Roteiro de atividade prática

Nome: Turma: .

**Título da atividade: Gerando notas sintéticas**

**Objetivos**

Visualizar os resultados e explorar o código discutido em aula.

**Lista de materiais**

* Um computador com *browser* de internet.
* Caderno para anotações.
* 1 caneta.

**Procedimento experimental**

1. Acesse o site Colab do Google e crie um notebook.
2. Copie o código abaixo no bloco de notas e cole-o no notebook.

import numpy as np

import tensorflow as tf

# Definir o tamanho do batch

batch\_size = 16

# Definir o tamanho da sequência de notas

seq\_len = 128

# Definir o número de notas

n\_notes = 128

# Definir a função de perda

def loss\_fn(real, fake):

return tf.reduce\_mean(tf.math.square(real - fake))

# Definir o gerador

class Generator(tf.keras.Model):

def \_\_init\_\_(self):

super(Generator, self).\_\_init\_\_()

self.dense1 = tf.keras.layers.Dense(128, activation='relu')

self.dense2 = tf.keras.layers.Dense(128, activation='relu')

self.dense3 = tf.keras.layers.Dense(n\_notes, activation='tanh')

def call(self, inputs):

x = inputs

x = self.dense1(x)

x = self.dense2(x)

x = self.dense3(x)

return x

# Definir o discriminador

class Discriminator(tf.keras.Model):

def \_\_init\_\_(self):

super(Discriminator, self).\_\_init\_\_()

self.dense1 = tf.keras.layers.Dense(128, activation='relu')

self.dense2 = tf.keras.layers.Dense(128, activation='relu')

self.dense3 = tf.keras.layers.Dense(1, activation='sigmoid')

def call(self, inputs):

x = inputs

x = self.dense1(x)

x = self.dense2(x)

x = self.dense3(x)

return x

# Instanciar os modelos

generator = Generator()

discriminator = Discriminator()

# Compilar os modelos

generator.compile(loss=loss\_fn, optimizer='adam')

discriminator.compile(loss='binary\_crossentropy', optimizer='adam')

# Criar o loop de treinamento

for epoch in range(100):

# Gerar um batch de notas aleatórias

real\_notes = np.random.randint(0, n\_notes, (batch\_size, seq\_len))

# Passar as notas aleatórias pelo discriminador

real\_output = discriminator(real\_notes)

# Gerar um batch de notas sintéticas

fake\_notes = generator(np.random.randint(0, 1, (batch\_size, seq\_len)))

# Passar as notas sintéticas pelo discriminador

fake\_output = discriminator(fake\_notes)

# Atualizar o discriminador

discriminator.train\_on\_batch(real\_notes, np.ones((batch\_size, 1)))

discriminator.train\_on\_batch(fake\_notes, np.zeros((batch\_size, 1)))

# Atualizar o gerador

generator.train\_on\_batch(np.random.randint(0, 1, (batch\_size, seq\_len)), np.ones((batch\_size, 1)))

# Gerar notas sintéticas

generated\_notes = generator(np.random.randint(0, 1, (1, seq\_len)))

# Mostrar as notas sintéticas

print(generated\_notes)

1. Clique em “Play” para executar
2. Compreenda o papel do gerador e do discriminador no contexto de GAN, bem como a função de perda e o processo de treinamento.
3. Execute o código para gerar sequências de notas musicais sintéticas.
4. Observe a saída do código (as notas geradas) e analise a qualidade e a viabilidade das sequências musicais produzidas.
5. Identifique pontos fortes e fracos do modelo atual, considerando a qualidade das notas musicais geradas.
6. Proponha melhorias ou ajustes no modelo, como alterações nas camadas do gerador ou discriminador, ajustes nos parâmetros de treinamento ou mesmo sugestões para aprimorar a função de perda para obter resultados mais realistas.
7. Discuta como esse modelo pode ser integrado em um produto de software mais amplo para geração de música.
8. Apresente ideias sobre como esse sistema poderia ser utilizado em contextos reais, como na criação de trilhas sonoras para jogos, filmes, ou como uma ferramenta para compositores e produtores musicais.

|  |
| --- |
|  |
|  |
|  |
|  |