**Experimento de Machine Learning usando técnicas de LSTM para gerar melodias originais baseado em um arquivo MIDI.**

**- O protocolo MIDI**

Lá nos anos 70, no auge do Jazz, Funk e Soul, os instrumentos musicais eletrônicos foram se tornando cada vez mais populares e acessíveis.   
  
Em contrapartida, os sintetizadores analógicos mais antigos usufruíam do sistema monofônico, do qual reproduz o som de apenas uma nota por vez (não importa quantas você aperte ao mesmo tempo).  
  
Nesse sistema, a força produzida é dedicada a apenas essa nota, o que gera uma tensão que permitia que outros instrumentos pudessem ser controlados através dela. E os fabricantes como Yamaha, Korg, etc utilizavam esta tensão para possibilitar ligar/desligar outros instrumentos em conjunto, de modo que um dispositivo poderia controlar um ou mais outros dispositivos, por exemplo.  
  
Mas este sistema era inadequado para controle de sintetizadores digitais e polifônicos mais recentes. Por isso, alguns fabricantes criaram sistemas que permitiram a interligação do seu próprio equipamento.  
Mas eram sistemas exclusivos, impossibilitando os sistemas de um fabricante de sincronizar com os de outro.

O protocolo MIDI solucionou esse (e muitos outros) problemas, unificou o sistema de comunicação, revolucionou a indústria musical e não é à toa, esse tipo de comunicação permite que com apenas um computador e um teclado MIDI você controle vários instrumentos diferentes, simulando com excelência sintetizadores clássicos, drum-machines, sinfonias... Além de tornar as configurações de palco muito mais portáteis.  
  
Mas o mais empolgante de tudo isso e o motivo pelo qual esse artigo está sendo escrito é a quantidade de arquivos MIDI que é gerada de forma aberta pela comunidade. Seja de composições próprias, orquestas, ou um MIDI generativo via Machine Learning, que será o caso aqui.

**- Machine Learning com LSTM**

LSTM é uma arquitetura específica de rede neural recorrente amplamente utilizadas em tarefas que envolvem dados sequenciais, como reconhecimento de fala, tradução automática, geração de texto...

Isso por causa de sua capacidade única de lidar com dependências de longo prazo e preservar informações relevantes ao longo do tempo. Aqui estão algumas razões pelas quais as LSTMs são eficazes em tarefas envolvendo sequências temporais, como geração de melodias midi, por exemplo:

1. **Memória de Longo Prazo:**
   * A capacidade de manter informações em sua memória de longo prazo permite que as LSTMs capturem dependências temporais de longo alcance em sequências. Isso é crucial em tarefas onde eventos distantes no tempo podem afetar o resultado final.
2. **Portões de Controle:**
   * As LSTMs utilizam portões de controle, como o portão de esquecimento, o portão de entrada e o portão de saída. Esses mecanismos de portão permitem que a rede aprenda quais informações reter e quais descartar em diferentes pontos ao longo de uma sequência.
3. **Prevenção de Desaparecimento do Gradiente:**
   * As LSTMs são projetadas para resolver o problema do desaparecimento do gradiente, que é comum em redes recorrentes mais simples. A estrutura das LSTMs permite um fluxo mais eficiente de gradientes durante o treinamento, facilitando a aprendizagem de dependências temporais complexas.
4. **Aprendizado Hierárquico:**
   * LSTMs têm a capacidade de aprender representações hierárquicas em dados sequenciais. Isso significa que elas podem capturar padrões em diferentes níveis de abstração, tornando-as úteis para tarefas complexas que exigem compreensão profunda da estrutura sequencial dos dados.
5. **Aplicações Específicas:**
   * Tarefas como reconhecimento de fala, tradução automática, geração de texto e previsão de séries temporais envolvem a compreensão de padrões temporais complexos e dependências a longo prazo. As LSTMs, com sua capacidade de modelar efetivamente sequências temporais, mostraram-se particularmente eficazes nessas aplicações.
6. **Adaptação a Diferentes Comprimentos de Sequências:**
   * LSTMs podem lidar com sequências de comprimentos variáveis, tornando-as versáteis para tarefas onde as entradas podem ter diferentes extensões temporais.

**- O experimento**

Este projeto utiliza o TensorFlow e Keras para criar um modelo LSTM que aprende padrões a partir de dados MIDI para gerar novas melodias.

O TensorFlow é uma biblioteca de aprendizado de máquina de baixo nível que fornece a infraestrutura para construção e treinamento de modelos e o Keras é a API padrão para construir modelos de Machine Learning com o TensorFlow.

Vamos analisar o código:

pythonCopy code

import numpy as np from music21 import converter, instrument, stream, note, chord from tensorflow.keras.models import Sequential from tensorflow.keras.layers import LSTM, Dense

1. **Importação de Bibliotecas:**
   * O código começa importando as bibliotecas necessárias. **numpy** é utilizado para manipulação numérica, **music21** para processar dados MIDI, e **tensorflow** e **keras** para construir e treinar o modelo LSTM.

pythonCopy code

# Carregar dados MIDI midi\_path = 'caminho\_para\_seu\_arquivo\_midi.mid' midi\_stream = converter.parse(midi\_path) notes = [] for element in midi\_stream.flat: if isinstance(element, note.Note): notes.append(str(element.pitch)) elif isinstance(element, chord.Chord): notes.append('.'.join(str(n) for n in element.normalOrder))

1. **Carregamento de Dados MIDI:**
   * O código carrega um arquivo MIDI especificado em **midi\_path** usando a biblioteca **music21**. Ele itera sobre os elementos do stream MIDI, extrai as notas e acordes, convertendo-os para uma representação de string que inclui a altura da nota ou o número dos acordes.

pythonCopy code

# Preparar sequências para o modelo n\_vocab = len(set(notes)) network\_input, network\_output = prepare\_sequences(notes, n\_vocab)

1. **Preparação de Sequências para o Modelo:**
   * A função **prepare\_sequences** converte as sequências de notas em dados numéricos que podem ser usados para treinar o modelo. Isso inclui mapear notas para números inteiros, criar sequências de entrada e saída, e normalizar os dados.

pythonCopy code

# Criar e treinar o modelo model = create\_network(network\_input, n\_vocab) model.fit(network\_input, network\_output, epochs=50, batch\_size=64)

1. **Criação e Treinamento do Modelo:**
   * A função **create\_network** constrói o modelo LSTM usando a biblioteca Keras. O modelo é treinado com os dados preparados usando **model.fit**.

pythonCopy code

# Gerar novas melodias generated\_notes = generate\_notes(model, network\_input, pitchnames, n\_vocab)

1. **Geração de Novas Melodias:**
   * A função **generate\_notes** utiliza o modelo treinado para gerar novas sequências de notas. O ponto de partida é escolhido aleatoriamente.

pythonCopy code

# Criar um stream MIDI a partir das notas geradas offset = 0 output\_notes = [] for pattern in generated\_notes: if ('.' in pattern) or pattern.isdigit(): notes\_in\_chord = pattern.split('.') notes = [] for current\_note in notes\_in\_chord: new\_note = note.Note(int(current\_note)) new\_note.storedInstrument = instrument.Piano() notes.append(new\_note) new\_chord = chord.Chord(notes) new\_chord.offset = offset output\_notes.append(new\_chord) else: new\_note = note.Note(pattern) new\_note.offset = offset new\_note.storedInstrument = instrument.Piano() output\_notes.append(new\_note) offset += 0.5 # Você pode ajustar o espaçamento entre as notas geradas # Salvar as notas geradas em um novo arquivo MIDI midi\_stream\_generated = stream.Stream(output\_notes) midi\_stream\_generated.write('midi', fp='output\_generated.mid')

1. **Criação de Melodia MIDI a partir das Notas Geradas:**
   * O código cria um stream MIDI a partir das notas geradas e salva-as em um novo arquivo MIDI.

É importante notar que este é um exemplo básico e pode precisar de ajustes para atender às suas necessidades específicas. Por exemplo, você pode experimentar com diferentes arquiteturas de modelo, hiperparâmetros e estratégias de geração para melhorar a qualidade das melodias geradas. O código fornece uma base que você pode expandir e ajustar conforme necessário.