi cumprido, conforme descrito no Capítulo 4. Além disso, ressalta-se que todos os algoritmos implementados se encontram registrados nos apêndices desta tese, contribuindo para que próximos estudos possam utilizá-los como referência, comparação ou ponto de partida. Essa contribuição ganha relevância no contexto em que a vasta maioria das aplicações de GANs são focadas no processamento de imagens, vídeos e sons, como discutido na seção 1.2.2. Dessa forma, os códigos implementados podem servir de exemplo para outras aplicações com dados tabulares.

Por fim, o terceiro objetivo específico estabelece a necessidade de avaliação dos resultados do método MDE-GANs em casos teóricos e reais. Tal objetivo foi alcançado com a definição e avaliação dos objetos de estudo descritos nos Capítulos 5 e 6. Além disso, o trabalho contribui ao demonstrar a execução da proposta de MDE-GANs, o que foi registrado no Apêndice G. Ressalta-se que a escolha dos casos foi realizada seguindo-se os critérios para tipos de processos e problemas que devem ser inclusos, ou seja, abrangendo dados homogêneos ou heterogêneos e independentes ou dependentes.

Os casos teóricos paramétricos foram utilizados mais diretamente na resposta às questões de pesquisa e na avaliação das hipóteses formuladas. Já os casos não paramétricos e os reais serviram principalmente como demonstração da utilização do método para distribuições com características de complexidade singulares.

7.3. Considerações finais

Atualmente, o autor desta tese atua de forma regular em projetos de SED, assumindo ou acompanhando atividades de modelagem de dados de entrada. No dia a dia da área, em vários momentos são observados os desafios citados neste trabalho. Em especial, destaca-se a dificuldade de construção de MDEs que incluam dependências, o que demanda sempre um esforço de definição de uma arquitetura para o modelo de simulação de forma a contornar a

limitação das DEPs. Além disso, não é raro de se observar que, mesmo após horas ou dias de análise e tratamento dos dados coletados, as ferramentas de modelagem baseadas em DEPs não retornem distribuições que representem bem os dados reais. Na vivência do modelador, trata-se de um momento que demanda tempo e esforço para encontrar uma solução.

Em linha com os resultados obtidos para as hipóteses formuladas, o presente trabalho pode representar uma alternativa para facilitar a condução da etapa de modelagem de dados de entrada, permitindo contornar pelo próprio método de modelagem as dificuldades acima mencionadas. Com os rápidos avanços em Aprendizado de Máquina, espera-se que surjam cada vez mais oportunidades de sinergias entre as áreas.

A proposta tem o potencial de reduzir o tempo de modelagem de dados de entrada, por considerar uma ferramenta que pode ser utilizada para diferentes tipos de dados e complexidades. Além disso, tem o potencial de reduzir o próprio tempo de modelagem computacional nos projetos em que os dados apresentem dependências. Isso, pois, na prática de modelagem, modeladores dispendem tempo para criar lógicas e *inputs* condicionais com o objetivo de forçar a ocorrência de dependências nos sorteios e driblar a falta de suporte dos pacotes de simulação a distribuições multivariadas. Com a utilização de MDE-GANs, essa prática pode ser reduzida ou eliminada.

Dentre os desafios enfrentados para a elaboração desta tese, ressalta-se o tempo para pesquisa e entendimento sobre GANs e sua forma de implementação. Como afirmado anteriormente, a literatura apresenta grande quantidade de estudos focados em imagens, que utilizam arquiteturas de RNAs diferentes da necessária para a finalidade da tese. Foram diversas versões, adaptações e protótipos até a obtenção de resultados estáveis. Contribuíram neste processo de aprendizado a abrangência e a quantidade de conteúdos disponibilizados de forma generosa pela comunidade da área, especialmente em plataformas como *Stack Overflow*, *GitHub e Medium*, além do repositório de artigos *arXiv*, gerido pela Universidade Cornell. Para o autor da tese, que não tinha conhecimentos em Python antes do trabalho, foi uma oportunidade de grande desenvolvimento.

Nesse sentido, destaca-se ainda que a proposta formulada inclui uma série de ferramentas com papeis importantes e que agregam valor à solução implementada, sendo possível citar o teste C2ST, as ferramentas para tratamento dos dados reais e a Análise de Componentes Principais como direcionador para definição de hiperparâmetros das RNAs.

7.4. Sugestões para trabalhos futuro

A partir deste trabalho, foram identificadas as seguintes oportunidades para trabalhos futuros:

- Como discutido na seção 1.5.3.2, o presente trabalho não abrange a modelagem de dados de entrada que apresentem autocorrelação. Essa decisão foi tomada, pois outros tipos de arquiteturas de RNAs são mais adequados para esse tipo de dados, tais como *Long Short-Term Memory* (LSTM) e *Recurrent Neural Network* (RNN), capazes de lidar com séries temporais e sequências de dados. Estudos futuros podem explorar essas arquiteturas e tipos de dados;
- Além disso, desde o trabalho seminal sobre GANs, diversas variações foram propostas, tais como *Conditional* GANs (cGANs), para geração de dados condicionais; *Information Maximizing* GANs (infoGANs) e Wasserstein GANs (WGAN) com melhorias no processo de aprendizagem; e variações para obtenção de imagens com alta resolução, tais como BigGAN, ProGAN e StyleGAN (ALDAUSARI *et al.*, 2023). Novos estudos podem avaliar a aplicabilidade e as vantagem dessas variações, além de comparar os resultados obtidos com os resultados da GAN original, utilizada no presente trabalho;
- Outras técnicas generativas podem ser encontradas na literatura, tais como *Variational Autoenconders* (VAEs) e métodos baseados em fluxos (XUE *et al.*, 2022). Seus resultados podem ser avaliados e comparados com os obtidos nesta tese;
- Recomenda-se, também, a utilização de métodos *ensemble*, combinando GANs e DEPs;
- O presente estudo adota hiperparâmetros citados como bons pela literatura ou que apresentaram resultados positivos em testes exploratórios conduzidos durante o desenvolvimento da proposta. Novas abordagens podem ser propostas para otimização dos hiperparâmetros, visando melhorar os resultados obtidos por MDE-GANs;
- Em relação à hipótese H1, para um dos casos reais, a proposta de MDE-GANs não atingiu acurácias fortes. Recomenda-se o estudo de dados com características semelhantes, para compreensão e melhoria do método proposto;
- Em relação às hipóteses H2 e H3, notou-se que, pelo menos nesta proposta, GANs e DEPs possuem áreas específicas onde se destacam. Nesse sentido, novos estudos podem explorar a sinergia entre DEPs e GANs em uma proposta de utilização conjunta, com escolha automática pelo MDE que forneça a melhor qualidade de representação;

- Por fim, em relação à hipótese H4, foi possível confirmar que o tamanho amostral para treinamento das GANs impacta a qualidade de representação obtida para os MDEs. Entretanto, outros estudos devem ser conduzidos para explorar com maior detalhe este fenômeno, contribuindo para a definição de regras para cálculo do tamanho amostral necessário para cada caso.

Apêndice A – Módulo *gandata*

Este apêndice reúne os códigos desenvolvidos para ler e preparar os dados a serem modelados.

```
""" Read and prepare real data
This module includes:
* Data (class): comprises methods and attributes for real data

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022

"""

# Imports
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA

# Data class
class Data():

    """ Comprises methods and attributes for real data
    The following steps are carried out to prepare real data for GAN training:
    1. Dropping missing data
    2. Scaling data
    3. Calculating the number of data dimensions (PCA)
    """

    def __init__(self):
        """
        Attributes
        ____________
        """
        self._dataset_name = "dataset"
        # Dataset versions
        self.original_data = None
        self.real_data = None
        self.scaled_data = None
        # Data characteristics
        self._sample_size = None
        self._ncomponents = None
        self._nvariables = None
        # Columns
        self._original_columns = None
        self._columns = None
        self._int_columns = None
        # Scaler
        self._scaler = None

    def read_excel(self, excel_file_name, sample_size = None):
        """ Read excel files

        Parameters
        ____________
        excel_file_name: string
            the name of the excel file, including path and file extension
        sample_size: integer
            (optional) the number of rows to randomly sample from the excel file

        Returns
        ____________
        self.original_data: pandas.DataFrame
            original data (sampled, if applicable)
        self._dataset_name: string
            excel file name
```

```
63.
64.         """
65.         df = pd.read_excel(excel_file_name)
66.         if sample_size is not None and sample_size < df.shape[0]:
67.             self.original_data = df.sample(sample_size)
68.         else:
69.             self.original_data = df.sample(frac=1)
70.         self._dataset_name = excel_file_name.rsplit("\\",1)[1].rsplit(".", 1)[0]
71.
72.     def prepare_data(self, scaler_estimator):
73.         """ Prepare data for GAN training
74.
75.         Parameters
76.         _____________
77.         scaler_estimator: sklearn.preprocessing scaler instance
78.             MinMaxScaler() or RobustScaler()
79.
80.         Returns
81.         _____________
82.         # Dataset versions
83.         self.real_data: pandas.DataFrame
84.             real data w/o missing observations and w/ dummy variables for categorical
    data
85.         self.scaled_data: pandas.DataFrame
86.             scaled data
87.         # Data characteristics
88.         self._sample_size: integer
89.             number of observations
90.         self._ncomponents: integer
91.             number of data dimensions (PCA)
92.         self._nvariables: integer
93.             number of variables
94.         # Columns
95.         self._original_columns: list
96.             list of original column names
97.         self._columns: list
98.             list of column names after the creation of dummy variables
99.         self._int_columns: list
100.              list of column names with integer data (for rounding after GAN training)
101.          # Scaler
102.          self._scaler: sklearn.preprocessing scaler instance
103.              fitted scaler
104.
105.          """
106.          # Get copy of original data
107.          real_data = self.original_data.copy()
108.
109.          # Force columns to be string
110.          real_data.rename(columns=lambda x: str(x), inplace=True)
111.
112.          # Get original column names
113.          original_column_names = real_data.columns
114.
115.          # Report and drop missing data
116.          n_dropped_observations = np.sum(real_data.isna().sum(axis=1) > 0)
117.          if n_dropped_observations > 0:
118.              print('\033[93m' + "Missing data by variable (%):")
119.              print(round(100*real_data.isna().sum(axis=0)/real_data.shape[0],1))
120.              print("")
121.              print("Total dropped observations:")
122.              print(np.sum(real_data.isna().sum(axis=1) > 0))
123.              print("" + '\033[0m')
124.          real_data = real_data.dropna()
125.
126.          # Identify int columns (this identification will enable rounding them after
    generating)
127.          int_columns = real_data.select_dtypes(include=['integer']).columns
128.
```

```
129.            # Scale data
130.            scaled_data = pd.DataFrame(scaler_estimator.fit_transform(real_data.copy()),
    columns = real_data.columns)
131.
132.            # PCA
133.            pca = PCA(n_components=real_data.shape[1])
134.            pca.fit(scaled_data)
135.            ncomponents = 0
136.            for n, ratio in enumerate(pca.explained_variance_ratio_):
137.                if sum(pca.explained_variance_ratio_[0:(n+1)]) >= 0.99:
138.                    ncomponents = n + 1
139.                    break
140.
141.            # Save attributes
142.            # Dataset versions
143.            self.real_data = real_data
144.            self.scaled_data = scaled_data
145.            # Data characteristics
146.            self._sample_size = real_data.shape[0]
147.            self._nvariables = real_data.shape[1]
148.            self._ncomponents = ncomponents
149.            # Columns
150.            self._original_columns = original_column_names
151.            self._columns = real_data.columns
152.            self._int_columns = int_columns
153.            # Scaler
154.            self._scaler = scaler_estimator
155.
```

Apêndice B – Módulo *gan*

Este apêndice reúne os códigos desenvolvidos para configurar a GAN, treiná-la e avaliar os resultados obtidos pelo gerador.

```
""" Build, train and evaluate GANs
This module includes:
* GAN (class): A class w/ methods to build, train and evaluate GANs. As well as methods to generate synthetic data.

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022

"""

# Imports
import numpy as np
import pandas as pd
import os
from pathlib import Path
import time
import tensorflow as tf
from tensorflow.keras.layers import Dense, LeakyReLU, Dropout, Input, Concatenate
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras import optimizers, losses, metrics
from tensorflow.keras.utils import plot_model
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from collections import namedtuple
import xlsxwriter
from c2st import C2ST_knn
from utilities import model_to_C2ST_accuracy, C2ST_to_model_accuracy, define_efficient_batch
import ganplot

class GAN():

    """The GAN class comprises methods to:
    1. Define and train GANs
    2. Evaluate Generators (calling C2ST)
    3. Sample new synthetic data from Generators

    """

    def __init__(self, data):

        """ Build GAN

        Parameters
        ____________
        data: Data()
            Instance of the data class with already prepared data

        Returns
        ____________
        self.data: Data
            Reference to instance of the data class with already prepared data
        self._latent_space_size
            Latent space size for the generator input (as a function of the number of real data dimensions)
        self.GAN: tensorflow.keras.models.Model
            Artificial neural network for the entire GAN
        self.generator: tensorflow.keras.models.Model
            Artificial neural network for the generator
```

```
57.         self.discriminator: tensorflow.keras.models.Model
58.             Artificial neural network for the discriminator
59.         self._output_path
60.             Create a folder for saving training results
61.         self.standard_noise
62.             A standard sample of the latent space
63.         self.epoch
64.             Initialize the epoch counter
65.
66.         """
67.
68.         # Assign data as attribute
69.         self.data = data
70.
71.         # Create folder to save results
72.         self._output_path = f'{Path().absolute()}\\{time.strftime("%Y%m%d-
    %H%M%S")}_{self.data._dataset_name}'
73.         os.mkdir(self._output_path)
74.
75.         # Define latent space size
76.         # Rule recommended by this study: 3 * number of principal components that explain
    >99% of data variation
77.         self._latent_space_size = 3*data._ncomponents
78.
79.         # Define generator and discriminator networks
80.         # Rule recommended by this study:
81.         # density = 8 * number of principal components that explain >99% of data
    variation
82.         self.generator = self._define_generator(density=8*data._ncomponents)
83.         self.discriminator = self._define_discriminator(density=8*data._ncomponents)
84.
85.         # Define GAN
86.         self.GAN = self._define_GAN(self.generator, self.discriminator)
87.
88.         # Define standard noise
89.         # This noise will be used to generate synthetic data and, with them, evaluate the
    generator during training
90.         # This is important to avoid the multiple testing problem if
91.         self.standard_noise = self.generate_noise(sample_size=data._sample_size)
92.
93.     def _build_base_ann(
94.         self,
95.         name,
96.         input_shape,
97.         density,
98.         hidden_layers,
99.         output_shape,
100.          output_activation,
101.          ):
102.
103.          """ Build base networks for both generator and discriminator
104.          The activation function of hidden layers is the Leaky ReLU function (alpha =
    0.2).
105.          Other hyperparameters are defined according to the network type.
106.
107.          Parameters
108.          ____________
109.          name: string
110.              network name
111.          input_shape: integer
112.              network input size. For the generator, the latent space size. For the
    discriminator, the number of data variables.
113.          density: integer
114.              number of neurons in each hidden layer
115.          hidden_layers: integer
116.              number of hidden layers
117.          output_shape: integer
```

```
118.              network output size. For the generator, the number of data variables. For
    the discriminator, 1.
119.          output_activation: string
120.              name of the activation function of the output layers. For the
    discriminator, 'sigmoid'. For the generator, it depends on the scaler function domain
    chosen in the stage of real data preparation.
121.
122.          Returns
123.          _____________
124.          model: tensorflow.keras.models.Model
125.              Base artificial neural network for the generator
126.
127.          """
128.
129.          # Define Leaky ReLU
130.          lrelu = lambda x: tf.keras.activations.relu(x, alpha=0.2)
131.
132.          # Input layer
133.          inputs = Input(shape=(input_shape,), name = ("input_latent" if name ==
    "generator" else "input_data"))
134.
135.          # Create base network
136.          base = Dense(density, activation=lrelu, name = "input_layer")(inputs)
137.          for hidden_layer in range(hidden_layers):
138.              x = Dense(density, activation=lrelu, name =
    f"hidden_layer_{hidden_layer+1}")(base)
139.              base = x
140.          outputs = Dense(output_shape, activation=output_activation, name =
    "output_layer")(base)
141.          model = Model(inputs=inputs, outputs=outputs, name=name)
142.
143.          return model
144.
145.      def _define_discriminator(
146.          self,
147.          density,
148.          hidden_layers = 3,
149.          alpha = 0.0002,
150.          beta = 0.5,
151.          beta2 = 0.999,
152.          epsilon = 1e-8
153.          ):
154.
155.          """ Define the discriminator network
156.          Standard values for hyperparameters are defined according to good values
    recommended by the literature.
157.
158.          Parameters
159.          _____________
160.          density: integer
161.              number of neurons in each hidden layer
162.          hidden_layers: integer
163.              (optional) the number of hidden layers
164.          alpha, beta, beta2, epsilon: numeric
165.              (optional) hyperparameter for the Adam Optimizer
166.
167.          Returns
168.          _____________
169.          discriminator: tensorflow.keras.models.Model
170.              Artificial neural network for the discriminator
171.          Saves a visual representation of the model
172.          """
173.
174.          # Build discriminator
175.          discriminator =
    self._build_base_ann(name='discriminator',input_shape=self.data._nvariables,
    density=density,
176.                                       hidden_layers=hidden_layers, output_shape=1,
```

```
177.                                    output_activation="sigmoid")
178.
179.          # Define discriminator optimizer
180.          optimizer = optimizers.Adam(learning_rate=alpha, beta_1=beta, beta_2=beta2,
    amsgrad=False, epsilon=epsilon)
181.
182.          # Compile discriminator
183.          discriminator.compile(loss="binary_crossentropy", optimizer=optimizer)
184.          discriminator.trainable = False
185.
186.          # Print discriminator summary and save plotted model
187.          print("")
188.          print("="*100)
189.          print(discriminator.summary())
190.          plot_model(discriminator, to_file=self._output_path + "\\" +
    '_discriminator.png', show_shapes=True, show_layer_names=True, expand_nested=True)
191.
192.          return discriminator
193.
194.      def _define_generator(
195.          self,
196.          density,
197.          hidden_layers = 3
198.          ):
199.
200.          """ Define the generator network
201.          Standard values for hyperparameters are defined according to good values
    recommended by the literature.
202.
203.          Parameters
204.          ____________
205.          density: integer
206.              number of neurons in each hidden layer
207.          hidden_layers: integer
208.              (optional) the number of hidden layers
209.
210.          Returns
211.          ____________
212.          generator: tensorflow.keras.models.Model
213.              Artificial neural network for the generator
214.          Saves a visual representation of the model
215.
216.          """
217.
218.          # Define output layer activation function according to chosen scaler
219.          if isinstance(self.data._scaler, MinMaxScaler):
220.              output_activation = "sigmoid"
221.          elif isinstance(self.data._scaler, RobustScaler):
222.              output_activation = "linear"
223.          else:
224.              output_activation = "linear"
225.              print("Error: for data scaling, you must choose either MinMaxScaler or
    RobustScaler.")
226.
227.          # Build generator
228.          generator =
    self._build_base_ann(name='generator',input_shape=self._latent_space_size,
    density=density,
229.                                      hidden_layers=hidden_layers,
    output_shape=self.data._nvariables,
230.                                      output_activation=output_activation)
231.
232.          # Print generator summary and save plotted model
233.          print("")
234.          print("="*100)
235.          print(generator.summary())
236.          plot_model(generator, to_file=self._output_path + "\\" + '_generator.png',
    show_shapes=True, show_layer_names=True, expand_nested=True)
```

```
237.
238.          return generator
239.
240.      def _define_GAN(
241.          self,
242.          generator,
243.          discriminator,
244.          alpha = 0.0002,
245.          beta = 0.5,
246.          beta2 = 0.999,
247.          epsilon = 1e-8
248.          ):
249.
250.          """ Compile the GAN network
251.          Standard values for hyperparameters are defined according to good values
    recommended by the literature.
252.
253.          Parameters
254.          _____________
255.          generator: tensorflow.keras.models.Model
256.              the already defined gerator model
257.          discriminator: tensorflow.keras.models.Model
258.              the already defined discriminator model
259.          alpha, beta, beta2, epsilon: numeric
260.              (optional) hyperparameter for the Adam Optimizer
261.
262.          Returns
263.          _____________
264.          GAN: tensorflow.keras.models.Model
265.              Artificial neural network for the entire GAN
266.          Saves a visual representation of the model and the model itself (.h5)
267.          """
268.
269.          # Define GAN/generator optimizer
270.          optimizer = optimizers.Adam(learning_rate=alpha, beta_1=beta, beta_2=beta2,
    amsgrad=False, epsilon = epsilon)
271.
272.          # Combine and compile cGAN
273.          gen_noise = generator.input
274.          gen_output = generator.output
275.          gan_output = discriminator(gen_output)
276.          GAN = Model(gen_noise, gan_output, name = "GAN")
277.          GAN.compile(loss='binary_crossentropy', optimizer=optimizer)
278.
279.          # Save plotted model
280.          plot_model(GAN, to_file=self._output_path + "\\" + '_GAN.png',
    show_shapes=True, show_layer_names=True, expand_nested=False)
281.          GAN.save(f'{self._output_path}\\model.h5')
282.
283.          return GAN
284.
285.      def generate_noise(self, sample_size):
286.
287.          """ Generate noise (sample latent variables) to input to the generator
288.          Noise matrix shape = [sample size, latent space size]
289.
290.          Parameters
291.          _____________
292.          sample_size: integer
293.              number of observations that will be created with the noise
294.
295.          Returns
296.          _____________
297.          noise: tensorflow.python.framework.ops.EagerTensor
298.              A tensor of the specified shape filled with random normal values.
299.          """
300.
301.          # Generate noise (generator input)
```

```
302.          noise = tf.random.normal(shape=[sample_size, self._latent_space_size])
303.
304.          return noise
305.
306.      def generate_synthetic_sample(self, sample_size, scaled = False, noise = None,
    post_processing = False):
307.
308.          """ Create synthetic samples using the generator
309.          For training purposes, synthetic data should be:
310.              * Scaled, since the real data is also scaled for training
311.              * Created from non-standard noise, to favor sample diversity accross epochs
312.              * Not post processed, to ensure that the loss function is differentiable
313.          For evaluation purposes, synthetic data should be:
314.              * Scaled, since the knn classifier requires scaled data
315.              * Created from standard noise, to avoid the multiple testing problem. This
    way, synthetic samples accross epochs will present a positive correlation.
316.              * Post processed, because, even if normalized, the real data maintains its
    discrete characteristics, if the variables are integers. This way, we allow a fairer
    evaluation.
317.          For application purposes, synthetic data should be:
318.              * Unscaled, presenting the same ranges observed in the real process
319.              * Created from non-standard noise, to favor sample diversity
320.              * Post processed, to ensure that integer variables are actually integer
321.
322.          Parameters
323.          _____________
324.          sample_size: integer
325.              number of observations that will be created
326.          scaled: boolean
327.              if the new observations should be scaled according the normalization
    performed for the real data during data preparation
328.          noise: tensorflow.python.framework.ops.EagerTensor
329.              (optional) the latent space sample to be used as input to the generator. If
    none, the function gets a new sample.
330.          post_processing: boolean
331.              if the new observations should be rounded (integer variables)
332.
333.          Returns
334.          _____________
335.          synthetic_sample: pandas.DataFrame
336.              DataFrame with new synthetic data.
337.          """
338.
339.          # Generator network
340.          generator = self.generator
341.          # Real data scaler
342.          scaler = self.data._scaler
343.          # List of integer variables
344.          int_cols = self.data._int_columns
345.
346.          # Generate noise if necessary
347.          if (noise is None):
348.              noise = tf.random.normal(shape=[sample_size, self._latent_space_size])
349.          else:
350.              noise = noise[0:sample_size]
351.
352.          # Generate synthetic sample
353.          synthetic_sample = generator(noise).numpy()
354.
355.          # If user wants unscaled data, unscaled synthetic data
356.          # This step should also be performed if post processing is necessary. I.e.,
    rounding can only by done with observations in the original range.
357.          if post_processing or not scaled:
358.              # Unscale data
359.              synthetic_sample = scaler.inverse_transform(synthetic_sample)
360.          # If it is necessary to round data
361.          if (post_processing):
```

```
362.                synthetic_sample = pd.DataFrame(synthetic_sample, columns =
    self.data._columns)
363.                synthetic_sample.loc[:,int_cols] =
    np.round(synthetic_sample.loc[:,int_cols],0)
364.            # Scale data
365.            if post_processing and scaled:
366.                synthetic_sample = scaler.transform(synthetic_sample)
367.
368.            # To pandas.DataFrame
369.            synthetic_sample = pd.DataFrame(synthetic_sample, columns=self.data._columns)
370.
371.            return synthetic_sample
372.
373.        def train(self, target_accuracy,
374.                  label_real = 0.9, label_synthetic = 0.0, batch_size = 64,
375.                  max_epochs = 10000, alfa = 0.05, checkpoints = True, evaluation_interval
    = 512, plot_interval = 64, earlystopping = False):
376.
377.            """ Train the GAN
378.            This method comprises the following activities:
379.            * Training GANs for up to max_epochs
380.            * Evaluating the generator quality of representation (using the C2ST) every
    evaluation_interval epochs
381.            * Plotting synthetic vs real data and GAN losses every plot_interval epochs
382.
383.            Parameters
384.            _____________
385.            target_accuracy: float
386.                target input model accuracy
387.            label_real, label_synthetic: float
388.                (optional) values for the labels that indicate if an observation is real or
    synthetic. Standard values consider one-side label smoothing for training stabilization.
389.            batch_size: integer
390.                (optional) the desired batch size
391.            max_epochs: integer
392.                (optional) the maximum number of epochs for training
393.            alfa: numeric
394.                (optional) significance level por C2ST (GAN evaluation)
395.            checkpoints: boolean
396.                (optional) if True, the training process will save checkpoints every
    checkpoints_interval (if the generator presents a better performance)
397.            evaluation_interval: integer
398.                (optional) number of epochs between evaluations of the generator
    performance
399.            plot_interval: integer
400.                (optional) number of epochs between plots of synthetic data and losses
401.            earlystopping: boolean
402.                (optional) if true, the training process will be stopped earlier if the
    generator accuracy has reached the target accuracy
403.
404.            Returns
405.            _____________
406.            _best_model_accuracy: float
407.                the best model accuracy identified during the training process
408.            _epochs: list
409.                list of epochs when data plots were created (according to plot_interval)
410.            Saves images for the data and losses plots
411.            Creates a video to demonstrate the results over epochs
412.            Prints the C2ST results during and after training
413.
414.            """
415.
416.            # Start time
417.            start = time.time()
418.
419.            # Initialize model accuracy (worst possible)
420.            self._best_model_accuracy = 0
421.
```

```
422.          # Initialize pvalue (worst possible)
423.          pvalue = 1
424.
425.          # Initalize vectors of discriminator and generator losses accross epochs
426.          D_losses = []
427.          G_losses = []
428.
429.          # Initalize attribute to count epochs
430.          self.epoch = 0
431.
432.          # Initialize attribute to record epochs with saved graphs
433.          self._epochs = []
434.
435.          # Define batch size
436.          batch_size = define_efficient_batch(batch_size, self.data._sample_size)
437.
438.          # Get training data and create batches
439.          scaled_data = self.data.scaled_data
440.          tf_dataset = tf.data.Dataset.from_tensor_slices(scaled_data)
441.          batches = tf_dataset.batch(batch_size, drop_remainder=True).prefetch(1)
442.
443.          # Grab the separate GAN components
444.          generator = self.generator
445.          discriminator = self.discriminator
446.
447.          # For every epoch
448.          while (self.epoch < max_epochs or (earlystopping and pvalue < alfa)):
449.
450.              # Update epoch attribute
451.              self.epoch += 1
452.
453.              # Print epoch
454.              if (self.epoch == 1 or self.epoch%evaluation_interval == 0):
455.                  print(f"Currently on Epoch {self.epoch}")
456.
457.              # For every batch in the dataset
458.              for batch in batches:
459.
460.                  # Cast it to float 32
461.                  batch = tf.dtypes.cast(batch, tf.float32)
462.
463.                  ####################################
464.                  ## TRAINING THE DISCRIMINATOR ######
465.                  ####################################
466.
467.                  # Generate synthetic sample based on noise input
468.                  synthetic_sample =
    self.generate_synthetic_sample(sample_size=batch_size, scaled=True)
469.                  synthetic_sample = synthetic_sample.to_numpy()
470.                  # Concatenate synthetic sample against the real dataframe
471.                  X = tf.concat([synthetic_sample, batch], axis=0)
472.                  # Define labels (real or synthetic) for synthetic and real samples to
    train the discriminator (according label smoothing parameter)
473.                  # We want the discriminator to distinguish real and synthetic samples
474.                  y_discriminator = tf.constant([[label_synthetic]] * batch_size +
    [[label_real]] * batch_size)
475.
476.                  # Train the discriminator on this batch
477.                  discriminator.trainable = True
478.                  D_loss = discriminator.train_on_batch(X, y_discriminator)
479.                  # Record discriminator loss
480.                  D_losses.append(D_loss)
481.
482.                  ####################################
483.                  ## TRAINING THE GENERATOR     ######
484.                  ####################################
485.
486.                  # Create noise (latent variables)
```

```
487.                     noise = self.generate_noise(sample_size=batch_size)
488.
489.                     # Define labels to train the generator (according to label smoothing
    parameter)
490.                     # We want the discriminator to believe that the samples are real
491.                     y_generator = tf.constant([[label_real]] * batch_size)
492.
493.                     # Train the generator on this batch
494.                     # For stability, now the discriminator will not be trained
495.                     discriminator.trainable = False
496.                     G_loss = self.GAN.train_on_batch(noise, y_generator)
497.                     # Record generator loss
498.                     G_losses.append(G_loss)
499.
500.                 # Check lack-of-fit using C2ST
501.                 if (self.epoch == 1 or self.epoch%evaluation_interval == 0 or self.epoch ==
    max_epochs):
502.                     # Evaluate generator according to user defined target accuracy
503.                     model_accuracy, pvalue, _ =
    self.evaluate_generator(target_accuracy=target_accuracy,
504.                                                                         alfa=alfa)
505.
506.                     # If current model presents the best performance until now
507.                     if checkpoints:
508.                         if (model_accuracy >= self._best_model_accuracy):
509.                             # Save model weights
510.                             generator.save_weights(self._output_path + "\\checkpoint")
511.                             self._best_model_accuracy = model_accuracy
512.
513.                 # Plot data
514.                 if (self.epoch == 1 or self.epoch%plot_interval == 0 or self.epoch ==
    max_epochs):
515.                     # Plot losses
516.                     ganplot.plot_losses(D_losses=D_losses, G_losses=G_losses,
517.                                         path=self._output_path, epoch=self.epoch,
    formats=["jpg"])
518.                     # Generate sample
519.
    synthetic_data=self.generate_synthetic_sample(sample_size=self.data._sample_size,
520.                                         scaled=False, noise=self.standard_noise,
    post_processing=True)
521.
522.                     # Plot distribution
523.                     ganplot.plot_data(synthetic_data=synthetic_data,
    real_data=self.data.real_data,
524.                                       path=self._output_path, epoch=self.epoch,
    formats=["jpg"])
525.
526.                     # Append this epoch id: this vector will be used to create a movie with
    epochs' exported images
527.                     self._epochs.append(self.epoch)
528.
529.             # Report results
530.             print("="*100)
531.             print("Training has been completed")
532.             print("="*100)
533.
534.             # Load weights from the best epoch
535.             if checkpoints:
536.                 self.generator.load_weights(self._output_path + "\\checkpoint")
537.
538.             # Evaluate generator for standard accuracies
539.             _ = self.evaluate_generator(alfa=alfa)
540.
541.             print("="*100)
542.
543.             # Calculate training time
544.             end = time.time()
```

```
545.          training_time = end - start
546.
547.          # Return generator
548.          return generator, training_time
549.
550.      def evaluate_generator(self, alfa, target_accuracy = None):
551.
552.          """ Evaluate generator quality of representation
553.          This method comprises the following steps:
554.          * Generate synthetic samples using the generator
555.          * Concatenate them with real data
556.          * Pass test data and labels (synthetic e real) to C2ST
557.
558.          Parameters
559.          _____________
560.          target_accuracy: float
561.              target input model accuracy
562.          alfa: numeric
563.              significance level por C2ST (GAN evaluation)
564.
565.          Returns
566.          _____________
567.          Result: namedtuple (with results of the C2ST)
568.              model_accuracy: numeric
569.                  model accuracy calculated with a knn classifier
570.              pvalue: numeric
571.                  the pvalue of the C2ST for one hypothesized model accuracy
572.              h_accuracy_class: string
573.                  the accuracy class. If target_accuracy is defined, the tested accuracy
    class is the user hypothesis.
574.                  Otherwise, the test will consider the interpretation of the standard
    hypothesis (none to almost perfect)
575.          """
576.
577.          # Sample size for evaluation
578.          sample_size = self.data._sample_size
579.          # Standard noise
580.          standard_noise = self.standard_noise
581.          # Scaled data (real) for comparison
582.          scaled_data = self.data.scaled_data
583.
584.          # Named tuple for results
585.          Result = namedtuple('evaluation',['model_accuracy', 'pvalue',
    'h_accuracy_class'])
586.
587.          # Generate scaled synthetic sample based on standard noise input
588.          synthetic_sample = self.generate_synthetic_sample(sample_size=sample_size,
    scaled=True,
589.                                                            noise=standard_noise,
    post_processing=True)
590.          synthetic_sample = synthetic_sample.to_numpy()
591.
592.          # Create concatenated test samples
593.          # Test data = classes + other columns
594.          test_data = np.concatenate((synthetic_sample, scaled_data))
595.          # Test label = real or synthetic (0 or 1)
596.          test_label = np.concatenate((np.zeros(sample_size), np.ones(sample_size)))
597.
598.          # C2ST_knn is designed to separate synthetic and real samples
599.          # If synthetic and real samples come from the same distribution, test accuracy
    should remain near chance-level
600.          model_accuracy, pvalue, h_accuracy_class = C2ST_knn(test_data, test_label,
    target_accuracy, alfa)
601.
602.          print(f"Input model accuracy = {round(100*model_accuracy,1)}%.")
603.          print(f"For target accuracy = {h_accuracy_class}, p-value =
    {round(pvalue,3)}.")
604.
```

```
605.            return Result(model_accuracy, pvalue, h_accuracy_class)
606.
```

Apêndice C – Módulo *c2st*

Este apêndice reúne os códigos desenvolvidos para implementar o teste classificador para duas amostras (C2ST) baseado no classificador k-NN.

```
""" C2ST
This module includes:
* C2ST_knn (function): a function to perform the knn-based Classifier 2-Sample Test (C2ST)

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022

"""

# Imports
from numpy import ceil
from math import sqrt
from sklearn.utils import shuffle
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import norm
from collections import namedtuple
from utilities import round_up_to_odd, model_to_C2ST_accuracy, C2ST_to_model_accuracy

# knn-based C2ST
def C2ST_knn(test_data, labels, h_model_accuracy = None, alfa = 0.05):
    """ Perform the knn-based C2ST
    The test runs a knn classification using cross-validation.
    k (number of neighbors) is defined as the smallest odd number greather than or equal to sqrt(n) (an usual rule of thumb).
    Where n is the total number of observations.
    The accuracy is tested against standard or user-defined hypothesis using the one proportion test (normal approximation).

    Parameters
    ____________
    test_data: numpy.ndarray
        multidimensional array composed by real and synthetic data
    labels: numpy.ndarray
        vector of numeric labels indicating whether each observation of test_data is real (1) or synthetic (0)
    h_model_accuracy: numeric
        (optional) the hypothesized model accuracy (between 0 and 1). If none, the test will use a set of standard hypothesis (none to almost perfect)
    alfa: numeric
        (optional) the significance level (standard = 0.05)

    Returns
    ____________
    Result: namedtuple
        model_accuracy: numeric
            model accuracy calculated with a knn classifier
        pvalue: numeric
            the pvalue of the C2ST for one hypothesized model accuracy
        h_accuracy_class: string
            the accuracy class. If h_model_accuracy is defined, the tested accuracy class is the user hypothesis.
            Otherwise, the test will consider the interpretation of the standard hypothesis (none to almost perfect)
    """

    # Reference accuracies and interpretations for the set of standard hypothesis
```

```
53.     reference_accuracies = {
54.         0.59: 'A - Almost perfect (>82%)',
55.         0.68: 'B - Strong (64% - 82%)',
56.         0.825: 'C - Moderate (35% - 64%)',
57.         0.925: 'D - Weak (15% - 35%)',
58.         0.98: 'E - Minimal (4% - 15%)',
59.         1.00: 'F - None (0% - 4%)'
60.     }
61.
62.     # Named tuple for the test results
63.     Result = namedtuple('C2ST_results', ['model_accuracy', 'pvalue', 'h_accuracy_class'])
64.
65.     # Check if user defined an hypothesis for the model accuracy
66.     if h_model_accuracy is None:
67.         h_accuracies = reference_accuracies
68.     else:
69.         h_accuracies = {
70.             model_to_C2ST_accuracy(h_model_accuracy): f'User hypothesis:
    {round(100*h_model_accuracy,1)}%'}
71.
72.     # Defines the knn classifier
73.     knn = KNeighborsClassifier(n_neighbors = round_up_to_odd(sqrt(test_data.shape[0])))
74.
75.     # Shuffle test data and labels
76.     test_data, labels = shuffle(test_data, labels, random_state=0)
77.
78.     # Calculate the C2ST accuracy using cross-validation
79.     scores = cross_val_score(estimator=knn, X=test_data, y=labels, cv=5)
80.     accuracy = scores.mean()
81.     # And translates it to model accuracy
82.     model_accuracy = C2ST_to_model_accuracy(accuracy)
83.
84.     # Perform the hypothesis tests
85.     for h_accuracy, h_accuracy_class in h_accuracies.items():
86.         # Test statistic and pvalue for one proportion test (normal approximation)
87.         test_statistic = (accuracy - h_accuracy)/sqrt(h_accuracy*(1-
    h_accuracy)/(test_data.shape[0]))
88.         pvalue = norm.cdf(test_statistic, 0, 1)
89.         # If test is considering a set of hypothesis, checks if the actual one is already
    accepted
90.         if (pvalue <= alfa):
91.             return Result(model_accuracy=model_accuracy, pvalue=pvalue,
    h_accuracy_class=h_accuracy_class)
92.     return Result(model_accuracy=model_accuracy, pvalue=pvalue,
    h_accuracy_class=h_accuracy_class)
93.
```

Apêndice D – Módulo *savemodel*

Este apêndice reúne os códigos desenvolvidos para traduzir e salvar o modelo de dados de entrada, ou seja, a equação da rede geradora, acoplada a transformações conforme o tratamento dos dados modelados.

```
""" Save input model
This module includes:
* save_input_model (function): translates the generator to the language used by FlexSim

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022
"""

# Imports
from tensorflow.keras.layers import Dense
import xlsxwriter

def save_input_model(model, path, file_name, scaler = None, int_columns = None, column_headers = None):

    """ Saves the input model

    Translates the generator to the language used by FlexSim. The equation comprises:
    * Sampling of latent space variables
    * Neurons of hidden and output layers
    * Reverse scaling
    * Post processing, if one or more variables should be integers
    * Returning one observation (vector) of the input variables
    Equation components are denoted L{i}N{ji}, where L{i} are each generator layer and N[ji] each neuron of L{i}.

    This function is applicable only if:
    * The scaler is MinMaxScaler (w/ output activation function = sigmoid) or RobustScaler (w/ output activation function = linear)
    * The activation function of hidden layers is Leaky ReLU with slope 0.2
    * The latent variables follow standard normal distributions

    Parameters
    ____________
    model: tensorflow.keras.models.Model
        trained generator
    path: string
        file path
    file_name: string
        model name for saving the files
    scaler: sklearn.preprocessing scaler instance
        (optional) scaler fitted with real data (for reverse scaling)
    int_columns: list
        (optional) names of columns that contains integer data (for rounding operations)
    column_headers: list
        (optional) name of columns of the real dataset. If int_columns is not none, this parameter is mandatory

    Returns
    ____________
    Saves the generator equation to txt and xlxs files.

    """

    # Model layers
    dense_layers = [layer for layer in model.layers if isinstance(layer, Dense)]
```

```
    # Initialize vector of equation components (strings)
    equation_components = []

    # Initialize inputs (latent space)
    inputsNum = model.input_shape[1]
    for neuronNum in range(inputsNum):
        equation_components.append(f'double L0N{neuronNum} = normal(0,1);')

    # Initialize variables (neurons)
    for layerNum, layer in enumerate(dense_layers):
        neurons = layer.units
        for toNeuronNum in range(neurons):
            equation_components.append(f'double L{layerNum+1}N{toNeuronNum} = 0;')

    # Dump biases and weights
    for layerNum, layer in enumerate(dense_layers):
        weights = layer.get_weights()[0]
        biases = layer.get_weights()[1]

        for toNeuronNum, bias in enumerate(biases):
            equation_components.append(f'L{layerNum+1}N{toNeuronNum} += {bias};')

        for fromNeuronNum, wgt in enumerate(weights):
            for toNeuronNum, wgt2 in enumerate(wgt):
                equation_components.append(f'L{layerNum+1}N{toNeuronNum} += {wgt2} *
L{layerNum}N{fromNeuronNum};')

        # Activation function
        for toNeuronNum, _ in enumerate(biases):
            neuron = f'L{layerNum+1}N{toNeuronNum}'
            # If this is not the last layer, activation function = leaky ReLU
            if (layerNum != (len(dense_layers)-1)):
                equation_components.append(f'{neuron} = ({neuron} < 0)*(0.2*{neuron}) +
({neuron} >= 0)*({neuron});')
            # Else, activation function = linear or sigmoid
            else:
                if layer.get_config()["activation"] == "sigmoid":
                    equation_components.append(f'{neuron} = 1 /(1 + Math.exp(-
{neuron}));')

    # Reverse scaling and rounding, if applicable
    if scaler is not None:
        # Scaler class name
        scaler_class = scaler.__class__.__name__
        # Output layer
        output_layer = dense_layers[-1]
        id_output_layer = len(dense_layers)
        biases = output_layer.get_weights()[1]
        # For each neuron in the output layer
          for neuronNum, _ in enumerate(biases):
            neuron = f'L{id_output_layer}N{neuronNum}'
            if scaler_class == "MinMaxScaler":
                equation_components.append(f'{neuron} =
{neuron}*{scaler.data_range_[neuronNum]} + {scaler.data_min_[neuronNum]};')
            if scaler_class == "RobustScaler":
                equation_components.append(f'{neuron} =
{neuron}*{scaler.scale_[neuronNum]} + {scaler.center_[neuronNum]};')
            if int_columns is not None and column_headers is not None:
                if column_headers[neuronNum] in int_columns:
                    equation_components.append(f'{neuron} = Math.round({neuron},0);')

    # Print return
    returnText = []
    lastLayerNum = len(dense_layers) - 1
    outputsNum = model.output_shape[1]
    for neuronNum in range (outputsNum):
        returnText.append(f'L{lastLayerNum+1}N{neuronNum}')
    equation_components.append('return [' + ', '.join(returnText) + '];')
```

```
117.        equation = "\n".join(equation_components)
118.
119.        with open(path + "\\" + file_name + ".txt", "w+") as text_file:
120.            text_file.write(equation)
121.
122.        with xlsxwriter.Workbook(path + "\\" + file_name + ".xlsx") as workbook:
123.            worksheet = workbook.add_worksheet('input_model')
124.            worksheet.write(0, 0, equation)
```

Apêndice E – Módulo *ganplot*

Este apêndice reúne os códigos desenvolvidos para plotar gráficos dos dados reais e sintéticos, comparando-os para avaliação visual da qualidade de ajuste.

```
""" Plot results
This module includes:
* plot_losses (function): plot generator and discriminator losses over epochs
* plot_data (function): plot histograms (for univariate data) or pairplots (for multivariate data) over epochs
* qqplot (function): plot quantile-quantile graphs over epochs, for each input variable
* create_video (function): create videos from the images saved by the other functions

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022
"""

# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
import imageio

def plot_losses(D_losses, G_losses, path, epoch = None, formats = ["jpg"], dpi=300):
    """ Plot generator and discriminator losses over epochs

    Parameters
    ____________
    D_losses: list
        vector of discriminator losses for each training epoch
    G_losses: list
        vector of generator losses for each training epoch
    path: string
        path to save the images
    epoch: integer
        (optional) number of the actual epoch, for the definition of the graph title
    formats: list
        (optional) image formats to be saved
    dpi: integer
        (optional) image quality

    Returns
    ____________
    Saves the plots
    """

    # Create new figure
    fig = plt.figure()
    # Plot losses
    plt.plot(D_losses, label = "D. loss", zorder=50, alpha=0.7)
    plt.plot(G_losses, label = "G. loss", zorder=40, alpha=0.7)
    # Define axis titles
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    # Include legend
    plt.legend(loc = 1)
    # Set title
    fig.suptitle(f'GAN losses, epoch {epoch}', fontsize=12)
    # Save plots
    for format in formats:
```

```
58.         plt.savefig(f"{path}\\Epoch {epoch} - GAN losses.{format}", format=format,
    dpi=dpi) #, bbox_inches='tight')
59.     plt.close()
60.
61. def qqplot(synthetic_data, real_data, path, epoch = None, formats = ["jpg"], dpi=300):
62.     """ Plot quantile-quantile graphs over epochs, for each input variable
63.
64.     Parameters
65.     ____________
66.     synthetic_data: pandas.DataFrame
67.         synthetic data created by the generator
68.     real_data: pandas.DataFrame
69.         real data
70.     path: string
71.         path to save the images
72.     epoch: integer
73.         (optional) number of the actual epoch, for the definition of the graph title
74.     formats: list
75.         (optional) image formats to be saved
76.     dpi: integer
77.         (optional) image quality
78.
79.     Returns
80.     ____________
81.     Saves the plots
82.     """
83.
84.     # Real data quantiles
85.     quantiles_real = real_data.quantile(q=np.arange(0.01,1.00,0.01))
86.     # Synthetic data quantiles
87.     quantiles_synthetic = synthetic_data.quantile(q=np.arange(0.01,1.00,0.01))
88.     # For each input variable
89.     for col in quantiles_real.columns:
90.         # Plot quantiles
91.         sns.scatterplot(x=quantiles_real[col],y=quantiles_synthetic[col])
92.         # Axis limits
93.         axis_limits = np.array([plt.xlim(), plt.ylim()])
94.         axis_limits = axis_limits.flatten()
95.         # Use same min e max limits for both axis, for better visualization
96.         min_lim = np.min(axis_limits)
97.         max_lim = np.max(axis_limits)
98.         # Set axis limits
99.         plt.xlim([min_lim, max_lim])
100.          plt.ylim([min_lim, max_lim])
101.          # Draw diagonal line
102.          plt.axline((min_lim, min_lim), (max_lim, max_lim), linewidth=1.5, color='r')
103.          # Define axis titles
104.          plt.xlabel("Quantiles for real data")
105.          plt.ylabel("Quantiles for synthetic data")
106.          # Set title
107.          plt.title(f"qq-plot for {col}, epoch {epoch}")
108.          # Include grid
109.          plt.grid()
110.          # Save plots
111.          for format in formats:
112.              plt.savefig(f"{path}\\Epoch {epoch} - qq-plot for {col}.{format}",
    format=format, dpi=dpi) #, bbox_inches='tight')
113.          plt.close()
114.
115.  def plot_data(synthetic_data, real_data, path, epoch = None, formats = ["jpg"],
    dpi=300):
116.      """ Call functions to plot losses and qqplots and generate histogram/pairplots over
    epochs
117.
118.      Parameters
119.      ____________
120.      synthetic_data: pandas.DataFrame
121.          synthetic data created by the generator
```

```
122.      real_data: pandas.DataFrame
123.          real data
124.      path: string
125.          path to save the images
126.      epoch: integer
127.          (optional) number of the actual epoch, for the definition of the graph title
128.      formats: list
129.          (optional) image formats to be saved
130.      dpi: integer
131.          (optional) image quality
132.      Returns
133.      _____________
134.      Saves the plots
135.      """
136.
137.      # Real data
138.      real_data_copy = real_data.copy()
139.      # Columns
140.      columns = synthetic_data.columns
141.      # Hue (for histogram and pairplot)
142.      hue = '_Type'
143.      hue_order = ['Real', 'Synthetic']
144.
145.      # Create quantile-quantile plot
146.      qqplot(synthetic_data=synthetic_data, real_data=real_data_copy,
147.             epoch=epoch, path=path, formats=formats, dpi=dpi)
148.
149.      # Create auxiliar column for data type
150.      synthetic_data['_Type'] = 'Synthetic'
151.      real_data_copy['_Type'] = 'Real'
152.      # And concatenate synthetic and real data for distribution plot
153.      df = pd.concat([synthetic_data, real_data_copy])
154.      df.reset_index(inplace=True, drop=True)
155.
156.      # Distribution plot according to the number of variables
157.      if len(columns) == 1:
158.          # Univariate
159.          sns.displot(data=df, x=columns[0], hue=hue, hue_order=hue_order, palette="hls")
160.      else:
161.          # Multivariate
162.          sns.pairplot(data=df, vars=columns, hue=hue, hue_order=hue_order,
    diag_kind='hist', plot_kws = {"s":3})
163.
164.      # Set plot title
165.      plt.title(f"Histogram, epoch {epoch}")
166.
167.      # Save images
168.      for format in formats:
169.          plt.savefig(f"{path}\\Epoch {epoch} - histogram.{format}", format=format,
    dpi=dpi) #, bbox_inches='tight')
170.
171.      plt.close()
172.
173.  def create_video(variables, epochs, path, fps = 60):
174.      """ Create videos from the images saved by the other functions
175.
176.      Parameters
177.      _____________
178.      variables: list
179.          list of input variables (column headers)
180.      epochs: list
181.          list of epochs with saved plots
182.      path: string
183.          path to save the images
184.      fps: integer
185.          (optional) frames per second
186.      Returns
187.      _____________
```

```
188.        Saves the videos
189.        """
190.
191.        # List of images of losses and histograms/pairplots
192.        images_losses = []
193.        images_histogram = []
194.        # For each epoch with saved plots
195.        for epoch in epochs:
196.            # Append images to lists
197.            images_losses.append(imageio.imread(f"{path}\\Epoch {epoch} - GAN losses.jpg"))
198.            images_histogram.append(imageio.imread(f"{path}\\Epoch {epoch} -
    histogram.jpg"))
199.        # Create videos for losses and histrograms/pairplots
200.        imageio.mimsave(f"{path}\\Losses.mp4", images_losses, fps=fps)
201.        imageio.mimsave(f"{path}\\Histograms.mp4", images_histogram, fps=fps)
202.
203.        # For each input variable
204.        for variable in variables:
205.            # List of images of qqplots
206.            images_qqplot = []
207.            # For each epoch with saved images
208.            for epoch in epochs:
209.                # Append image to list
210.                images_qqplot.append(imageio.imread(f"{path}\\Epoch {epoch} - qq-plot for
    {variable}.jpg"))
211.            # Create video for qqplot
212.            imageio.mimsave(f"{path}\\q-q plots for {variable}.mp4", images_qqplot,
    fps=fps)
213.
214.
```

Apêndice F – Módulo *utilities*

Este apêndice reúne funções de apoio.

```
""" Utility functions for other modules
This module includes:
* C2ST_to_model_accuracy (function): translates CS2T accuracy into model accuracy
* model_to_C2ST_accuracy (function): translates model accuracy into C2ST accuracy
* round_up_to_odd (function): rounds a numeric value up to the next odd value
* define_efficient_batch (function): finds an efficient batch size to maximize the use of real data during GAN training

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022

"""

# Imports
from numpy import ceil

def C2ST_to_model_accuracy(C2ST_accuracy):
    """ Translates CS2T accuracy into model accuracy

    Parameters
    ____________
    C2ST_accuracy: numeric
        an C2ST accuracy value between 0 and 1

    Returns
    ____________
    model_accuracy: numeric
        an model accuracy value between 0 and 1
    """
    model_accuracy = 1 - 2*max(0, C2ST_accuracy - 0.5)
    return model_accuracy

def model_to_C2ST_accuracy(model_accuracy):
    """ Translates model accuracy into C2ST accuracy

    Parameters
    ____________
    model_accuracy: numeric
        an model accuracy value between 0 and 1

    Returns
    ____________
    C2ST_accuracy: numeric
        an C2ST accuracy value between 0 and 1
    """
    C2ST_accuracy = 0.5 + (1 - model_accuracy)/2
    return C2ST_accuracy

def round_up_to_odd(f):
    """ Rounds a numeric value up to the next odd value

    Parameters
    ____________
    f: numeric
        a float number

    Returns
    ____________
    odd: numeric
        the next odd value
    """
```

```
62.     odd = int(ceil(f) // 2 * 2 + 1)
63.     return odd
64.
65. def define_efficient_batch(desired_batch_size, real_data_sample_size):
66.     """ Finds an efficient batch size to maximize the use of real data during GAN
    training.
67.     During GAN training, the last batch should be dropped in the case it has fewer than
    batch_size elements.
68.     This function finds the batch size closest to the one desired by the user and for
    which the sample is multiple.
69.
70.     Parameters
71.     _____________
72.     desired_batch: integer
73.         the desired batch size
74.     real_data_sample_size: integer
75.         the number of observations in the real dataset
76.
77.     Returns
78.     _____________
79.     efficient_batch_size: integer
80.         the batch size that maximizes the use of real data during GAN training
81.     """
82.     # The complete sample comprises n real observations and n synthetic observations
83.     total_sample_size = real_data_sample_size*2
84.     mod = total_sample_size%desired_batch_size
85.     if (mod == 0):
86.         efficient_batch_size = desired_batch_size
87.     else:
88.         if (mod <= desired_batch_size/2):
89.             efficient_batch_size = desired_batch_size + mod/4
90.         else:
91.             efficient_batch_size = desired_batch_size - (desired_batch_size-mod)/4
92.     return int(efficient_batch_size)
93.
```

Apêndice G – Exemplo de utilização da proposta

Este apêndice demonstra por meio de códigos a utilização da proposta para a modelagem dos dados do estudo de caso “círculo” do conjunto de dados Datasaurus.

```
""" Example of MDE-GANs application for modeling Datasaurus 'circle' dataset

Author: Afonso Teberga <afonso.teberga@gmail.com>

Created: 31th July, 2022

"""

# Imports
import gandata
from gan import GAN
from savemodel import save_input_model
from sklearn.preprocessing import RobustScaler

# Create instance of gandata
data = gandata.Data()
# Read dataset
data.read_excel("D:\MyData\datasaurus_circle.xlsx")
# Prepare data using RobustScaler
data.prepare_data(RobustScaler())
# Create GAN instance
myGAN = GAN(data)
# Train model
generator, training_time = myGAN.train(target_accuracy=0.95, max_epochs=32)
# Save input model
save_input_model(model=generator, path="D:\MyData", file_name="myModel",
scaler=data._scaler)
```

Apêndice H – Exemplo de MDE exportado

Neste apêndice está demonstrado um exemplo de modelo de dados de entrada exportado pelo algoritmo desenvolvido e compatível com a linguagem de programação utilizada pelo FlexSim.

```
/**MDE-GAN para */
double L0N0 = normal(0,1);
double L0N1 = normal(0,1);
double L0N2 = normal(0,1);
double L0N3 = normal(0,1);
double L0N4 = normal(0,1);
double L0N5 = normal(0,1);
double L1N0 = 0;
double L1N1 = 0;
double L1N2 = 0;
double L1N3 = 0;
double L1N4 = 0;
double L1N5 = 0;
double L1N6 = 0;
double L1N7 = 0;
double L1N8 = 0;
double L1N9 = 0;
double L1N10 = 0;
double L1N11 = 0;
double L1N12 = 0;
double L1N13 = 0;
double L1N14 = 0;
double L1N15 = 0;
double L2N0 = 0;
double L2N1 = 0;
double L2N2 = 0;
double L2N3 = 0;
double L2N4 = 0;
double L2N5 = 0;
double L2N6 = 0;
double L2N7 = 0;
double L2N8 = 0;
double L2N9 = 0;
double L2N10 = 0;
double L2N11 = 0;
double L2N12 = 0;
double L2N13 = 0;
double L2N14 = 0;
double L2N15 = 0;
double L3N0 = 0;
double L3N1 = 0;
double L3N2 = 0;
double L3N3 = 0;
double L3N4 = 0;
double L3N5 = 0;
double L3N6 = 0;
double L3N7 = 0;
double L3N8 = 0;
double L3N9 = 0;
double L3N10 = 0;
double L3N11 = 0;
double L3N12 = 0;
double L3N13 = 0;
double L3N14 = 0;
double L3N15 = 0;
double L4N0 = 0;
double L4N1 = 0;
double L4N2 = 0;
double L4N3 = 0;
double L4N4 = 0;
double L4N5 = 0;
double L4N6 = 0;
double L4N7 = 0;
```

```
64. double L4N8 = 0;
65. double L4N9 = 0;
66. double L4N10 = 0;
67. double L4N11 = 0;
68. double L4N12 = 0;
69. double L4N13 = 0;
70. double L4N14 = 0;
71. double L4N15 = 0;
72. double L5N0 = 0;
73. double L5N1 = 0;
74. L1N0 += 0.10695146024227142;
75. L1N1 += 0.15107855200767517;
76. L1N2 += 0.31228408217430115;
77. L1N3 += 0.11467897891998291;
78. L1N4 += -0.35768815875053406;
79. L1N5 += 0.005103187169879675;
80. L1N6 += 0.0564936101436615;
81. L1N7 += 0.127553790807724;
82. L1N8 += 0.3167121112346649;
83. L1N9 += 0.12223763018846512;
84. L1N10 += -0.20307724177837372;
85. L1N11 += 0.21562713384628296;
86. L1N12 += 0.06732279807329178;
87. L1N13 += 0.04384152591228485;
88. L1N14 += 0.4164145588874817;
89. L1N15 += 0.22508396208286285;
90. L1N0 += -0.22982384264469147 * L0N0;
91. L1N1 += -0.2993376851081848 * L0N0;
92. L1N2 += -0.10530037432909012 * L0N0;
93. L1N3 += 0.057228244841098785 * L0N0;
94. L1N4 += -0.11664190888404846 * L0N0;
95. L1N5 += 0.10137046128511429 * L0N0;
96. L1N6 += -0.08825188130140305 * L0N0;
97. L1N7 += 0.17702944576740265 * L0N0;
98. L1N8 += 0.19243526458740234 * L0N0;
99. L1N9 += 0.01259008888155222 * L0N0;
100.L1N10 += 0.2020404338836667 * L0N0;
101.L1N11 += 0.10556872189044952 * L0N0;
102.L1N12 += -0.10050386935472488 * L0N0;
103.L1N13 += -0.15728741884231567 * L0N0;
104.L1N14 += -0.24279437959194183 * L0N0;
105.L1N15 += -0.05281732976436615 * L0N0;
106.L1N0 += 0.011374481953680515 * L0N1;
107.L1N1 += 0.012680852785706 52 * L0N1;
108.L1N2 += 0.061035554856061935 * L0N1;
109.L1N3 += -0.5566778779029846 * L0N1;
110.L1N4 += -0.34872329235076904 * L0N1;
111.L1N5 += -0.5759904980659485 * L0N1;
112.L1N6 += -0.20488493144512177 * L0N1;
113.L1N7 += 0.5132859349250793 * L0N1;
114.L1N8 += 0.5044444799423218 * L0N1;
115.L1N9 += 0.3347271978855133 * L0N1;
116.L1N10 += 0.21563908457756042 * L0N1;
117.L1N11 += -0.6818026304244995 * L0N1;
118.L1N12 += -0.26811328530311584 * L0N1;
119.L1N13 += 0.03895058482885361 * L0N1;
120.L1N14 += -0.3465336561203003 * L0N1;
121.L1N15 += -0.34398165345191956 * L0N1;
122.L1N0 += -0.15703889727592468 * L0N2;
123.L1N1 += -0.21630965173244476 * L0N2;
124.L1N2 += -0.38472095131874084 * L0N2;
125.L1N3 += 0.24583406746387482 * L0N2;
126.L1N4 += 0.3419954180717468 * L0N2;
127.L1N5 += 0.33855751156806946 * L0N2;
128.L1N6 += 0.20404241979122162 * L0N2;
129.L1N7 += -0.3291204869747162 * L0N2;
130.L1N8 += -0.4123314321041107 * L0N2;
131.L1N9 += -0.06844470649957657 * L0N2;
132.L1N10 += -0.14698632061481476 * L0N2;
133.L1N11 += 0.2565721869468689 * L0N2;
134.L1N12 += 0.5071045160293579 * L0N2;
135.L1N13 += -0.10526486486196518 * L0N2;
136.L1N14 += -0.18371325731277466 * L0N2;
```

```
L1N15 += 0.23232080042362213 * L0N2;
L1N0 += 0.20447495579719543 * L0N3;
L1N1 += 0.2815406322479248 * L0N3;
L1N2 += -0.04649937152862549 * L0N3;
L1N3 += -0.14370255172252655 * L0N3;
L1N4 += 0.41664841771125793 * L0N3;
L1N5 += -0.21590293943881989 * L0N3;
L1N6 += 0.2542453706264496 * L0N3;
L1N7 += -0.44490960240364075 * L0N3;
L1N8 += -0.5119102597236633 * L0N3;
L1N9 += -0.1671714037656784 * L0N3;
L1N10 += -0.3551088869571686 * L0N3;
L1N11 += -0.3046431243419647 * L0N3;
L1N12 += 0.5667809247970581 * L0N3;
L1N13 += 0.2836381793022156 * L0N3;
L1N14 += 0.09366948157548904 * L0N3;
L1N15 += 0.1597094088792801 * L0N3;
L1N0 += 0.12294935435056686 * L0N4;
L1N1 += 0.15778015553951263 * L0N4;
L1N2 += 0.3015754818916321 * L0N4;
L1N3 += -0.10170638561248779 * L0N4;
L1N4 += -0.31483250856399536 * L0N4;
L1N5 += -0.10501230508089066 * L0N4;
L1N6 += -0.19875580072402954 * L0N4;
L1N7 += 0.26886969804763794 * L0N4;
L1N8 += 0.34499943256378174 * L0N4;
L1N9 += 0.14017176628112793 * L0N4;
L1N10 += -0.06932993233203888 * L0N4;
L1N11 += -0.06997498869895935 * L0N4;
L1N12 += -0.4757643938064575 * L0N4;
L1N13 += 0.002804114483296871 * L0N4;
L1N14 += 0.25116702914237976 * L0N4;
L1N15 += -0.14723894000053406 * L0N4;
L1N0 += 0.33346912264823914 * L0N5;
L1N1 += 0.45317715406417847 * L0N5;
L1N2 += 0.5663143396377563 * L0N5;
L1N3 += 0.44253379106521606 * L0N5;
L1N4 += 0.17396727204322815 * L0N5;
L1N5 += 0.22006401419639587 * L0N5;
L1N6 += 0.19433265924453735 * L0N5;
L1N7 += -0.4985322952270508 * L0N5;
L1N8 += -0.3874005973339081 * L0N5;
L1N9 += -0.6937238574028015 * L0N5;
L1N10 += -0.0927705317735672 * L0N5;
L1N11 += 0.5778990983963013 * L0N5;
L1N12 += -0.30824729800224304 * L0N5;
L1N13 += 0.24292810261249542 * L0N5;
L1N14 += 0.8659984469413757 * L0N5;
L1N15 += 0.20406357944011688 * L0N5;
L1N0 = (L1N0 < 0)*(0.2*L1N0) + (L1N0 >= 0)*(L1N0);
L1N1 = (L1N1 < 0)*(0.2*L1N1) + (L1N1 >= 0)*(L1N1);
L1N2 = (L1N2 < 0)*(0.2*L1N2) + (L1N2 >= 0)*(L1N2);
L1N3 = (L1N3 < 0)*(0.2*L1N3) + (L1N3 >= 0)*(L1N3);
L1N4 = (L1N4 < 0)*(0.2*L1N4) + (L1N4 >= 0)*(L1N4);
L1N5 = (L1N5 < 0)*(0.2*L1N5) + (L1N5 >= 0)*(L1N5);
L1N6 = (L1N6 < 0)*(0.2*L1N6) + (L1N6 >= 0)*(L1N6);
L1N7 = (L1N7 < 0)*(0.2*L1N7) + (L1N7 >= 0)*(L1N7);
L1N8 = (L1N8 < 0)*(0.2*L1N8) + (L1N8 >= 0)*(L1N8);
L1N9 = (L1N9 < 0)*(0.2*L1N9) + (L1N9 >= 0)*(L1N9);
L1N10 = (L1N10 < 0)*(0.2*L1N10) + (L1N10 >= 0)*(L1N10);
L1N11 = (L1N11 < 0)*(0.2*L1N11) + (L1N11 >= 0)*(L1N11);
L1N12 = (L1N12 < 0)*(0.2*L1N12) + (L1N12 >= 0)*(L1N12);
L1N13 = (L1N13 < 0)*(0.2*L1N13) + (L1N13 >= 0)*(L1N13);
L1N14 = (L1N14 < 0)*(0.2*L1N14) + (L1N14 >= 0)*(L1N14);
L1N15 = (L1N15 < 0)*(0.2*L1N15) + (L1N15 >= 0)*(L1N15);
L2N0 += 0.17679420113563538;
L2N1 += 0.006330722942948341;
L2N2 += 0.06173498183488846;
L2N3 += 0.07773300477027893;
L2N4 += -0.19657908380031586;
L2N5 += 0.1203998550772667;
L2N6 += -0.04519252851605415;
L2N7 += 0.1557326763868332;
```

```
210.L2N8 += 0.005584236700087786;
211.L2N9 += 0.16022086143493652;
212.L2N10 += 0.014386706054210663;
213.L2N11 += -0.06648766994476318;
214.L2N12 += 0.2501501142978668;
215.L2N13 += -0.15278171002864838;
216.L2N14 += -0.31032320857048035;
217.L2N15 += 0.17848365008831024;
218.L2N0 += 0.05770610272884369 * L1N0;
219.L2N1 += -0.3312543034553528 * L1N0;
220.L2N2 += 0.26787906885147095 * L1N0;
221.L2N3 += 0.22516362369060516 * L1N0;
222.L2N4 += -0.11554186046123505 * L1N0;
223.L2N5 += 0.08067085593938828 * L1N0;
224.L2N6 += -0.39513009786605835 * L1N0;
225.L2N7 += 0.1240372508764267 * L1N0;
226.L2N8 += -0.04911091923713684 * L1N0;
227.L2N9 += -0.13974222540855408 * L1N0;
228.L2N10 += 0.29250985383987427 * L1N0;
229.L2N11 += 0.09724147617816925 * L1N0;
230.L2N12 += -0.46773436665534973 * L1N0;
231.L2N13 += 0.3524459898471832 * L1N0;
232.L2N14 += -0.3210906982421875 * L1N0;
233.L2N15 += 0.2880183458328247 * L1N0;
234.L2N0 += -0.34795090556144714 * L1N1;
235.L2N1 += 0.03338991105556488 * L1N1;
236.L2N2 += -0.3075248897075653 * L1N1;
237.L2N3 += -0.274406760931015 * L1N1;
238.L2N4 += 0.15475469827651978 * L1N1;
239.L2N5 += 0.26768404245376587 * L1N1;
240.L2N6 += -0.49094054102897644 * L1N1;
241.L2N7 += 0.028505055233836174 * L1N1;
242.L2N8 += -0.06668546050786972 * L1N1;
243.L2N9 += 0.36161336302757263 * L1N1;
244.L2N10 += 0.2600594162940979 * L1N1;
245.L2N11 += 0.3629295527935028 * L1N1;
246.L2N12 += 0.1485145539045334 * L1N1;
247.L2N13 += -0.0021461190190166235 * L1N1;
248.L2N14 += 0.005098492838442326 * L1N1;
249.L2N15 += -0.034030601382255554 * L1N1;
250.L2N0 += -0.0391601137816906 * L1N2;
251.L2N1 += -0.010936625301837921 * L1N2;
252.L2N2 += 0.18275630474090576 * L1N2;
253.L2N3 += 0.3049706220626831 * L1N2;
254.L2N4 += -0.45533838868141174 * L1N2;
255.L2N5 += -0.3663052022457123 * L1N2;
256.L2N6 += -0.08985108137130737 * L1N2;
257.L2N7 += -0.5425071716308594 * L1N2;
258.L2N8 += 0.41227108240127563 * L1N2;
259.L2N9 += 0.3168027698993683 * L1N2;
260.L2N10 += 0.2967996299266815 * L1N2;
261.L2N11 += -0.07119596749544144 * L1N2;
262.L2N12 += -0.11170432716608047 * L1N2;
263.L2N13 += 0.19518640637397766 * L1N2;
264.L2N14 += 0.12980951368808746 * L1N2;
265.L2N15 += 0.054811716079711914 * L1N2;
266.L2N0 += -0.3502625823020935 * L1N3;
267.L2N1 += -0.1835256814956665 * L1N3;
268.L2N2 += 0.4954283535480499 * L1N3;
269.L2N3 += -0.3010351061820984 * L1N3;
270.L2N4 += 0.18212735652923584 * L1N3;
271.L2N5 += 0.29956647753715515 * L1N3;
272.L2N6 += -0.5056031346321106 * L1N3;
273.L2N7 += -0.5457834601402283 * L1N3;
274.L2N8 += 0.09153841435909271 * L1N3;
275.L2N9 += 0.07436641305685043 * L1N3;
276.L2N10 += 0.09260119497776031 * L1N3;
277.L2N11 += 0.15551665425300598 * L1N3;
278.L2N12 += 0.04957097768783569 * L1N3;
279.L2N13 += -0.023344125598669052 * L1N3;
280.L2N14 += 0.13784736394882202 * L1N3;
281.L2N15 += -0.2876944839954376 * L1N3;
282.L2N0 += 0.1606389284133911 * L1N4;
```

```
283.L2N1 += -0.4025626480579376 * L1N4;
284.L2N2 += -0.30045565962791443 * L1N4;
285.L2N3 += 0.3013535439968109 * L1N4;
286.L2N4 += 0.40357065200805664 * L1N4;
287.L2N5 += -0.5737048983573914 * L1N4;
288.L2N6 += -0.49805545806884766 * L1N4;
289.L2N7 += -0.3256787955760956 * L1N4;
290.L2N8 += 0.1094847247004509 * L1N4;
291.L2N9 += -0.21911053359508514 * L1N4;
292.L2N10 += -0.003080683061853051 * L1N4;
293.L2N11 += 0.30889296531677246 * L1N4;
294.L2N12 += -0.07911940664052963 * L1N4;
295.L2N13 += 0.24989058077335358 * L1N4;
296.L2N14 += 0.16285710036754608 * L1N4;
297.L2N15 += -0.3022889494895935 * L1N4;
298.L2N0 += -0.28101444244384766 * L1N5;
299.L2N1 += -0.20572058856487274 * L1N5;
300.L2N2 += -0.2225760817527771 * L1N5;
301.L2N3 += -0.14510147273540497 * L1N5;
302.L2N4 += -0.10811035335063934 * L1N5;
303.L2N5 += -0.16449134051799774 * L1N5;
304.L2N6 += -0.2929359972476959 * L1N5;
305.L2N7 += 0.024756887927651405 * L1N5;
306.L2N8 += -0.07946359366178513 * L1N5;
307.L2N9 += 0.22264565527439117 * L1N5;
308.L2N10 += 0.16576510667800903 * L1N5;
309.L2N11 += -0.0850202664732933 * L1N5;
310.L2N12 += -0.30605876445770264 * L1N5;
311.L2N13 += 0.3915550708770752 * L1N5;
312.L2N14 += 0.003285677172243595 * L1N5;
313.L2N15 += 0.20451219379901886 * L1N5;
314.L2N0 += -0.030159620568156242 * L1N6;
315.L2N1 += 0.1858397275209427 * L1N6;
316.L2N2 += -0.1649150848388672 * L1N6;
317.L2N3 += -0.4628114104270935 * L1N6;
318.L2N4 += -0.08313047140836716 * L1N6;
319.L2N5 += 0.20562468469142914 * L1N6;
320.L2N6 += 0.1421421766281128 * L1N6;
321.L2N7 += 0.15699365735054016 * L1N6;
322.L2N8 += 0.006358363199979067 * L1N6;
323.L2N9 += 0.4300260841846466 * L1N6;
324.L2N10 += -0.2400440275669098 * L1N6;
325.L2N11 += 0.3235061466693878 * L1N6;
326.L2N12 += -0.08647476881742477 * L1N6;
327.L2N13 += 0.12329480797052383 * L1N6;
328.L2N14 += -0.44321781396865845 * L1N6;
329.L2N15 += -0.04824201017618179 * L1N6;
330.L2N0 += 0.5093513131141663 * L1N7;
331.L2N1 += -0.3560653626918793 * L1N7;
332.L2N2 += -0.3375144898891449 * L1N7;
333.L2N3 += 0.30770376324653625 * L1N7;
334.L2N4 += 0.4049086868876297 * L1N7;
335.L2N5 += -0.5029560923576355 * L1N7;
336.L2N6 += 0.14504803717136383 * L1N7;
337.L2N7 += 0.21584244072437286 * L1N7;
338.L2N8 += -0.391072541475296 * L1N7;
339.L2N9 += -0.36760565638542175 * L1N7;
340.L2N10 += -0.5727357864379883 * L1N7;
341.L2N11 += -0.1719699501991272 * L1N7;
342.L2N12 += -0.21411685645580292 * L1N7;
343.L2N13 += 0.08139368891716003 * L1N7;
344.L2N14 += -0.24580691754817963 * L1N7;
345.L2N15 += 0.11999417841434479 * L1N7;
346.L2N0 += 0.5634073615074158 * L1N8;
347.L2N1 += -0.40097424387931824 * L1N8;
348.L2N2 += -0.23243677616119385 * L1N8;
349.L2N3 += 0.48356908559799194 * L1N8;
350.L2N4 += 0.1367521584033966 * L1N8;
351.L2N5 += -0.46602946519851685 * L1N8;
352.L2N6 += 0.44398805499076843 * L1N8;
353.L2N7 += 0.04208742082118988 * L1N8;
354.L2N8 += -0.538705587387085 * L1N8;
355.L2N9 += -0.2424192726612091 * L1N8;
```

```
356.L2N10 += -0.39745572209358215 * L1N8;
357.L2N11 += -0.5120688676834106 * L1N8;
358.L2N12 += 0.22384895384311676 * L1N8;
359.L2N13 += -0.023232607170939445 * L1N8;
360.L2N14 += -0.04658878594636917 * L1N8;
361.L2N15 += 0.15242432057857513 * L1N8;
362.L2N0 += 0.23125603795051575 * L1N9;
363.L2N1 += -0.03248037025332451 * L1N9;
364.L2N2 += 0.3404187560081482 * L1N9;
365.L2N3 += 0.338119238615036 * L1N9;
366.L2N4 += 0.06784860044717789 * L1N9;
367.L2N5 += -0.3604899048805237 * L1N9;
368.L2N6 += 0.5275251865386963 * L1N9;
369.L2N7 += -0.2950654923915863 * L1N9;
370.L2N8 += 0.18397144973278046 * L1N9;
371.L2N9 += -0.10071340948343277 * L1N9;
372.L2N10 += -0.376166969537735 * L1N9;
373.L2N11 += -0.3159080147743225 * L1N9;
374.L2N12 += 0.4072623550891876 * L1N9;
375.L2N13 += -0.25698918104171753 * L1N9;
376.L2N14 += 0.38342610001564026 * L1N9;
377.L2N15 += -0.1546621471643448 * L1N9;
378.L2N0 += 0.018402159214019775 * L1N10;
379.L2N1 += 0.22939200699329376 * L1N10;
380.L2N2 += 0.0024337663780897856 * L1N10;
381.L2N3 += 0.1490049660205841 * L1N10;
382.L2N4 += -0.019596518948674202 * L1N10;
383.L2N5 += -0.18999312818050385 * L1N10;
384.L2N6 += 0.19315117597579956 * L1N10;
385.L2N7 += -0.13927359879016876 * L1N10;
386.L2N8 += 0.2111838012933731 * L1N10;
387.L2N9 += 0.005274228751659393 * L1N10;
388.L2N10 += 0.04173974320292473 * L1N10;
389.L2N11 += -0.3581170439720154 * L1N10;
390.L2N12 += -0.11603749543428421 * L1N10;
391.L2N13 += 0.18729156255722046 * L1N10;
392.L2N14 += -0.04747970774769783 * L1N10;
393.L2N15 += 0.3290242552757263 * L1N10;
394.L2N0 += 0.16847556829452515 * L1N11;
395.L2N1 += -0.25987428426742554 * L1N11;
396.L2N2 += -0.11968894302845001 * L1N11;
397.L2N3 += 0.034469403326511138 * L1N11;
398.L2N4 += -0.2749740183353424 * L1N11;
399.L2N5 += 0.023488597944378853 * L1N11;
400.L2N6 += -0.37660592794418335 * L1N11;
401.L2N7 += -0.05553797259926796 * L1N11;
402.L2N8 += -0.07840488106012344 * L1N11;
403.L2N9 += 0.2781227231025696 * L1N11;
404.L2N10 += -0.05063411593437195 * L1N11;
405.L2N11 += 0.356528639793396 * L1N11;
406.L2N12 += -0.06185542792081833 * L1N11;
407.L2N13 += 0.17664770781993866 * L1N11;
408.L2N14 += -0.1873238980770111 * L1N11;
409.L2N15 += -0.07917268574237823 * L1N11;
410.L2N0 += -0.008521895855665207 * L1N12;
411.L2N1 += 0.048466138541698456 * L1N12;
412.L2N2 += -0.019899416714906693 * L1N12;
413.L2N3 += -0.20542198419570923 * L1N12;
414.L2N4 += -0.3991868197917938 * L1N12;
415.L2N5 += 0.32260191440582275 * L1N12;
416.L2N6 += 0.5366027355194092 * L1N12;
417.L2N7 += 0.4739528298377991 * L1N12;
418.L2N8 += 0.27437660098075867 * L1N12;
419.L2N9 += -0.042137227952480316 * L1N12;
420.L2N10 += 0.34712591767311096 * L1N12;
421.L2N11 += 0.06477636098861694 * L1N12;
422.L2N12 += -0.2738102674484253 * L1N12;
423.L2N13 += -0.15033164620399475 * L1N12;
424.L2N14 += -0.44238758087158203 * L1N12;
425.L2N15 += 0.10562589764595032 * L1N12;
426.L2N0 += 0.2178991436958313 * L1N13;
427.L2N1 += 0.1752876192331314 * L1N13;
428.L2N2 += 0.20447257161140442 * L1N13;
```

```
L2N3 += -0.313519150018692 * L1N13;
L2N4 += -0.015861256048083305 * L1N13;
L2N5 += 0.1745581328868866 * L1N13;
L2N6 += -0.19811765849590302 * L1N13;
L2N7 += -0.13086453080177307 * L1N13;
L2N8 += 0.020774593576788902 * L1N13;
L2N9 += 0.25806137919425964 * L1N13;
L2N10 += -0.1779647320508957 * L1N13;
L2N11 += 0.14141340553760529 * L1N13;
L2N12 += 0.15813317894935608 * L1N13;
L2N13 += 0.27996134757995605 * L1N13;
L2N14 += -0.3020434081554413 * L1N13;
L2N15 += -0.20093949139118195 * L1N13;
L2N0 += -0.23682399094104767 * L1N14;
L2N1 += 0.1917727142572403 * L1N14;
L2N2 += -0.45070600509643555 * L1N14;
L2N3 += -0.43676474690437317 * L1N14;
L2N4 += -0.13385841250419617 * L1N14;
L2N5 += -0.07069883495569229 * L1N14;
L2N6 += -0.5159294605255127 * L1N14;
L2N7 += -0.3574669659137726 * L1N14;
L2N8 += -0.3048723638057709 * L1N14;
L2N9 += 0.36185944080352783 * L1N14;
L2N10 += -0.2696084976196289 * L1N14;
L2N11 += 0.13282902538776398 * L1N14;
L2N12 += -0.48728886246681213 * L1N14;
L2N13 += 0.4316358268260956 * L1N14;
L2N14 += -0.13133111596107483 * L1N14;
L2N15 += -0.00013701118587050587 * L1N14;
L2N0 += -0.25107884407043457 * L1N15;
L2N1 += -0.09783823043107986 * L1N15;
L2N2 += 0.364907830953598 * L1N15;
L2N3 += 0.07639680802822113 * L1N15;
L2N4 += -0.05702289566397667 * L1N15;
L2N5 += 0.3102017045021057 * L1N15;
L2N6 += -0.03223973885178566 * L1N15;
L2N7 += 0.0314396470785141 * L1N15;
L2N8 += 0.4074648916721344 * L1N15;
L2N9 += 0.4272627532482147 * L1N15;
L2N10 += -0.23092342913150787 * L1N15;
L2N11 += 0.4370119869709015 * L1N15;
L2N12 += 0.23755663633346558 * L1N15;
L2N13 += 0.0093726497143507 * L1N15;
L2N14 += 0.26084619760513306 * L1N15;
L2N15 += 0.17538942396640778 * L1N15;
L2N0 = (L2N0 < 0)*(0.2*L2N0) + (L2N0 >= 0)*(L2N0);
L2N1 = (L2N1 < 0)*(0.2*L2N1) + (L2N1 >= 0)*(L2N1);
L2N2 = (L2N2 < 0)*(0.2*L2N2) + (L2N2 >= 0)*(L2N2);
L2N3 = (L2N3 < 0)*(0.2*L2N3) + (L2N3 >= 0)*(L2N3);
L2N4 = (L2N4 < 0)*(0.2*L2N4) + (L2N4 >= 0)*(L2N4);
L2N5 = (L2N5 < 0)*(0.2*L2N5) + (L2N5 >= 0)*(L2N5);
L2N6 = (L2N6 < 0)*(0.2*L2N6) + (L2N6 >= 0)*(L2N6);
L2N7 = (L2N7 < 0)*(0.2*L2N7) + (L2N7 >= 0)*(L2N7);
L2N8 = (L2N8 < 0)*(0.2*L2N8) + (L2N8 >= 0)*(L2N8);
L2N9 = (L2N9 < 0)*(0.2*L2N9) + (L2N9 >= 0)*(L2N9);
L2N10 = (L2N10 < 0)*(0.2*L2N10) + (L2N10 >= 0)*(L2N10);
L2N11 = (L2N11 < 0)*(0.2*L2N11) + (L2N11 >= 0)*(L2N11);
L2N12 = (L2N12 < 0)*(0.2*L2N12) + (L2N12 >= 0)*(L2N12);
L2N13 = (L2N13 < 0)*(0.2*L2N13) + (L2N13 >= 0)*(L2N13);
L2N14 = (L2N14 < 0)*(0.2*L2N14) + (L2N14 >= 0)*(L2N14);
L2N15 = (L2N15 < 0)*(0.2*L2N15) + (L2N15 >= 0)*(L2N15);
L3N0 += -0.1382434368133545;
L3N1 += 0.19612865149974823;
L3N2 += 0.04839078336954117;
L3N3 += 0.2587492763996124;
L3N4 += 0.20156085491180420;
L3N5 += -0.06945056468248367;
L3N6 += -0.0634329691529274;
L3N7 += 0.04598114266991615;
L3N8 += -0.19554494321346283;
L3N9 += 0.14107164740562440;
L3N10 += 0.10985288023948670;
L3N11 += 0.14249937236309052;
```

```
L3N12 += 0.22885167598724365;
L3N13 += 0.17413517832756042;
L3N14 += 0.11446884274482727;
L3N15 += 0.14770843088626862;
L3N0 += -0.3571985065937042 * L2N0;
L3N1 += 0.02134987711906433 * L2N0;
L3N2 += 0.0010602392721921206 * L2N0;
L3N3 += 0.26345106959342957 * L2N0;
L3N4 += -0.16193781793117523 * L2N0;
L3N5 += -0.46559762954711914 * L2N0;
L3N6 += 0.29794028401374817 * L2N0;
L3N7 += -0.29329684376716614 * L2N0;
L3N8 += 0.18876175582408905 * L2N0;
L3N9 += 0.2422236055135727 * L2N0;
L3N10 += -0.47511038184165955 * L2N0;
L3N11 += -0.07183124125003815 * L2N0;
L3N12 += 0.19672632217407227 * L2N0;
L3N13 += -0.4812087416648865 * L2N0;
L3N14 += -0.2723715901374817 * L2N0;
L3N15 += 0.05961834266781807 * L2N0;
L3N0 += 0.04137653857469559 * L2N1;
L3N1 += 0.12581832706928253 * L2N1;
L3N2 += 0.35010325908660889 * L2N1;
L3N3 += -0.33500921726226807 * L2N1;
L3N4 += 0.2604817748069763 * L2N1;
L3N5 += -0.15981127321720123 * L2N1;
L3N6 += -0.04922771453857422 * L2N1;
L3N7 += 0.474800705909729 * L2N1;
L3N8 += -0.15208567678928375 * L2N1;
L3N9 += -0.2299967259168625 * L2N1;
L3N10 += 0.5246976613998413 * L2N1;
L3N11 += 0.367596685886383060 * L2N1;
L3N12 += -0.18308788537979126 * L2N1;
L3N13 += -0.021581396460533142 * L2N1;
L3N14 += 0.15836447477340698 * L2N1;
L3N15 += 0.20048822462558746 * L2N1;
L3N0 += 0.05744394287467003 * L2N2;
L3N1 += 0.21983812749385834 * L2N2;
L3N2 += 0.2439107894897461 * L2N2;
L3N3 += 0.18535441160202026 * L2N2;
L3N4 += 0.22732110321521760 * L2N2;
L3N5 += 0.13020850718021393 * L2N2;
L3N6 += 0.23385535180568695 * L2N2;
L3N7 += 0.38549560308456420 * L2N2;
L3N8 += 0.10627221316099167 * L2N2;
L3N9 += -0.2602340579032898 * L2N2;
L3N10 += -0.09422171115875244 * L2N2;
L3N11 += 0.11869414150714874 * L2N2;
L3N12 += 0.30785647034644080 * L2N2;
L3N13 += -0.14745964109897614 * L2N2;
L3N14 += 0.40133270621299744 * L2N2;
L3N15 += -0.1145213171839714 * L2N2;
L3N0 += 0.10642563551664352 * L2N3;
L3N1 += -0.04214849695563316 * L2N3;
L3N2 += 0.05041886866092682 * L2N3;
L3N3 += 0.17356614768505096 * L2N3;
L3N4 += 0.20931403338909150 * L2N3;
L3N5 += -0.3770342171192169 * L2N3;
L3N6 += 0.16613689064979553 * L2N3;
L3N7 += -0.34158971905708313 * L2N3;
L3N8 += 0.10829898715019226 * L2N3;
L3N9 += -0.3875363767147064 * L2N3;
L3N10 += -0.4394950866699219 * L2N3;
L3N11 += -0.20239609479904175 * L2N3;
L3N12 += -0.35394546389579773 * L2N3;
L3N13 += -0.11751729995012283 * L2N3;
L3N14 += -0.5463056564331055 * L2N3;
L3N15 += -0.3135393559932709 * L2N3;
L3N0 += -0.28981301188468933 * L2N4;
L3N1 += -0.33739003539085390 * L2N4;
L3N2 += -0.5748686194419861 * L2N4;
L3N3 += -0.00844578631222248 * L2N4;
L3N4 += -0.2456510365009308 * L2N4;
```

```
575.L3N5 += -0.2824607491493225 * L2N4;
576.L3N6 += -0.29571032524108887 * L2N4;
577.L3N7 += 0.058940671384334564 * L2N4;
578.L3N8 += 0.14209988713264465 * L2N4;
579.L3N9 += 0.09123063832521439 * L2N4;
580.L3N10 += 0.3202662765979767 * L2N4;
581.L3N11 += -0.42352962493896484 * L2N4;
582.L3N12 += -0.19888398051261902 * L2N4;
583.L3N13 += 0.09190085530281067 * L2N4;
584.L3N14 += -0.46251940727233887 * L2N4;
585.L3N15 += -0.16827233135700226 * L2N4;
586.L3N0 += 0.5522328615188599 * L2N5;
587.L3N1 += 0.2655816376209259 * L2N5;
588.L3N2 += 0.37725451588630676 * L2N5;
589.L3N3 += -0.48780906200408936 * L2N5;
590.L3N4 += -0.3540497124195099 * L2N5;
591.L3N5 += 0.4198920726776123 * L2N5;
592.L3N6 += -0.2076953500509262 * L2N5;
593.L3N7 += 0.2884843051433563 * L2N5;
594.L3N8 += 0.004320287145674229 * L2N5;
595.L3N9 += -0.20550771057605743 * L2N5;
596.L3N10 += -0.03721917048096657 * L2N5;
597.L3N11 += 0.08911026269197464 * L2N5;
598.L3N12 += -0.01540807168930769 * L2N5;
599.L3N13 += 0.3924897313117981 * L2N5;
600.L3N14 += 0.3849309980869293 * L2N5;
601.L3N15 += -0.07808060944080353 * L2N5;
602.L3N0 += -0.33675384521484375 * L2N6;
603.L3N1 += -0.6206623911857605 * L2N6;
604.L3N2 += 0.35769352316856384 * L2N6;
605.L3N3 += -0.11998724937438965 * L2N6;
606.L3N4 += 0.06724672019481659 * L2N6;
607.L3N5 += 0.4829581379890442 * L2N6;
608.L3N6 += -0.03369337320327759 * L2N6;
609.L3N7 += 0.347551554441452 * L2N6;
610.L3N8 += 0.48171529173851013 * L2N6;
611.L3N9 += -0.44640645384788513 * L2N6;
612.L3N10 += 0.08103439956903458 * L2N6;
613.L3N11 += -0.022263145074248314 * L2N6;
614.L3N12 += -0.6401853561401367 * L2N6;
615.L3N13 += -0.671088457107544 * L2N6;
616.L3N14 += -0.48582491278648376 * L2N6;
617.L3N15 += 0.10360168665647507 * L2N6;
618.L3N0 += 0.0014727929374203086 * L2N7;
619.L3N1 += -0.2423618733882904 * L2N7;
620.L3N2 += 0.2194976657629013 * L2N7;
621.L3N3 += 0.28361907601356506 * L2N7;
622.L3N4 += 0.2838708162307739 * L2N7;
623.L3N5 += 0.5440695285797119 * L2N7;
624.L3N6 += -0.009988315403461456 * L2N7;
625.L3N7 += 0.21081134676933289 * L2N7;
626.L3N8 += 0.3444746434688568 * L2N7;
627.L3N9 += 0.27468448877334595 * L2N7;
628.L3N10 += 0.4907085597515106 * L2N7;
629.L3N11 += 0.31230196356773376 * L2N7;
630.L3N12 += -0.24469681084156036 * L2N7;
631.L3N13 += 0.2603408694267273 * L2N7;
632.L3N14 += -0.029392430558800697 * L2N7;
633.L3N15 += -0.10995861887931824 * L2N7;
634.L3N0 += 0.1165304183959961 * L2N8;
635.L3N1 += 0.2234722524881363 * L2N8;
636.L3N2 += 0.08469350636005402 * L2N8;
637.L3N3 += 0.022502796724438667 * L2N8;
638.L3N4 += -0.12030418962240219 * L2N8;
639.L3N5 += 0.06210501492023468 * L2N8;
640.L3N6 += 0.11933216452598572 * L2N8;
641.L3N7 += 0.3708968758583069 * L2N8;
642.L3N8 += 0.06535357236862183 * L2N8;
643.L3N9 += 0.1640915423631668 * L2N8;
644.L3N10 += 0.3303592801094055 * L2N8;
645.L3N11 += -0.21231667697429657 * L2N8;
646.L3N12 += -0.3944900631904602 * L2N8;
647.L3N13 += 0.0078439274802804 * L2N8;
```

```
648.L3N14 += 0.2547006607055664 * L2N8;
649.L3N15 += -0.07699832320213318 * L2N8;
650.L3N0 += 0.12484333664178848 * L2N9;
651.L3N1 += 0.31649425625801086 * L2N9;
652.L3N2 += -0.5525079369544983 * L2N9;
653.L3N3 += -0.2778518795967102 * L2N9;
654.L3N4 += -0.09515687823295593 * L2N9;
655.L3N5 += 0.2642917335033417 * L2N9;
656.L3N6 += -0.36963823437690735 * L2N9;
657.L3N7 += -0.4540131688117981 * L2N9;
658.L3N8 += 0.13532429933547974 * L2N9;
659.L3N9 += 0.26353153586387634 * L2N9;
660.L3N10 += 0.06454047560691833 * L2N9;
661.L3N11 += 0.19579829275608063 * L2N9;
662.L3N12 += 0.25653067231178284 * L2N9;
663.L3N13 += 0.18897102773189545 * L2N9;
664.L3N14 += -0.008177964016795158 * L2N9;
665.L3N15 += 0.3299933075904846 * L2N9;
666.L3N0 += 0.13074007630348206 * L2N10;
667.L3N1 += 0.12725403904914856 * L2N10;
668.L3N2 += -0.2988448143005371 * L2N10;
669.L3N3 += -0.26221880316734314 * L2N10;
670.L3N4 += 0.17160861194133759 * L2N10;
671.L3N5 += -0.01189424004405737 * L2N10;
672.L3N6 += -0.12631048262119293 * L2N10;
673.L3N7 += -0.04201395437121391 * L2N10;
674.L3N8 += 0.41847825050354004 * L2N10;
675.L3N9 += 0.3042084872722626 * L2N10;
676.L3N10 += -0.1940954476594925 * L2N10;
677.L3N11 += -0.3219843804836273 * L2N10;
678.L3N12 += 0.08962688595056534 * L2N10;
679.L3N13 += 0.1664917916059494 * L2N10;
680.L3N14 += 0.35668015480041504 * L2N10;
681.L3N15 += 0.2616043984889984 * L2N10;
682.L3N0 += 0.5691145062446594 * L2N11;
683.L3N1 += -0.3395065665245056 * L2N11;
684.L3N2 += 0.2180025726556778 * L2N11;
685.L3N3 += -0.580183744430542 * L2N11;
686.L3N4 += -0.1675146073102951 * L2N11;
687.L3N5 += -0.0739881843328476 * L2N11;
688.L3N6 += 0.2215356081724167 * L2N11;
689.L3N7 += -0.07224279642105103 * L2N11;
690.L3N8 += 0.3084300458431244 * L2N11;
691.L3N9 += 0.26272913813591003 * L2N11;
692.L3N10 += 0.15894770622253418 * L2N11;
693.L3N11 += -0.229538694024086 * L2N11;
694.L3N12 += -0.4525930881500244 * L2N11;
695.L3N13 += -0.22633862495422363 * L2N11;
696.L3N14 += 0.05507354438304901 * L2N11;
697.L3N15 += -0.28062114119529724 * L2N11;
698.L3N0 += -0.3823719918727875 * L2N12;
699.L3N1 += -0.10920268297195435 * L2N12;
700.L3N2 += 0.24867026507854462 * L2N12;
701.L3N3 += 0.26264122128486633 * L2N12;
702.L3N4 += -0.08192107081413269 * L2N12;
703.L3N5 += -0.4174318313598633 * L2N12;
704.L3N6 += 0.16444681584835052 * L2N12;
705.L3N7 += 0.07736343145370483 * L2N12;
706.L3N8 += -0.3478460907936096 * L2N12;
707.L3N9 += 0.015326039865612984 * L2N12;
708.L3N10 += -0.030439483001828194 * L2N12;
709.L3N11 += -0.31529808044433594 * L2N12;
710.L3N12 += -0.1999271661043167 * L2N12;
711.L3N13 += 0.4183700680732727 * L2N12;
712.L3N14 += -0.005649076774716377 * L2N12;
713.L3N15 += 0.25491365790367126 * L2N12;
714.L3N0 += 0.033453501760959625 * L2N13;
715.L3N1 += -0.3639487624168396 * L2N13;
716.L3N2 += 0.072977215051651 * L2N13;
717.L3N3 += 0.28790241479873666 * L2N13;
718.L3N4 += -0.16715751588344574 * L2N13;
719.L3N5 += 0.07180178165435791 * L2N13;
720.L3N6 += -0.3357159197330475 * L2N13;
```

```
721.L3N7 += -0.3211514353752136 * L2N13;
722.L3N8 += 0.3580110967159271 * L2N13;
723.L3N9 += -0.42893582582473755 * L2N13;
724.L3N10 += 0.22627228498458862 * L2N13;
725.L3N11 += 0.2898480296134949 * L2N13;
726.L3N12 += -0.30167967081069946 * L2N13;
727.L3N13 += -0.2011551409959793 * L2N13;
728.L3N14 += -0.18055231869220734 * L2N13;
729.L3N15 += -0.30982595682144165 * L2N13;
730.L3N0 += 0.15761002898216248 * L2N14;
731.L3N1 += 0.046880777925252914 * L2N14;
732.L3N2 += -0.22242245078086853 * L2N14;
733.L3N3 += -0.4491525888442993 * L2N14;
734.L3N4 += 0.09435659646987915 * L2N14;
735.L3N5 += -0.36560654640197754 * L2N14;
736.L3N6 += 0.013651047833263874 * L2N14;
737.L3N7 += 0.2526959180831909 * L2N14;
738.L3N8 += -0.44560906291007996 * L2N14;
739.L3N9 += -0.3281446695327759 * L2N14;
740.L3N10 += -0.22270435094833374 * L2N14;
741.L3N11 += -0.3328462541103363 * L2N14;
742.L3N12 += -0.13076414167881012 * L2N14;
743.L3N13 += -0.4397849142551422 * L2N14;
744.L3N14 += 0.024773811921477318 * L2N14;
745.L3N15 += -0.11596976965665817 * L2N14;
746.L3N0 += -0.23320047557353973 * L2N15;
747.L3N1 += 0.03683893755078316 * L2N15;
748.L3N2 += 0.2799874246120453 * L2N15;
749.L3N3 += -0.2818835973739624 * L2N15;
750.L3N4 += -0.2886163890361786 * L2N15;
751.L3N5 += -0.17245499789714813 * L2N15;
752.L3N6 += -0.23338083922863007 * L2N15;
753.L3N7 += 0.0888524129986763 * L2N15;
754.L3N8 += -0.2095157653093338 * L2N15;
755.L3N9 += -0.15619774162769318 * L2N15;
756.L3N10 += 0.14850804209709167 * L2N15;
757.L3N11 += -0.18590521812438965 * L2N15;
758.L3N12 += 0.22117145359516144 * L2N15;
759.L3N13 += -0.17634332180023193 * L2N15;
760.L3N14 += 0.3316672742366791 * L2N15;
761.L3N15 += 0.415356308221817 * L2N15;
762.L3N0 = (L3N0 < 0)*(0.2*L3N0) + (L3N0 >= 0)*(L3N0);
763.L3N1 = (L3N1 < 0)*(0.2*L3N1) + (L3N1 >= 0)*(L3N1);
764.L3N2 = (L3N2 < 0)*(0.2*L3N2) + (L3N2 >= 0)*(L3N2);
765.L3N3 = (L3N3 < 0)*(0.2*L3N3) + (L3N3 >= 0)*(L3N3);
766.L3N4 = (L3N4 < 0)*(0.2*L3N4) + (L3N4 >= 0)*(L3N4);
767.L3N5 = (L3N5 < 0)*(0.2*L3N5) + (L3N5 >= 0)*(L3N5);
768.L3N6 = (L3N6 < 0)*(0.2*L3N6) + (L3N6 >= 0)*(L3N6);
769.L3N7 = (L3N7 < 0)*(0.2*L3N7) + (L3N7 >= 0)*(L3N7);
770.L3N8 = (L3N8 < 0)*(0.2*L3N8) + (L3N8 >= 0)*(L3N8);
771.L3N9 = (L3N9 < 0)*(0.2*L3N9) + (L3N9 >= 0)*(L3N9);
772.L3N10 = (L3N10 < 0)*(0.2*L3N10) + (L3N10 >= 0)*(L3N10);
773.L3N11 = (L3N11 < 0)*(0.2*L3N11) + (L3N11 >= 0)*(L3N11);
774.L3N12 = (L3N12 < 0)*(0.2*L3N12) + (L3N12 >= 0)*(L3N12);
775.L3N13 = (L3N13 < 0)*(0.2*L3N13) + (L3N13 >= 0)*(L3N13);
776.L3N14 = (L3N14 < 0)*(0.2*L3N14) + (L3N14 >= 0)*(L3N14);
777.L3N15 = (L3N15 < 0)*(0.2*L3N15) + (L3N15 >= 0)*(L3N15);
778.L4N0 += -0.10279684513807297;
779.L4N1 += 0.15791498124599457;
780.L4N2 += 0.25793352723121643;
781.L4N3 += -0.03789704293012619;
782.L4N4 += 0.1018124669790268;
783.L4N5 += -0.030132973566651344;
784.L4N6 += -0.018574362620711327;
785.L4N7 += 0.17065364122390747;
786.L4N8 += -0.014979061670601368;
787.L4N9 += 0.15551349520683289;
788.L4N10 += 0.10987450927495956;
789.L4N11 += 0.05578535795211792;
790.L4N12 += -0.042319606989622116;
791.L4N13 += 0.06159653514623642;
792.L4N14 += -0.07502947002649307;
793.L4N15 += 0.23776261508464813;
```

```
794.L4N0 += -0.3042299449443817 * L3N0;
795.L4N1 += -0.08992042392492294 * L3N0;
796.L4N2 += 0.2473558783531189 * L3N0;
797.L4N3 += -0.30529484152793884 * L3N0;
798.L4N4 += -0.27267178893089294 * L3N0;
799.L4N5 += -0.2504669427871704 * L3N0;
800.L4N6 += 0.1914198398590088 * L3N0;
801.L4N7 += 0.24154053628444672 * L3N0;
802.L4N8 += 0.01840278133749962 * L3N0;
803.L4N9 += -0.27541935443878174 * L3N0;
804.L4N10 += -0.4617244601249695 * L3N0;
805.L4N11 += 0.029117664322257042 * L3N0;
806.L4N12 += -0.5476120114326477 * L3N0;
807.L4N13 += 0.2800254523754120 * L3N0;
808.L4N14 += -0.3815304636955261 * L3N0;
809.L4N15 += 0.3064645230770111 * L3N0;
810.L4N0 += -0.32791340351104736 * L3N1;
811.L4N1 += -0.4680458903312683 * L3N1;
812.L4N2 += -0.25855520367622375 * L3N1;
813.L4N3 += -0.449394166469574 * L3N1;
814.L4N4 += 0.24352765083312988 * L3N1;
815.L4N5 += 0.3240308165550232 * L3N1;
816.L4N6 += 0.11554232239723206 * L3N1;
817.L4N7 += 0.2175474315881729 * L3N1;
818.L4N8 += 0.25936147570610046 * L3N1;
819.L4N9 += 0.26015588641166687 * L3N1;
820.L4N10 += 0.02512439899146557 * L3N1;
821.L4N11 += 0.22006841003894806 * L3N1;
822.L4N12 += 0.3466735780239105 * L3N1;
823.L4N13 += 0.18109099566936493 * L3N1;
824.L4N14 += -0.23333613574504852 * L3N1;
825.L4N15 += 0.06726907193660736 * L3N1;
826.L4N0 += -0.37198296189308167 * L3N2;
827.L4N1 += 0.18546625971794128 * L3N2;
828.L4N2 += -0.035813186317682266 * L3N2;
829.L4N3 += 0.5018053650856018 * L3N2;
830.L4N4 += -0.25264281034469604 * L3N2;
831.L4N5 += -0.4145321846008301 * L3N2;
832.L4N6 += -0.37041181325912476 * L3N2;
833.L4N7 += 0.2591787874698639 * L3N2;
834.L4N8 += 0.18826128542423248 * L3N2;
835.L4N9 += 0.46188238263130190 * L3N2;
836.L4N10 += -0.0011942574055865407 * L3N2;
837.L4N11 += 0.029556266963481903 * L3N2;
838.L4N12 += 0.29482099413871765 * L3N2;
839.L4N13 += 0.22567620873451233 * L3N2;
840.L4N14 += -0.26305466890335083 * L3N2;
841.L4N15 += 0.1513768583536148 * L3N2;
842.L4N0 += -0.17867521941661835 * L3N3;
843.L4N1 += 0.17903047800064087 * L3N3;
844.L4N2 += 0.5415840148925781 * L3N3;
845.L4N3 += 0.04278906434774399 * L3N3;
846.L4N4 += -0.5337427258491516 * L3N3;
847.L4N5 += 0.0883788913488388 * L3N3;
848.L4N6 += 0.10025151818990707 * L3N3;
849.L4N7 += 0.3708134889602661 * L3N3;
850.L4N8 += -0.39275363087654114 * L3N3;
851.L4N9 += -0.37449851632118225 * L3N3;
852.L4N10 += -0.0641036182641983 * L3N3;
853.L4N11 += 0.10486611723899841 * L3N3;
854.L4N12 += -0.4606698751449585 * L3N3;
855.L4N13 += -0.6014769077301025 * L3N3;
856.L4N14 += -0.1911541372537613 * L3N3;
857.L4N15 += -0.37440726161003113 * L3N3;
858.L4N0 += -0.007192388642579317 * L3N4;
859.L4N1 += 0.28271713852882385 * L3N4;
860.L4N2 += -0.039387889206409454 * L3N4;
861.L4N3 += 0.0837821215391159 * L3N4;
862.L4N4 += -0.19599097967147827 * L3N4;
863.L4N5 += -0.0726005882024765 * L3N4;
864.L4N6 += 0.225996732711792 * L3N4;
865.L4N7 += -0.023202598094940186 * L3N4;
866.L4N8 += -0.5435519814491272 * L3N4;
```

```
867.L4N9 += 0.5045763850212097 * L3N4;
868.L4N10 += 0.22518721222877502 * L3N4;
869.L4N11 += 0.25041210651397705 * L3N4;
870.L4N12 += -0.03935902193188667 * L3N4;
871.L4N13 += -0.179063081741333 * L3N4;
872.L4N14 += 0.4342765212059021 * L3N4;
873.L4N15 += 0.026507778093218803 * L3N4;
874.L4N0 += 0.2663697600364685 * L3N5;
875.L4N1 += -0.0018828896572813392 * L3N5;
876.L4N2 += -0.1280936747789383 * L3N5;
877.L4N3 += -0.11305790394544601 * L3N5;
878.L4N4 += -0.21856465935707092 * L3N5;
879.L4N5 += -0.12604062259197235 * L3N5;
880.L4N6 += -0.13841690123081207 * L3N5;
881.L4N7 += 0.46392887830734253 * L3N5;
882.L4N8 += 0.08000923693180084 * L3N5;
883.L4N9 += 0.03824658691883087 * L3N5;
884.L4N10 += -0.266900897026062 * L3N5;
885.L4N11 += 0.22264039516448975 * L3N5;
886.L4N12 += 0.37111738324165344 * L3N5;
887.L4N13 += -0.027566198259592056 * L3N5;
888.L4N14 += 0.02938178740441799 * L3N5;
889.L4N15 += -0.42185133695602417 * L3N5;
890.L4N0 += 0.29898586869239807 * L3N6;
891.L4N1 += -0.14541780948638916 * L3N6;
892.L4N2 += -0.27532172203063965 * L3N6;
893.L4N3 += -0.21912546455860138 * L3N6;
894.L4N4 += 0.3010154366493225 * L3N6;
895.L4N5 += -0.3492240309715271 * L3N6;
896.L4N6 += -0.2525809407234192 * L3N6;
897.L4N7 += -0.3483433127403259 * L3N6;
898.L4N8 += 0.33524978160858154 * L3N6;
899.L4N9 += -0.10973123461008072 * L3N6;
900.L4N10 += -0.16688592731952667 * L3N6;
901.L4N11 += 0.029517879709601402 * L3N6;
902.L4N12 += 0.3873053193092346 * L3N6;
903.L4N13 += -0.0462777242064476 * L3N6;
904.L4N14 += -0.11421070992946625 * L3N6;
905.L4N15 += -0.2070520669221878 * L3N6;
906.L4N0 += 0.3075375556945801 * L3N7;
907.L4N1 += 0.10763643682003021 * L3N7;
908.L4N2 += -0.5241452455520630 * L3N7;
909.L4N3 += 0.150213360786438 * L3N7;
910.L4N4 += -0.001960835652425885 * L3N7;
911.L4N5 += -0.296752005815506 * L3N7;
912.L4N6 += 0.34313252568244934 * L3N7;
913.L4N7 += -0.36954477429389954 * L3N7;
914.L4N8 += 0.21957489848136902 * L3N7;
915.L4N9 += 0.3372625410556793 * L3N7;
916.L4N10 += 0.1687035709619522 * L3N7;
917.L4N11 += 0.2929879426956177 * L3N7;
918.L4N12 += 0.37600839138031006 * L3N7;
919.L4N13 += -0.13636067509651184 * L3N7;
920.L4N14 += -0.05392468720674515 * L3N7;
921.L4N15 += -0.6844295263290405 * L3N7;
922.L4N0 += 0.34815266728401184 * L3N8;
923.L4N1 += -0.16833215951919556 * L3N8;
924.L4N2 += 0.04015199840068817 * L3N8;
925.L4N3 += 0.04711028188467026 * L3N8;
926.L4N4 += -0.49758127331733704 * L3N8;
927.L4N5 += 0.14790017902851105 * L3N8;
928.L4N6 += 0.33119845390319824 * L3N8;
929.L4N7 += -0.45287027955055237 * L3N8;
930.L4N8 += -0.4567713439464569 * L3N8;
931.L4N9 += -0.1752093881368637 * L3N8;
932.L4N10 += -0.32791298627853394 * L3N8;
933.L4N11 += -0.25801825523376465 * L3N8;
934.L4N12 += -0.07112789154052734 * L3N8;
935.L4N13 += -0.40849462151527405 * L3N8;
936.L4N14 += 0.11466901004314423 * L3N8;
937.L4N15 += -0.5255426168441772 * L3N8;
938.L4N0 += -0.0615440234541893 * L3N9;
939.L4N1 += -0.20890595018863678 * L3N9;
```

```
940.L4N2 += -0.23270288109779358 * L3N9;
941.L4N3 += 0.037157028913497925 * L3N9;
942.L4N4 += 0.3010448217391968 * L3N9;
943.L4N5 += 0.35691574215888977 * L3N9;
944.L4N6 += 0.18634939193725586 * L3N9;
945.L4N7 += -0.13035942614078522 * L3N9;
946.L4N8 += -0.10051868110895157 * L3N9;
947.L4N9 += -0.14435309171676636 * L3N9;
948.L4N10 += 0.16194963455200195 * L3N9;
949.L4N11 += -0.1747329980134964 * L3N9;
950.L4N12 += -0.4485771656036377 * L3N9;
951.L4N13 += 0.38867810368537903 * L3N9;
952.L4N14 += -0.3508818447589874 * L3N9;
953.L4N15 += 0.4837839901447296 * L3N9;
954.L4N0 += 0.29934829473495483 * L3N10;
955.L4N1 += 0.4503791928291321 * L3N10;
956.L4N2 += 0.16204175353050232 * L3N10;
957.L4N3 += -0.42737695574760437 * L3N10;
958.L4N4 += 0.4001122713088989 * L3N10;
959.L4N5 += -0.21404752135276794 * L3N10;
960.L4N6 += -0.11587720364332199 * L3N10;
961.L4N7 += -0.2663756012916565 * L3N10;
962.L4N8 += 0.15669432282447815 * L3N10;
963.L4N9 += -0.028887908905744553 * L3N10;
964.L4N10 += -0.17380522191524506 * L3N10;
965.L4N11 += 0.4687311053276062 * L3N10;
966.L4N12 += -0.07087238132953644 * L3N10;
967.L4N13 += 0.39287230372428894 * L3N10;
968.L4N14 += -0.21029318869113922 * L3N10;
969.L4N15 += -0.013741837814450264 * L3N10;
970.L4N0 += 0.16435788571834564 * L3N11;
971.L4N1 += 0.08403605967760086 * L3N11;
972.L4N2 += -0.05445760115981102 * L3N11;
973.L4N3 += 0.10574876517057419 * L3N11;
974.L4N4 += 0.3810519874095917 * L3N11;
975.L4N5 += 0.05195411667227745 * L3N11;
976.L4N6 += 0.0693483054637909 * L3N11;
977.L4N7 += 0.5135672092437744 * L3N11;
978.L4N8 += 0.3642987310886383 * L3N11;
979.L4N9 += 0.14184749126434326 * L3N11;
980.L4N10 += 0.045661091804504395 * L3N11;
981.L4N11 += 0.22443990409374237 * L3N11;
982.L4N12 += -0.05587758868932724 * L3N11;
983.L4N13 += 0.41659480333328247 * L3N11;
984.L4N14 += -0.02905641496181488 * L3N11;
985.L4N15 += -0.3484033942222595 * L3N11;
986.L4N0 += 0.01942712813615799 * L3N12;
987.L4N1 += -0.22140032052993774 * L3N12;
988.L4N2 += 0.5372694730758667 * L3N12;
989.L4N3 += -0.24309112131595612 * L3N12;
990.L4N4 += -0.11624891310930252 * L3N12;
991.L4N5 += 0.2973751127719879 * L3N12;
992.L4N6 += 0.25134599208831787 * L3N12;
993.L4N7 += -0.5398023724555969 * L3N12;
994.L4N8 += 0.1352599710226059 * L3N12;
995.L4N9 += -0.8173204660415649 * L3N12;
996.L4N10 += 0.3369413912296295 * L3N12;
997.L4N11 += 0.3331140875816345 * L3N12;
998.L4N12 += -0.11380849778652191 * L3N12;
999.L4N13 += -0.6723803877830505 * L3N12;
1000.  L4N14 += 0.001012888504192233 * L3N12;
1001.  L4N15 += 0.6846782565116882 * L3N12;
1002.  L4N0 += -0.3544190526008606 * L3N13;
1003.  L4N1 += -0.21943989396095276 * L3N13;
1004.  L4N2 += 0.4904841184616089 * L3N13;
1005.  L4N3 += -0.216970294713974 * L3N13;
1006.  L4N4 += 0.4417608082294464 * L3N13;
1007.  L4N5 += 0.39225807785987854 * L3N13;
1008.  L4N6 += 0.2154502123594284 * L3N13;
1009.  L4N7 += 0.2707475721836090 * L3N13;
1010.  L4N8 += 0.3496413826942444 * L3N13;
1011.  L4N9 += -0.20159955322742462 * L3N13;
1012.  L4N10 += 0.11551422625780106 * L3N13;
```

```
1013.  L4N11 += 0.22457800805568695 * L3N13;
1014.  L4N12 += 0.36334407329559326 * L3N13;
1015.  L4N13 += -0.27501460909843445 * L3N13;
1016.  L4N14 += 0.05708436295390129 * L3N13;
1017.  L4N15 += 0.16410838067531586 * L3N13;
1018.  L4N0 += 0.06192992627620697 * L3N14;
1019.  L4N1 += 0.2060519903898239 * L3N14;
1020.  L4N2 += -0.22711500525474548 * L3N14;
1021.  L4N3 += -0.4394078552722931 * L3N14;
1022.  L4N4 += 0.5193581581115723 * L3N14;
1023.  L4N5 += 0.3824387788772583 * L3N14;
1024.  L4N6 += 0.3549005091190338 * L3N14;
1025.  L4N7 += 0.14405304193496704 * L3N14;
1026.  L4N8 += -0.10383375734090805 * L3N14;
1027.  L4N9 += 0.4612460136413574 * L3N14;
1028.  L4N10 += -0.3076947331428528 * L3N14;
1029.  L4N11 += 0.335836261510849 * L3N14;
1030.  L4N12 += 0.10754292458295822 * L3N14;
1031.  L4N13 += 0.4505842626094818 * L3N14;
1032.  L4N14 += -0.18685144186019897 * L3N14;
1033.  L4N15 += -0.03428778424859047 * L3N14;
1034.  L4N0 += -0.10494151711463928 * L3N15;
1035.  L4N1 += 0.12356604635715485 * L3N15;
1036.  L4N2 += 0.06731285154819489 * L3N15;
1037.  L4N3 += -0.2025236338376999 * L3N15;
1038.  L4N4 += -0.35279810428619385 * L3N15;
1039.  L4N5 += 0.09739355742931366 * L3N15;
1040.  L4N6 += 0.2687724232673645 * L3N15;
1041.  L4N7 += 0.169214129447937 * L3N15;
1042.  L4N8 += -0.23263177275657654 * L3N15;
1043.  L4N9 += 0.03271646797657013 * L3N15;
1044.  L4N10 += 0.26014745235443115 * L3N15;
1045.  L4N11 += 0.22325725853443146 * L3N15;
1046.  L4N12 += 0.29510530829429626 * L3N15;
1047.  L4N13 += 0.1972631812095642 * L3N15;
1048.  L4N14 += 0.12533129751682281 * L3N15;
1049.  L4N15 += 0.21697965264320374 * L3N15;
1050.  L4N0 = (L4N0 < 0)*(0.2*L4N0) + (L4N0 >= 0)*(L4N0);
1051.  L4N1 = (L4N1 < 0)*(0.2*L4N1) + (L4N1 >= 0)*(L4N1);
1052.  L4N2 = (L4N2 < 0)*(0.2*L4N2) + (L4N2 >= 0)*(L4N2);
1053.  L4N3 = (L4N3 < 0)*(0.2*L4N3) + (L4N3 >= 0)*(L4N3);
1054.  L4N4 = (L4N4 < 0)*(0.2*L4N4) + (L4N4 >= 0)*(L4N4);
1055.  L4N5 = (L4N5 < 0)*(0.2*L4N5) + (L4N5 >= 0)*(L4N5);
1056.  L4N6 = (L4N6 < 0)*(0.2*L4N6) + (L4N6 >= 0)*(L4N6);
1057.  L4N7 = (L4N7 < 0)*(0.2*L4N7) + (L4N7 >= 0)*(L4N7);
1058.  L4N8 = (L4N8 < 0)*(0.2*L4N8) + (L4N8 >= 0)*(L4N8);
1059.  L4N9 = (L4N9 < 0)*(0.2*L4N9) + (L4N9 >= 0)*(L4N9);
1060.  L4N10 = (L4N10 < 0)*(0.2*L4N10) + (L4N10 >= 0)*(L4N10);
1061.  L4N11 = (L4N11 < 0)*(0.2*L4N11) + (L4N11 >= 0)*(L4N11);
1062.  L4N12 = (L4N12 < 0)*(0.2*L4N12) + (L4N12 >= 0)*(L4N12);
1063.  L4N13 = (L4N13 < 0)*(0.2*L4N13) + (L4N13 >= 0)*(L4N13);
1064.  L4N14 = (L4N14 < 0)*(0.2*L4N14) + (L4N14 >= 0)*(L4N14);
1065.  L4N15 = (L4N15 < 0)*(0.2*L4N15) + (L4N15 >= 0)*(L4N15);
1066.  L5N0 += 0.11865844577550888;
1067.  L5N1 += 0.08648092299699783;
1068.  L5N0 += -0.9808340668678284 * L4N0;
1069.  L5N1 += -0.3647302687168121 3 * L4N0;
1070.  L5N0 += 0.06612902879714966 * L4N1;
1071.  L5N1 += 0.27155381441116333 * L4N1;
1072.  L5N0 += -0.4041330814361572 * L4N2;
1073.  L5N1 += -0.020141318440437317 * L4N2;
1074.  L5N0 += 0.09316546469926834 * L4N3;
1075.  L5N1 += -0.2297641634941101 * L4N3;
1076.  L5N0 += -0.3166204094886 78 * L4N4;
1077.  L5N1 += -0.579929769039154 * L4N4;
1078.  L5N0 += -0.3578570783138275 * L4N5;
1079.  L5N1 += -0.6084591746330261 * L4N5;
1080.  L5N0 += -0.37403687834739685 * L4N6;
1081.  L5N1 += 0.2920810282230377 * L4N6;
1082.  L5N0 += 0.7258944511413574 * L4N7;
1083.  L5N1 += 0.16416528820991516 * L4N7;
1084.  L5N0 += 0.1309511363506317 * L4N8;
1085.  L5N1 += -0.44276902079582214 * L4N8;
```

```
1086.  L5N0 += 0.64991694688797 * L4N9;
1087.  L5N1 += -0.07398493587970734 * L4N9;
1088.  L5N0 += -0.8991274833679199 * L4N10;
1089.  L5N1 += 0.46880438923835754 * L4N10;
1090.  L5N0 += -0.23681333661079407 * L4N11;
1091.  L5N1 += 0.22694268822669983 * L4N11;
1092.  L5N0 += 0.43771591782569885 * L4N12;
1093.  L5N1 += -0.34575265645980835 * L4N12;
1094.  L5N0 += 0.7361483573913574 * L4N13;
1095.  L5N1 += -0.21339885890483856 * L4N13;
1096.  L5N0 += 0.21572721004486084 * L4N14;
1097.  L5N1 += -0.33751699328422546 * L4N14;
1098.  L5N0 += -0.8946645855903625 * L4N15;
1099.  L5N1 += -0.16859190165996552 * L4N15;
1100.  L5N0 = L5N0*22.071470008174998 + 53.32557334555;
1101.  L5N1 = L5N1*59.47851954972501 + 50.7623379055;
1102.  print(L5N0, L5N1);
```

REFERÊNCIAS

ABDELMEGID, M. A. *et al.* The roles of conceptual modelling in improving construction simulation studies: A comprehensive review. **Advanced Engineering Informatics**, [*s. l.*], v. 46, n. June, 2020.

ABU ALFEILAT, Haneen Arafat *et al.* Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. **Big Data**, [*s. l.*], v. 7, n. 4, p. 221–248, 2019.

AGNESE, Jorge. A survey and taxonomy of adversarial neural networks for text-to-image synthesis. **WIREs Data Mining Knowl Discov**, [*s. l.*], n. April 2019, p. 1–26, 2020.

AHMED, Ali; PAGE, John; OLSEN, John. Enhancing Six Sigma methodology using simulation techniques. **International Journal of Lean Six Sigma**, [*s. l.*], v. 11, n. 1, p. 211–232, 2020.

ALARSAN, Fajr Ibrahem; YOUNES, Mamoon. Best Selection of Generative Adversarial Networks Hyper-Parameters Using Genetic Algorithm. **SN Computer Science**, [*s. l.*], v. 2, n. 4, 2021.

ALDAUSARI, Nuha *et al.* Video Generative Adversarial Networks: A Review. **ACM Computing Surveys**, [*s. l.*], v. 55, n. 2, 2023.

ALI, Hazrat *et al.* The role of generative adversarial networks in brain MRI: a scoping review. **Insights into Imaging**, [*s. l.*], v. 13, n. 1, 2022. Disponível em: https://doi.org/10.1186/s13244-022-01237-0.

ALQAHTANI, Hamed; KAVAKLI-THORNE, Manolya; KUMAR, Gulshan. Applications of Generative Adversarial Networks (GANs): An Updated Review. **Archives of Computational Methods in Engineering**, [*s. l.*], v. 28, n. 2, p. 525–552, 2021. Disponível em: https://doi.org/10.1007/s11831-019-09388-y.

ALZUBAIDI, Laith *et al.* **Review of deep learning: concepts, CNN architectures, challenges, applications, future directions**. [*S. l.*]: Springer International Publishing, 2021-. ISSN 21961115.v. 8 Disponível em: https://doi.org/10.1186/s40537-021-00444-8.

AVERILL M. LAW & ASSOCIATES. **ExpertFit**. [*S. l.*], 2022. Disponível em: http://www.averill-law.com/distribution-fitting/. Acesso em: 7 ago. 2022.

BANKS, Jerry (org.). **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. 1. ed. New York: John Wiley and Sons Inc., 1998.

BERTRAND, J Will M; FRANSOO, Jan C. Operations management research methodologies using quantitative modeling. **INTERNATIONAL JOURNAL OF OPERATIONS & PRODUCTION MANAGEMENT**, [*s. l.*], v. 22, n. 2, p. 241–264, 2002.

BILLER, Bahar. Copula-based multivariate input models for stochastic simulation. **Operations Research**, [*s. l.*], v. 57, n. 4, p. 878–892, 2009.

BILLER, Bahar; GUNES, Canan. Copula-based multivariate input modeling. **Surveys in Operations Research and Management Science**, [*s. l.*], v. 17, n. 2, p. 69–84, 2012. Disponível

em: http://dx.doi.org/10.1016/j.sorms.2012.04.001.

BILLER, Bahar; GUNES, Canan. Introduction to Simulation Input Modeling. *In*: , 2010. **Winter Simulation Conference**. [*S. l.: s. n.*], 2010.

BILLER, Bahar; NELSON, Barry L. ANSWERS TO THE TOP TEN INPUT MODELING QUESTIONS. *In*: , 2002. **Proceedings of the 2002 Winter Simulation Conference**. [*S. l.: s. n.*], 2002. p. 35–40.

BORJI, Ali. Pros and cons of GAN evaluation measures. **Computer Vision and Image Understanding**, [*s. l.*], v. 179, n. October 2018, p. 41–65, 2019. Disponível em: https://doi.org/10.1016/j.cviu.2018.10.009.

BRO, Rasmus; SMILDE, Age K. Principal component analysis. **Analytical Methods**, [*s. l.*], v. 6, n. 9, p. 2812–2831, 2014.

CAIRO, Alberto. **Download the Datasaurus**. [*S. l.*], 2016. Disponível em: http://www.thefunctionalart.com/2016/08/download-datasaurus-never-trust-summary.html. Acesso em: 31 ago. 2020.

CEN, Wang; HERBERT, Emily A; HAAS, Peter J. NIM: GENERATIVE NEURAL NETWORKS FOR SIMULATION INPUT MODELING. *In*: , 2019. **Proceedings of the 2019 Winter Simulation Conference**. [*S. l.: s. n.*], 2019.

CEN, Wang; HERBERT, Emily A; HAAS, Peter J. Nim: Modeling and Generation of Simulation Inputs via Generative Neural Networks. *In*: , 2020. **Proceedings of the 2020 Winter Simulation Conference**. [*S. l.: s. n.*], 2020. p. 584–595.

CHENG, Russell. History of Input Modeling. *In*: , 2017. **Proceedings of the 2017 Winter Simulation Conference**. [*S. l.: s. n.*], 2017. p. 181–201.

CHENG, R *et al.* Simulation: The past 10 years and the next 10 years. *In*: , 2017. **Proceedings - Winter Simulation Conference**. [*S. l.: s. n.*], 2017. p. 2180–2192.

CHOROMANSKA, Anna; HENAFF, Mikael; MATHIEU, Michael. The Loss Surfaces of Multilayer Networks. *In*: , 2015. **Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)**. [*S. l.: s. n.*], 2015.

CORLU, Canan G.; AKCAY, Alp; XIE, Wei. Stochastic simulation under input uncertainty: A Review. **Operations Research Perspectives**, [*s. l.*], v. 7, n. March, p. 100162, 2020. Disponível em: https://doi.org/10.1016/j.orp.2020.100162.

COURTENAY, Lloyd A.; GONZÁLEZ-AGUILERA, Diego. Geometric morphometric data augmentation using generative computational learning algorithms. **Applied Sciences (Switzerland)**, [*s. l.*], v. 10, n. 24, p. 1–25, 2020.

CYBENKO, G. Approximation by Superpositions of a Sigmoidal Function. **Mathematics of Control, Signals, and Systems**, [*s. l.*], v. 2, p. 303–314, 1989.

DE PAULA FERREIRA, William; ARMELLINI, Fabiano; DE SANTA-EULALIA, Luis Antonio. Simulation in industry 4.0: A state-of-the-art review. **Computers and Industrial**

Engineering, [*s. l.*], v. 149, n. January, p. 106868, 2020. Disponível em: https://doi.org/10.1016/j.cie.2020.106868.

DE SANTIS, Alberto *et al.* Determining the optimal piecewise constant approximation for the nonhomogeneous Poisson process rate of Emergency Department patient arrivals. **Flexible Services and Manufacturing Journal**, [*s. l.*], p. 1–34, 2021.

DOS SANTOS, Carlos Henrique *et al.* Decision support in productive processes through DES and ABS in the Digital Twin era: a systematic literature review. **International Journal of Production Research**, [*s. l.*], v. 60, n. 8, p. 2662–2681, 2022.

DOUDAREVA, Evgueniia; CARTER, Michael. Discrete event simulation for emergency department modelling: A systematic review of validation methods. **Operations Research for Health Care**, [*s. l.*], v. 33, p. 100340, 2022. Disponível em: https://doi.org/10.1016/j.orhc.2022.100340.

DUAN, Yanqing; EDWARDS, John S; DWIVEDI, Yogesh K. International Journal of Information Management Artificial intelligence for decision making in the era of Big Data – evolution , challenges and research agenda. **International Journal of Information Management**, [*s. l.*], v. 48, n. February, p. 63–71, 2019. Disponível em: https://doi.org/10.1016/j.ijinfomgt.2019.01.021.

EROL, Selim *et al.* Tangible Industry 4 . 0 : a scenario-based approach to learning for the future of production. **Procedia CIRP**, [*s. l.*], v. 54, p. 13–18, 2016. Disponível em: http://dx.doi.org/10.1016/j.procir.2016.03.162.

FARAJZADEH-ZANJANI, Maryam *et al.* Generative Adversarial Networks: A Survey on Training, Variants, and Applications. **Intelligent Systems Reference Library**, [*s. l.*], v. 217, n. 8, p. 7–29, 2022.

FLEXSIM. **FlexSim Documentation**. [*S. l.*], 2022. Disponível em: https://docs.flexsim.com. Acesso em: 7 ago. 2022.

FOSTER, David. **Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play**. 1. ed. [*S. l.*]: O'Reilly Media, 2019.

FRANKISH, Keith; RAMSEY, William M. (org.). **The Cambridge Handbook of Artificial Intelligence**. Cambridge: CAMBRIDGE UNIVERSITY PRESS, 2014.

GABRIEL, Gustavo Teodoro *et al.* Good practices and deficiencies in conceptual modelling: A systematic literature review. **Journal of Simulation**, [*s. l.*], v. 16, n. 1, p. 84–100, 2020. Disponível em: https://doi.org/10.1080/17477778.2020.1764875.

GARCIA-GARCIA, J *et al.* Software Process Simulation Modeling : Systematic literature review. **Computer Standards & Interfaces**, [*s. l.*], v. 70, n. August 2019, p. 103425, 2020. Disponível em: https://doi.org/10.1016/j.csi.2020.103425.

GEER MOUNTAIN SOFTWARE. **Stat::Fit**. [*S. l.*], 2022. Disponível em: https://www.geerms.com/Fitting-Distributions.html. Acesso em: 7 ago. 2022.

GEER MOUNTAIN SOFTWARE. **Stat:Fit**. [*S. l.*], 2022. Disponível em:

https://www.geerms.com/Fitting-Distributions.html. Acesso em: 7 ago. 2022.

GOIENETXEA URIARTE, Ainhoa; NG, Amos H.C.; URENDA MORIS, Matías. Bringing together Lean and simulation: a comprehensive review. **International Journal of Production Research**, [*s. l.*], v. 58, n. 1, p. 87–117, 2020. Disponível em: https://doi.org/10.1080/00207543.2019.1643512.

GOODFELLOW, Ian J *et al.* Generative Adversarial Nets. **arXiv**, [*s. l.*], p. 1–9, 2014.

GOODFELLOW, I. Generative Adversarial Networks. *In*: , 2016, Barcelona. **Proceedings of the Thirtieth Conference on Neural Information Processing Systems**. Barcelona: [*s. n.*], 2016.

GREASLEY, Andrew; EDWARDS, John Steven. Enhancing discrete-event simulation with big data analytics: A review. **Journal of the Operational Research Society**, [*s. l.*], v. 72, n. 2, p. 247–267, 2021. Disponível em: https://doi.org/10.1080/01605682.2019.1678406.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2. ed. [*S. l.*]: Prentice Hall, 1998.

HE, Kaiming *et al.* Deep Residual Learning for Image Recognition. *In*: , 2016. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S. l.: s. n.*], 2016. p. 1–9.

HERNANDEZ, Mikel *et al.* Synthetic data generation for tabular health records: A systematic review. **Neurocomputing**, [*s. l.*], v. 493, p. 28–45, 2022. Disponível em: https://doi.org/10.1016/j.neucom.2022.04.053.

JAQUES, P A *et al.* Evaluation of gowns and coveralls used by medical personnel working with Ebola patients against simulated bodily fluids using an Elbow Lean Test. **Journal of Occupational and Environmental Hygiene**, [*s. l.*], v. 13, n. 11, p. 881–893, 2016. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84988416310&doi=10.1080%2F15459624.2016.1186279&partnerID=40&md5=0396465d7fba95de06b4a9dc6220f62e.

JEONG, Jiwoong J. *et al.* Systematic Review of Generative Adversarial Networks (GANs) for Medical Image Classification and Segmentation. **Journal of Digital Imaging**, [*s. l.*], v. 35, n. 2, p. 137–152, 2022. Disponível em: https://doi.org/10.1007/s10278-021-00556-w.

KOLESNYK, A. S.; KHAIROVA, N. F. Justification for the Use of Cohen's Kappa Statistic in Experimental Studies of NLP and Text Mining. **Cybernetics and Systems Analysis**, [*s. l.*], v. 58, n. 2, p. 280–288, 2022.

KOTU, Vijay; DESHPANDE, Bala. **Data Science**. 2. ed. [*S. l.*]: Elsevier, 2019.

KRAUS, Mathias; FEUERRIEGEL, Stefan; OZTEKIN, Asil. Deep learning in business analytics and operations research : Models , applications and managerial implications R. **European Journal of Operational Research**, [*s. l.*], v. 281, n. 3, p. 628–641, 2020. Disponível em: https://doi.org/10.1016/j.ejor.2019.09.018.

KUHL, Michael E *et al.* PROBABILISTIC INPUT PROCESSES FOR SIMULATION. *In*: ,

2008. **Proceedings of the 2008 Winter Simulation Conference**. [*S. l.: s. n.*], 2008. p. 48–61.

KUSIAK, Andrew. Convolutional and generative adversarial neural networks in manufacturing. **International Journal of Production Research**, [*s. l.*], v. 58, n. 5, p. 1594–1604, 2020. Disponível em: https://doi.org/00207543.2019.1662133.

KUSIAK, Andrew. Smart manufacturing. **International Journal of Production Research**, [*s. l.*], v. 7543, p. 1–10, 2018. Disponível em: https://doi.org/10.1080/00207543.2017.1351644.

LAW, Averill M. A Tutorial on How to Select Simulation Input Probability Distributions. *In*: , 2016, Tucson, AZ. **Proceedings of the 2016 Winter Simulation Conference**. Tucson, AZ: [*s. n.*], 2016.

LAW, Averill. **Simulation Modeling and Analysis**. 5. ed. New York: McGraw-Hill Education, 2014.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, [*s. l.*], v. 521, p. 436–444, 2015.

LEEMIS, Larry. SEVEN HABITS OF HIGHLY SUCCESSFUL INPUT MODELERS. *In*: , 1997. **Proceedings of the 1997 Winter Simulation Conference**. [*S. l.: s. n.*], 1997.

LEI, Lei; HU, Jian Qiang; ZHU, Chenbo. Discrete-event stochastic systems with copula correlated input processes. **IISE Transactions**, [*s. l.*], v. 54, n. 4, p. 321–331, 2022. Disponível em: http://dx.doi.org/10.1080/24725854.2021.1943571.

LEMLEY, J.; BAZRAFKAN, S.; CORCORAN, P. Smart Augmentation Learning an Optimal Data Augmentation Strategy. **IEEE Access**, [*s. l.*], 2017.

LI, Haodong *et al.* Identification of deep network generated images using disparities in color components. **Signal Processing**, [*s. l.*], v. 174, p. 107616, 2020. Disponível em: https://doi.org/10.1016/j.sigpro.2020.107616.

LIU, Tianyi; LIN, Yifan; ZHOU, Enlu. A Bayesian Approach to Online Simulation Optimization with Streaming Input Data. *In*: , 2021, Atlanta, GA, USA. **Proceedings of the 2021 Winter Simulation Conference**. Atlanta, GA, USA: [*s. n.*], 2021.

LONGO, Francesco; NICOLETTI, Letizia; PADOVANO, Antonio. Computers & Industrial Engineering Smart operators in industry 4 . 0 : A human-centered approach to enhance operators ' capabilities and competencies within the new smart factory context. **Computers & Industrial Engineering**, [*s. l.*], v. 113, p. 144–159, 2017. Disponível em: https://doi.org/10.1016/j.cie.2017.09.016.

MATPLOTLIB. [*S. l.*], 2022. Disponível em: https://matplotlib.org/. Acesso em: 7 ago. 2022.

MCCULLOGH, Warren; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, [*s. l.*], v. 5, p. 115–133, 1943.

MCHUGH, M L. Interrater reliability: The kappa statistic. **Biochemia Medica**, [*s. l.*], v. 22, n. 3, p. 276–282, 2012. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84867276477&partnerID=40&md5=18d317c55cde90e818e94af7625b7c73.

MEHRA, Akash *et al.* Implicit vs. Explicit Style Transfer? A Comparison of GAN Architectures for Continuous Path Keyboard Input Modeling. **European Signal Processing Conference**, [*s. l.*], v. 2021-Augus, n. 15, p. 1396–1400, 2021.

MEHRA, Akash *et al.* Leveraging GANs to Improve Continuous Path Keyboard Input Models. *In*: , 2020, Cupertino, California. **IEEE International Conference On Acoustics, Speech and Signal Processing - Proceeedings 2020**. Cupertino, California: [*s. n.*], 2020. p. 8174–8178.

MIGUEL, Paulo Augusto Cauchick. Estudo de caso na engenharia de produção: estruturação e recomendações para sua condução. **Produção**, [*s. l.*], v. 17, n. 1, p. 216–229, 2007.

MONTEVECHI, J A B *et al.* Conceptual modeling in simulation projects by mean adapted IDEF: An application in a Brazilian tech company. *In*: , 2010, Baltimore, MD. **Proceedings - Winter Simulation Conference**. Baltimore, MD: [*s. n.*], 2010. p. 1624–1635.

MONTEVECHI, José Arnaldo Barra *et al.* Input Data Modeling: An Approach Using Generative Adversarial Networks. *In*: , 2021. **Proceedings of the 2021 Winter Simulation Conference**. [*S. l.: s. n.*], 2021. p. 1–12.

MONTGOMERY, Douglas; RUNGER, George. **Applied Statistics and Probability for Engineers**. 6. ed. Hoboken: John Wiley and Sons Inc., 2013.

MORGAN, Lucy E *et al.* A Spline-based method for modelling and generating a Nonhomogeneous poisson process. *In*: , 2019. **Proceedings of the 2019 Winter Simulation Conference**. [*S. l.: s. n.*], 2019. p. 356–367.

MOTYL, Barbara *et al.* How will change the future engineers ' skills in the Industry 4 . 0 framework ? A questionnaire survey. **Procedia Manufacturing**, [*s. l.*], v. 11, n. June, p. 1501–1509, 2017. Disponível em: http://dx.doi.org/10.1016/j.promfg.2017.07.282.

MOURTZIS, Dimitris. Simulation in the design and operation of manufacturing systems: state of the art and new trends. **International Journal of Production Research**, [*s. l.*], v. 58, n. 7, p. 1927–1949, 2020.

NADKARNI, Prakash. **Clinical Research Computing: A Practitioner's Handbook**. 1st. ed. London, UK: Academic Press, 2016.

NELSON, Barry L. *et al.* Reducing simulation input-model risk via input model averaging. **INFORMS Journal on Computing**, [*s. l.*], v. 33, n. 2, p. 672–684, 2021.

NIE, Jing *et al.* Prediction of Liquid Magnetization Series Data in Agriculture Based on Enhanced CGAN. **Frontiers in Plant Science**, [*s. l.*], v. 13, n. June, p. 1–14, 2022.

NITANDA, Atsushi; SUZUKI, Taiji. Gradient layer: Enhancing the convergence of adversarial training for generative models. *In*: , 2018, Playa Blanca, Lanzarote, Canary Islands. **21st International Conference on Artificial Intelligence and Statistics, AISTATS 2018**. Playa Blanca, Lanzarote, Canary Islands: [*s. n.*], 2018. p. 1008–1016.

NUMPY. [*S. l.*], 2022. Disponível em: https://numpy.org/. Acesso em: 7 ago. 2022.

OXFORD UNIVERSITY PRESS. **OxfordLanguages**. [*S. l.*], 2022. Disponível em:

https://languages.oup.com/google-dictionary-pt/. Acesso em: 7 ago. 2022.

PANDAS. [*S. l.*], 2022. Disponível em: https://pandas.pydata.org/. Acesso em: 7 ago. 2022.

PEDAMONTI, Dabal. Comparison of non-linear activation functions for deep neural networks on. **arXiv**, [*s. l.*], n. 3, 2018.

PYTHON SOFTWARE FOUNDATION. **PyPi**. [*S. l.*], 2022. Disponível em: www.pypi.org. Acesso em: 7 ago. 2022.

PYTHON SOFTWARE FOUNDATION. **Python 3.10.6 documentation**. [*S. l.*], 2022. Disponível em: https://docs.python.org/3/. Acesso em: 7 ago. 2022.

ROBERTSON, N; PERERA, T. Automated data collection for simulation ?. **Simulation Practice and Theory**, [*s. l.*], v. 9, p. 349–364, 2002.

ROY, Sumanta; VENKATESAN, Shanmugam Prasanna; GOH, Mark. Healthcare services : A systematic review of patient-centric logistics issues using simulation. **Journal of the Operational Research Society**, [*s. l.*], v. 0, n. 0, p. 1–23, 2020. Disponível em: https://doi.org/10.1080/01605682.2020.1790306.

SALMON, Andrew *et al.* Operations Research for Health Care A structured literature review of simulation modelling applied to Emergency Departments : Current patterns and emerging trends. **Operations Research for Health Care**, [*s. l.*], v. 19, p. 1–13, 2018. Disponível em: https://doi.org/10.1016/j.orhc.2018.01.001.

SARGENT, R G. Verification and validation of simulation models. **Journal of Simulation**, [*s. l.*], v. 7, n. 1, p. 12–24, 2013.

SCIKIT-LEARN. [*S. l.*], 2022. Disponível em: www.scikit-learn.org. Acesso em: 7 ago. 2022.

SCIPY. [*S. l.*], 2022. Disponível em: https://scipy.org/. Acesso em: 7 ago. 2022.

SEABORN. [*S. l.*], 2022. Disponível em: https://seaborn.pydata.org/. Acesso em: 7 ago. 2022.

SEYEDZADEH, Saleh *et al.* Tuning of Machine Learning Models for Prediction of Building Energy Loads. **Sustainable Cities and Society**, [*s. l.*], v. 47, p. 1–18, 2019.

SINGH, Dalwinder; SINGH, Birmohan. Investigating the impact of data normalization on classification performance. **Applied Soft Computing**, [*s. l.*], v. 97, p. 105524, 2020. Disponível em: https://doi.org/10.1016/j.asoc.2019.105524.

SKOOGH, Anders; JOHANSSON, Björn. Mapping of Time-Consumption During Input Data Management Activities. **Simulation Notes Europe**, [*s. l.*], v. 19, n. 2, p. 39–46, 2009.

SORIN, Vera *et al.* Creating Artificial Images for Radiology Applications Using Generative Adversarial Networks (GANs) – A Systematic Review. **Academic Radiology**, [*s. l.*], n. 9, 2020.

STACHENFELD, Kimberly L; BOTVINICK, Matthew M; GERSHMAN, Samuel J. The hippocampus as a predictive map. **Nature Neuroscience**, [*s. l.*], v. 20, n. 11, 2018.

TENSORFLOW. [*S. l.*], 2022. Disponível em: www.tensorflow.org. Acesso em: 7 ago. 2022.

THARA, T. D.K.; PREMA, Prema Sudha; XIONG, Fan. Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. **Pattern Recognition Letters**, [*s. l.*], v. 128, p. 544–550, 2019. Disponível em: https://doi.org/10.1016/j.patrec.2019.10.029.

THIS ARTWORK DOES NOT EXIST. [*S. l.*], 2022. Disponível em: thisartworkdoesnotexist.com. Acesso em: 7 ago. 2022.

THIS CAT DOES NOT EXIST. [*S. l.*], 2022. Disponível em: thiscatdoesnotexist.com. Acesso em: 7 ago. 2022.

THIS PERSON DOES NOT EXIST. [*S. l.*], 2022. Disponível em: thispersondoesnotexist.com. Acesso em: 7 ago. 2022.

TOLOSANA, Ruben *et al.* Deepfakes and beyond : A Survey of face manipulation and fake detection. **Information Fusion**, [*s. l.*], v. 64, n. December 2019, p. 131–148, 2020.

TOUTOUH, Jamal; HEMBERG, Erik; O'REILLY, Una May. Spatial evolutionary generative adversarial networks. **GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference**, [*s. l.*], p. 472–480, 2019.

UHLIG, Tobias; ROSE, Oliver; RANK, Sebastian. Evaluation of modeling tools for autocorrelated input processes. *In*: , 2016. **Proceedings of the 2016 Winter Simulation Conference**. [*S. l.: s. n.*], 2016. p. 1048–1059.

VÁZQUEZ-SERRANO, Jesús Isaac; PEIMBERT-GARCÍA, Rodrigo E.; CÁRDENAS-BARRÓN, Leopoldo Eduardo. Discrete-event simulation modeling in healthcare: A comprehensive review. **International Journal of Environmental Research and Public Health**, [*s. l.*], v. 18, n. 22, 2021.

VOLOVOI, Vitali. Simulation of Maintenance Processes in the Big Data Era. *In*: , 2016. **Proceedings of the 2016 Winter Simulation Conference**. [*S. l.: s. n.*], 2016. p. 1872–1883.

WRIGHT, Louise; DAVIDSON, Stuart. How to tell the difference between a model and a digital twin. **Advanced Modeling and Simulation in Engineering Sciences**, [*s. l.*], v. 7, n. 1, 2020. Disponível em: https://doi.org/10.1186/s40323-020-00147-4.

XUE, Yuan *et al.* Deep image synthesis from intuitive user input: A review and perspectives. **Computational Visual Media**, [*s. l.*], v. 8, n. 1, p. 3–31, 2022.

YANG, Li; SHAMI, Abdallah. On hyperparameter optimization of machine learning algorithms: Theory and practice. **Neurocomputing**, [*s. l.*], v. 415, p. 295–316, 2020. Disponível em: https://doi.org/10.1016/j.neucom.2020.07.061.

YE, Yunyang *et al.* Evaluating performance of different generative adversarial networks for large-scale building power demand prediction. **Energy and Buildings**, [*s. l.*], v. 269, p. 112247, 2022. Disponível em: https://doi.org/10.1016/j.enbuild.2022.112247.

YI, Xin; WALIA, Ekta; BABYN, Paul. Generative Adversarial Network in Medical Imaging:

A Review. **arXiv**, [*s. l.*], 2019.

ZHANG, Xiange. Application of discrete event simulation in health care : a systematic review. **BMC Health Services Research**, [*s. l.*], p. 1–11, 2018.

ZHANG, James *et al.* Dynamic time warp-based clustering: Application of machine learning algorithms to simulation input modelling. **Expert Systems with Applications**, [*s. l.*], v. 186, n. June 2020, p. 0–2, 2021.

ZHANG, Daniel *et al.* **The AI Index 2021 Annual Report**. Stanford, CA: [*s. n.*], 2021.

ZHANG, Guoqiang; PATUWO, B Eddy; HU, Michael Y. Forecasting with artificial neural networks : The state of the art. **International Journal of Forecasting**, [*s. l.*], v. 14, p. 35–62, 1998.