UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JOÃO LEONARDO HARRES DALL AGNOL

GERAÇÃO PROCEDURAL DE MELODIAS POR GÊNERO USANDO APRENDIZADO DE MÁQUINA

PATO BRANCO

2022

JOÃO LEONARDO HARRES DALL AGNOL

GERAÇÃO PROCEDURAL DE MELODIAS POR GÊNERO USANDO APRENDIZADO DE MÁQUINA

PROCEDURAL GENERATION OF MELODIES BY GENRE USING MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Érick Oliveira Rodrigues

PATO BRANCO

2022

JOÃO LEONARDO HARRES DALL AGNOL

GERAÇÃO PROCEDURAL DE MELODIAS POR GÊNERO USANDO APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de Aprovação: 07 de dezembro de 2022.

Prof. Me. Vínicius Pegorini
Universidade Tecnológica Federal do Paraná

Prof. Dr. Jefferson Tales Oliva
Universidade Tecnológica Federal do Paraná

Prof. Dr. Érick Oliveira Rodrigues
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2022

RESUMO

A geração procedural de melodias consiste no processo de automatização da composição de sequências de notas musicais. Para a ciência da computação, principalmente nas áreas voltadas ao aprendizado de máquina, esse processo pode ser atingido por meio da amostragem estatística uma vez que o conjunto das tonalidades dos sistemas musicais contemporâneos e as relações entre as notas são organizadas devido suas propriedades físicas e matemáticas. Todavia, devido a natureza estritamente computacional de grande parte das implementações descritas pela literatura atual, a aprovação das peças resultantes tanto por especialistas quanto pelo público geral é ainda muito baixa. Dessa forma, o presente trabalho implementa uma plataforma web que contempla a geração procedural melodias por gênero utilizando Redes Neurais (devido suas capacidades de identificação de tendências, o que lhes tornam aptas a compreenderem e aplicarem regras de múltiplos sistemas musicais), e a avaliação das melodias geradas. Considerando os três gêneros disponibilizados na plataforma, o sistema conseguiu atingir uma avaliação positiva em quase um terço das melodias geradas.

Palavras-chave: melodia; geração; procedural; aprendizado de máquina.

ABSTRACT

The procedural generation of melodies can be understood as the processes through which the composition of musical notes sequences can be automated. For computer science, especially in the field of machine learning, this goal can be achieved through statistical sampling since the set of keys of contemporary musical systems and the relationships between notes are organized due to both their physical and mathematical properties. However, due to the strictly computational nature of most of the implementations described in the current literature, the approval rating of such melodies both by specialists and the general public is still very low. Thus, the present work implements a web platform that contemplates the procedural generation of melodies by genre using Neural Networks (due to their capacity to identify trends, which makes them able to understand and apply rules of multiple musical systems), and the rating of the generated melodies. Considering the three genres available on the platform, the system managed to reach a positive rating in almost a third of the melodies generated.

Keywords: melody; generation; procedural; machine learning.

LISTA DE CÓDIGOS-FONTE

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

GAN	Redes Generativas Adversárias
HTTP	Protocolo de Transferência de Hipertexto
IA	Inteligência Artificial
LSTM	Memória de Longo e Curto Prazo
MIDI	Interface Digital de Instrumentos Musicais
RNN	Redes Neurais Recorrentes
SQL	Linguagem de Consulta Estruturada
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

O processo de geração procedural de melodias não é uma área recente na Ciência da Computação. Até no campo da matemática teórica já foram demonstrados diversos modelos estatísticos capazes da "imitação de estilo" e da "classificação de novas composições" (CONKLIN, 2003). Todavia, o interesse da computação não se limita apenas a confecção (e ao auxilio a confecção) musical, mas também nas oportunidades de exploração estatística que os diversos sistemas musicais oferecem. Além disso, a Ciência de Dados e, em especial, as pesquisas focadas na construção de modelos de Inteligência Artificial (IA), é particularmente dependente de técnicas de amostragem estatística devido a forma como os algoritmos descobrem (ou seja, aprendem) as características determinantes de um certo objeto de estudo, e também como os mesmos geram novos dados a partir de um conjunto de entradas, como pinturas a partir de uma frase, ou músicas a partir de um gênero.

Portanto, o estudo da composição melódica por algoritmos de aprendizagem de máquina pode se mostrar produtivo devido as capacidades de identificação de tendencias destes modelos, possibilitando a construção de ferramentas de apoio ao trabalho criativo, como a geração de melodias sem direitos autorais. Alinhado a esse ponto, a construção de ferramentas de apoio à tomada de decisão também pode facilitar a captação de uma grande quantidade de dados úteis para o treinamento de novos modelos de IA devido, principalmente, aos benefícios trazidos ao usuário pelo sistema.

As Redes Neurais Recorrentes (RNN), na arquitetura Memória de Longo e Curto Prazo (LSTM), são particularmente apropriadas para a finalidade da composição de melodias por serem organizadas em torno da construção de um contexto de dados (ou seja, memória de longo prazo) que pode ser utilizado no processamento de novas entradas. Roberts *et al.* (2018a) e Huang *et al.* (2019) descrevem alguns modelos de IA capazes da composição musical, todavia, apesar do grande potencial da área e das extensas bases de dados focadas na produção musical, o estudo de estilização de composições musicais é ainda relativamente precário, como demonstrado por Li-Chia Yang, Chou e Yi-Hsuan Yang (2017).

Sendo assim, este trabalho propõe a implementação de um sistema de geração de melodias discriminadas por gênero disponibilizado em uma plataforma web com suporte para avaliação do público geral, visando explorar a aplicação de técnicas de aprendizado de máquina no auxílio a composição. Também é prevista a consequente produção de um banco de dados com avaliações manuais das composições geradas pelo algoritmo, de forma que a inteligência deverá

aprimorar suas melodias de acordo com o retorno dos usuários, reduzindo o uso inapropriado dos vieses e tendências de cada gênero.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Implementar um sistema capaz da geração procedural de melodias por gênero musical por intermédio de técnicas de IA, que consiga armazenar os dados das avaliações dos usuários, fornecendo uma plataforma web de apoio a composição musical que também possibilite a coleta de dados relacionados a criação de melodias.

1.1.2 Objetivos Específicos

- Coletar e classificar melodias por gênero que constituirão as bases de dados de treinamento e de testes.
- Aplicar um modelo de Aprendizado de Máquina na geração procedural de melodias por gênero musical.
- Desenvolver uma interface web para interação e avaliação das melodias compostas pelo modelo.

1.2 JUSTIFICATIVA

Para a Ciência da Computação, a pesquisa relacionada a música e, mais especificamente, a geração de procedural de melodias e até mesmo músicas completas se mostra interessante devido tanto às características matemáticas envolvidas na estruturação de melodias - como o agrupamento dos tons musicais em conjuntos discretos de frequências e as diferentes combinações (harmoniosas) derivadas dessa característica dos sistemas musicais contemporâneos - quanto os benefícios de auxílio a tomada de decisão dentro do contexto do trabalho criativo (voltado à produção musical), facilitando a interação do público com a plataforma proposta.

As técnicas de Aprendizado de Máquina são particularmente adequadas para a resolver os problemas da composição musical automatizada devido a dificuldade de confeccionar progressões (de notas musicais) adequadas utilizando métodos puramente randomizados, uma vez que esses

métodos não conseguem comportar as regras e tendencias dos sistemas musicais. A aplicação destas regras e tendencias é importante devido o fato de que a construção de uma melodia bem estruturada depende da aplicação de múltiplos conhecimentos prévios sobre as relações entre os diferentes tons, as progressões, e os ritmos do sistema musical abordado, além da compreensão do contexto da melodia gerada, possibilitando a construção de sequências tonais coerentes.

Dessa forma, o presente trabalho propõe a implementação de um sistema de geração e avaliação de melodias, de código aberto, que pode atuar não apenas como ferramenta de apoio a composição musical, gerando melodias (por gênero) que podem ser utilizadas na construção de peças musicais maiores, como também proporcionar uma plataforma de coleta de dados de treinamento de IAs voltadas à mesma finalidade.

1.3 ESTRUTURA DO TRABALHO

O Capítulo 1 apresenta o trabalho, contendo também os objetivos e a justificativa. No Capítulo 2 encontra-se a fundamentação teórica do projeto com base na literatura dos temas abordados. O Capítulo 3 trata tanto dos materiais utilizados quanto do método empregado, enquanto que o Capítulo 4 apresenta os resultados do trabalho, e o Capítulo 5 contém a conclusão da dissertação. A última parte contém a lista de referências utilizadas.

2 REFERENCIAL TEÓRICO

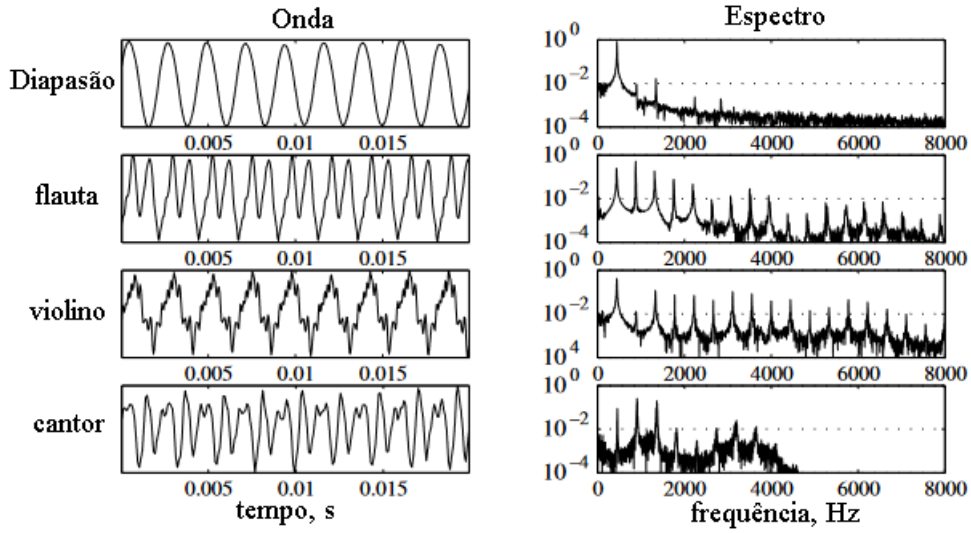
2.1 MELODIA

Para a música ocidental, a melodia pode ser definida como uma sucessão de notas musicais (APEL, 2003), sendo que essa sequência só pode ser classificada como "coerente" (ou afinada) quando todas as notas estão dentro do conjunto de frequências esperadas em relação ao tom da obra (LIMB, 2006), e que uma nota desafinada é aquela cuja frequência não é prevista pelas relações das demais notas da peça. Esta percepção de afinação é comumente atribuída a fatores culturais, como explica Bartlett (1993) ao analisar um comercial de rádio:

> É uma afirmação de que o conhecimento da estrutura tonal das melodias, uma noção do tom em que as melodias são tocadas, é amplamente compartilhado em nossa cultura. Podemos ser incapazes de cantar no "tom", ou tocar uma melodia simples no piano, mas conseguimos identificar uma melodia atonal, ou uma nota desafinada em uma melodia tonal. E não só podemos identificar melodias atonais e notas desafinadas, também parecemos identificá-las automaticamente, sem nada que se assemelhe a uma análise crítica consciente da música. (BARTLETT, 1993, p. 39).

As notas musicais são comumente decompostas em dois elementos principais: tom e duração. Isso significa que cada nota musical é uma onda sonora que ressoa por um determinado intervalo de tempo, entretanto, o tom não é a onda física (com amplitude e frequência) mas sim a percepção atribuída a onda (LYON; SHAMMA, 1996). Como demonstra a Figura 1, as propriedades da onda de uma mesma nota (no caso, Lá) depende de sua fonte, seja esta um instrumento ou não.

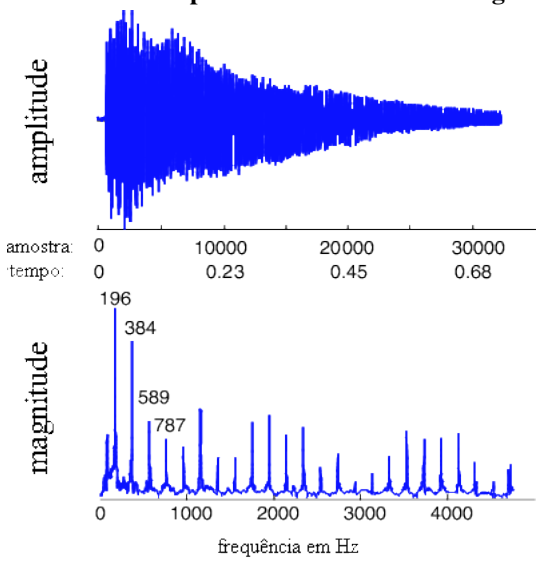
Figura 1 – Análise da nota Lá tocada em instrumentos diferentes (440 Hz)



Fonte: Petersen (2004).

Como explica Petersen (2004), o formato das ondas pode parecer completamente diferente, porém, o intervalo (distância) entre os tons permanece o mesmo. Chamado de harmônica, segundo Sethares (2005) esse fenômeno acontece devido às variações nas vibrações de cada instrumento causadas pelas propriedades materiais do mesmo - como a extensão das cordas, dimensões e material do corpo, entre outras características. As harmônicas podem ser calculadas multiplicando um número inteiro pela frequência fundamental (pura) produzida pelo instrumento (PETERSEN, 2004), sendo que a curva de dissonância formada pelo espectro harmônico do instrumento (conjunto de harmônicas) revela os intervalos escalares de cada tom (SETHARES, 2005), que são nada mais que a relação entre as frequências dos tons. A Figura 2 demonstra o espectro da corda G (Sol) tocada em uma guitarra, sendo que, conforme observa Sethares (2005), a nota ocorre na frequência 196 Hz, e as harmônicas acontecem próximas dos valores 384 Hz, 589 Hz, 787 Hz, etc.

Figura 2 – Onda e espectro da corda G de uma guitarra



Fonte: Sethares (2005).

É notório perceptivamente, essa diferença entre os diversos espectros dos instrumentos é chamada de timbre (SETHARES, 2005), e pode influenciar nas propriedades da onda atribuída a cada nota. Também é notável outro impacto da frequência na percepção de tom: quanto maior a frequência (mais hertz), mais aguda é a nota.

Ao total, as frequências podem ser divididas em sete notas naturais, além das alterações "bemol" (b) e "sustenido" (#), como, por exemplo, o Lá bemol (Ab) e o Lá sustenido (A#). O Quadro 1 lista a relação de notas e nomes como utilizadas pelo sistema de música ocidental, sendo que as inúmeras variações (menor, maior, maior sustenida, entre outras) derivam deste catálogo de notas.

Quadro 1 – Relação das notas naturais e seus respectivos nomes

Nome	Dó	Ré	Mi	Fá	Sol	Lá	Si
Acorde	C	D	E	F	G	A	B

Fonte: Autoria própria (2022).

As propriedades tonais discutidas acima definem a paleta de sons das composições, todavia, uma vez que cada nota precisa ocorrer dentro de um espaço de tempo, a definição de melodia não pode ser isolada do conceito de ritmo. Apel (2003) define ritmo como "tudo relativo às qualidades temporais (duração) do som musical", mas na prática esse conceito é composto por duas propriedades principais: tempo e compasso (GORDON, 1976). O tempo possui um significado físico, pois corresponde às batidas (pulsações) por minuto, enquanto que o compasso se refere à divisão da peça (obra/melodia) em inúmeras porções iguais de batidas. Destas definições deriva à noção de assinatura temporal da música.

A assinatura temporal segue a notação de fração, onde o numerador sinaliza a quantidade de batidas por compasso e o denominador representa a quantidade de batidas que formam uma nota (GORDON, 1976). Essa característica possibilita a composição de diversas melodias de mesma duração com uma quantidade variável de notas independente do tom escolhido. Algumas das assinaturas de tempo mais recorrentes na música ocidental são apresentadas pela Tabela 1.

Tabela 1 – Assinaturas de tempo mais comuns

Assinatura	Tipo	Aplicações comuns
4/4	Simples	Mais popular entre os gêneros musicais
2/2	Simples	Marchas e orquestras
3/4	Simples	Valsas
6/8	Composta	Pop, rock e cantigas
12/8	Composta	Gospel

Fonte: Adaptado de Musical U (2022) e Troiano (2022).

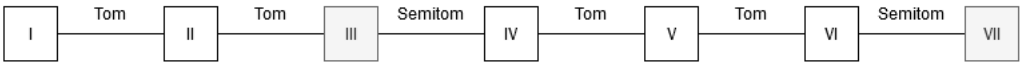
Tendo em vista as definições acima, podem ser elencados quatro pontos referentes à composição de melodias.

1. A melodia precisa de um ritmo.
2. A melodia precisa de uma ou mais notas.
3. Cada nota precisa de uma duração.
4. Para ser coerente, a melodia não pode conter notas desafinadas.

Portanto, o problema da construção de uma melodia coerente pode ser resolvido utilizando as escalas definidas na música ocidental. As escalas são sequências de notas ordenadas de acordo com suas relações, isto é, de acordo com a relação de suas frequências, ou ainda, sua distância musical (TYMOCZKO, 2004). O que isso significa é que o sistema de música

ocidental (assim como outros sistemas) já prevê diversas sequências de tons e semitons que podem formar melodias coesas. As escalas diatônicas, um dos tipos mais comuns encontrados da música ocidental, são formadas por sete notas, totalizando cinco intervalos de um tom, e dois intervalos de um semitom (LEMES, 2020), como apresentado pela Figura 3.

Figura 3 – Estrutura das escalas diatônicas maiores

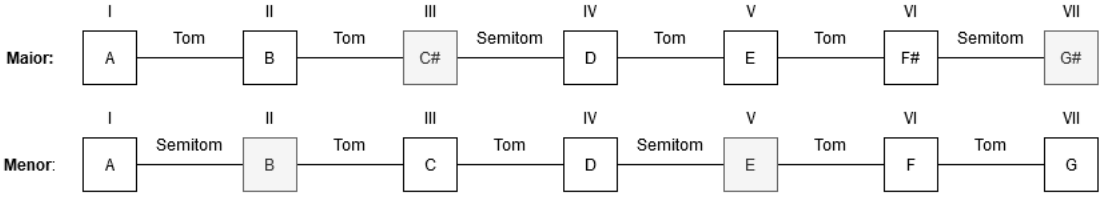


Fonte: Autoria própria (2022).

Segundo Lemes (2020), as escalas diatônicas podem ser classificadas entre maior e menor. Nas escalas maiores a terceira nota, a mediante, é uma terça maior que a nota tônica (primeira nota), enquanto nas escalas menores a mediante é uma terça menor que a nota tônica.

Contrapondo apenas as escalas da Figura 4, observa-se que a mediante da escala menor é exatamente um tom menor que a mediante da escala maior (LEMES, 2020), o que implica na reorganização da sequência de semitons na estrutura das escalas menores, visto que agora estes intervalos ocupam as posições do segundo e quinto tom. Apesar da diferença estrutural ser sútil, a percepção das escalas maiores e menores difere radicalmente uma vez que as melodias compostas com escalas maiores são geralmente reconhecidas como “alegres”, enquanto que as escalas menores formam melodias “tristes” (WILLIMEK, 2014).

Figura 4 – Escalas diatônicas no tom Lá (A) maior e menor



Fonte: Autoria própria (2022).

Como exemplificado pelo experimento de Bartlett e Dowling (1988), para a composição de uma melodia escalar podemos utilizar qualquer sequência de notas desde que o conjunto de todos os tons utilizados seja previsto por pelo menos uma das escalas diatônicas. Outra consideração importante é a de que, geralmente, as melodias mais “agradáveis” são concluídas pela nota inicial, e isso significa que dentro de uma única escala existem múltiplas possibilidades de melodias, sequencialmente distintas, para cada de suas notas. Todavia, apesar de ser possível a inicialização da sequência com qualquer nota da escala selecionada, é mais usual começar uma melodia escalar pela nota tônica. Por fim, para que seja formada uma melodia satisfatória é

preciso trabalhar dentro do conceito de harmonia tonal, ou seja, se faz necessária a composição com acordes.

Apel (2003) define harmonia como “a estrutura cordal (ou vertical) de uma composição musical”, a implicação disso é inversa à melodia: a harmonia trabalha com notas tocadas simultaneamente (empilhadas), enquanto que a melodia se preocupa apenas a organização horizontal das notas (enfileiradas). Dessa definição deriva o conceito de acorde, “O som simultâneo de três ou mais tons” (APEL, 2003), e portanto, tem-se que o estudo da harmonia tonal, também chamado de análise harmônica, consiste no estudo dos acordes ou harmonias contidas em uma progressão musical, ou até em composições musicais completas (APEL, 2003).

Mais especificamente, o campo da harmonia tonal define quatro tipos de tríades dentro das escalas (DEVOTO, 1998):

1. Maior: Composta pela tônica, terça maior, e quinta justa (intervalo de dois tons).
2. Menor: Composta pela tônica, terça menor, e quinta justa.
3. Aumentada: Composta pela tônica, terça maior e quinta aumentada (intervalo de quatro tons).
4. Diminuta: Composta pela tônica, terça menor e quinta diminuta (intervalo de três tons).

Na música ocidental, tríades cuja a nota tônica se encontra no conjunto dos tons I, IV e V são as mais frequentes (tonais), seguidas das tríades dos tons II, III e VI (modais), sendo VII a tônica menos frequente (APEL, 2003). Apesar da movimentação entre as tríades não ser restrita a uma regra específica, padrões de movimentação costumam definir diferentes gêneros musicais, como exibe a Tabela 2.

Tabela 2 – Progressões comuns no Jazz

Progressão	Nome popular	Exemplo
V-I (or V-i)	"Five-One"	G7 - C
ii-V	"Two-Five"	Dm - G7
ii-V-I	"Two-Five-One"	Dm - G7 - C
I-IV	"One-Four"	C - F
IV-iv	"Nostalgia Chord"	F - Fm

Fonte: adaptada de Walsh (2020).

Organizadas as escalas em conjuntos matemáticos discretos e, ainda mais, mapeadas as progressões de acordes mais corriqueiras, a composição de uma melodia escalar com ou sem gênero acaba se assimilando muito a geração de músicas completas. Isso acontece devido ao processo estatístico que pode ser aplicado na seleção de notas, ou partes no caso da música, afinal, como afirma Conklin (2003), “(...) a geração de músicas é equiparada ao problema da

amostragem de um modelo estatístico, ou equivalentemente, a exploração de um espaço de busca com um modelo estatístico usado para avaliação". Essa observação se torna válida uma vez que, como demonstrado, é possível calcular a sequência de notas mais provável dado um intervalo de tempo de uma música, ou ainda, dado uma escala e uma nota inicial.

2.2 INTELIGÊNCIA ARTIFICIAL

De modo geral, este trabalho compreende IA como programas de computador capazes de imitar a inteligência humana, isto é, máquinas que passam no teste de Turing (SCHUETT, 2019). Todavia, na área de IA o sentido atribuído a palavra "inteligência" é mais restrito do que o significado comumente associado à palavra. Como definida por Dicio (2022), inteligência diz respeito à "faculdade de conhecer, compreender e aprender", porém para a IA pode ser considerada apenas como a capacidade de resolução de problemas. Mais especificamente, grande parte do trabalho contemporâneo na área de Aprendizado de Máquina é focado no que se chama de "IA estreita", ou seja, focada apenas na resolução de um problema específico (como a classificação de melodias por gênero), ao invés de constituir uma "inteligência geral" (PENNACHIN; GOERTZEL, 2007).

Esta distinção fica mais evidente ao considerar como as tecnologias de IA resolvem a questão do aprendizado: apesar das diversas tentativas de simular o cérebro humano, alguns dos métodos mais populares de IA são baseados em teorias estatísticas, sendo que a máquina não tanto "aprende" a resolver problemas quanto descobre a melhor configuração de pesos para os parâmetros de uma longa função com inúmeras dimensões (isto é, ao menos em grande parte de implementações de Redes Neurais), capaz de prever uma distribuição radicalmente próxima à distribuição real. Como explica Ghahramani (2015):

> A principal ideia por trás da estrutura probabilística para Aprendizado de Máquina é que a aprendizagem pode ser pensada como inferência de modelos plausíveis para explicar os dados observados. (...) Para fazer inferências sobre dados não observados partindo dos dados observados, o sistema de aprendizado precisa fazer algumas suposições; reunidas, essas suposições constituem um modelo. Um modelo pode ser muito simples e rígido, como um modelo clássico de regressão linear estatística, ou complexo e flexível, como uma rede neural grande e profunda, ou mesmo um modelo com muitos parâmetros. (GHAHRAMANI, 2015, p. 452).

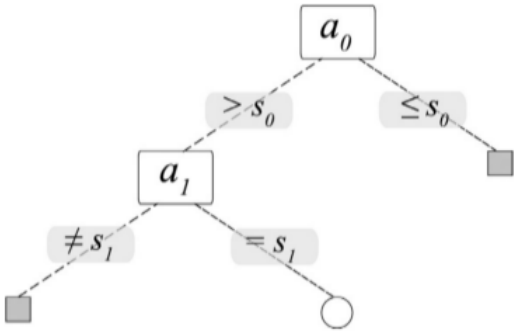
Sendo a IA um modelo matemático complexo capaz de traçar as relações entre os dados e fazer inferências partindo desse "conhecimento", ou seja, capaz de descrever previsões estatísticas, fica claro que o processo de Aprendizado de Máquina é o esforço matemático no qual

estas relações são descobertas - ou seja, o aprendizado sobre dados e experiências. A distinção importante a ser feita aqui é de que as tecnologias contemporâneas de IA imitam a inteligência humana sem a simular, como exemplificam Pennachin e Goertzel (2007), as IAs precisam apenas simular (ou ainda, dar a impressão de) uma inteligência humana sem copiar seus processos.

2.2.1 Classificação e Reconhecimento de Padrões

O problema da classificação de dados por meio da identificação de padrões (como, por exemplo, a identificação de um objeto em uma imagem a partir de suas características visuais), é um dos principais aspectos dos algoritmos de Aprendizado de Máquina, afinal, a inteligência deve ser capaz de tirar conclusões e fazer inferências sobre os dados analisados. Esta tarefa pode ser realizada por múltiplas abordagens distintas, incluindo desde algoritmos mais simples, como árvores de decisão, até redes neurais complexas, porém, para grande parte das implementações da literatura a ideia aplicada permanece a mesma: a decomposição do problema em problemas menores (NIELSEN, 2015), ou ainda, a fragmentação do questionamento geral "a que classe pertence este dado?" em perguntas cada vez mais específicas.

Figura 5 – Árvore de decisão



Fonte: (RODRIGUES; CONCI; LIATSIS, 2018).

As árvores de decisão demonstram essa característica de forma particularmente visual devido a sua estrutura hierárquica, na qual a classificação se inicia com uma pergunta geral, também chamada de nó inicial ou raiz, e escorre até os nós folhas que são as classes resultantes. Para fazer esse caminho, o dado precisa percorrer uma sequência de nós que representam, cada um, perguntas distintas (acerca de seus atributos) que podem ser respondidas com "sim ou não" (KINGSFORD; SALZBERG, 2008). A estrutura de uma árvore de decisão genérica pode ser vista na Figura 5, onde os nós de decisão recebem o rótulo a_n e as conexões (galhos) s_n representam

uma condição (igual, maior, diferente, etc). Ao final o objeto recebe uma classe, representada pelos círculos e quadrados (que não dão origem a nenhum outro nó folha).

Outra abordagem notável é a classificação via Naive Bayes, que aplica os cálculos da soma e produto (GHAHRAMANI, 2015) para definir a probabilidade de um elemento pertencer a uma classe y dado a probabilidade de cada entrada de seu vetor de atributos x (RISH *et al.*, 2001).

$$P(x|y) = \Pi_{(i=1)}^{n} P(x_i|y) \tag{1}$$

Sendo x_i o valor i do vetor x, a probabilidade da classe y é obtida por meio do produto das probabilidades de y dado cada um dos atributos de x. De acordo com Rish *et al.* (2001), apesar das suposições impostas pelo método, "Naive Bayes provou ser eficaz em muitas aplicações práticas, incluindo a classificação de texto, o diagnóstico médico, e o gerenciamento de desempenho de sistemas".

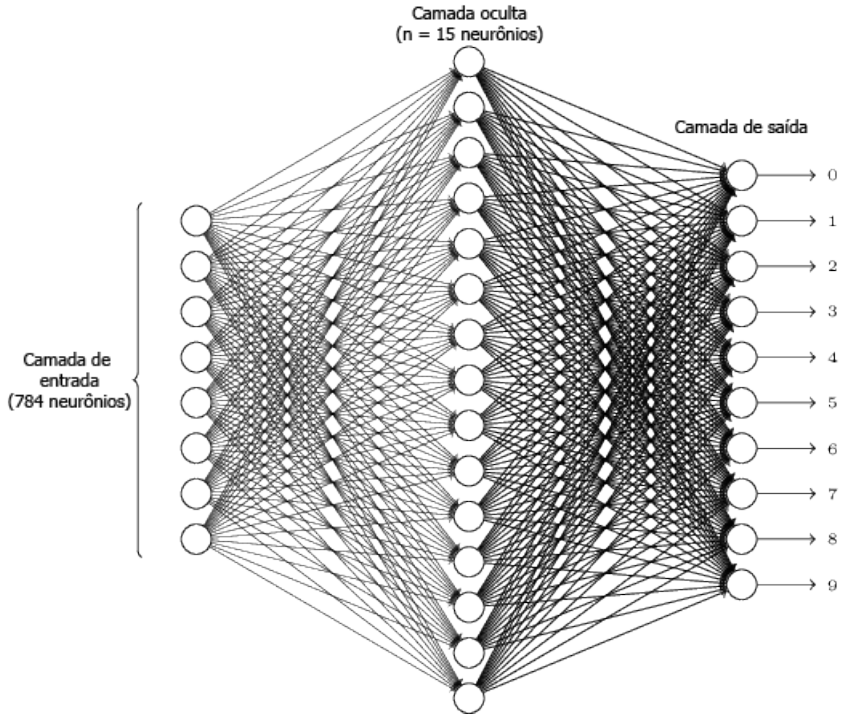
Em contrapartida, a estrutura sequencial dos métodos discutidos acima, as Redes Neurais são um tipo de classificador que utiliza diversas camadas sequenciais compostas por múltiplos neurônios paralelos que tomam diferentes decisões sobre os dados processados. As camadas de neurônios (ou nós) recebem múltiplos valores de entrada (um por nó de entrada), porém a rede retorna apenas um único resultado (NIELSEN, 2015), que identifica a classe do elemento analisado. Esses nós podem aplicar diferentes funções de ativação, como as funções diferenciáveis sigmoide (HAN; MORAGA, 1995) que transformam uma entrada (numérica) em um valor entre 0 e 1. Além disso, as camadas de nós podem ser estruturadas de diferentes maneiras, como a arquitetura RNN (HOCHREITER; SCHMIDHUBER, 1997) descrita na Sub-Seção 2.2.2, entretanto, as principais características dos nós destes classificadores permanecem inalteradas:

- Peso: Cada relação (ou sinapse) entre neurônios possui um peso w atribuído (RODRIGUES; CONCI; LIATSIS, 2018), sendo que algumas relações podem interferir mais ou menos no resultado. Em razão disso, os pesos determinam quais entradas são mais importantes para a assertividade da predição.
- Limiar: Para que a ativação de um neurônio seja válida, o resultado de seu cálculo precisa exceder um limite t (NIELSEN, 2015).
- Viés: Mede a facilidade com que o neurônio é ativado, definido como $b = -t$ (NIELSEN, 2015).

A proposta da classificação via Redes Neurais é treinar o modelo de forma que

a configuração de pesos e limites produza as predições com a menor taxa de perda (erro) (NIELSEN, 2015). Em outras palavras, o treinamento da rede deve conduzi-la a atribuir os maiores pesos para as conexões de nós mais importantes, ou ainda, que melhor definem a classe. A estrutura dessa abordagem é apresentada pela Figura 6, no caso, a IA foi projetada para reconhecer dígitos manuscritos e recebe como entrada uma matriz de inteiros que corresponde uma figura preta (valor 1) e branca (valor 0) de 784 pixels.

Figura 6 – Estrutura de uma rede neural



Fonte: (NIELSEN, 2015).

Portanto, os 784 neurônios de entrada (um para cada pixel) fazem 15 conexões cada (um por neurônio da camada oculta), sendo que cada neurônio da camada de saída recebe um valor entre 0 e 1 de acordo com as conexões ativas (a soma de todos os neurônios de saída é 1). Como determina Nielsen (2015), para essa rede um neurônio de entrada se torna ativo apenas se o valor recebido é 1, logo o valor de cada nó de saída corresponde as transformações aplicadas pela camada oculta nos valores recebidos por meio de suas conexões com os neurônios de entrada. Por sua vez, o resultado final é determinado pelo neurônio de saída que recebeu a maior probabilidade.

Segundo Rodrigues, Conci e Liatsis (2018) a relação entre as camadas de entrada e saída são descobertas pelas Redes Neurais por meio da junção de três elementos: a representação das camadas intermediárias, as funções de ativação dos neurônios, e os pesos atribuídos para as conexões entre os neurônios. Desse modo, a saída final de uma Rede Neural é definida não

somente pelos caminhos percorridos entre a entrada e a saída, como também pelo peso total de cada trajeto. Isso significa que a entrada de uma classe qualquer pode ativar diversos caminhos simultaneamente, porém a saída deve ser única, e portanto é reconhecida como o caminho ativo com maior peso.

Sendo assim, a fórmula de ativação de um nó de saída (não necessariamente da última camada) é descrita como:

$$a^{(1)} = \sigma(Wa^{(0)} + b) \tag{2}$$

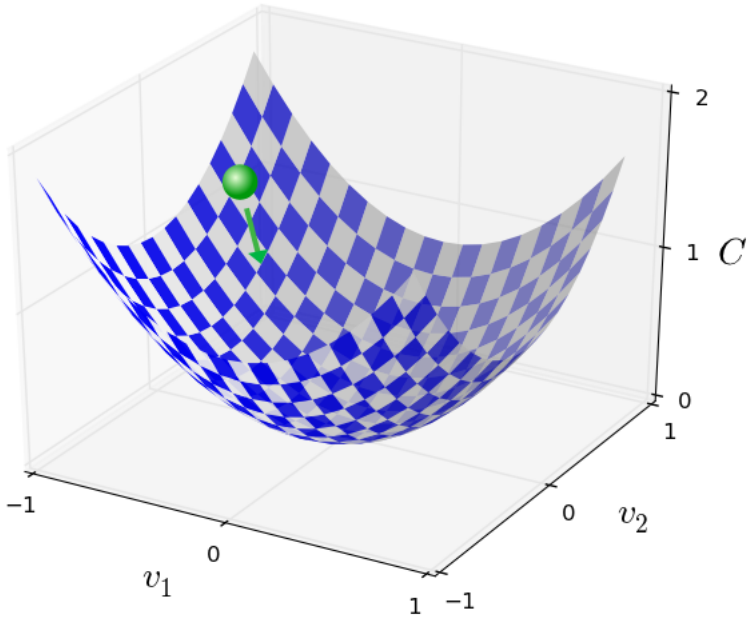
Onde σ representa a função de ativação do neurônio, W representa a matriz de todos os vetores de pesos w das conexões da penúltima camada com o neurônio de saída, $a^{(0)}$ o vetor das ativações da camada, e b representa o vetor dos vieses de cada neurônio (NIELSEN, 2015).

Como os pesos e vieses são fatores determinantes na construção do resultado retornado pelas Redes Neurais, o processo de treinamento (aprendizado) consiste no ajuste dos pesos e vieses de acordo com a média de erro da IA dado seu conjunto de decisões ao receber dados de treinamento. Segundo Géron (2019), funções de perda/custo (C) como a Raiz do Erro Quadrático Médio (RMSE) são utilizadas para o propósito de calcular a perda do modelo, contabilizando a diferença entre o resultado esperado e o resultado recebido.

O objetivo do treinamento de uma Rede Neural é, portanto, a minimização de C. Em razão disso, se faz necessária a aplicação um algoritmo de otimização conhecimento como "gradiente descendente", que tem como função orientar as alterações nos parâmetros da IA de forma a reduzir o resultado de C (GÉRON, 2019). Uma característica importante do gradiente descendente é que as alterações executadas nos pesos e vieses são executadas ao longo de "passos", ou seja, de forma iterativa. Portanto, o Gradiente Descente não descreve uma descida direta ao resultado mínimo de C, mas sim uma progressão em etapas que pode ou não atingir o mínimo valor possível.

A Figura 7 ilustra esse algoritmo representando o universo de resultados de C como um vale. O gradiente deve guiar as alterações na IA (representado pelo vetor) para que o ponto mais baixo possa ser atingido, sendo seus cálculos determinados pela "taxa de aprendizado" recebida (GÉRON, 2019). Esse hiper-parâmetro repassado a IA determina a intensidade dos passos tomados pelo gradiente, e pode causar dois problemas: uma taxa de aprendizagem muito baixa força a execução de muitas iterações para atingir um resultado agradável, enquanto que uma taxa de aprendizagem muito alta pode fazer com que as alterações ultrapassam o ponto

Figura 7 – Gradiente descendente



Fonte: (NIELSEN, 2015).

mínimo sem nunca atingí-lo.

Outro ponto importante é a existência de “pontos mínimos locais” que podem causar flutuações no valor de C, que podem ser erroneamente interpretadas como resultado de uma taxa de aprendizagem muito alta. De forma a acelerar o processo do gradiente, e reduzir (mas não solucionar) o impacto dos pontos mínimos locais, Nielsen (2015) sugere o uso de gradientes descendentes estocásticos.

Os gradientes descendentes estocásticos calculam seus passos com base em instâncias aleatórias dos dados de treinamento (GÉRON, 2019), o que ocasiona pequenas flutuações que acabam por resultar em uma configuração de pesos e vieses não ideais que, ainda assim, produzem bons resultados. Isso significa que o processo de treinamento ocorre mais rápido, mas ainda não é possível garantir que o ponto mínimo absoluto será encontrado.

$$w_k \rightarrow w_k^{'} = w_k - \frac{n}{m}\Sigma_j \frac{\partial C_{x_j}}{\partial_{wk}} \tag{3}$$

$$b_k \rightarrow b_k^{'} = b_k - \frac{n}{m}\Sigma_j \frac{\partial C_{x_j}}{\partial_{bk}} \tag{4}$$

A fórmula do gradiente descendente estocástico de pesos é dada pela Equação 3, enquanto a Equação 4 descreve o cálculo referente aos vieses. Dessa forma, as alterações determinadas pelo gradiente são calculadas com base na soma dos produtos das derivadas parciais de C, considerando a taxa de aprendizagem n para a quantidade de instâncias selecionadas m, e o parâmetro que esta sendo ajustado (w ou b). Segundo Nielsen (2015), o cálculo das derivadas parciais (de pesos e vieses) de C é executado pelo algoritmo de retro-propagação aplicado no modelo.

Como explicam Goodfellow, Bengio e Courville (2016), o fluxo normal de uma Rede Neural é direto (avança para frente entre as camadas de neurônios), já a retro-propagação "permite que a informação do custo flua para trás através da rede para calcular o gradiente". Dessa forma, a retro-propagação pode ser definida como o processo de cálculo do gradiente através da aplicação da função de custo em uma camada de neurônios i para descobrir a diferença entre as entradas esperadas por i e as entradas recebidas da camada $i - 1$.

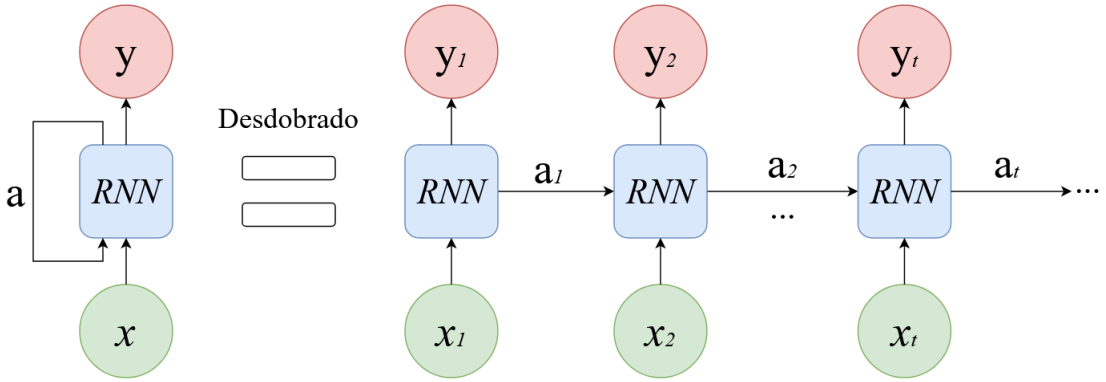
Sendo assim, a retro-propagação se inicia na camada de saída e é executada até a camada de entrada, realizando o fluxo contrário das redes de "alimentação direta".

2.2.2 Redes Neurais Recorrentes e Memória de Longo e Curto Prazo

As RNN são modelos de IA voltados a estimativa baseadas em contexto. Hochreiter e Schmidhuber (1997) definem as RNNs como Redes Neurais que utilizam conexões de retorno (*feedback*) para construir o contexto de suas entradas recentes considerando as ativações causadas por elas. Sendo assim, este contexto pode ser utilizado pela rede para realizar previsões dentro de um processo dito "cíclico" devido sua natureza de reutilização das ativações causadas pelas entradas passadas como parte das novas entradas recebidas (SAK; SENIOR; BEAUFAYS, 2014).

Essas conexões de retorno permitem ao modelo RNN trabalhar não somente com os pesos e vieses comuns às Redes Neurais gerais (memória de longo prazo), como também manter controle do contexto da entrada recebida (memória de curto prazo). O fluxo pelo qual uma RNN constrói seu contexto é demonstrado pela Figura 8, onde o vetor de ativações a é produzido pelas entradas x, e modificado ao longo do tempo t de acordo com as saídas y. Dessa forma, a cada ciclo de execução a RNN mantém uma coleção de dados sobre as ativações das entradas anteriores que podem ser influenciar o processamento de entradas futuras, uma vez que estas são compreendidas dentro de um contexto de informações (como a progressão de tons de uma melodias).

Figura 8 – Fluxo de uma RNN



Fonte: Autoria própria (2022).

É importante ressaltar que o vetor a é repassado como entrada para as execuções futuras da rede, sendo utilizado no cálculo das saídas futuras até que todas as entradas x sejam processadas. Afshine Amidi e Shervine Amidi (2022) exemplificam este processo na Equação 5, na qual W_{ya} e b_y representam os coeficientes de pesos e vieses que compõem o contexto de entradas, saídas e ativações da rede, enquanto g_2 se refere a função de ativação utilizada.

$$y^t = g_2(W_{ya}a^t + b_y) \tag{5}$$

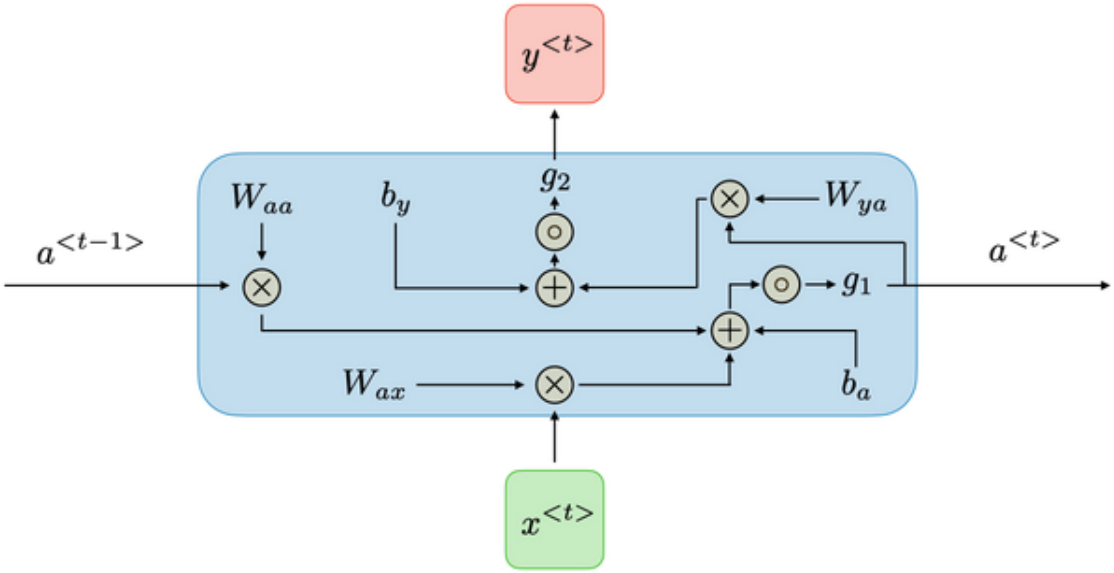
Portanto, y é o produto da ativação causada pela soma das ativações atuais e passadas. Já a pode ser calculado através da Equação 6, que o define como o valor resultante da ativação gerada pela soma entre três coeficientes: de ativações passadas, da ativação atual, e o coeficiente de vieses atual.

$$a^t = g_1(W_{aa}a^{t-1} + W_{ax}x^t + b_a) \tag{6}$$

Destas equações deriva a estrutura interna das RNNs, demonstrada pela Figura 9. Destaca-se na imagem a interação dos coeficientes e vieses na determinação da saída (resultado) y, e no vetor a repassado a nova execução $t + 1$. Seguindo o fluxo de informações, a entrada x obtida no tempo t (que, portanto, configura uma parcela da entrada completa repassada ao modelo), é transformada pela multiplicação com o vetor de pesos W_{ax}, e com o vetor de pesos W_{aa} (transformado pelo "contexto" contido nas ativações $a^{<t-1>}$). O produto destas operações é repassado a função de ativação $g1$, compondo o novo vetor de ativações $a^{<t>}$ ao mesmo tempo que, após transformado pela multiplicação com o vetor de pesos W_{ya} e somado ao viés b_y, o resultado de $g1$ é repassado a função de ativação $g2$, produzindo a saída y do tempo t ($y^{<t>}$).

Apesar das possíveis aplicações de RNNs em diversos campos da computação, como

Figura 9 – Estrutura de uma RNN



Fonte: (AMIDI, A.; AMIDI, S., 2022).

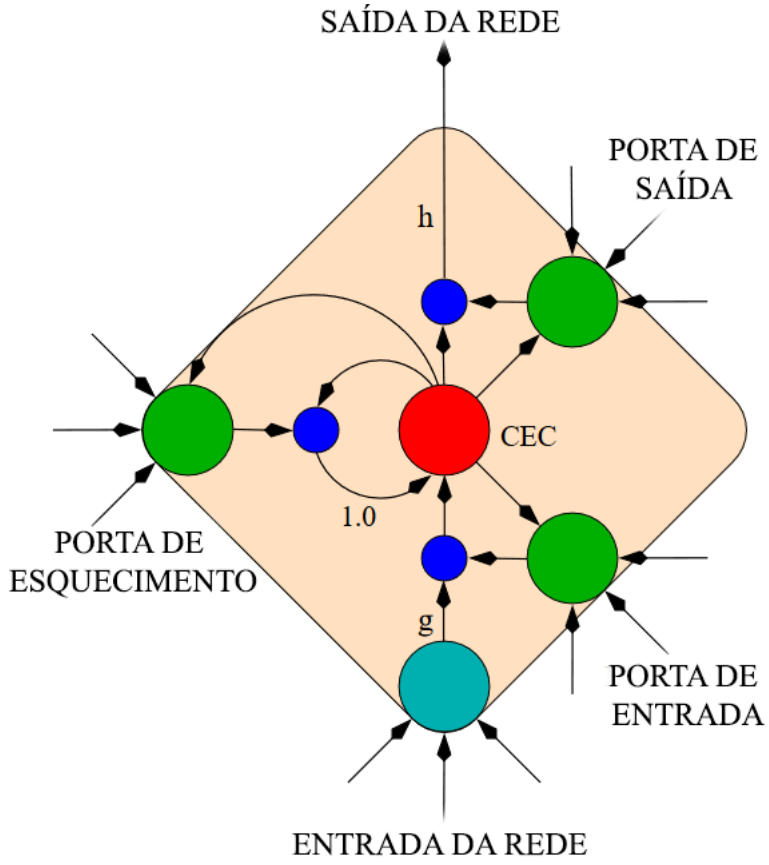
o processamento de imagens, reconhecimento de voz, e até a composição musical. Hochreiter e Schmidhuber (1997) observam que "os algoritmos (mais populares) que definem o que deve ou não ser mantido na memória de curto prazo são muito lentos ou não funcionam". Isso se deve a forma pela qual esses algoritmos lidam com a retro-propagação, fazendo com que os sinais de erro estourem (tende ao infinito) ou sumam (tende a zero) com a passagem do tempo. Como solução para o problema da retro-propagação, Hochreiter e Schmidhuber (1997) propõe a arquitetura LSTM, que força o fluxo constante de erro por meio do uso de blocos de memória.

Segundo Graves e Schmidhuber (2005), os blocos de memória LSTM são compostos de diversas células de memória e três unidades multiplicativas (entrada, saída, e esquecimento), que atuam como portas para a comunicação da rede com as células. O objetivo da comunicação via portas é, justamente, "proteger o conteúdo das células de memória de perturbações causadas por entradas (ou saídas) irrelevantes" (HOCHREITER; SCHMIDHUBER, 1997).

Como ilustrado pela Figura 10, o funcionamento interno de uma LSTM (nesse caso com uma única célula de memória) se baseia na multiplicação das entradas e saídas pelas suas respectivas portas (onde g e h são funções de ativação). Além disso, o fluxo de erro é mantido pelo chamado "carrossel de erro contínuo" (CEC), que é o processo de aplicação da porta de esquecimento no estado interno da célula (GRAVES; SCHMIDHUBER, 2005).

Sak, Senior e Beaufays (2014) explicam que é através desse processo de limpeza do estado da célula antes do recebimento da entrada recorrente que a arquitetura LSTM impede o sumiço do gradiente (assim como são amenizadas as chances de ocorrência de um estouro).

Figura 10 – Estrutura de um bloco LSTM



Fonte: Adaptado de Graves e Schmidhuber (2005).

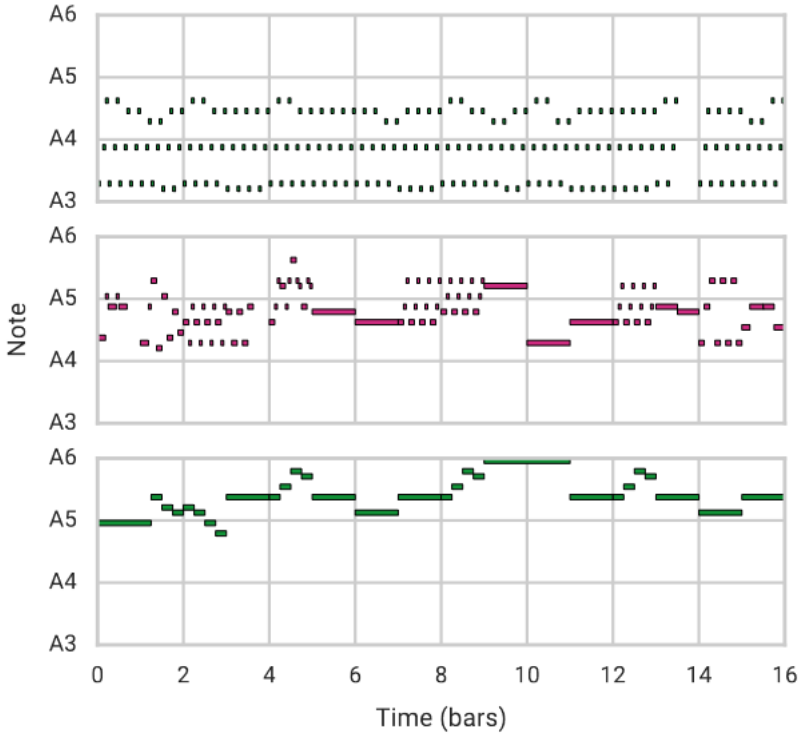
2.2.3 LSTM Bidirecionais na Composição de Músicas

Confirmando as expectativas de Hochreiter e Schmidhuber (1997) quanto as capacidades das LSTM na composição de melodia, Roberts *et al.* (2018a) propõe o modelo MusicVAE que utiliza um conjunto de redes LSTM bidirecionais para replicar os dados recebidos no treinamento (autocodificador), assim como gerar novas sequências similares.

Redes LSTM Bidirecionais são, essencialmente, um par de Redes Neurais que trabalham sobre uma mesma entrada em direções opostas (SCHUSTER; PALIWAL, 1997). Isso significa que enquanto a primeira rede fará um fluxo de tempo positivo (do início ao fim da entrada), a segunda rede executará um fluxo de tempo negativo (do fim ao início), assim permitindo que a LSTM faça uso de conhecimentos futuros - presentes no fim da entrada recebida - para realizar suas previsões.

Segundo Roberts *et al.* (2018a), a composição musical pode ser executada através da utilização de um autocodificador bidirecional e um decodificador hierárquico, resolvendo o problema do espaço latente frequentemente encontrado em outros autocodificadores variáves

Figura 11 – Demonstração de uma peça gerada por um modelo MusicVAE



Fonte: (ROBERTS *et al.*, 2018a).

(VAE). Como demonstra a Figura 11, o modelo proposto permite a composição de músicas em múltiplas faixas (uma por instrumento).

No modelo MusicVAE, assim como nos VAE tradicionais, o autocodificador bidirecional processa uma sequência de entrada $x = x_1, x_2,..., x_T$ para gerar um estado h_T. Todavia, diferindo de outros VAE, esse estado é então repassado a duas camadas LSTM que calculam dois parâmetros de distribuição, μ (Equação 7) e θ (Equação 8) onde W e b representam as matrizes de peso e vetores de viés (ROBERTS *et al.*, 2018a).

$$\mu = W_{h\mu} h_T + b_\mu \tag{7}$$

$$\theta = \log \exp W_{h\theta} h_T + b_\theta + 1 \tag{8}$$

Utilizando o contexto provido por μ e θ, o decodificador hierárquico RNN tem a função de produzir uma sequência de saída $y = y_1, y_2,..., y_T$. Para realizar essa tarefa, Roberts *et al.* (2018a) delegam um "condutor" LSTM para a tarefa de geração de vetores $c = c_1, c_2,..., c_U$ baseados nos parâmetros de distribuição recebidos, onde U são subdivisões arbitrárias (sem sobreposição) da sequência de saída esperada. Os vetores c são então utilizados pela camada final do decodificador RNN para construir o retorno do modelo.

Dessa forma, o decodificador hierárquico RNN proposto por Roberts *et al.* (2018a) é composto de uma camada LSTM "condutora", que irá receber os parâmetros μ e θ gerados pelo codificador, e uma camada LSTM "decodificadora", que produzirá uma sequência de saída y com base nos múltiplos vetores c repassados pelo condutor. Portanto, a geração de novas sequências musicais se torna possível uma vez que um ruído é repassado ao modelo.

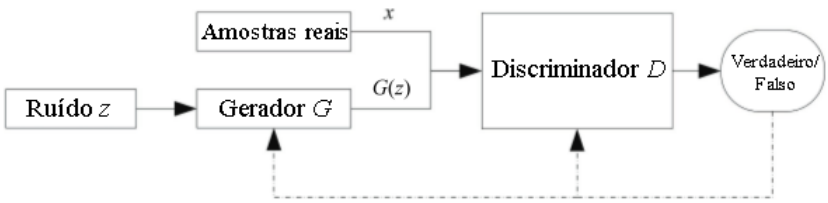
2.2.4 Rede Adversária Generativa

Redes Generativas Adversárias (GAN) é um *framework* de Redes Neurais para estimação estatística voltado para a criação de novos dados. Como proposto por Goodfellow, Pouget-Abadie *et al.* (2014), "o modelo generativo se opõe a um adversário: um modelo discriminativo que aprende a determinar se uma amostra é da distribuição do modelo ou da distribuição de dados". Dessa maneira, as redes GANs são ditas adversárias pois contrapõe duas redes neurais com objetivos opostos:

- Discriminador (D): Tem como objetivo diferenciar os dados reais dos dados falsos produzidos pelo gerador. Esta rede se especializa em detecção de falsificações e subsequente classificação das entradas.
- Gerador (G): A rede generativa tem como objetivo enganar o discriminador. Esta rede se especializa na "falsificação" dos dados.

O fluxo base das GANs, no qual D classifica uma sequência de amostras reais e falsas (geradas por G) para a condição "amostra real?", pode ser acompanhado na Figura 12. Neste caso, assim como em grande parte das GANs, a única entrada do modelo é um ruído z, que tem como objetivo randomizar o resultado produzido por G. Junto a produção da rede geradora, uma amostra real é resgatada da base, sendo que a rede D atribui um valor binário (verdadeiro ou falso) a cada entrada recebida, ou seja, marca a amostra como real (verdadeira, proveniente da base real) ou falsa (falsificada por G).

Figura 12 – Fluxo de uma GAN



Fonte: (WANG, K. *et al.*, 2017).

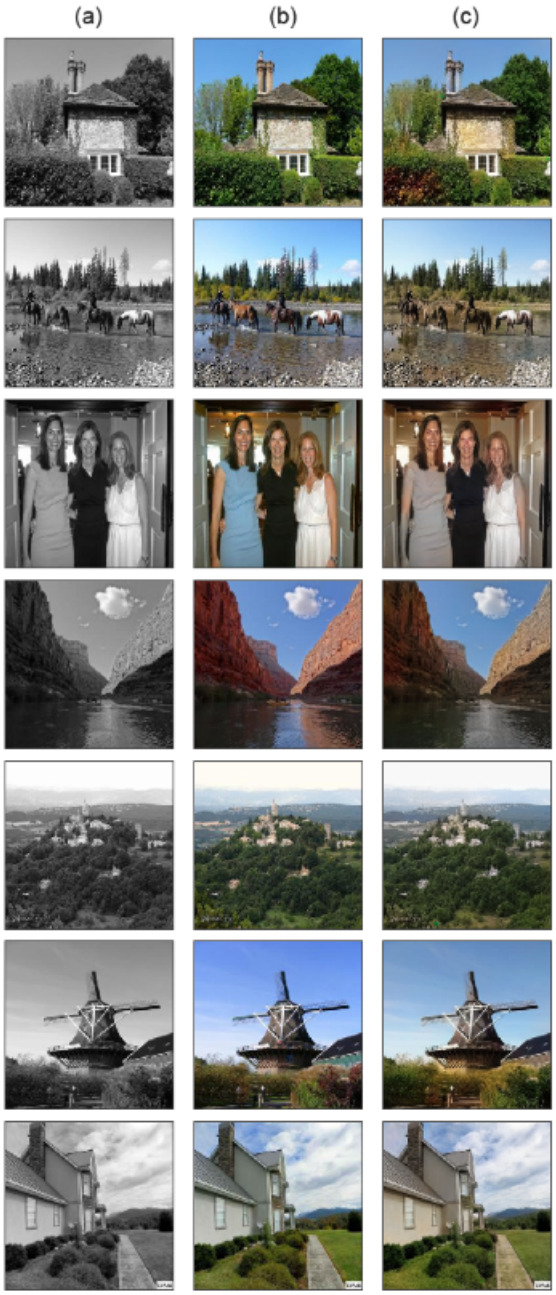
Segundo Kunfeng Wang *et al.* (2017), o processo de otimização de uma GAN é análogo ao método minimax, onde os jogadores (no caso, D e G) optam pela redução da perda. O algoritmo da competição minimax das GANs é descrito como:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \tag{9}$$

Onde p_g é a distribuição dos dados x, z representa o parâmetro de ruído, G (função $G(z;\theta_g)$) e D (função $D(x;\theta_d)$) são as funções discriminatórias das duas redes multicamadas de nós (G e D). Sendo $D(x)$ a probabilidade de que x veio da distribuição real ao invés de p_g, durante todo processo de treinamento D tenta maximizar sua capacidade de discriminação (GOODFELLOW; POUGET-ABADIE *et al.*, 2014), enquanto que G inicialmente assume o objetivo de maximizar $\log D(G(z))$ para depois minimizar $\log(1 - D(G(z)))$. Isso se deve ao fato de G começar com "falsificações" ruins, e essa troca inicial "resulta no mesmo ponto da dinâmica de G e D, mas fornece gradientes muito melhores no início do aprendizado" (GOODFELLOW; POUGET-ABADIE *et al.*, 2014). É importante ressaltar que a decisão final do discriminador é sempre binária, o dado só pode ser classificado como real ou falso.

Quando aplicadas na solução de problemas reais, como a colorização de imagens preto e branco (Figura 13), as GANs apresentam resultados muito próximos aos processos manuais. Destacam-se também suas aplicações na geração de faces (WANG, X. *et al.*, 2022) e na composição de melodias (YANG, L.-C.; CHOU; YANG, Y.-H., 2017).

Figura 13 – Colorização de imagens por GAN. Em sequência: (a) grayscale, (b) imagem original, e (c) colorização por GAN



Fonte: Adaptado de Nazeri, Ng e Ebrahimi (2018).

2.2.4.1 Trabalhos relacionados

Conforme discutido anteriormente, a composição de músicas não é uma área nova na computação. Ademais, conforme analisam Li-Chia Yang, Chou e Yi-Hsuan Yang (2017) e Li, Jang e Sung (2019), também existem diversas implementações de LSTM, GANs, e outras categorias de redes neurais que abordam esse problema. A Tabela 3 compara a proposta desse

trabalho com os demais métodos da literatura.

Tabela 3 – Comparação entre modelos recentes de composição musical com aprendizado de máquina

	MelodyRNN	**DeepBach**	**MusicVAE**	**MidiNET**	**C-RNN-GAN**	**Proposta**
Modelo	RNN	RNN	LSTM	CNN/GAN	RNN/GAN	LSTM
Gênero	Não	Bach (artista)	Não	Não	Não	Múltiplos
Multicanal	Não	Sim	Sim	Sim	Não	Não
Código aberto	Sim	Não	Sim	Sim	Sim	Sim
Plataforma web	Não	Não	Sim	Não	Não	Sim

Fonte: Autoria própria (2022), adaptado de Li-Chia Yang, Chou e Yi-Hsuan Yang (2017).

É notável que a composição por diversos gêneros ainda é precária, sendo que os trabalhos focam na estruturação geral (sem gênero) ou na reprodução de um único estilo em específico, como é o caso do DeepBach. Por esse motivo, este trabalho explora a composição por gênero musical como uma das entradas recebidas pelo processo de geração procedural. Os gêneros disponíveis são listados pela Tabela 4.

Outro fator, ainda mais pertinente, é a falta de interação humana com os algoritmos apresentados. Dentre estes, apenas Roberts *et al.* (2018a) propõe um sistema de interação amigável ao usuário final, sendo disponibilizada uma ferramenta de geração de músicas em múltiplas faixas com foco no auxílio a composição. Em razão disso, é proposta a disponibilização do acesso a geração de melodias completas por intermédio da interface web descrita na subseção 3.2.3 com foco no uso pelo publico geral, sem a necessidade de conhecimentos musicais prévios. Dessa forma, torna-se possível que as avaliações sejam levadas em conta no treinamento da IA.

Além das características listadas acima, também é importante destacar que tanto o código quanto a base de melodias avaliadas estão disponíveis para uso da comunidade geral (DALLAGNOL, 2022).

3 MATERIAIS E MÉTODO

3.1 FERRAMENTAS E MATERIAIS

O Quadro 2 apresenta as ferramentas e materiais empregados no trabalho.

Quadro 2 – Materiais

Ferramenta / Tecnologia	Versão	Finalidade
Python	3.10.2	Linguagem de programação
Flask	0.12.2	Framework de desenvolvimento *front-end*
Angular	13	Framework de desenvolvimento *front-end*
PostgreSQL	12	Banco de dados
Visual Studio Code	1.65.2	Implementação de códigos em Python e Angular
Google Colab	2022	Implementação de códigos em Python
TensorFlow	2.7.0	Framework de aprendizado de máquina
Angular Material	13.2.6	Biblioteca de componentes visuais
Magenta Music	1.11.3	Biblioteca de componentes voltada a interação com MIDI
Bach Doodle Dataset	-	Base de dados

Fonte: Autoria própria (2022).

O TensorFlow e o Google Colab são aplicados no algoritmo de geração procedural de melodias, apresentado na Seção 3.2. O TensorFlow foi escolhido devido ao suporte a uma grande quantidade de bibliotecas de aprendizado de máquina, incluindo ferramentas do Magenta Music. De sua parte, o Google Colab foi selecionado devido dois diferenciais importantes: a necessidade de compartilhamento do código, e o ambiente de desenvolvimento na forma de "notebooks", que são documentos web dinâmicos que possibilitam a escrita e execução de código Python sem configurações prévias, assim como o acesso ao processamento via placa gráfica (COLAB, 2022).

O Angular, Angular Material, Magenta Music e Visual Studio Code compõe a lista das ferramentas e materiais da plataforma web (*front-end*), descrita na subseção 3.2.3. O Angular, e sua biblioteca Angular Material, foram selecionados devido o foco no desenvolvimento voltado a componentes, que permite a separação de regras de comunicação com a aplicação de persistência e o algoritmo de geração de melodias. Além disso, o Angular também suporta a bibliotecas Javascript, possibilitando o uso dos componentes de visualização e reprodução de melodias do Magenta Music.

Por sua vez, o Python é utilizado tanto na construção das aplicações do servidor (*front-end*), que faz a comunicação com o banco de dados e disponibiliza o serviço de geração das melodias da IA, quanto na implementação do próprio algoritmo de geração de melodias. Está linguagem foi selecionada pela razão de ser a base do *framework* TensorFlow.

Para construir as aplicações do servidor com Python, o *framework* Flask foi escolhido por ser uma biblioteca focada apenas na criação de serviços web (FLASK, 2022), adequada aos serviços propostos pelo trabalho (que são compostos por poucos métodos de comunicação entre cliente e servidor). Estas aplicações poderão se conectar com o banco de dados da aplicação, montada no sistema gerenciador PostgreSQL, que armazenará as melodias geradas pela IA.

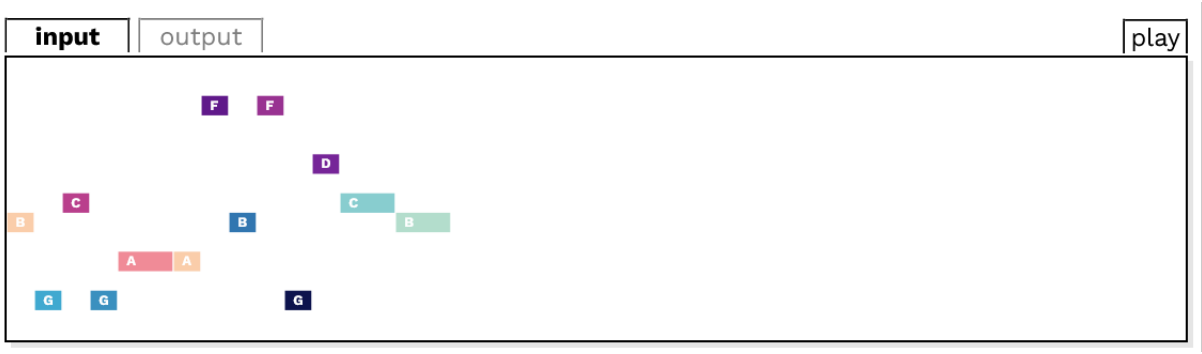
Em razão da necessidade de treinar a IA com dados de melodias reais, o Bach Doodle Dataset foi escolhido como a fonte de dados da IA por disponibilizar lotes de arquivos prontos para o uso nas rotinas de treinamento de redes implementadas com TensorFlow.

Conforme apresentado por Huang *et al.* (2019), a base de dados de composições Bach Doodle contém aproximadamente vinte e um milhões de melodias submetidas por usuários ao redor do mundo. Os dados da composição seguem os campos declarados pela estrutura NoteSequence (MAGENTA, 2020), sendo que cada item armazenado pela base contém dois campos principais:

- input_sequence: a composição submetida pelo usuário. Esse dado pode ser reconstruído na estrutura NoteSequence (serializada).
- loops_listened: a quantidade de reproduções da composição, isto é, quantas vezes a melodia foi ouvida pelos usuário do Bach Doodle.

A Figura 14 exibe a reconstrução de uma melodia em formato NoteSequence conforme a demonstração disponibilizada pelo Bach Doodle Dataset (MAGENTA, 2022). Essa visualização pode ser implementada através dos componentes do Magenta Music.

Figura 14 – Melodia extraída de uma NoteSequence



Fonte: Adaptado de Magenta (2022).

O Quadro 3 apresenta as propriedades da estrutura NoteSequence utilizadas no processo de geração e discriminação deste trabalho.

As propriedades *time_signature* e *tempo* correspondem a assinatura temporal e as batidas por minuto, respectivamente. É importante notar que a assinatura temporal é composta

por dois campos numéricos, numerador e denominador, ou seja, segue a notação formal explicada durante a Seção 2.1. Por sua vez, o tempo possui o campo numérico *qpm*, que corresponde às batidas por minuto (também abreviada para bpm). Cada NoteSequence também possui uma lista de notas que são armazenadas na propriedade *note*, na qual *pitch* e *pitch_name* correspondem ao tom da nota (frequência numérica e a letra correspondente), e *start_time* e *end_time* sinalizam o intervalo de segundos pelos quais a nota ressoa. Por fim, o tom da melodia é armazenado no campo de texto *key*, da propriedade *key_signatures*, e o campo *genre* dos metadados (*metadata*) rotula o gênero da peça.

Quadro 3 – Propriedades da NoteSequence

Nome	Principais propriedades	Finalidade
time_signature	numerator, denominator	Assinatura de tempo da melodia.
key_signatures	key	Tom da melodia.
note	pitch, pitch_name, start_time, end_time	Sequência das notas da melodia com as respectivas durações.
metadata	genre	Metadados gerais da melodia.
tempo	qpm	Batidas por minuto

Fonte: Autoria própria (2022).

3.2 MÉTODO

O objetivo da IA proposta é a geração de melodias com duração variável que sigam as estruturas adequadas de cada gênero *g* informado como entrada para a processo de composição. Para tanto, são utilizadas três redes distintas em um modelo MusicVAE: um codificador LSTM bidirecional, um condutor LSTM, e um decodificador hierárquico (categórico).

A seguir são definidos os conceitos gerais da IA, na sequência são explicadas as particularidades de cada rede e por fim o método proposto é comparado com as demais implementações da literatura.

3.2.1 LSTM para Composição de Melodias Escalares

De acordo com o método proposto por Silla, Koerich e Kaestner (2008), três características fundamentais podem ser extraídas de um sinal de áudio: timbre, tom e ritmo. Como a instrumentalização não faz parte do escopo do projeto, temos então que o treinamento das redes trabalha apenas com a extração das informações de frequência e ritmo contidas nas estruturas NoteSequence, sendo o timbre de piano uma informação fixa. Também é válido ressaltar que

não foi necessário trabalhar diretamente com o processamento de áudio, uma vez que a melodia deriva da NoteSequence, sendo essa a estrutura interpretada como som por seguir o formato de Interface Digital de Instrumentos Musicais (MIDI), que como explicam Brunner *et al.* (2018), "é um padrão de notação musical similar a partitura", ou seja, não contém som, apenas instruções para reprodução do mesmo.

Dessa forma, a geração de melodias executada pela IA baseia-se no processo de amostragem do espaço latente - ou seja, na combinação de características ditas "convencionais" (ROBERTS *et al.*, 2018b) - para compor uma sequência de notas considerada adequada. Portanto, fica a cargo exclusivo da IA a seleção das características físicas da composição, como as batidas por minuto, a assinatura temporal da peça, a escala utilizada e também o tom da melodia.

Como definido pelo diagrama da Figura 15, o processo de geração de uma melodia é iniciado com três parâmetros de entrada:

- Gênero g: Um número inteiro de 1 a 3, definido de acordo com a Tabela 4.
- Tamanho t: Um número inteiro constante que dita a quantidade máxima de notas presentes no resultado final. O valor de t é sempre igual a noventa, resultando em melodias com duração entre dez e doze segundos (em média).
- Ruído r: Gerado pelo MusicVAE como um vetor que possui *z_size* elementos, onde *z_size* é igual à quantidade de variáveis acessadas pelo modelo (vetores latentes).

Enquanto o t e r são repassados para o codificador diretamente, o gênero g é interpretado como a "temperatura" do modelo, que se refere ao hiperparâmetro da função softmax da IA.

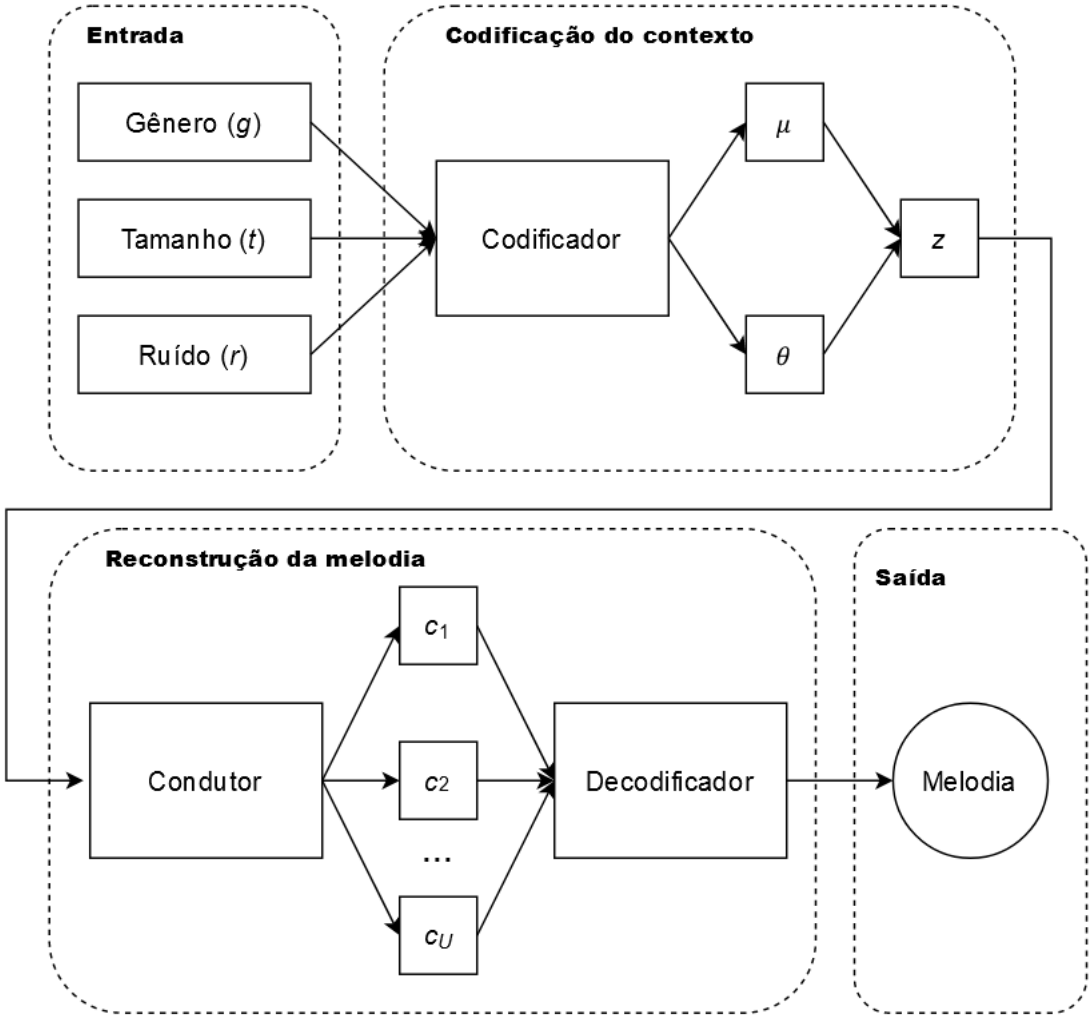
$$q_i = \frac{exp(z_i/T)}{\Sigma_j exp(z_j/T)} \tag{10}$$

Segundo Goodfellow, Bengio e Courville (2016), quando utilizadas dentro de um modelo a função de softmax pode alterar as variáveis internas de modo que a distribuição final pode ser afetada. Como Hinton, Vinyals e Dean (2015) demonstram pela Equação 10 (de cálculo softmax), quanto mais alta for a temperatura T mais raros são os usos de características não convencionais, sendo q a probabilidade de uma classe i e z o vetor de predições/características (também conhecido como logits) da rede.

Seguindo estas observações é estabelecida a seguinte distinção entre os gêneros musicais: o gênero "Clássica" recebe a menor temperatura possibilitando combinações complexas inusitadas, o gênero "Rock" oferece a opção de meio termo, enquanto que o "Pop" está mais restrito a amostragem das características mais populares por possuir a maior temperatura dos três. Os

valores utilizados para cada gênero são apresentados pela Tabela 4.

Figura 15 – Diagrama da geração de uma melodia



Fonte: Autoria própria (2022).

Tabela 4 – Gêneros correspondentes a cada entrada g

Inteiro repassado	Gênero correspondente
1	Clássica
2	Rock
3	Pop

Fonte: Autoria própria (2022).

Como explicado durante a Sub-Seção 2.2.3, com essas entradas o codificador produz um vetor de contexto z, que é utilizado pelo condutor na produção de U vetores c. Os vetores c são recebidos pela rede LSTM decodificadora, que reconstrói uma melodia NoteSequence com até t notas. Essa NoteSequence pode ser reproduzida como som por seguir o padrão MIDI.

O padrão MIDI prevê uma estrutura de mensagens que contém uma série de instruções sobre a reprodução de uma peça musical específica, podendo definir desde os tons a serem

reproduzidos, até os tempos, intensidades, e outras características de cada nota presente na peça (PINTO, 2000). Estas mensagem são enviadas dos computadores a instrumentos ou aparelhos capazes de produzir sons, como sintetizadores e fones de ouvido.

Sendo assim, o processo de geração proposto determina a progressão da melodia, assim como as propriedades temporais de cada noda, apenas pelo que o modelo LSTM considera provável, não sendo imposta nenhuma restrição senão quanto a extensão máxima da peça. Em contraste com as demais propostas da literatura, como comparadas por Li-Chia Yang, Chou e Yi-Hsuan Yang (2017), isso implica na expansão da versatilidade dentro do processo de composição, tanto para a formação de melodias escalares quanto para estruturação temporal da peça, pois o modelo não possui um gênero fixo nem uma assinatura temporal constante mas ainda procura adequar suas produções as características de múltiplos gêneros musicais.

3.2.2 Treinamento do Modelo e Aprendizagem de Máquina

O treinamento da IA é feito com base nos dados disponibilizados pelo Bach Doodle, que são extraídos de forma aleatória de um arquivo registros sem considerar qualquer metadado de gênero ou popularidade (que podem não estar preenchidos em todas as melodias). Ao fim do treinamento, a IA terá construído um contexto (espaço latente) de progressões musicais que será utilizado pelo processo de geração como fonte de amostras que são, provavelmente, adequadas para continuar a sequência em composição. Portanto, o objetivo dessa rotina é fazer com que o modelo "aprenda as principais características da base de treinamento" e "exclua as possibilidades pouco convencionais" (ROBERTS *et al.*, 2018b).

Como demonstrado pelo Código 1, para executar o treinamento são selecionados dez lotes contendo dois mil e quarenta e oito (variável "BATCH_SIZE") exemplos aleatórios entre os registros contidos na base de treinamento (função "prepare_training_set()", que utiliza o método $shuffle$ do TensorFlow para embaralhar os dados considerando a quantidade total de lotes, armazenada na variável "SHARDS_LEN"). Estes lotes são armazenados em um arquivo temporário, criado no caminho definido pela variável "OUTPUT_PATH" durante a execução da função de escrita do TFRecordWriter (também disponibilizado pelo TensorFlow), que é nomeado de acordo com o número do lote (1, 2, 3, etc). Já a função "get_train_batch_path()" é chamada na criação dos lotes, e antes do início de cada passo do treinamento, pois recebe um índice (por padrão, o número do passo atual do treinamento) para retornar o caminho do arquivo que contém o lote de dados correspondente.

Código-fonte 1 – Funções preparação da base de dados

```
...
def get_train_batch_path(index = current_step):
  return OUTPUT_PATH + "batch-%i.tfrecord"%(index)

def prepare_training_set(dataset_path):
  tr_raw_set = tf.data.TFRecordDataset(dataset_path)
  tr_data = tr_raw_set.shuffle(BATCH_SIZE *
    SHARDS_LEN)

  for i in range(0, SHARDS_LEN):
    writer = tf.data.experimental.TFRecordWriter(
        get_train_batch_path(i))
    writer.write(tr_data.shard(SHARDS_LEN, i))
...
```

Fonte: Autoria própria (2022).

Individualmente, estes lotes são repassados para a função “train()”, disponibilizada pelo MusicVae do Magenta como demonstra o Código 2. Essa função, que é executada uma vez por lote, tem como responsabilidade calcular e aplicar os gradientes (estocásticos) com base nos resultados obtidos para o lote recebido. A rotina completa é executada dez vezes, também chamadas de dez épocas (variável “EPOCHS”), sendo criado um ponto de restauração dos parâmetros da IA a cada lote processado (vide parâmetro “checkpoints_to_keep”). A função “train()”, definida pelo código deste trabalho, controla os passos do treinamento (um passo por lote, sendo que estes são reiniciados com o valor zero a cada nova época) alimentando a variável “current_step” que altera o lote resgatado pela função “get_dataset()” (esta função retorna os dados do treinamento conforme o lote definido pela configuração do modelo). É durante os ciclos definidos por essa função que o método de treino do MusicVAE é chamado, sendo sua execução a única responsabilidade da função “train_step()” - que, portanto, se refere à execução de um único passo do treinamento.

Dessa forma, cada execução completa do treinamento processa cerca de 204.800 melodias de duração variável. De sua parte, o modelo de geração procedural (variável “config”, incluindo também a função “get_model_config()”) será detalhado na Sub-Seção 4.2.3.

Além da rotina de treinamento descrita, também é necessário salientar que, após a coleta das avaliações dos usuários, é possível aplicar uma segunda fase de treinamento fazendo uso exclusivo das melodias geradas pela IA que foram bem avaliadas pelos usuários. Com essa alteração no universo de dados de treinamento, a LSTM é condicionada a considerar as propriedades das combinações bem avaliadas como as características mais prováveis de uma

Código-fonte 2 – Funções treinamento da IA

```
    ...
    def train_step():
        run_dir = os.path.expanduser('training_dir')
        music_vae_train.train(
            train_dir=os.path.join(run_dir, 'train'),
            config=config,
            dataset_fn=get_dataset,
            num_steps=hparams.batch_size,
            checkpoints_to_keep=1,
            task=0
        )

    def train():
      for i in range(0, EPOCHS):
        current_step = 0;
        for j in range (0, SHARDS_LEN):
          current_step = j;
          try:
            train_step()
          except:
            pass
    ...
```

Fonte: Autoria própria (2022).

melodia. Com isso, pode-se esperar que as melodias geradas obtenham uma maior taxa de aprovação.

Outro ponto importante do processo descrito, é que apesar do parâmetro gênero g não ser utilizado no treinamento, o mesmo tem a finalidade de alterar a configuração de hiper-parâmetros do modelo (modificando as probabilidades associadas as características do espaço latente) durante o processo de geração procedural. Destoando das propostas da literatura como as redes CycleGAN (BRUNNER *et al.*, 2018), o MIDINet (YANG, L.-C.; CHOU; YANG, Y.-H., 2017), e os modelos pré-treinados do MusicVAE (ROBERTS *et al.*, 2018a), essa entrada permite a suposição da existência de uma relação entre as probabilidades das características encontradas pela IA e suas frequências de uso na construção de obras pelos gêneros definidos - ou seja, entende-se que, por exemplo, músicas “Pop” fazem uso de combinações mais prováveis (e portanto mais populares) que músicas do gênero “Clássica”. Isso possibilita que o treinamento seja executando sobre quaisquer melodias (sem quaisquer restrições), porém a geração procedural passa a trabalhar com diversas combinações de variáveis internas para satisfazer a restrição imposta pelo parâmetro g.

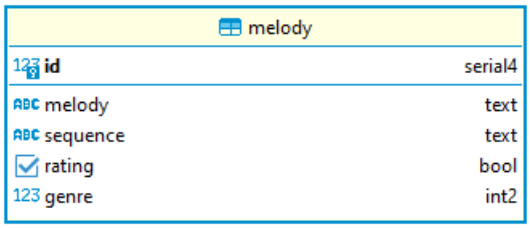
3.2.3 Plataforma Web

A interface web é separada em duas partes, o *front-end* implementado em Python, e o *front-end* implementado em Typescript/Angular.

O *front-end* em Python disponibiliza o serviço de geração e persistência das melodias geradas pela LSTM. O objeto retornado pelo método GET (geração de melodias) é um arquivo texto que contém todas as informações da NoteSequence gerada pela IA, sendo que as propriedades são nomeadas de acordo com o modelo implementado por Huang *et al.* (2019). O método POST, por sua vez, recebe a NoteSequence gerada juntamente com sua avaliação, gênero, e o objeto JSON montado pelo *front-end* apartir do texto da melodia recebido via o método GET. Esse objeto é chamado de NoteSequenceDTO pois designa a interface de transporte dos dados.

Como mostra a Figura 16, no banco de dados PostgreSQL a avaliação ("rating") e o gênero ("genre") são armazenada na mesma tabela da NoteSequence ("sequence"), assim como o texto do JSON montado pelo *front-end* ("melody"). Dessa forma, é possível resgatar as respostas dos usuários como um valor binário verdadeiro ou falso, o gênero como seu inteiro correspondente (seguindo a relação da Tabela 4), e a melodia completa como um texto.

Figura 16 – Diagrama do banco de dados



Fonte: Autoria própria (2022).

O *front-end* utiliza os componentes Angular Material para controles gerais, e um componente customizado do Magenta para reprodução das NoteSequence obtidas. Também é importante ressaltar que o esquema de cores foi gerado partindo do guia de cores da biblioteca Angular Material.

Para agilizar o uso da interface, o NoteSequenceDTO é enviado do *front-end* para a aplicação em Python no instante em que o usuário avalia a melodia, evitando cliques específicos para esse processo uma vez que o mesmo não é de interesse direto do usuário (o interesse do mesmo é apenas na geração da melodia). Esse processo ocorre no momento da avaliação pois é a partir desse estágio que os valores de todos os três campos do banco de dados são conhecidos.

Devido o fluxo descrito, apenas melodias avaliadas constam na base de dados utilizada

pela segunda fase de treinamento da LSTM. A página montada segue o protótipo apresentado pela Figura 17.

Figura 17 – Protótipo da interface web (baixa fidelidade)

Fonte: Autoria própria (2022).

4 RESULTADO

4.1 ESCOPO DO SISTEMA

O objetivo do sistema proposto é a composição de melodias (MIDI) geradas de acordo com o gênero musical repassado como entrada para uma rede neural. Dessa forma, a IA deve selecionar as características de sua produção (tom, progressão e ritmo) considerando sua probabilidade de aceitação para o gênero recebido, sendo o universo de gêneros musicais disponíveis composto somente pelos tipos “Clássica”, “Rock”, e “Pop”.

Visando aprimorar a qualidade das composições, o treinamento da rede neural foi estruturado em duas etapas: sobre composições manuais, e sobre composições artificiais (produzidas pela própria IA) avaliadas manualmente. O modelo LSTM foi escolhido devidos suas capacidades de reter conhecimentos de longo termo, trabalhando dentro de um contexto de entradas relacionadas. Esse funcionamento se mostra produtivo na composição de melodias devido à natureza de dependência entre os tons de uma sequência qualquer, como explicado na Seção 2.1. Além disso, outras características melódicas como o ritmo e a intensidade das notas dependem do contexto de inserção da nota, sendo fundamental a manutenção de estado da IA para realizar escolhas não apenas adequadas, mais “interessantes” - ou seja, compor sequências com variações complexas que fazem sentido para o tom/ritmo selecionado.

Por conta do modelo escolhido, ao total são empregadas três redes distintas: uma rede para codificação, uma rede para decodificação, e uma rede condutora herdada do modelo MusicVAE.

Para que sejam avaliadas as composições da IA, também é disponibilizada uma interface web para interação com as redes LSTM, que não exige nenhum conhecimento prévio sobre aprendizado de máquina nem composição musical. São as funcionalidades disponibilizadas pela interface web:

- Gerar melodia, cuja requisição é enviada para rede neural.
- Ouvir melodia, quando a interface interpreta o MIDI gerado como som.
- Avaliar melodia, cuja requisição é persistida na base de dados de composições avaliadas.

Não faz parte do escopo do trabalho a imitação da composição manual, porém a distribuição das produções da IA deverá ser próxima da distribuição real das melodias compostas manualmente, e portanto é previsto que o sistema irá se adequar às regras formais da composição

através da análise estatística da distribuição de tons, escalas, ritmos e progressões presentes no estrato de melodias encontradas na base de composições reais. Estima-se também que, posteriormente, a IA priorizará as combinações com a maior probabilidade de aprovação pelos usuários da interface web, sendo esse comportamento aprendido por meio da mesma análise estatística refeita em cima dos dados da base de melodias avaliadas.

4.2 APRESENTAÇÃO DO SISTEMA

Esta seção apresenta os resultados preliminares do trabalho. A subseção 4.2.1 apresenta a interface web desenvolvida.

4.2.1 Interface Web

Conforme elencado na seção 4.1, a interface web, apresentada pela Figura 18, permite três tipos distintos de ação: geração, reprodução e avaliação de melodias.

Figura 18 – Interface web (início)

Fonte: Autoria própria (2022).

A geração de melodias, cujo fluxo é ilustrado pela Figura 19, é a etapa inicial de interação com o sistema, e também a única que se comunica com a IA.

Figura 19 – Fluxo de geração de melodias pela interface web

Fonte: Autoria própria (2022).

Primeiramente é necessário que o usuário selecione um gênero na caixa de seleção, e na sequência, envie uma requisição POST para o *front-end* através do botão “Gerar melodia”. Após gerada a melodia, o usuário pode visualizar a NoteSequence resultante dentro da caixa de exibição, sendo que a partir desse momento também é disponibilizada a função de reproduzir a melodia (MIDI), e de avaliá-la de forma positiva ou negativa.

Dessa maneira, todo fluxo da interface se baseia no recebimento de uma NoteSequence interpretada pelo *front-end* partindo de um arquivo JSON que armazena uma matriz de notas. Essa matriz pode ou não ser persistida de no banco de dados de acordo com a realização do processo de avaliação.

A matriz de notas, como exibida pelo Cód. 3, é composta de diversas entradas com três propriedades principais: “startTime” e “endTime”, que identificam os tempos de início e término da nota respectivamente, e a propriedade “pitch”, que identifica o tom (numérico) que deve ser utilizado pelos dispositivos MIDI no momento de reprodução do som. Durante o processo de reprodução da melodia, esta sequência de notas é repassada ao “Player” do Magenta Music, como apresentado pelo Cód. 4.

Código-fonte 3 – JSON da melodia

```
...
    "notes": [
                {
                    "endTime": 2,
                    "pitch": 64,
                    "pitchName": "E",
                    "quantizedEndStep": "8",
                    "quantizedStartStep": "4",
                    "startTime": 1,
                    "velocity": 100
                },
                {
                    "endTime": 3,
                    "pitch": 71,
                    ...
                }
                ...
    ]
...
```

Fonte: Autoria própria (2022).

O reprodutor do Magenta Music, instanciado no projeto como “midiPlayer”, é utilizado pela função “play()” que, por sua vez, é chamada pelo botão “Tocar”. Assim como o método de reprodução do Player, “start()”, a função “play()” é assíncrona pois deve aguardar a conclusão

Código-fonte 4 – Reprodução da NoteSequence pelo Player

```
    ...
    public async play() {
        if (!this.noteSequence) { return; }
        this.isPlaying = !this.isPlaying;

        if (!this.midiPlayer.isPlaying()) {
            this.isPlaying = true;
            await this.midiPlayer.start({
                notes: this.noteSequence.notes,
                totalTime: this.noteSequence.totalTime
            });
        } else {
            this.midiPlayer.stop();
        }

        this.isPlaying = false;
    }
    ...
```

Fonte: Autoria própria (2022).

da melodia para reconfigurar a interface entre os botões “Tocar” (quando a variável booleana “isPlaying” é falsa), e ”Parar“ (quando a variável “isPlaying” é verdadeira).

Caso a função “play()” seja executada enquanto o Player ainda está reproduzindo a melodia (também é chamada pelo botão “Parar”), a reprodução é interrompida através do método “stop()”, do próprio Player. Este método força o retorno prematuro do método “start()”, fazendo com que a primeira execução assíncrona da função “play()” seja finalizada e a variável “isPlaying” receba o valor “false”. Dessa forma é restaurado o estado inicial da interface.

Conforme mostrado durante a Sub-Seção 3.2.3, para que a melodia seja reproduzida pelo Player o *front-end* precisa executar a conversão da NoteSequence recebida como texto para o padrão JSON. Essa conversão é de cargo da classe PlayerUtil, que recebe o texto na função pública “convertToInputSequence()” e retorna um objeto NoteSequence serializável.

Como mostra o Código 5, os campos de tempo total da obra, tempo (ritmo), e notas são extraídos do texto através da busca e recorte de pedaços do texto recebido (funções “get”). É importante ressaltar que, como será visto na Sub-Seção 4.3, a IA. pode gerar notas com tons abaixo do trigésimo sexto oitavo (do modelo MIDI), que são filtrados do modelo por não corresponderem ao intervalo de frequências esperados pelo componente de visualização do Magenta.

Em seguida os dados recortados são processados pelas funções “parseStringAsNumber()”

Código-fonte 5 – Conversão de NoteSequence (texto) para objeto serializável

```
    ...
    public convertToInputSequence(str: string): any {
        ...
        return {
            notes: this.getNotes(str),
            tempos: [this.getTempo(str)],
            totalTime: this.getTotalTime(str),
            ticksPerQuarter: this.getTicksPerQuarter(str),
            totalQuantizedSteps: this.currentQuantizedStep,
            quantizationInfo: { stepsPerQuarter: 4 },
        };
    }
    ...
```

Fonte: Autoria própria (2022).

(campos númericos gerais) ou "getNote()" (campos da nota, executado uma vez para cada nota da melodia), obtendo-se uma lista de notas com informações de tom e tempo (início e fim) como mostra o Código 6, além de outras diversas propriedades numéricas correspondentes às características gerais da melodia.

Código-fonte 6 – Conversão de notas da NoteSequence

```
    ...
    private getNote(str: string, lastIndex: number):
        ...
        return {
            note: {
                pitch: this.getPropNumber(noteStr, 'pitch'),
                velocity: this.getPropNumber(noteStr, 'velocity'),
                startTime: this.getPropNumber(noteStr, 'start_time'),
                endTime: this.getPropNumber(noteStr, 'end_time'),
                quantizedStartStep: this.currentQuantizedStep,
                quantizedEndStep: this.currentQuantizedStep
                    + this.STEP
            },
            lastIndex: noteEnd
        };
    }
    ...
```

Fonte: Autoria própria (2022).

Além disso, também é necessário computar as batidas do MIDI (steps). A quantidade de batidas de cada nota é definida pela constante "STEP", não sendo possível realizar alterações individuais. Dessa forma, cada nota recebe um total de quatro batidas, sendo o total de batidas da melodia igual a $STEP \cdot totalNotes$, onde a variável $totalNotes$ armazena a quantidade total de

notas da peça.

4.2.2 Aplicação Python

A aplicação Python recebe as requisições da interface web, redirecionando-as para a IA no caso da geração de melodias, ou executando operações de inserção no banco de dados quando uma melodia é avaliada.

Todas as comunicações com outras aplicações externas são realizadas por meio da biblioteca “Flask”, que permite a execução contínua de um código Python com roteamento de operações, ou pela biblioteca “requests”, que permite a execução de requisições HTTP para outros serviços. Dessa forma, a aplicação permanece operante porém inativa até que uma de suas rotas seja requisitada pela interface web.

Código-fonte 7 – Método de geração de melodias na aplicação Python

```
1     ...
2     def request_melody(genre):
3         payload = {'genre': genre}
4         response = requests.get(get_ai_connection()+'/get-sample',
5             params=payload)
6         return Response(response.text, mimetype='text/plain')
7     ...
```

Fonte: Autoria própria (2022).

Sendo assim, a função “request_melody()” do Código 7 executa a requisição para o serviço da IA, retornando a NoteSequence na forma de texto (a ser interpretado pela interface) como retornado pela aplicação localizada no caminho armazenado pela variável retornada pela função “get_ai_connection()”. Esta variável é lida do banco, estando armazenada em uma tabela separada das melodias chamada “aiurl”, que contém apenas uma coluna do tipo texto.

Já a conexão com o banco de dados PostgreSQL é realizada por meio da biblioteca “psycopg2”, que controla as sessões abertas, permitindo a execução de operações de escrita e leitura em bancos PostgreSQL como demonstrado pela função “insert_melody()” do Código 8. A função em questão recebe os campos enviados pela interface, como a sequência gerada pela IA (parâmetro “sequence”) e a avaliação do usuário (parâmetro “rating”), para persistir a melodia na base de dados do trabalho por meio do comando SQL “INSERT”. Para executar este comando SQL, uma variável chamada “query” é criada, contendo o não só o comando como também as entradas a serem substituídas pelos valores reais (identificadas pelos caracteres “%s”). A “query”

é executada pelo cursor (função “execute”) criado com base na conexão estabelecida entre a aplicação e o banco (pela biblioteca “psycopg2”), por meio das configurações encontradas na variável “params”.

Código-fonte 8 – Método de persistência de melodias na aplicação Python

```
    ...
    def insert_melody(melody, sequence, rating, genre):
        conn = None
        try:
            params = config()
            conn = psycopg2.connect(**params)
            cur = conn.cursor()

            query = "INSERT INTO melody VALUES " +
                "(nextval('seq_melody'), %s, %s, %s, %s)"
            cur.execute(query, (melody, sequence, rating, genre))
        except (Exception, psycopg2.DatabaseError) as error:
            raise Exception(error)
        finally:
            if conn is not None:
                conn.commit()
                conn.close()
    ...
```

Fonte: Autoria própria (2022).

Ao final desse processo, a conexão é fechada, e as alterações executadas por ela são mantidas pelo banco. Essas ações são de responsabilidade dos métodos “commit()” e “close()”, da conexão disponibilizada pela biblioteca “psycopg2’.

4.2.3 Modelo LSTM

O modelo LSTM foi implementado em Python utilizando a biblioteca Magenta e suas dependências - mais notavelmente o TensorFlow, que fornece múltiplas funcionalidades de criação e treinamento de redes neurais. Além disso, a própria IA é também um serviço “Flask”, recebendo as requisições enviadas pela aplicação Python e retornando suas produções.

A definição da IA, que pode ser acompanhada pelo Código 9, se baseia na criação de um modelo MusicVAE que possui uma rede LSTM codificadora e outra rede LSTM decodificadora, como declarado na Sub-Seção 3.2.1.

Além das redes “BidirectionalLstmEncoder” e “CategoricalLstmDecoder”, também é necessário repassar os hiper-parâmetros do modelo, e definir um conversor de NoteSequence

Código-fonte 9 – Definição da IA

```
    ...
    model = MusicVAE(lstm_models.BidirectionalLstmEncoder(),
                     lstm_models.CategoricalLstmDecoder())

    hparams = merge_hparams(
        lstm_models.get_default_hparams(),
        HParams(
            batch_size=BATCH_SIZE,
            max_seq_len=32,
            z_size=512,
            enc_rnn_size=[2048],
            dec_rnn_size=[2048, 2048, 2048],
            learning_rate=0.005,
            ))

    data_converter = data.OneHotMelodyConverter(
        skip_polyphony=False,
        max_bars=100,
        slice_bars=2,
        steps_per_quarter=4)

    def get_model_config():
      return Config(
        model=model,
        hparams=hparams,
        data_converter=data_converter,
        train_examples_path=get_train_batch_path(),)
    ...
```

Fonte: Autoria própria (2022).

("OneHotMelodyConverter"), que pode codificar (ou decodificar) esses objetos, transformando-os em registros binários consumidos pelas funções de treinamento do TensorFlow. Uma taxa de aprendizagem de 0.005 é utilizada pois permite que o modelo evolua de forma mais acelerada que os modelos implementados por Roberts *et al.* (2018a), com o risco de não atingir um gradiente mínimo desejável. Esta taxa de aprendizagem alta foi escolhida tendo em vista que o modelo trabalha com poucos registros (dois mil e quarenta e oito) por execução, sendo definido esse limite de tamanho do lote devido às restrições de uso de memória volátil impostas pelo Google Colab.

Outro ponto importante a ser ressaltado, é a definição da função "get_model_config()", que retorna um objeto "Config" do MusicVAE, contendo a definição das redes, dos hiperparâmetros, do conversor de dados, e o caminho do lote atual. Este objeto é utilizado pelo próprio MusicVAE durante a execução de sua função "train()", servindo principalmente para o resgate da

instância do modelo a ser treinado, e do lote de dados de treinamento a ser utilizado pelo passo em processamento.

Como explicado durante a Sub-Seção 3.2.2, este modelo é treinado utilizado a função "train()" do MusicVAE, que calcula e aplica os gradientes com base na perda do modelo em relação aos lotes de dois mil e quarante e oito registros aleatórios extraídos da base Bach Doodle. Os registros de treinamento são retornados pela função "get_dataset()", chamada pela própria biblioteca do Magenta, enquanto que os pontos de restauração dos parâmetros do modelo são exportados para a pasta no caminho declarado no parâmetro "checkpoint_dir_or_path".

Por fim, o modelo treinado pode ser restaurado utilizado uma instância da classe "TrainedModel" do MusicVAE. Como exposto pelo Código 10, essa classe recebe a configuração da IA assim como um ponto de restauração de parâmetros, sendo que o modelo treinado torna possível a geração de novas NoteSequences quando executado a função "generate_sample()". Esta função recebe a temperatura do modelo de acordo com o gênero enviado pela aplicação Python, e realiza a amostragem do espaço latente para montar e retornar o arquivo de texto que será convertido pela interface web.

Código-fonte 10 – Utilização da IA treinada

```
1   ...
2   trained_model = TrainedModel(
3       config=get_model_config(),
4       batch_size=4,
5       checkpoint_dir_or_path='/content/model.ckpt-0')
6
7   def generate_sample(genre=1.0):
8       return trained_model.sample(
9           n=1,
10          length=90,
11          temperature=genre)
12  ...
```

Fonte: Autoria própria (2022).

Como demonstrado pelo Código 11, este método é disponibilizado para chamadas GET HTTP via o ponto de entrada "/get-sample", que espera um único parâmetro de rota chamado "genre". É importante ressaltar que, para o funcionamento correto do serviço, esse parâmetro deve conter um inteiro referente ao gênero escolhido.

Código-fonte 11 – Serviço Flask da IA

```
1    ...
2    app = Flask(__name__)
3    CORS(app)
4
5    @app.route('/get-sample', methods=['GET'])
6    def get_melody_sample():
7        genre = request.args.get('genre')
8        sequences = generate_sample(genre)
9        return str(sequences[0])
```

Fonte: Autoria própria (2022).

4.3 MELODIAS GERADAS

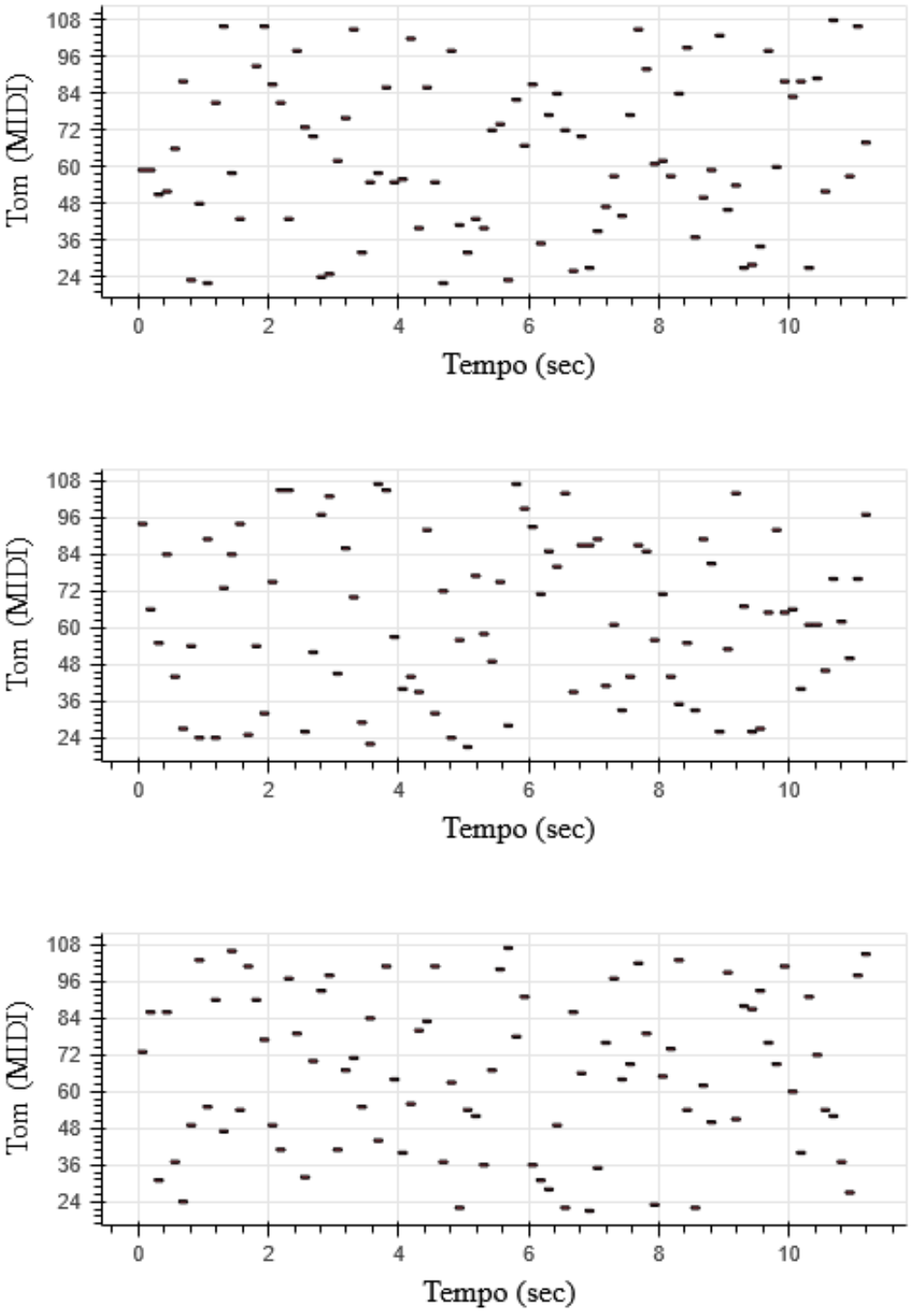
Quando as melodias produzidas pelo modelo são dispostas em um gráfico cartesiano frequência (oitavos) x tempo é possível analisar os resultados da IA. Quanto a conformidade de ritmos, tons, e repetições utilizadas nas composições. Como pode ser observado pela Figura 20, as melodias do gênero "clássica", geradas com o modelo treinado com os dados da base Bach Doodle, contém uma grande variação frequências que acabam por não respeitar as restrições tonais de cada peça. Alguns exemplos gerados pelo modelo podem ser ouvidos no vídeo (de demonstração do sistema) que está contido no repositório do projeto.

Como será discutido na Seção 4.4, isso não impede que o modelo produza sequências que contém seções coerentes (possivelmente ocasionando avaliações favoráveis dos usuários). Entretanto, é evidente que a IA terá dificuldade em produzir sequências longas sem preencher as lacunas com notas indevidas devido a tendência de compor progressões com tons que podem variar em até oitenta e quatro oitavos por segundo.

Em comparação a um trecho da composição "Cannon in D" do músico clássico Pachelbel, apresentada pela Figura 21 (onde o eixo x representa os oitavos em MIDI, e o eixo y representa as batidas), é possível notar que apesar dos saltos entre frequências serem possíveis, idealmente a IA. teria que percorrer estas distâncias de forma gradual ao invés de produzir quedas seguidas de crescentes drásticos.

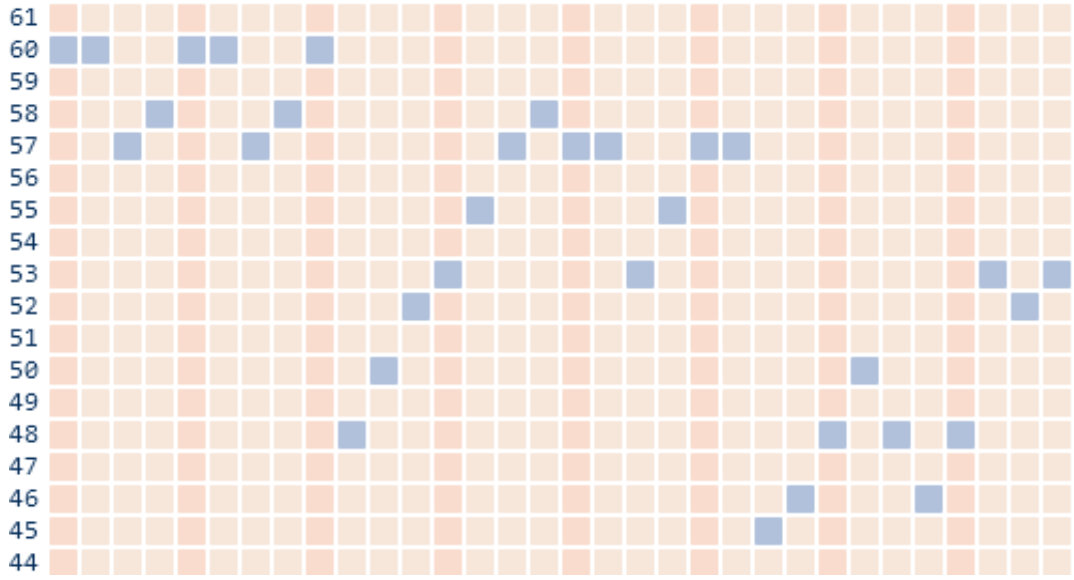
Outro componente fundamental das melodias, na qual as composições do modelo ainda se mostram precárias, é a persistência de notas ao longo do tempo. É possível verificar que raramente algum tom dura por mais do que um passo - ou seja, cada nota da composição soa brevemente por algumas frações de segundo até que um novo tom (por vezes em um oitavo muito distante) o substituí.

Figura 20 – Melodias clássicas geradas pelo modelo



Fonte: Autoria própria (2022).

Figura 21 – Trecho da peça "Canon in D" em MIDI



Fonte: (MAGENTA, 2022).

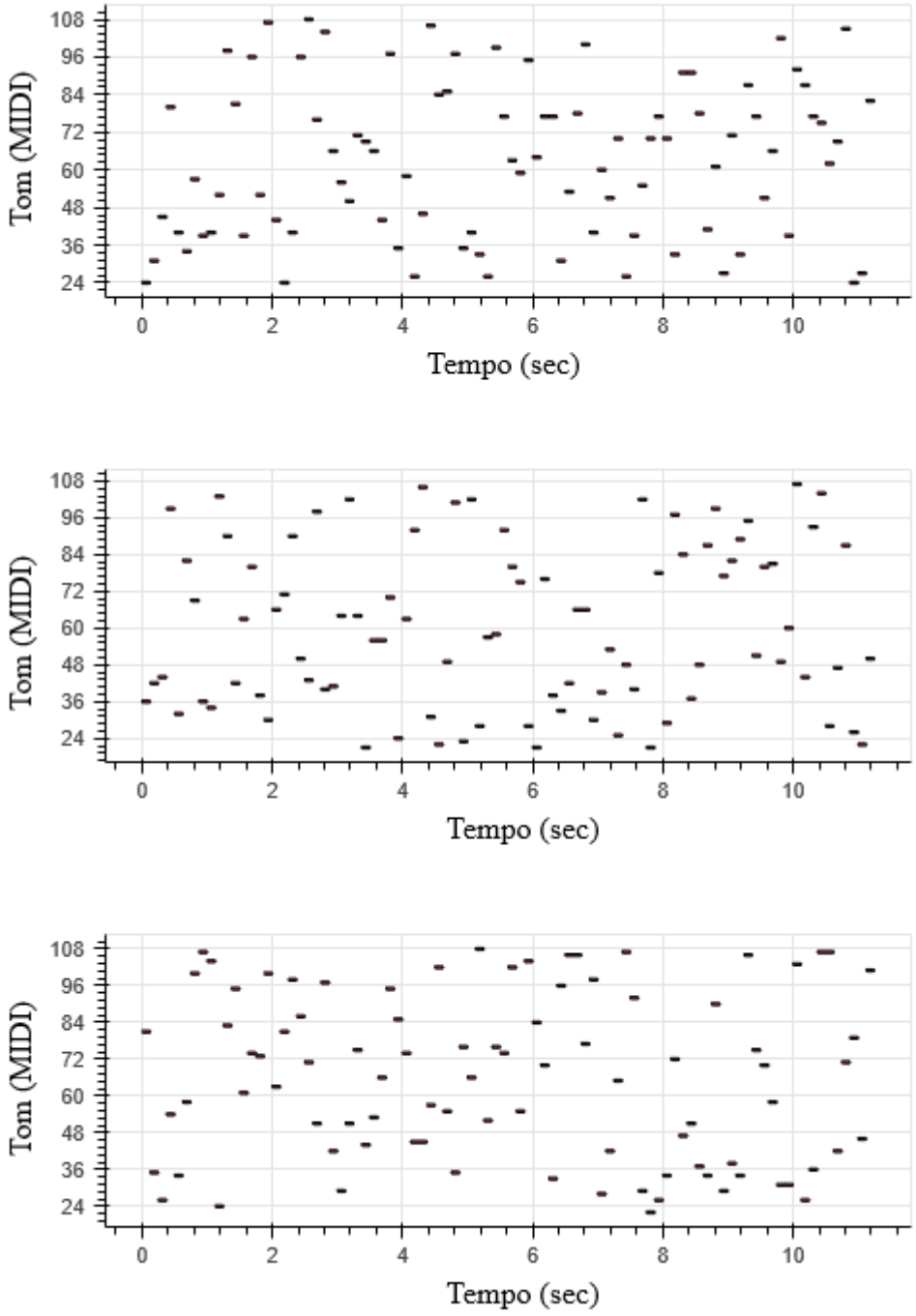
A composição por gênero também se demonstra deficitária quanto a adequação aos modelos de cada tipo: a Figura 22 e a Figura 23 exibem alguns exemplos de resultados obtidos a partir das temperaturas relacionadas a cada opção de gênero. Apesar das composições não se mostrarem suficientemente diferentes, as alterações nos hiper-parâmetros das redes se provou efetiva uma vez que as melodias dos gêneros "Rock" e "Pop" - cujas temperaturas correspondentes são as maiores, gerando melodias com as características mais populares - obtiveram a melhor relação entre melodias bem avaliadas para a quantidade de sequências geradas.

A principal diferença encontrada em algumas das melodias "Pop", como o segundo exemplo da Figura 23, é a presença de alguns crescentes graduais. Parte dessa deficiência pode ser atribuída tanto a base de dados, que não contém exemplos suficientes para o treinamento por gênero (devido o campo ser opcional sem qualquer rigorosidade em seu preenchimento), quanto a falta de um controle adicional de estilização. Uma possível alternativa é o uso de redes neurais adicionais voltadas ao aprimoramento das sequências geradas pelas LSTMs implementadas, como é comum nos processos de automação da coloração de imagens por aprendizado de máquina (FANG *et al.*, 2021).

Apesar dos problemas citados, as redes implementadas ainda se mostraram capazes de gerar pequenas sequências consideradas agradáveis pelos usuários (vide resultados apresentados pela Seção 4.4), principalmente quando os "saltos" para tons mais agudos (entende-se, abaixo do trigésimo sexto oitavo) são amenizados conforme o processo de conversão descrito na Sub-Seção 4.2.1.

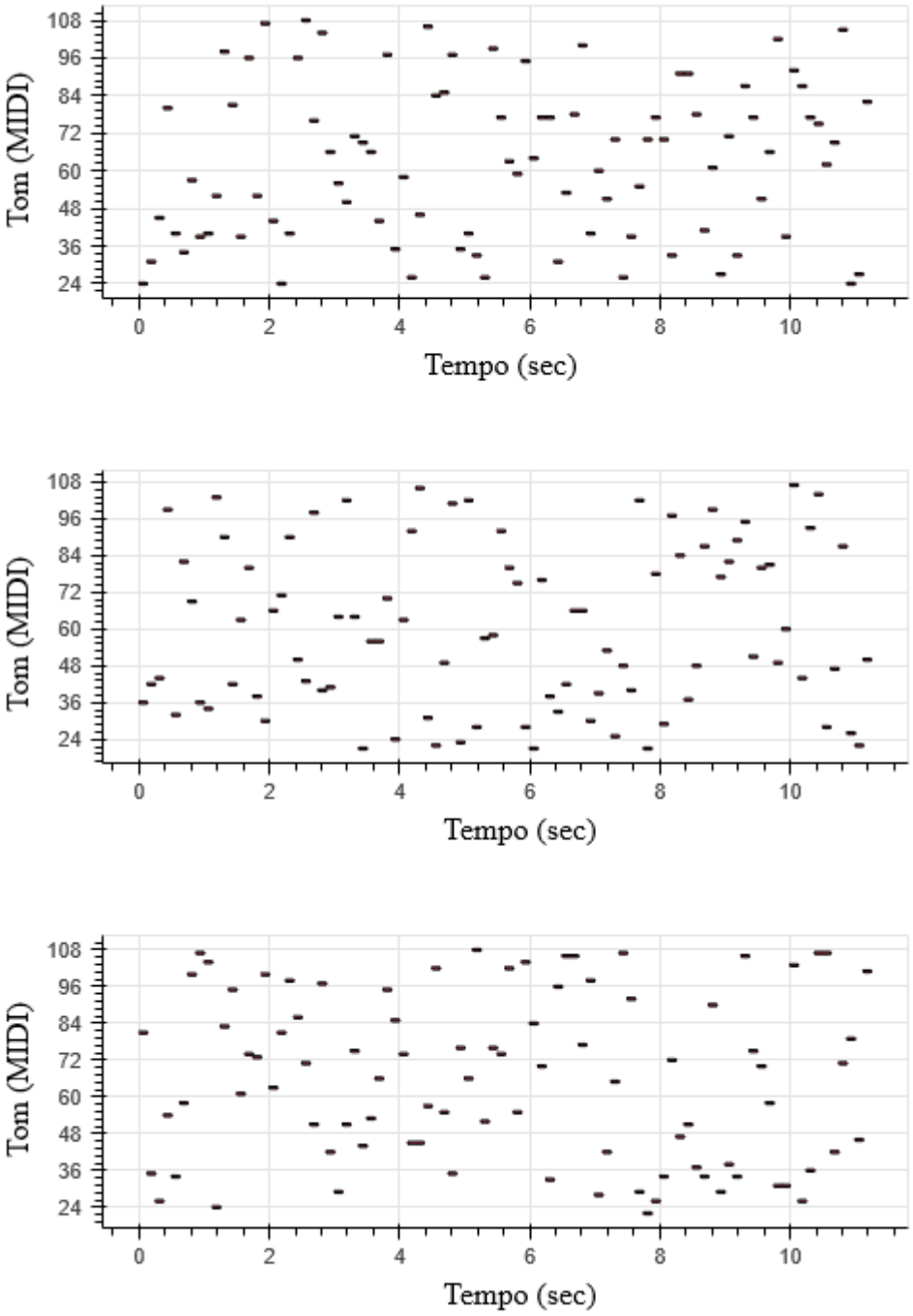
Um fator promissor, encontrado em todos os gêneros (ou seja, independente da temperatura), é a tendência a criar pequenos agrupamentos de tons sequenciais com pouca variação de oitavos, ocasionando a formação de “escadas” nos gráficos. Dessa forma, com o objetivo de sanitizar as produções, é possível fazer uso de algoritmos de limpeza e filtragem das frequências mais destoantes dado certos intervalos de tolerância. O resultado final dessa técnica é uma sequência que agrupa pequenas progressões de tons próximos.

Figura 22 – Exemplos de melodias do gênero “Rock” geradas pela I.A.



Fonte: Autoria própria (2022).

Figura 23 – Exemplos de melodias do gênero “Pop” geradas pela I.A.



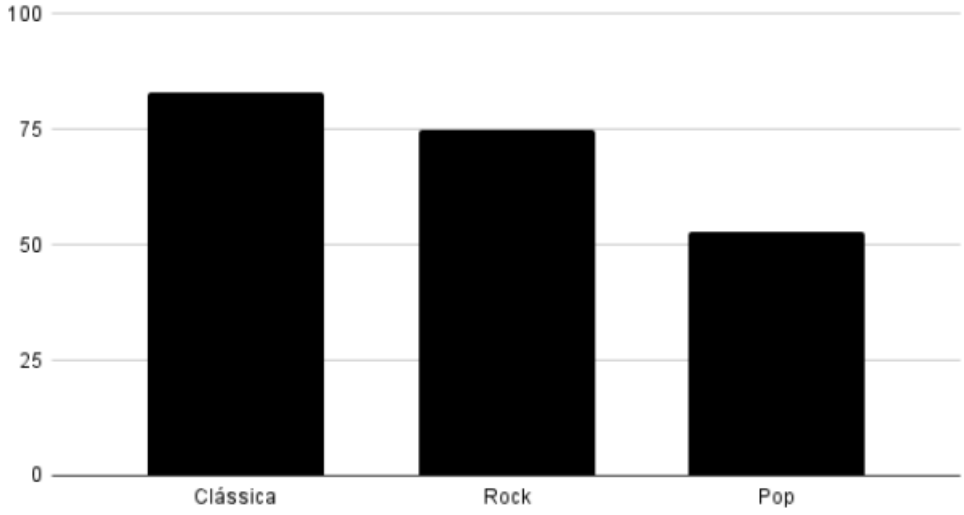
Fonte: Autoria própria (2022).

4.4 AVALIAÇÕES DOS USUÁRIOS

O acesso a plataforma foi divulgado por meio de múltiplas redes sociais, não havendo nenhum controle sobre o público avaliador. Além disso, não foi realizada a coleta de nenhum tipo de dado pessoal dos usuários, como o grau de conhecimento musical, ou indicativos do gosto pessoal de cada avaliador. Dessa forma, os resultados apresentados se tratam de uma estimativa baseada na opinião de indivíduos anônimos, e não podem ser tomados como definitivos. Também é importante ressaltar que, para obter métricas definitivas sobre a performance da plataforma, ainda é necessária a implementação de um sistema mais rigoroso de avaliação, preferencialmente em um ambiente controlado, para que também sejam contabilizadas as particularidades do público avaliador (como a experiência com a música, preferências de gênero, entre outras características).

Ao total, foram avaliadas duzentas e sete melodias dividas entre os três gêneros disponibilizados pela interface web. Como mostra o gráfico da Figura 24, o gênero mais popular foi a música "Clássica" com setenta e oito avaliações, seguido do gênero "Rock" com setenta e cinco avaliações. Por sua vez, o gênero "Pop" (que figurava no final da lista do campo de seleção) foi o menos selecionado pelo usuários, obtendo apenas cinquenta e quatro avaliações.

Figura 24 – Quantidade de melodias geradas (agrupamento por gênero)



Fonte: Autoria própria (2022).

Entre todos os gêneros, 37,78% das melodias receberam uma avaliação positiva, sendo que as sequências geradas para o gênero "Rock" foram as mais bem recebidas (42,67% das melodias "Rock" foram bem avaliadas). Segundo os resultados apresentados na Tabela 5, apesar do modelo trabalhar melhor com temperaturas > 1, existe pouca diferença na aprovação das

melodias geradas a partir de valores maiores que dois, e não é possível afirmar que temperaturas ainda maiores resultariam em taxas de aprovação significativamente maiores.

Tabela 5 – Taxa de aprovação das melodias geradas

	Bem avaliadas	Mal avaliadas	Total	Proporção de avaliações positivas
Clássica	25	53	78	32,05%
Rock	32	43	75	42,67%
Pop	21	33	54	38,89%

Fonte: Autoria própria (2022).

Outro ponto importante evidenciado pela Tabela 5 é o sucesso do sistema em prover um serviço de geração de melodias que consegue compor sequências aceitáveis com certa constância. Como até mesmo o gênero com a menor taxa de aprovação (as melodias "clássicas") obteve boas avaliações em cerca de 32% dos casos, pode-se afirmar que, para cada três melodias geradas pela IA, ao menos uma apresentará (pelo menos) uma progressão de notas considerada satisfatória pelo usuário.

Todavia, como o escopo do trabalho se limitou a composição de sequências que utilizam o timbre de piano padrão do MIDI, a inserção direta das melodias geradas (mesmo aquelas consideradas satisfatórias pelos usuários) na composição de peças maiores (como músicas completas) pode se mostrar um tanto inadequada sem que, ao menos, seja aplicada alguma de transformação de som como distorções, efeitos de ressonância, entre outras alternativas. Junto a isso, também é possível realizar a reprodução manual das sequências por meio de um instrumento musical, sendo que a interface web facilita esse processo pois disponibiliza a visualização dos tons da melodia anotados de acordo com as cifras citadas da Seção 2.1.

Portanto, o resultado das avaliações dos usuários aponta para um modelo capaz de produzir melodias curtas com certa competência, porém sem grande adequação aos gêneros definidos. Ainda assim, o sistema proposto configura uma plataforma capaz da geração procedural de melodias que pode ser melhorada por meio da implementação de diversas técnicas de refinamento e estilização das sequências musicais, com ou sem a adição de outras redes neurais.

4.5 DISCUSSÃO DE RESULTADOS

Quando abordada como um problema de combinação matemática, a geração procedural de melodias se mostra uma área promissora para o desenvolvimento de tecnologias de aprendizagem de máquina. Todavia, a vasta paleta de tons da música ocidental (e suas quase infinitas combinações) ainda configuram grandes obstáculos para a obtenção de melodias comparáveis

as produções manuais, principalmente quando o objetivo da geração de melodias é produzir sequências de notas adequadas a gêneros musicais específicos. Em especial, a área de estilização automatizada de melodias (e músicas em geral) se mostra ainda muito precária, dificultando a criação de amostras que aderem as restrições de um certo conjunto de gêneros.

O método discutido no presente trabalho - que implementa o processo de composição através da amostragem de espaço latente por redes LSTM, e a modificação dos hiper-parâmetros dessas redes para produzir combinações associadas a certos gêneros pré-definidos - consegue atingir uma avaliação positiva da audiência em até 37,78% dos casos, mas ainda sofre com problemas constantes de progressões (de notas) mal-formadas, causando a geração de melodias "atonais". Por vezes, porém, as produções da IA implementada são constituídas de curtas passagens harmoniosas que são eventualmente interrompidas por transições para tons muito distantes (ou seja, que estão há cinco ou mais dezenas de oitavos de distância). Essa característica das melodias geradas indica a possibilidade de atingir taxas de aprovação maiores com modificações no espaço latente da IA, seja pela execução de uma rotina de treinamento com dados mais rigorosos (como bases de dados constituídas exclusivamente de composições profissionais), eliminando as combinações indevidas (e alongando as progressões adequadas), ou pela implementação de redes neurais adicionais que auxiliem na retenção das características mais importantes das melodias processadas. Além disso, o sistema completo (contemplando a interface web, a aplicação de servidor e a IA) consegue prover um serviço de apoio a composição profissional e recreativa, possibilitando a criação de melodias sem direitos autorais com duração média de onze segundos que podem ser utilizadas como base para a construção de outras peças musicais compostas manualmente. É importante ressaltar que, para esse fim, o sistema facilita a reprodução das melodias em instrumentos musicais devido a visualização cifrada, e também permite que as mesmas sejam gravadas e editadas por programas externos.

Dessa forma, o presente trabalho disponibiliza uma plataforma amigável ao público geral que facilita a inserção de métodos de apoio a decisão na composição musical. Em conjunto a isso, é importante ressaltar que os resultados produzidos podem, em retorno, serem utilizados no avanço das tecnologias de geração procedural de melodias, aprofundando ainda mais as técnicas de apoio automatizado ao trabalho criativo.

5 CONCLUSÃO

O sistema implementado se mostrou capaz de gerar melodias distintas de forma procedural, com pequenos indícios de variações por gênero e, apesar da pesquisa de avaliação rudimentar sem controle da população avaliadora, aponta para uma plataforma de apoio a tomada de decisão no trabalho criativo que pode atingir resultados satisfatórios, gerando uma boa amostra (ou seja, uma amostra bem-avaliada) para cada três composições. Isso significa que, em um intervalo curto de tempo (por volta de um a dois minutos), um usuário pode extrair uma melodia de dez a onze segundos, sem direitos autoriais, que pode ser utilizada para variadas finalidades (como o auxílio a composição musical, e o uso na produção de conteúdo multimídia).

Devido as limitações de escopo e cronograma desde trabalho, porém, a plataforma ainda se encontra em estado rudimentar. Para melhor atender o público geral, é possível expandir o sistema provendo serviços como a edição de melodias geradas, assim como melhorar o algoritmo de geração procedural para que sejam geradas melodias mais complexas, mais coesas, e com maior consistência (reduzindo o empreendimento de tempo para obter uma boa amostra).

Todavia, a principal limitação do projeto diz respeito ao formato de avaliação empregado: a realização de um esquema de avaliação em um ambiente controlado - no qual são conhecidas as características do usuário avaliador - pode resultar na confecção de uma base dados de treinamento ainda mais completa que a base disponibilizada por este trabalho.

O uso destes dados no treinamento de outros modelos de IA, ou mesmo no treinamento da IA descrita na Sub-Seção 4.2.3, pode acabar produzindo sistemas com taxas de aprovação ainda maiores e, por consequência, possibilitar a implementação de uma plataforma de apoio a composição ainda mais completa.

5.1 TRABALHOS FUTUROS

Com base nos resultados obtidos, é possível identificar diversos pontos de deficiência do modelo implementado, assim como limitações da própria literatura atual, principalmente em relação a estilização de áudios de acordo com gêneros musicais.

De forma geral, a aplicação da arquitetura de GANs pode beneficiar a qualidade da geração procedural de melodias devido seu foco na criação de falsificações autênticas, como as implementações comparadas por Li-Chia Yang, Chou e Yi-Hsuan Yang (2017). Esse *framework* permite, inclusive, a utilização de redes recorrentes como as LSTMs implementadas nesse

trabalho, com o benefício de acelerar o processo de treinamento através da competição minimax. Ou seja, a arquitetura GAN possibilita a consolidação de um contexto/espaço latente ainda mais robusto em comparação ao resultado obtido, mesmo se o gerador utilizado for similar as redes neurais empregadas neste trabalho.

Alinhado a finalidade de melhorar as produções do modelo, a implementação de redes secundárias de pós-processamento para estilização - como o modelo proposto por Fang *et al.* (2021) para resolver o problema da cópia de estilos na coloração automatizada de ilustrações - pode ampliar a adequação das melodias a gêneros musicais pré-definidos, ou até mesmo na imitação do estilo musical empregado por certos artistas e/ou movimentos artísticos.

Porém, além das técnicas de aprendizado de máquina, também é possível realizar a adição de algoritmos de filtragem e controle das progressões de notas, como os processos citados durante o Capítulo 4. Em especial, a remoção de saltos muito grandes entre as frequências de uma sequência pode auxiliar na redução da quantidade de progressões atonais geradas pelo modelo atuando apenas no pós-processamento das melodias geradas. Associado a esse processo, a extensão da duração de certas notas (como o início e o fim das progressões) pode refinar as produções da IA ao mesmo tempo que as frequências mais destoantes são sobre-escritas, possivelmente aumentando a taxa de aprovação das composições produzidas, ou seja, resultando em melodias melhores.

Por fim, destacamos também a ampliação das funcionalidades do sistema disponibilizado, que pode conter desde ferramentas para a exportação das melodias geradas, como também possibilitar a edição das sequências de notas, e a submissão das melodias alteradas pelos usuários. Além disso, também é possível implementar um sistema de avaliação mais detalhado do que o proposto por este trabalho, que diferencia as melodias somente entre “boas” e “ruins”.

Somadas, estas funcionalidades podem configurar não só um sistema de apoio a composição muito amplo, como também uma plataforma de coleta de dados sobre a composição de melodias que poderá ser utilizada na pesquisa das múltiplas tecnologias associadas.

REFERÊNCIAS

AMIDI, A.; AMIDI, S. **Recurrent Neural Networks cheatsheet**. [*S. l.: s. n.*]. Disponível em: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Acesso em: 13 out. 2022.

APEL, W. **Harvard dictionary of music**. [*S. l.*]: Belknap Press of Harvard University Press, 2003.

BARTLETT, J. C. Psychology and music: The understanding of melody and rhythm. In: 1. ed. [*S. l.*]: Lawrence Erlbaum Associates, Inc., 1993. cap. 3.

BARTLETT, J. C.; DOWLING, W. J. Scale structure and similarity of melodies. **Music Perception**, University of California Press, v. 5, n. 3, p. 285–314, 1988.

BRUNNER, G. *et al.* Symbolic music genre transfer with cyclegan. In: IEEE. 2018 ieee 30th international conference on tools with artificial intelligence (ictai). [*S. l.: s. n.*], 2018. P. 786–793.

COLAB, G. **Olá, este é o Colaboratory**. [*S. l.*]: Google Colab. Disponível em: https://colab.research.google.com/?utm_source=scs-index. Acesso em: 29 ago. 2022.

CONKLIN, D. Music generation from statistical models. In: SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND CREATIVITY IN THE ARTS AND SCIENCES, abr. 2003, London, UK. **Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences**. London, England: Society for the Study of Artificial Intelligence e the Simulation of Behaviour, 2003. P. 30–35.

DALLAGNOL, J. L. H. **utfpr-melody-gan**. [*S. l.*]: GitHub, 2022. Disponível em: https://github.com/JoaoLeonardo/utfpr-melody-lstm. Acesso em: 15 nov. 2022.

DEVOTO, M. **Triad**. In: ENCYCLOPEDIA Britannica. [*S. l.: s. n.*], 1998.

DICIO, D. O. d. P. **Inteligência**. In: [*s. l.*]: Dicio, Dicionário Online de Português. Disponível em: https://www.dicio.com.br/inteligencia/. Acesso em: 19 mai. 2022.

FANG, T.-T. *et al.* Stylized-Colorization for Line Arts. In: 2020 25th International Conference on Pattern Recognition (ICPR). [*S. l.: s. n.*], 2021. P. 2033–2040. DOI: 10.1109/ICPR48806.2021.9412756.

FLASK. **Flask, web development one drop at a time**. [*S. l.*]: Flask. Disponível em: https://flask.palletsprojects.com/en/2.2.x/. Acesso em: 27 out. 2022.

GÉRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & Tensorflow**. [*S. l.*]: Alta Books, 2019. ISBN 8550803812.

GHAHRAMANI, Z. Probabilistic machine learning and artificial intelligence. **Nature**, v. 521, p. 452–9, mai. 2015. DOI: 10.1038/nature14541.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016.

GOODFELLOW, I. J.; POUGET-ABADIE, J. *et al.* **Generative Adversarial Networks**. [*S. l.*]: arXiv, 2014. DOI: 10.48550/ARXIV.1406.2661. Disponível em: https://arxiv.org/abs/1406.2661.

GORDON, E. **Tonal and rhythm patterns, an objective analysis**: a taxonomy of tonal patterns and rhythm patterns and seminal experimental evidence of their difficulty and growth rate. [*S. l.*]: State University of New York Press Albany, 1976. P. 31–37. ISBN 0873953541.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional LSTM networks. In: PROCEEDINGS. 2005 IEEE International Joint Conference on Neural Networks, 2005. [*S. l.: s. n.*], 2005. v. 4, 2047–2052 vol. 4. DOI: 10.1109/IJCNN.2005.1556215.

HAN, J.; MORAGA, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In: SPRINGER. INTERNATIONAL workshop on artificial neural networks. [*S. l.: s. n.*], 1995. P. 195–201.

HINTON, G.; VINYALS, O.; DEAN, J. **Distilling the Knowledge in a Neural Network**. [*S. l.*]: arXiv, 2015. DOI: 10.48550/ARXIV.1503.02531.

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

HUANG, C.-Z. A. *et al.* The Bach Doodle: Approachable music composition with machine learning at scale. In: INTERNATIONAL Society for Music Information Retrieval (ISMIR). [*S. l.: s. n.*], 2019. Disponível em: https://goo.gl/magenta/bach-doodle-paper.

KINGSFORD, C.; SALZBERG, S. What are decision trees? **Nature biotechnology**, v. 26, p. 1011–3, out. 2008. DOI: 10.1038/nbt0908-1011.

LEMES, F. **Escala maior e menor — o que são e para que servem?** [*S. l.: s. n.*], ago. 2020. Disponível em: https://musicaevinho.com.br/escala-maior-e-menor-o-que-sao-e-para-que-servem/. Acesso em: 2 mai. 2022.

LI, S.; JANG, S.; SUNG, Y. Automatic melody composition using enhanced GAN. **Mathematics**, Multidisciplinary Digital Publishing Institute, v. 7, n. 10, p. 883, 2019.

LIMB, C. J. Structural and functional neural correlates of music perception. **The Anatomical Record Part A: Discoveries in Molecular, Cellular, and Evolutionary Biology: An Official Publication of the American Association of Anatomists**, Wiley Online Library, v. 288, n. 4, p. 435–446, 2006.

LYON, R.; SHAMMA, S. Auditory Representations of Timbre and Pitch. In: 1. ed. [*S. l.*]: Spring, 1996. cap. 6. (Springer Handbook of Auditory Research).

MAGENTA. **note-seq**. [*S. l.*]: GitHub, 2020. Disponível em: https://github.com/magenta/note-seq. Acesso em: 19 mai. 2022.

MAGENTA. **What does the data look like?** [*S. l.*]: Magenta. Disponível em: https://magenta.github.io/bach-doodle-visualization/dataset.html. Acesso em: 19 mai. 2022.

MUSICAL U, T. **Introduction to Time Signatures**. [*S. l.: s. n.*]. Disponível em: https://www.musical-u.com/learn/introduction-to-time-signatures/. Acesso em: 9 mai. 2022.

NAZERI, K.; NG, E.; EBRAHIMI, M. Image Colorization Using Generative Adversarial Networks. In: ARTICULATED Motion and Deformable Objects. [*S. l.*]: Springer International Publishing, 2018. P. 85–94. DOI: 10.1007/978-3-319-94544-6_9. Disponível em: https://doi.org/10.1007%2F978-3-319-94544-6_9.

NIELSEN, M. A. **Neural Networks and Deep Learning**. [*S. l.*]: Determination Press, 2015.

PENNACHIN, C.; GOERTZEL, B. Contemporary Approaches to Artificial General Intelligence. In: **Artificial General Intelligence**. Edição: Ben Goertzel e Cassio Pennachin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. P. 1–30. ISBN 978-3-540-68677-4. DOI: 10.1007/978-3-540-68677-4_1. Disponível em: https://doi.org/10.1007/978-3-540-68677-4_1.

PETERSEN, M. R. Musical analysis and synthesis in Matlab. **College Mathematics Journal**, JSTOR, p. 396–401, 2004.

PINTO, T. A. **MIDI, Musical Instrument Digital Interface**. [*S. l.*]: USP, abr. 2000. Disponível em: http://www2.eca.usp.br/prof/iazzetta/tutor/midi/midi1.html. Acesso em: 15 nov. 2022.

RISH, I. *et al.* An empirical study of the naive Bayes classifier. In: 22. IJCAI 2001 workshop on empirical methods in artificial intelligence. [*S. l.: s. n.*], 2001. v. 3, p. 41–46.

ROBERTS, A. *et al.* A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. arXiv, 2018. DOI: 10.48550/ARXIV.1803.05428.

ROBERTS, A. *et al.* **MusicVAE: Creating a palette for musical scores with machine learning.** [*S. l.*]: Magenta, mar. 2018. Disponível em: https://magenta.tensorflow.org/music-vae. Acesso em: 17 out. 2022.

RODRIGUES, É.; CONCI, A.; LIATSIS, P. Morphological classifiers. **Pattern Recognition**, v. 84, p. 82–96, 2018. ISSN 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2018.06.010. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320318302206.

SAK, H.; SENIOR, A.; BEAUFAYS, F. **Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition**. [*S. l.*]: arXiv, 2014. DOI: 10.48550/ARXIV.1402.1128. Disponível em: https://arxiv.org/abs/1402.1128.

SCHUETT, J. A Legal Definition of AI. **SSRN Electronic Journal**, SSRN, set. 2019. DOI: 10.2139/ssrn.3453632.

SCHUSTER, M.; PALIWAL, K. Bidirectional recurrent neural networks. **IEEE Transactions on Signal Processing**, v. 45, n. 11, p. 2673–2681, 1997. DOI: 10.1109/78.650093.

SETHARES, W. **Tuning, Timbre, Spectrum, Scale**. [*S. l.*]: Spring, jan. 2005. ISBN 1852337974.

SILLA, C. N.; KOERICH, A. L.; KAESTNER, C. A. A machine learning approach to automatic music genre classification. **Journal of the Brazilian Computer Society**, Springer, v. 14, n. 3, p. 7–18, 2008.

TROIANO, W. **What is 6/8 Time Signature?** [*S. l.: s. n.*]. Disponível em: https://www.libertyparkmusic.com/what-is-6-8-time-signature/. Acesso em: 9 mai. 2022.

TYMOCZKO, D. Scale networks and Debussy. **Journal of Music Theory**, JSTOR, v. 48, n. 2, p. 219–294, 2004.

WALSH, J. **Jazz Chord Progressions — The Ultimate Guide**. [*S. l.: s. n.*], jun. 2020. Disponível em: https://jazz-library.com/articles/chord-progressions/. Acesso em: 3 mai. 2022.

WANG, K. *et al.* Generative adversarial networks: introduction and outlook. **IEEE/CAA Journal of Automatica Sinica**, IEEE, v. 4, n. 4, p. 588–598, 2017.

WANG, X. *et al.* **GAN-generated Faces Detection: A Survey and New Perspectives (2022)**. [*S. l.*]: arXiv, 2022. DOI: 10.48550/ARXIV.2202.07145. Disponível em: https://arxiv.org/abs/2202.07145.

WILLIMEK, D. Why do Minor Chords Sound Sad? The Theory of Musical Equilibration and the Emotions of Chords. **Journal of Psychology & Psychotherapy**, v. 04, jan. 2014. DOI: 10.4172/2161-0487.1000139.

YANG, L.-C.; CHOU, S.-Y.; YANG, Y.-H. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation. In: ISMIR. [*S. l.: s. n.*], 2017.

ÍNDICE REMISSIVO