Harmonizing AI: Music Generation using Recurrent Neural Networks

Vitor Albuquerque de Paula
Programa de Pós-Graduação em Sistemas Inteligentes
Universidade Federal de São Paulo (UNIFESP)
São Paulo, Brasil
vitor.ap@proton.pm

***Abstract*—This paper presents a methodical approach to generating music using Recurrent Neural Networks (RNNs), a type of neural network well-suited for sequential data processing. The core of our research involves transforming musical data, specifically notes, into a structured DataFrame format, enabling effective machine learning applications. We detail the process of converting this structured data into MIDI files, a standard in digital music representation. Our primary focus is on the development of a training dataset tailored for RNNs to capture the temporal and sequential nature of music. We discuss the choice of RNNs for this task, highlighting their advantages in handling time-series data. A significant part of our study is dedicated to the implementation of a windowing technique, essential for organizing the data to facilitate learning. The outcomes of our research showcase the ability of RNNs to learn from existing musical compositions and to generate new, cohesive musical pieces. This work contributes to the growing field of AI in music generation, offering insights and methodologies that could be beneficial for similar applications in AI-driven creative processes.**

***Index Terms*—Artificial Intelligence, Music Generation, Recurrent Neural Networks, MIDI, Data Transformation, Sequential Data, Machine Learning, Creative AI, Temporal Data Processing, Neural Network Architectures**

I. Introdução

A. Objetivo do Projeto

O propósito central deste projeto é explorar e demonstrar o potencial das Redes Neurais Recorrentes (RNNs) na geração de música. Especificamente, buscamos desenvolver um sistema que possa não apenas aprender padrões musicais a partir de um conjunto de dados existente, mas também gerar novas composições musicais que sejam harmonicamente ricas e esteticamente agradáveis. Este objetivo é alcançado através de uma série de etapas metodológicas, incluindo a transformação de dados de música em um formato estruturado, a utilização de arquivos MIDI para representação digital de música, e a formulação de um dataset de treinamento otimizado para RNNs. O projeto visa contribuir para o campo da inteligência artificial aplicada à criação artística, oferecendo novos insights e técnicas para a geração de música com redes neurais.

B. Contexto e Importância

A interseção entre a inteligência artificial e a criação artística tem ganhado notável atenção nos últimos anos. A capacidade das redes neurais de processar e aprender a partir de grandes volumes de dados tem aberto portas para novas formas de expressão criativa. No campo da música, este avanço tem implicações significativas, desde a composição assistida por IA até a geração autônoma de peças musicais completas. O uso de RNNs, em particular, é promissor devido à sua habilidade de lidar com sequências temporais, uma característica intrínseca da música. Neste contexto, nosso estudo se torna relevante, pois busca ampliar a compreensão de como as RNNs podem ser efetivamente aplicadas na geração de música. Além disso, ao desenvolver métodos que permitem às máquinas criar música que ressoa em um nível humano, estamos expandindo os limites da criatividade assistida por computador e abrindo novos caminhos para a inovação na arte e tecnologia.

II. Configuração e Preparação

A. Configuração do Ambiente

O desenvolvimento do projeto foi conduzido primariamente através do Jupyter Notebook, selecionado por sua praticidade e eficácia para experimentação interativa e visualização de dados em tempo real. Python, por sua predominância no campo científico e de machine learning, além de um ecossistema rico de bibliotecas, foi a linguagem de programação adotada.

Dentre o conjunto de ferramentas utilizadas, destacam-se o Pandas, para a manipulação e análise de dados, que nos permitiu preparar e estruturar as informações musicais de maneira eficaz, e o TensorFlow, uma biblioteca open-source para aprendizado de máquina, desenvolvida pelo Google Brain, que serviu como base para a construção e treinamento de nossas Redes Neurais Recorrentes. O Keras, interface de alto nível para o TensorFlow, foi também empregado para simplificar a arquitetura e o protótipo das redes.

O código deste projeto foi baseado em um código disponibilizado na pagina do TensorFlow [1], adaptado e expandido para adequar-se ao nosso conjunto de dados mais limitado em volume. Funções chave foram reescritas e otimizadas para promover a convergência do modelo em um contexto de dados restritos, um desafio significativo que exigiu um ajuste fino do modelo e função de perda.

B. Configuração do Ambiente

O desenvolvimento do projeto foi conduzido primariamente através do Jupyter Notebook, selecionado por sua praticidade e eficácia para experimentação interativa e visualização de dados

em tempo real. Python, por sua predominância no campo científico e de machine learning, além de um ecossistema rico de bibliotecas, foi a linguagem de programação adotada.

Dentre o conjunto de ferramentas utilizadas, destacam-se o Pandas, para a manipulação e análise de dados, que nos permitiu preparar e estruturar as informações musicais de maneira eficaz, e o TensorFlow, uma biblioteca open-source para aprendizado de máquina, desenvolvida pelo Google Brain, que serviu como a espinha dorsal para a construção e treinamento de nossas Redes Neurais Recorrentes. O Keras, interface de alto nível para o TensorFlow, foi também empregado para simplificar a arquitetura e o protótipo das redes.

O código deste projeto foi fortemente inspirado por um tutorial de geração de música do TensorFlow [1], adaptado e expandido para adequar-se ao nosso conjunto de dados mais limitado em volume. Funções chave foram reescritas e otimizadas para promover a convergência do modelo em um contexto de dados restritos, um desafio significativo que exigiu uma abordagem meticulosa de engenharia de características e ajuste fino do modelo.

C. Descrição do Dataset

Para este projeto, o dataset foi adquirido do Kaggle, uma plataforma conhecida por hospedar uma variedade de datasets para competições e pesquisas em ciência de dados e aprendizado de máquina. O conjunto de dados selecionado compreende aproximadamente 50 peças musicais do renomado compositor Frédéric Chopin. Chopin, um compositor e virtuose pianista do século XIX, é célebre por suas obras que exibem profundidade técnica e expressiva, tornando-o uma escolha ideal para explorar a complexidade musical através de aprendizado de máquina.

As músicas de Chopin, disponíveis no dataset em formato MIDI, oferecem uma rica variedade de harmonias, melodias e ritmos. O formato MIDI é particularmente adequado para este estudo, pois fornece informações detalhadas sobre as notas, duração, intensidade e instrumentação, aspectos cruciais para a análise musical. O processo de transformação destas composições MIDI em um formato de DataFrame permite uma análise mais aprofundada e facilita o treinamento da rede neural.

A escolha de focar nas obras de Chopin para treinar a rede neural tem um duplo propósito. Primeiramente, permite que o modelo aprenda a partir de composições de alta qualidade e complexidade técnica, potencialmente levando à geração de peças musicais que refletem o estilo e a sofisticação de Chopin. Em segundo lugar, concentra-se em um conjunto de obras coeso, o que pode ajudar a garantir que os padrões aprendidos pela rede neural sejam consistentes e estilisticamente relevantes.

D. Fonte dos Dados

Os dados são oriundos do Kaggle [2], uma plataforma popular para competições de ciência de dados e aprendizado de máquina.

III. Conversão de Notas Musicais para DataFrame

A. Processo de Conversão

A conversão de dados musicais, especificamente notas e acordes, para um formato de DataFrame é um passo crítico no nosso projeto de geração de música com redes neurais. Esse processo começa com a leitura dos arquivos MIDI, que contêm as composições de Chopin. Cada arquivo MIDI é uma representação digital que inclui informações detalhadas sobre as notas tocadas, incluindo o pitch (altura da nota), a duração, a intensidade e o tempo de início de cada nota ou acorde.

Utilizamos bibliotecas especializadas em Python, como a music21, para processar esses arquivos MIDI. Este processo envolve a extração das informações de cada nota e acorde, convertendo-as em uma representação textual ou numérica que pode ser tratada por algoritmos de aprendizado de máquina. Por exemplo, um dó central (C4) em um piano pode ser representado pelo número 60, enquanto a duração da nota pode ser quantificada em frações de tempo relativo ao andamento da peça.

Após a extração, estas informações são organizadas em um DataFrame, uma estrutura de dados tabular muito utilizada em análise de dados e aprendizado de máquina. Cada linha do DataFrame representa uma nota ou um acorde, com colunas correspondentes a diferentes características, como altura da nota, duração, intensidade, e outras propriedades musicais relevantes.

B. Importância da Conversão

A transformação de dados musicais em um DataFrame é fundamental por várias razões. Primeiramente, permite uma manipulação mais eficiente e uma análise detalhada dos dados musicais. Em um DataFrame, podemos aplicar facilmente operações como filtragem, agrupamento e visualização, o que é crucial para entender as tendências e padrões nas composições de Chopin.

Além disso, esta conversão facilita a aplicação de técnicas de aprendizado de máquina. Modelos de rede neural, como as RNNs, requerem dados de entrada em um formato estruturado e quantificado. Ao converter as notas musicais em um DataFrame, estamos efetivamente preparando os dados para serem alimentados na rede neural, permitindo que o modelo aprenda as complexidades e nuances das composições de Chopin.

Por fim, esta abordagem permite uma maior flexibilidade no tratamento dos dados. Podemos, por exemplo, alterar a granularidade dos dados, considerando características adicionais ou simplificando a representação conforme necessário para diferentes fases do projeto. Esta adaptabilidade é essencial para otimizar o desempenho do modelo e alcançar resultados de geração musical mais refinados e artisticamente expressivos.

IV. Criação de Arquivos MIDI

A. Metodologia de Criação

A etapa final do nosso projeto envolve a conversão dos dados de saída da rede neural, formatados em DataFrame,

de volta para arquivos MIDI. Este processo é essencial para transformar as sequências de notas e acordes geradas pela rede em composições musicais executáveis.

Utilizamos a biblioteca music21 para mapear cada elemento do DataFrame – que representa notas e acordes – em objetos MIDI. Estes objetos são organizados em uma estrutura sequencial para formar uma peça musical completa. O resultado é um arquivo MIDI, que encapsula a nova composição gerada pela rede.

1) Importância dos Arquivos MIDI: O uso de arquivos MIDI é crucial em nossa pesquisa. Este formato oferece uma representação digital que vai além das notas, incluindo elementos como tempo e intensidade, essenciais para uma reprodução fiel da música. A versatilidade do MIDI facilita a edição e reprodução das composições em uma ampla gama de plataformas, um aspecto importante para a validação e o teste do modelo.

Os arquivos MIDI permitem não apenas uma avaliação técnica das saídas do modelo, mas também fornecem um meio prático para audição e análise das composições geradas. Este aspecto é fundamental para avaliar a eficácia do modelo em produzir música que seja tanto tecnicamente correta quanto esteticamente agradável.

V. Formulação do Dataset de Treinamento

A preparação do dataset para treinamento é uma etapa crucial em qualquer projeto de aprendizado de máquina, e no nosso caso, onde lidamos com a geração de música, esta etapa assume uma complexidade adicional devido à natureza intrinsecamente sequencial e temporal da música.

1) Preparação do Dataset: Inicialmente, o conjunto de dados composto por músicas de Chopin em formato MIDI foi processado para extrair características musicais relevantes, como a altura das notas, duração, intensidade e tempo. Cada música foi então desmembrada em uma série de notas e acordes, que foram convertidos em um formato padronizado adequado para análise.

Após a conversão, as sequências de notas e acordes foram organizadas em um DataFrame. Para tornar esses dados mais digeríveis para o modelo de rede neural, implementamos uma técnica de ”tokenização”, onde cada nota ou acorde é representado por um token único. Esta abordagem reduz a complexidade do dataset e ajuda a rede neural a aprender mais eficientemente as relações entre as sequências de notas.

2) Técnicas e Abordagens Utilizadas: Uma das técnicas chave empregadas na preparação do dataset foi o ”janelamento”. Neste processo, o dataset foi dividido em uma série de sequências menores, ou ”janelas”, cada uma contendo um número fixo de notas. Esta abordagem é fundamental para treinar a rede neural, pois permite que ela aprenda a prever a próxima nota em uma sequência com base nas notas anteriores.

Além disso, foi necessário normalizar os dados para garantir que a rede neural tratasse todas as características com igual importância. A normalização envolveu ajustar a escala das características numéricas do dataset para que tivessem um intervalo comum, facilitando o processo de aprendizado da rede.

VI. Geração de Dataset com TensorFlow

A. Utilização do TensorFlow

O TensorFlow, uma das bibliotecas de aprendizado de máquina mais populares e poderosas, desempenha um papel fundamental na geração do dataset para o nosso projeto de geração de música com Redes Neurais Recorrentes (RNN). Utilizando o TensorFlow, transformamos os dados brutos de música em um formato estruturado e adequado para treinamento de modelos de aprendizado de máquina.

O primeiro passo nesse processo envolve a leitura e processamento dos arquivos MIDI, convertendo-os em sequências de dados representando notas, acordes e suas características. Utilizando as funcionalidades do TensorFlow, especificamente o módulo tf.data, criamos um dataset que pode ser eficientemente manipulado e preparado para treinamento. O tf.data oferece uma interface flexível e otimizada para a construção de pipelines de processamento de dados, permitindo operações como mapeamento, filtragem e transformação dos dados de forma eficaz.

1) Explicação Técnica: Tecnicamente, a geração do dataset no TensorFlow envolve a criação de um objeto Dataset que contém as sequências de notas musicais. Utilizamos funções como map, batch, e window para transformar estas sequências em formatos adequados para o treinamento da rede neural. Por exemplo, a técnica de janelamento, como mencionado anteriormente, é implementada para criar sequências de tamanho fixo, que são então usadas para prever a próxima nota na sequência.

Outro aspecto técnico importante é a otimização do carregamento e processamento dos dados, crucial para lidar com grandes volumes de informações musicais. Utilizando a capacidade do TensorFlow de realizar operações paralelas e a sua eficiente gestão de memória, conseguimos processar e preparar o dataset de forma rápida e eficiente.

2) Benefícios da Abordagem: A utilização do TensorFlow para a geração do dataset traz vários benefícios para o projeto:

- **Support Vector Machine (SVM)**:O TensorFlow é otimizado para operar com grandes conjuntos de dados, oferecendo um processamento rápido e eficiente, o que é crucial para o treinamento de modelos de aprendizado de máquina.
- **Flexibilidade**: A biblioteca permite uma ampla gama de operações de processamento de dados, oferecendo a flexibilidade necessária para ajustar e preparar o dataset de acordo com as especificidades do projeto.
- **Integração com Modelos de Aprendizado de Máquina**: Como o TensorFlow é uma biblioteca abrangente para aprendizado de máquina, ela oferece uma integração suave entre a preparação de dados e o treinamento de modelos, facilitando o desenvolvimento do sistema de geração de música.
- **Capacidade de Paralelização**: O TensorFlow suporta o processamento paralelo de dados, o que acelera significativamente a preparação do dataset, especialmente quando

lidamos com grandes volumes de dados. Isso foi vital para tornar o projeto viável.

VII. Aprendizado Baseado em RNN para Sequências Musicais

A. Escolha de Redes Neurais Recorrentes (RNN)

A decisão de utilizar Redes Neurais Recorrentes [3] [4] (RNN) para este projeto de geração de música baseia-se nas características únicas das RNNs em lidar com dados sequenciais. Música, por sua natureza, é uma forma de arte temporal, onde a relação entre notas consecutivas e padrões rítmicos desempenha um papel crucial na criação de harmonia e melodia. RNNs são particularmente adequadas para este tipo de dados, pois são projetadas para processar sequências de entrada, mantendo um 'estado' ou 'memória' das entradas anteriores. Essa capacidade de considerar informações passadas torna as RNNs ideais para prever a próxima nota em uma sequência musical, entendendo o contexto e a progressão até aquele ponto.

B. Breve Introdução a Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNNs) são uma classe de redes neurais especializadas no processamento de sequências de dados. Diferentemente das redes neurais tradicionais, que processam cada entrada de forma independente, as RNNs possuem a capacidade de manter um estado interno, ou "memória", que captura informações sobre entradas anteriores. Esta característica é fundamental para tarefas onde o contexto sequencial é importante, como é o caso da música.

Tecnicamente, uma RNN processa uma sequência de entrada passo a passo, mantendo um estado oculto que é atualizado a cada passo. Em cada etapa, a rede considera tanto a nova entrada quanto o estado oculto anterior para produzir uma saída e atualizar o estado. Esta capacidade de "lembrar" informações anteriores permite que a RNN capture dependências temporais e padrões sequenciais complexos.

C. Estrutura e Função de Perda do Modelo

Para a função de perda, utilizamos uma versão personalizada da Huber Loss [5], denominada HuberLossPositivePressure. A Huber Loss combina as propriedades do erro quadrático médio (MSE) para erros menores e do erro absoluto médio (MAE) para erros maiores, sendo eficaz ao lidar com outliers nos dados. Esta característica torna a Huber Loss uma escolha adequada para a previsão das características como step e duration do nosso modelo, especialmente considerando a possível presença de outliers.

A classe HuberLossPositivePressure estende a funcionalidade da Huber Loss padrão, introduzindo um termo adicional que penaliza previsões negativas. O parâmetro positive-pressure-weight é usado para controlar a intensidade dessa penalidade. Esta modificação é crucial para as características step e duration, onde valores negativos são impraticáveis e indesejáveis. A adição desse termo de pressão positiva garante que o modelo seja penalizado por prever valores que não fazem sentido no contexto musical, promovendo assim previsões mais precisas e realistas.

D. Arquitetura da RNN

A arquitetura da RNN em nosso projeto é focada na eficácia de lidar com as complexidades das sequências musicais. A rede é composta por uma camada de entrada seguida por uma camada LSTM (Long Short-Term Memory) e, finalmente, múltiplas camadas de saída para diferentes características da música.

- **Camada de Entrada**: Inicialmente, os dados são processados pela camada de entrada, que é preparada para lidar com a sequência de notas.
- **Camadas LSTM**:Após a camada de entrada, utilizamos duas camadas LSTM com 128 unidades. As células LSTM são essenciais para aprender e manter informações sobre sequências de notas ao longo do tempo, superando o problema do desvanecimento do gradiente e permitindo que a rede aprenda dependências de longo alcance.
- **Camadas de Saída**:A rede então se ramifica em múltiplas saídas, cada uma tratando de uma característica diferente da música, como pitch, duração e tempo. Essas camadas de saída são estruturadas para capturar e prever as nuances específicas de cada aspecto da sequência musical.

VIII. Técnica de Janelamento para Estruturação de Dados

Para preparar nossos dados para o treinamento da Rede Neural Recorrente (RNN), adotamos uma técnica conhecida como janelamento. Essa abordagem envolve segmentar o dataset em uma série de "janelas", onde cada janela encapsula uma sequência de notas de um comprimento predefinido, especificado pelo hiperparâmetro sequence length. Essencialmente, cada janela consiste em uma série de características de entrada (input features) e a próxima nota na sequência, que serve como rótulo (label).

A aplicação do janelamento é crucial para a preparação eficaz do dataset no contexto de aprendizado de máquina, particularmente para RNNs, que são projetadas para processar sequências de dados. Este método permite que o modelo aprenda as dependências temporais e padrões intrínsecos na música. A extensão do contexto temporal que o modelo leva em consideração para cada previsão é determinada pelo sequence length. Por exemplo, uma janela com um sequence length de 20 notas permite que a RNN aprenda padrões e dependências que abrangem essas 20 notas.

Além disso, o janelamento também ajuda a aumentar a quantidade de dados de treinamento disponíveis. Ao deslizar a janela sequencialmente sobre o conjunto de dados, criamos múltiplas sequências de treinamento a partir de uma única peça musical, cada uma começando em diferentes pontos na sequência original. Isso não só enriquece o conjunto de dados, mas também ajuda o modelo a aprender de maneira mais robusta, capturando diversas nuances e variações dentro de uma mesma composição musical.

O janelamento, portanto, não apenas transforma os dados musicais em um formato propício para as forças únicas das RNNs, mas também estabelece a base para um treinamento

efetivo e a geração subsequente de sequências musicais. Ao ajustar o sequence length, podemos controlar o equilíbrio entre a quantidade de contexto temporal fornecido ao modelo e a granularidade das previsões realizadas, o que é fundamental para otimizar o desempenho da rede na tarefa de geração de música.

IX. Interpretação dos Resultados

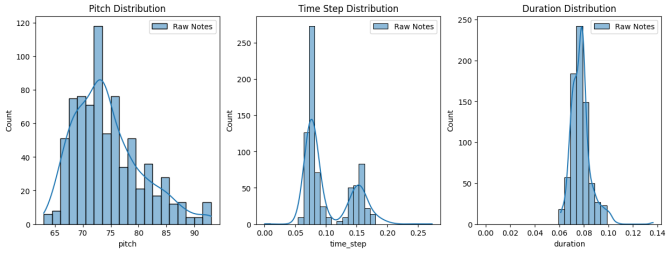

Fig. 1. Distribuição das características das notas de uma musica do Chopin

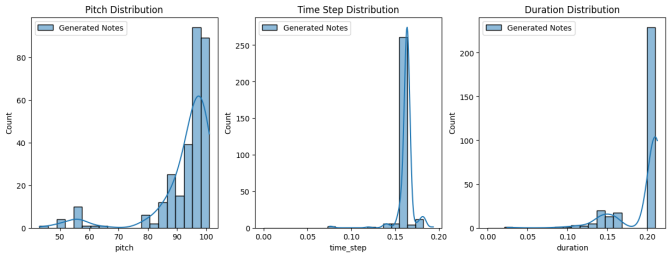

Fig. 2. Distribuição das características das notas de uma musica gerada pela rede

A análise das distribuições de características das notas geradas em comparação com uma amostra de música de referência revelou um desempenho promissor do modelo. As figuras 1 e 2 ilustram que o modelo capturou aspectos fundamentais da estrutura musical, como a distribuição de pitches e o ritmo indicado pelos passos de tempo. A coerência e coesão observadas nas sequências musicais geradas validam a capacidade do modelo de sintetizar música que mantém a essência estilística da base de dados original. No entanto, ainda há diferenças perceptíveis entre as composições geradas e as originais, o que sugere um potencial para melhorias. A ampliação da base de dados e a experimentação com diferentes topologias de rede podem aprimorar a capacidade do modelo de imitar estilos musicais específicos com maior precisão. Estes avanços poderiam, eventualmente, levar a um sistema de geração de música que não só emula, mas também expande o repertório musical humano com novas peças criativas e tecnicamente ricas.

X. Conclusão

Este estudo explorou a aplicabilidade das Redes Neurais Recorrentes na tarefa de geração de música, com foco particular nas obras de Frédéric Chopin. Os resultados quantitativos demonstraram que o modelo proposto é capaz de aprender e replicar a estrutura musical subjacente presente nos dados de treinamento, como evidenciado pelas distribuições de pitch, duration e time-step. Qualitativamente, as sequências geradas apresentaram uma coerência musical que ressoa com as características estilísticas das composições de Chopin, ainda que com margem para identificação de diferenças.

As perspectivas futuras para este trabalho incluem a expansão da base de dados de treinamento, o que poderia fornecer ao modelo uma variedade maior de exemplos musicais para aprender, potencialmente levando a uma geração de música mais diversificada e rica. Além disso, a experimentação com diferentes arquiteturas de rede, incluindo variações mais profundas e complexas, poderia oferecer novas maneiras de capturar as nuances da música. Investigar métodos de regularização aprimorados e técnicas de otimização também podem contribuir para a evolução do modelo, permitindo uma síntese musical que não apenas imita, mas também expande o repertório musical de maneiras criativas e inovadoras. Este estudo, portanto, estabelece um ponto de partida promissor para futuras incursões no campo emergente da composição musical assistida por inteligência artificial.

References

[1] TensorFlow. (n.d.). *Music generation with an RNN*. Recuperado em 15 de dezembro de 2023, de https://www.tensorflow.org/tutorials/audio/music-generation.

[2] Kaggle. (n.d.). *Classical Music MIDI*. Recuperado em 15 de dezembro de 2023, de https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Tech. Rep. ICS 8504, Institute for Cognitive Science, University of California, San Diego, California, Sept. 1985.

[4] M. I. Jordan, "Serial order: a parallel distributed processing approach," Tech. Rep. ICS 8604, Institute for Cognitive Science, University of California, San Diego, California, May 1986.

[5] P. J. Huber, "Robust Estimation of a Location Parameter," Annals of Statistics, vol. 53, no. 1, pp. 73–101, 1964. doi:10.1214/aoms/1177703732.