

EDUARDO OTTO RODRIGUES
LUCAS DE SANTANA SCOCCA

COMPOSITOR INTELIGENTE: COMPOSITOR MUSICAL
BASEADO EM INTELIGÊNCIA ARTIFICIAL

São Paulo
2021

EDUARDO OTTO RODRIGUES
LUCAS DE SANTANA SCOCCA

COMPOSITOR INTELIGENTE: COMPOSITOR MUSICAL
BASEADO EM INTELIGÊNCIA ARTIFICIAL

Trabalho apresentado à Escola
Politécnica para aprovação no curso de
Projeto de Formatura II.

Orientadores:
Ricardo Luis de Azevedo Rocha
Márcio Lobo Netto

São Paulo
2021

A handwritten signature in black ink, consisting of the letters 'R', 'L', 'A', and 'R' in a cursive, stylized font. The signature is positioned above a horizontal line.

Prof. Dr. Ricardo Luis de Azevedo da Rocha

RESUMO

Com o crescimento na rigurosidade de leis de direitos autorais, tem sido mais difícil e mais caro o acesso a músicas boas para serem usadas de trilha sonora ou jingle. Algumas soluções, como sites de músicas sem direitos autorais, já estão sendo criadas e vendidas para muitos pequenos produtores de entretenimento, mas a procura de uma solução melhor e mais interativa, que possibilitasse customização ativa das músicas por parte do cliente, levou à concepção do Compositor Inteligente.

O objetivo do compositor inteligente é proporcionar um auxiliar de composição interativo e facilmente acessível para que qualquer um possa compor música de todos os estilos, seja um toque de celular ou uma orquestra sinfônica em alguns cliques. Para isso serão usados e analisados 3 tipos de composição: Composição regrada, Composição com padrões e Composição inteligente.

A disponibilização do produto é feita via web application, já disponível no endereço compositor.inovax.com.br, de onde é possível fazer o download de arquivos MUSICXML e abri-los em um software de notação e composição musical, como o Musescore.

Palavras-chave: Música, Composição, Inteligência artificial.

ABSTRACT

With harsher copyright laws being enforced over the past years, access to good jingles and background music has gotten more difficult and more expensive. Some solutions, like copyright free song websites, are already in the market, but the search for a better and more interactive means, that allows active customization, has led to the making of the Smart Composer.

The Smart Composer's main objective is to be an interactive and easily accessible auxiliary composer, for anyone to use and compose songs of many different styles, from cellular phone ring tones to orchestras in a few clicks. To achieve this, 3 different composing methods are used: rule composing, pattern composing and smart composing.

The product can be accessed via web application, available at the address compositor.inovax.com.br, where it is possible to download MUSICXML files and open them in any notation software, such as Musescore.

Key words: Music, Composing, Artificial intelligence.

LISTA DE FIGURAS

- 1 Deep Dream - <https://deepdreamgenerator.com/#gallery>
- 2 Deep Dream - <https://deepdreamgenerator.com/#gallery>
- 3 MuseNet - <https://openai.com/blog/musenet/>
- 4 Esquema das entradas e saídas do compositor
- 5 Interface de usuário do projeto - compositor.inovax.com.br
- 6 Interface de usuário do projeto - compositor.inovax.com.br
- 7 Interface de usuário do projeto - compositor.inovax.com.br
- 8 Música composta pelo programa de composição
- 9 Exemplo de matriz simplificada utilizada no treinamento da rede neural
- 10 Modelo de gerador da rede neural
- 11 Modelo de discriminador da rede neural
- 12 Tokenização da rede NLP
- 13 Gráfico de convergência da rede NLP

SUMÁRIO

1	Introdução.....;	10
1.1	Motivação.....	11
1.2	Objetivo.....	12
1.3	Estado da Arte.....	12
2	Conceitos.....	14
2.1	Uma breve introdução à música.....	14
2.2	A relação entre a música e a matemática.....	14
2.3	Ferramentas musicais.....	15
2.3.1	Tom e semitom.....	15
2.3.2	Escala.....	15
2.3.3	Acorde.....	16
2.3.4	Compasso.....	17
2.4	Uso das ferramentas.....	17
2.5	Modelagem e aplicação dos conceitos.....	18
2.6	Matriz simplificada.....	19
3	Tecnologias utilizadas.....	20
3.1	Tratamento de músicas e partituras.....	20
3.2	Coleta dos dados.....	20
3.3	Processamento dos dados.....	21

3.4	Disponibilização do produto.....	22
4	Requisitos do projeto.....	23
5	Implementação do projeto.....	24
5.1	Estrutura.....	24
5.1.1	Interface de usuário.....	25
5.1.2	Processador de dados.....	25
5.1.3	Rede neural.....	25
5.1.4	Banco de dados.....	26
5.2	Funcionamento.....	26
5.2.1	Interface de usuário.....	26
5.2.2	Processador de dados.....	29
5.2.2.1	Módulo de composição.....	29
5.2.2.2	Módulo de leitura.....	30
5.2.2.3	Módulo de escrita.....	30
5.2.3	Rede neural.....	30
5.2.3.1	Rede GAN.....	31
5.2.3.2	Rede NLP.....	33
5.2.4	Banco de dados.....	34
5.3	Integração.....	34
5.3.1	Composição regrada.....	34
5.3.2	Composição com padrão.....	35

5.3.3	Composição inteligente.....	35
6	Análise dos resultados.....	37
6.1	Composição regrada.....	37
6.2	Composição com padrão.....	37
6.3	Composição inteligente.....	38
6.3.1	GAN.....	38
6.3.2	NLP.....	38
6.4	Aplicação web.....	39
6.5	Análise conjunta.....	39
7	Conclusões.....	40
7.1	Conclusões atuais.....	40
7.2	Trabalhos futuros.....	42

1 INTRODUÇÃO

A inteligência artificial foi criada com o intuito de simular como a mente humana funciona, via aprendizado e aplicação dos conhecimentos adquiridos, podendo assim resolver problemas de identificação de padrões e análise de dados de uma forma totalmente nova. Com a modelagem correta, foi possível utilizar inteligência artificial até na arte, produzindo desenhos originais e criativos. O Deep Dream Generator é um exemplo de rede neural capaz de produzir desenhos como os apresentados abaixo.



Figura 1



Figura 2

A música sempre foi uma parte muito importante tanto de entretenimento quanto de marketing: concertos, trilhas sonoras de filmes, música de fundo em jogos, o jingle que se escuta durante uma propaganda etc.

Dadas essas informações, despertam-se muitas novas possibilidades, por exemplo, a utilização de inteligência artificial na composição de músicas originais.

1.1 **Motivação**

Com o crescimento na rigurosidade de leis de direitos autorais, tem sido mais difícil e mais caro o acesso a músicas boas para serem usadas de trilha sonora ou jingle. Algumas soluções, como sites de músicas sem direitos autorais, já estão sendo criadas

e vendidas para muitos pequenos produtores de entretenimento, Exemplos são: <https://www.youtube.com/user/NoCopyrightSounds> e <https://www.bensound.com/>.

A procura de uma solução melhor e mais interativa, que possibilitasse customização ativa das músicas por parte do cliente, levou à concepção deste projeto.

1.2 **Objetivo**

O objetivo deste projeto é a criação de um software de composição de músicas através de inteligência artificial. O usuário pode fornecer alguns parâmetros, como estilo da música, tempo, instrumentos, uma melodia, uma sequência de acordes, entre outros, ou deixar que o software faça tudo sozinho, possibilitando então vários graus de interatividade.

Assim, o software será capaz de auxiliar tanto compositores que não têm muito tempo até a data de entrega a elaborar algo rapidamente, quanto produtores de vídeos e propagandas que não têm tempo para estudar teoria e composição musical.

1.3 **Estado da arte**

A área de aprendizado de máquina é ainda relativamente nova. Especialmente quando o escopo é arte com aprendizado de máquina, ainda se tem muito a explorar. O Musenet é um software em desenvolvimento que utiliza também redes neurais e aprendizado de máquina, porém não com o intuito de fornecer como produto uma música original “customizável”. A principal diferença entre o Musenet e este projeto está no fato do Musenet utilizar para composição e treino de músicas arquivos de áudio (no formato MIDI), enquanto o nosso projeto utiliza partituras, ou representações baseadas nas notas musicais que compõe a música (em vez do seu resultado sonoro).

MuseNet

We've created MuseNet, a deep neural network that can generate 4-minute musical compositions with 10 different instruments, and can combine styles from country to Mozart to the Beatles. MuseNet was not explicitly programmed with our understanding of music, but instead discovered patterns of harmony, rhythm, and style by learning to predict the next token in hundreds of thousands of MIDI files. MuseNet uses the same general-purpose unsupervised technology as GPT-2, a large-scale transformer model trained to predict the next token in a sequence, whether audio or text.

April 25, 2019



Figura 3

2 CONCEITOS

Neste capítulo serão abordados os conceitos fundamentais por trás do projeto, como uma introdução à base musical utilizada e a forte relação da música com a matemática.

2.1 Uma breve introdução à música

A música consiste em uma união de notas concorrentes e consecutivas que seguem certos padrões e sequências de forma que elas tenham um som agradável ao ouvido.

As duas características mais importantes da música são a harmonia, o som das notas, que, ao serem organizadas de uma certa forma produzem sons agradáveis, e o tempo, ou o ritmo, em que as notas são tocadas.

2.2 A relação entre a música e a matemática

Existe uma nota, chamada de Lá 440 ou A440, usada como base universal na hora de afinar instrumentos, sendo o 440 a frequência da nota, em Hertz. Essa definição é importantíssima na música, pois para que haja harmonia entre as notas é necessário que haja razões muito específicas entre elas.

As frequências utilizadas na música seguem uma progressão geométrica com razão $\sqrt[12]{2}$, tendo então o Lá sustenido acima do Lá 440 uma frequência de $440\sqrt[12]{2}Hz$. Esse padrão continua ao longo das notas consecutivas (Lá, Lá sustenido, Si, Dó, Dó sustenido, Ré, Ré sustenido, Mi, Fá, Fá sustenido, Sol e Sol sustenido) até o próximo Lá, de frequência 880Hz.

O ritmo da música tem características similares, mas segue uma progressão linear ao invés de exponencial, como no caso da harmonia. Devido a essa relação linear entre a duração das notas, existe um intervalo mínimo de tempo para qualquer música de batida constante cujos múltiplos compõem a duração de qualquer nota presente nesta música.

Assim, é possível representar uma música por uma escala exponencial de frequências em intervalos discretos de tempo, e essa relação é fundamental na hora de modelar as músicas de uma forma inteligível para as redes neurais.

2.3 Ferramentas musicais

Nesta seção serão definidas algumas ferramentas musicais importantes para o entendimento dos padrões musicais relevantes.

2.3.1 Tom e semitom

O tom e o semitom são medidas de distância entre as notas. O semitom equivale à distância mínima entre duas notas, ou seja, a uma razão de $\sqrt[12]{2}$. O tom equivale a dois semitons, ou seja, a uma razão de $\sqrt[12]{4}$.

2.3.2 Escala

Uma escala é um conjunto menor de notas que seguem um certo padrão a partir de uma nota base. Dois exemplos comuns são a escala maior, que a partir da nota base faz pulos consecutivos de tom, tom, semitom, tom, tom, tom, semitom até chegar na

próxima nota base; e a escala menor, que a partir da nota base faz pulos consecutivos de tom, semitom, tom, tom, semitom, tom, tom. Tomando como nota base um Dó, por exemplo, as duas escalas seriam respectivamente: Dó, Ré, Mi, Fá, Sol, Lá, Si, Dó; e Dó, Ré, Mi bemol, Fá, Sol, Lá bemol, Si bemol, Dó.

Existem vários diferentes tipos de escalas, cada uma fortemente correlacionada com seus respectivos estilos musicais.

Como a percepção humana é mais fortemente baseada na relação entre as notas, e não no valor absoluto da nota em si, é possível transpor essas sequências para qualquer nota base, mantendo-se a mesma sonoridade.

2.3.3 Acorde

Um acorde é um conjunto de notas, comumente de uma mesma escala, tocadas ao mesmo tempo. Assim como no caso das escalas, acordes têm pulos de tamanhos definidos entre suas notas, podendo ser aplicadas a qualquer nota base. Um acorde maior possui pulos consecutivos de 2 tons e um tom e meio; e um acorde menor possui pulos consecutivos de um tom e meio e dois tons. Tomando como nota base um Dó, um acorde maior de Dó possui um Dó, um Mi e um Sol, enquanto o acorde menor de Dó possui um Dó, um Mi bemol e um Sol.

Existem muitos tipos de acorde, como acordes com sexta ou acordes com sétima, em que junto às notas originais também é tocada a sexta ou a sétima nota da escala do respectivo acorde.

2.3.4 Compasso

O compasso é uma divisão fixa de tempo em uma música, em que há uma quantidade fixa de batidas e intervalos mínimos de tempo. Por exemplo, uma valsa tem seu compasso tipicamente dividido em três batidas, enquanto que uma música popular tipicamente tem seu compasso dividido em quatro batidas

2.4 Uso das ferramentas

É possível, apenas com as ferramentas citadas acima, compor uma música completa e agradável, seguindo algumas regras básicas.

Define-se uma escala em que a música será tocada. Essa definição é importante na hora de escrever a melodia da música, que deve em sua maioria se conformar à escala escolhida. Um bom exemplo disso é a mão direita do piano.

Define-se uma sequência de acordes para a música. Essa definição é importante na hora de escrever a base da música, e representa o caminho que a música toma em relação ao seu acorde base, devendo também em sua maioria se conformarem à escala escolhida. Um bom exemplo disso é a mão esquerda do piano.

Define-se o compasso da música. Essa definição é importante na hora de definir o ritmo da música. Isso influencia no espaçamento entre a mudança de acordes, em quantas notas são tocadas durante um acorde e na batida da música. Um bom exemplo disso é uma valsa, que possui compasso de 3/4, significando que cada compasso possui três notas.

Podemos então juntar tudo, e compor uma valsa com 3 notas por compasso cuja sequência de acordes é Dó maior, Lá menor, Fá maior e Sol maior, mudando de acorde a cada dois compassos e seguindo uma escala de Dó maior. A partir deste ponto, quase qualquer música produzida será agradável ao ouvido.

2.5 Modelagem e aplicação dos conceitos

A teoria musical possui uma profundidade muito maior que a apresentada neste relatório, mas a modelagem das músicas foi feita tomando em conta principalmente os conceitos básicos apresentados anteriormente.

Cada nota possui 5 parâmetros fundamentais:

- **O tom**, ou seja, a frequência da nota, representado em uma escala logarítmica, tomando uma nota arbitrária como base = 1, e cada número seguinte a potência de um semitom, ou $\sqrt[12]{2}$, sendo o 0 uma pausa;
- **A duração**, ou seja, o tempo que a nota se mantém, representada pela razão entre a duração da própria nota e a duração mínima presente na música, levando a uma representação discreta, devido à forte proporcionalidade presente no ritmo musical;
- **O tempo** em que a nota ocorre, representado pela razão entre o instante de tempo em que a nota ocorre e a duração mínima presente na música, sendo também uma representação discreta;
- **O instrumento**, representado por um valor arbitrário;
- **O volume**, representado por um valor arbitrário.

Esta modelagem permite representar quase todos os aspectos básicos de uma música, e é capaz de representar inteiramente o exemplo de composição na seção de uso das ferramentas.

2.6 Matriz simplificada

Após feita a modelagem da música em formato de matriz, é necessário encontrar uma forma de representar as cinco informações essenciais apresentadas anteriormente de uma forma inteligível a uma rede neural.

Com esse intuito foi criada então a matriz simplificada, uma matriz com um número fixo de linhas e colunas, representativa de um número fixo de intervalos mínimos de tempo com suas respectivas notas que são tocadas em cada canal.

Essas matrizes simplificadas são tanto as entradas como as saídas das duas redes neurais criadas neste projeto.

3 TECNOLOGIAS UTILIZADAS

Neste capítulo serão abordadas as principais tecnologias utilizadas para o funcionamento do projeto.

3.1 Tratamento de músicas e partituras

O tratamento de músicas e partituras é feito com o Musescore, um software de notação e composição musical, mas pode ser feito com qualquer software de mesma função (Finale etc).

Musescore - Software de notação e composição musical gratuito e open source. Possui várias funcionalidades essenciais ao projeto, como a conversão de partituras para formato MP3 (áudio) e formato Musicxml (texto) para a visualização das músicas a serem usadas como e as músicas compostas pela rede neural.

3.2 Coleta dos dados

A coleta dos dados é feita via uma varredura dos arquivos Musicxml feita por um programa escrito em C++. O programa de varredura foi o primeiro dos componentes criados para este projeto, e consiste na procura dos cinco parâmetros fundamentais citados no capítulo dois, para então convertê-los para formato matricial. Como no mercado não há conversores de MUSICXML para matriz, foi necessária a criação desta ferramenta como parte do projeto.

Code Blocks - Compilador de código C++ gratuito e open source. Foi utilizado para escrita e teste dos programas em C++.

Musicxml - Formato de escrita baseado em XML para representar músicas. Este formato é essencial por ser universal a todos os softwares de composição, permitindo flexibilidade na escolha do software por parte do usuário, e facilitando a obtenção de dados por parte do programa de leitura.

3.3 Processamento dos dados

O processamento dos dados é feito tanto em C++ quanto em Python. O processamento em Python utiliza as bibliotecas de Tensorflow/Keras para receber os dados processados e usá-los como base para o aprendizado de máquina no Google Colab. Os programas de processamento de dados consistem na manipulação das matrizes criadas pelo programa de varredura com o intuito de compor músicas ou convertê-las para matrizes simplificadas. No caso das matrizes simplificadas, elas então são enviadas para as redes neurais como base para o aprendizado profundo.

Tensorflow/Keras - Bibliotecas do python de inteligência artificial, utilizadas para treinar modelos de aprendizado profundo.

Google Colab - Plataforma online da Google destinada a compilar código, sendo a base das redes neurais utilizadas neste projeto.

3.4 Disponibilização do produto

A disponibilização do produto é feita via web application. São utilizados Docker containers com programas em Nodejs. A parte de configuração do servidor, instalação de sistemas operacionais, planejamento da estrutura de front-end e back-end e programação dos containers e do website também fez parte do que foi feito pelo projeto.

Docker - Software gratuito e open-source de containerização de aplicações. Com o Docker é possível rodar várias instâncias de um programa sem que interfiram entre si ou na máquina hospedeira, possibilitando estruturas facilmente executáveis e escaláveis.

Nodejs - Linguagem utilizada para a escrita de aplicações web. São utilizadas as bibliotecas Express, responsável pelo back-end do projeto; React, responsável pelo front-end do projeto; Axios, responsável pela conexão entre o back-end e o front-end; e NodeAddons, responsável pela conversão do código em C++ para funções em Nodejs.

Visual Studio Code - Software gratuito de edição de texto utilizado para a escrita e instalação do projeto em servidores remotos.

4 REQUISITOS DO PROJETO

O projeto possui quatro principais requisitos:

- **Músicas originais:** O programa deve utilizar uma base de dados, à qual podem ser adicionadas músicas a critério do usuário, e receber uma série de parâmetros opcionais para criar uma música original em pouco tempo. O programa possui formas de compor músicas: a partir de um conjunto de regras pré-estabelecidas (**Composição regrada**), a partir dos padrões de uma música específica (**Composição com padrão**), ou a partir do aprendizado profundo aplicado a uma grande quantidade de músicas (**Composição inteligente**).
- **Variedade de estilos musicais:** O programa deve ser capaz de escrever qualquer estilo musical dado um número adequado de músicas desse gênero na base de dados.
- **Compatibilidade com programas de notação e composição musical:** O programa deve ser capaz de receber e produzir arquivos compatíveis com qualquer programa de composição.
- **Interface simples:** O programa deve possuir uma interface simples e intuitiva de modo que qualquer um seja capaz de entender e compor música ao utilizá-la.

5 IMPLEMENTAÇÃO DO PROJETO

Neste capítulo serão abordados a estrutura e o funcionamento do projeto.

5.1 Estrutura

Nesta seção será descrita a estrutura do projeto. Uma versão resumida pode ser vista no diagrama abaixo:

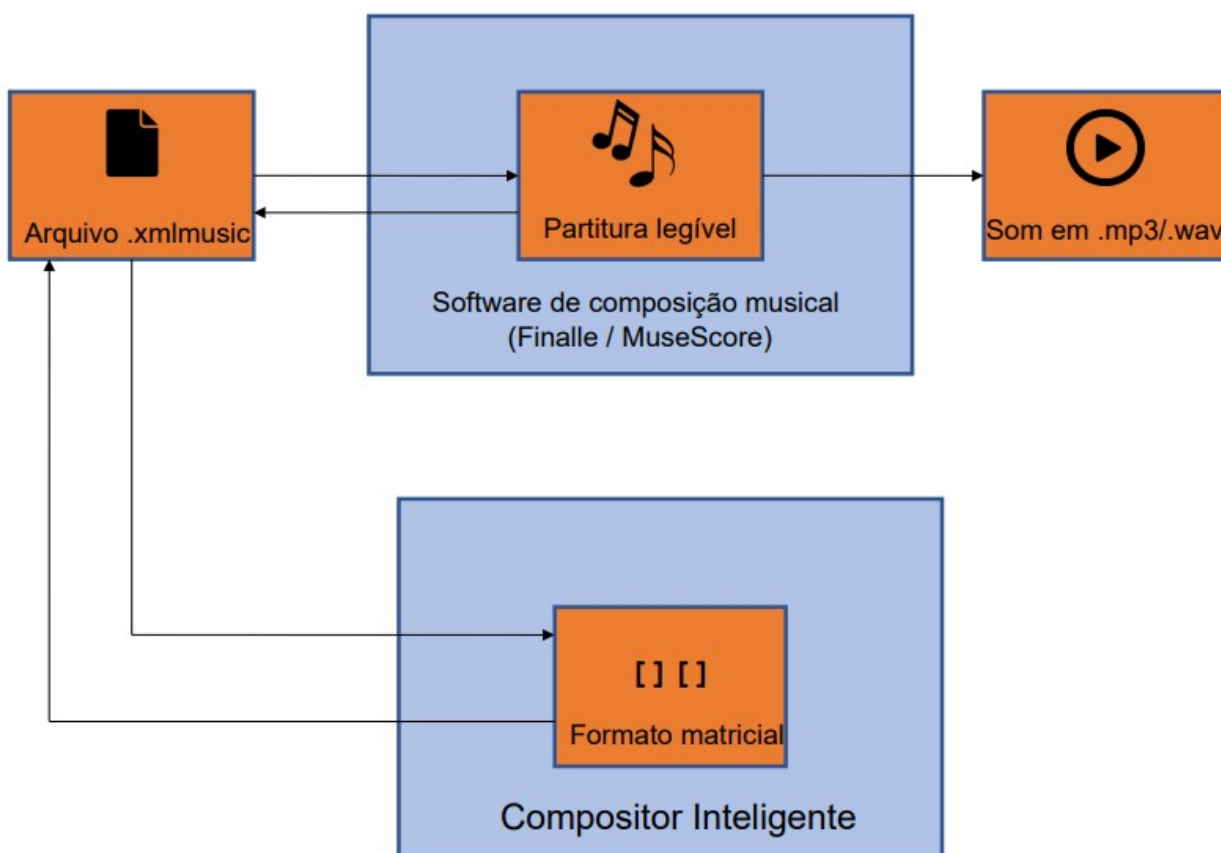


Figura 4

Softwares de notação e composição musical como o musescore são capazes de converter partituras para MUSICXML e MUSICXML para partituras, assim como

partituras para MP3, sendo então o responsável pela visualização dos resultados produzidos pelo compositor.

O compositor recebe como entrada e retorna como saída arquivos MUSICXML, e está dividido em 4 principais partes: A interface de usuário, o processador de dados, a rede neural e futuramente o banco de dados.

5.1.1 Interface de usuário

A interface de usuário é usada para a coleta de dados, como músicas que desejam ser adicionadas ao algoritmo de treinamento; e parâmetros do usuário, como estilo musical e número de instrumentos da música. Então, estes dados são enviados para o processador de dados.

5.1.2 Processador de dados

O processador de dados recebe as informações da interface de usuário, e têm duas principais funções, podendo devolver imediatamente uma música simples sem a ação da rede neural ou enviar os dados para a rede neural e devolver então a resposta da rede neural para a interface de usuário.

5.1.3 Rede neural

A rede neural é responsável pelo aprendizado profundo do projeto. Ela recebe as músicas já processadas pelo processador de dados de uma forma que fiquem mais

evidentes os padrões matemáticos, e retorna músicas originais compostas pelo gerador da rede neural. O modelo treinado também é salvo no banco de dados.

5.1.4 Banco de dados

O banco de dados é responsável pelo armazenamento dos modelos bem treinados de uma forma que facilite o acesso dos usuários a cada estilo musical.

5.2 Funcionamento

Nesta seção será descrito o funcionamento de cada uma das partes acima.

5.2.1 Interface de usuário

A interface de usuário constitui o front-end do projeto, e funciona em um contêiner independente do back-end. Os dois se comunicam via o método “post” e “send” do express em combinação com o axios para garantir a segurança do back-end, assim como a escalabilidade do projeto como um todo.

Atualmente, a interface de usuário, escrita em Nodejs com a biblioteca react, suporta uma das três funcionalidades propostas para o projeto: a composição regrada. Abaixo está uma imagem exemplificando a interface de usuário vista de um web browser:

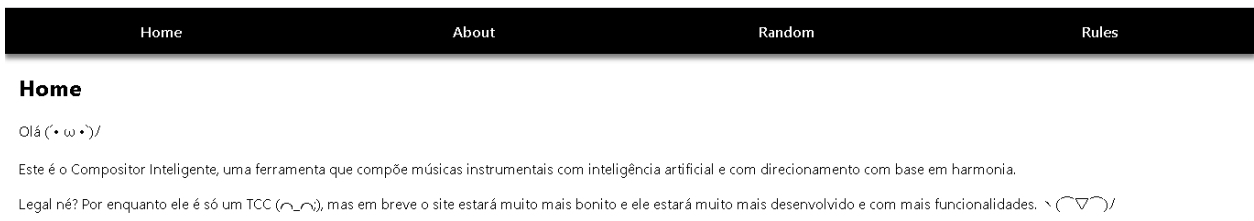


Figura 5

É possível selecionar as abas de composição aleatória e composição regrada e seguir as instruções para receber músicas originais em formato MUSICXML.

Random song generator

Esta funcionalidade é experimental. A música gerada aqui será completamente aleatória. A única entrada é o número de instrumentos. Provavelmente soará muito estranha (ᵀ ᵀ)

Figura 6

Rules song generator

Aqui a música será feita com um direcionamento. A melodia e a base são direcionadas, com determinadas regras e probabilidade. Como entradas, temos:

Primeira coluna: estilo da música. Atualmente os estilos disponíveis são:

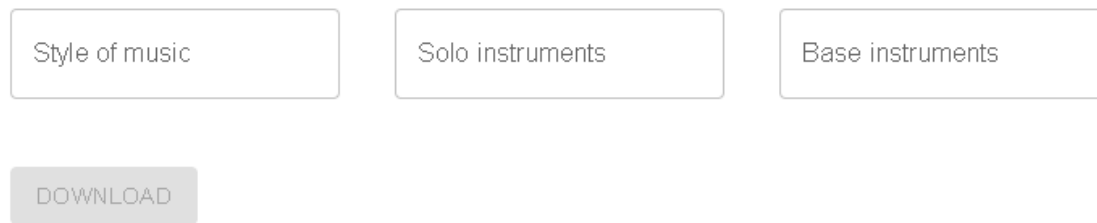
1: Dramática :(

2: Jingle

3: Flamenco

Segunda coluna: Número de instrumentos solo

Terceira coluna: Número de instrumentos base



The image shows a web interface for a song generator. It features three input fields arranged horizontally: 'Style of music', 'Solo instruments', and 'Base instruments'. Below these fields is a grey button labeled 'DOWNLOAD'. The text above the fields explains the parameters: 'Primeira coluna: estilo da música. Atualmente os estilos disponíveis são:' followed by a list of styles (Dramática :(, Jingle, Flamenco), 'Segunda coluna: Número de instrumentos solo', and 'Terceira coluna: Número de instrumentos base'.

Figura 7

Na imagem é possível ver a coleta dos parâmetros nas caixas (sendo eles, respectivamente, o estilo musical, o número de instrumentos principais e o número de instrumentos base), assim como o download realizado. Abaixo está uma imagem da composição feita pelo programa com os devidos dois instrumentos vista no Musescore.

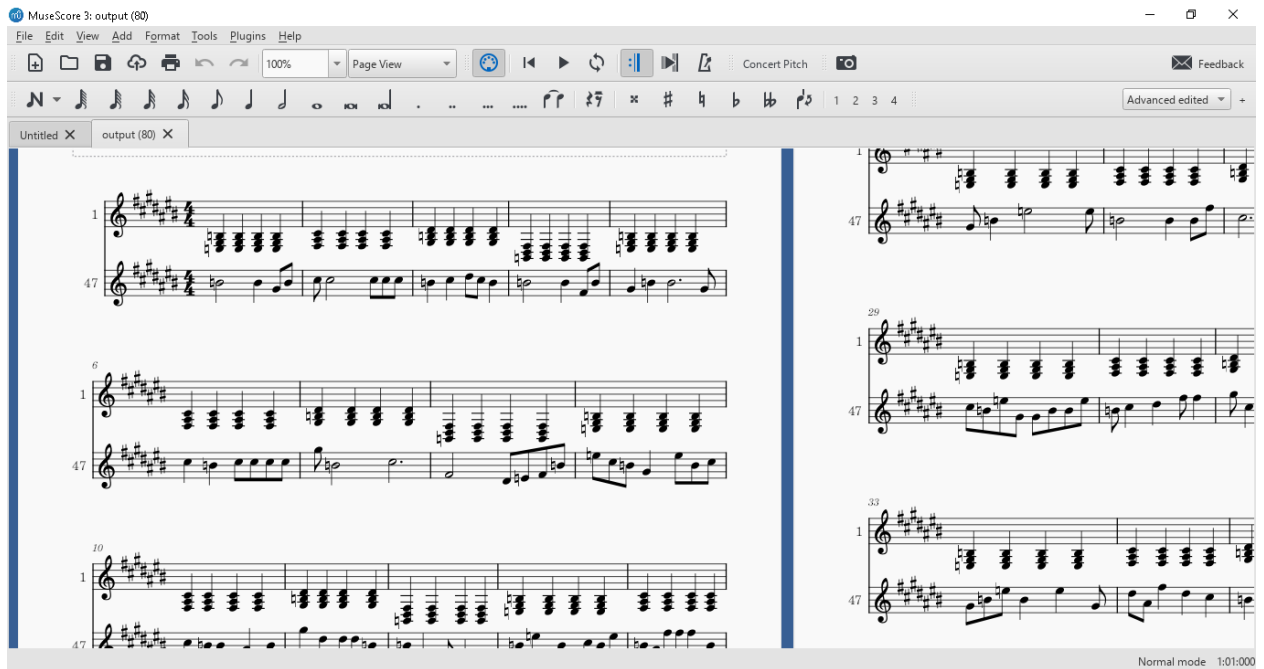


Figura 8

5.2.2 Processador de dados

O processador de dados constitui uma das três partes do back-end do projeto, e está contido em um segundo contêiner.

O processador de dados é dividido em três módulos: o módulo de composição, o módulo de leitura e o módulo de escrita

5.2.2.1 Módulo de composição

O módulo de composição é responsável pela composição regrada e pela composição com padrões. Ele recebe parâmetros, sejam eles números ou arquivos MUSICXML e os utiliza como base para compor músicas originais em formato MUSICXML.

5.2.2.2 Módulo de leitura

O módulo de leitura é responsável pela leitura de um arquivo MUSICXML e pela conversão da música em uma matriz simplificada, legível pela rede neural. Esta separação possibilita que novas modelagens para a rede neural sejam pensadas e implementadas sem afetar o funcionamento do resto do programa.

5.2.2.3 Módulo de escrita

O módulo de escrita é responsável pela conversão de matrizes simplificadas de volta para formato MUSICXML. Assim, é possível receber saídas tanto do módulo de leitura para confirmar o funcionamento das novas modelagens, e converter as composições feitas pelas redes neurais para uma partitura.

5.2.3 Rede neural

As redes neurais, escritas em Python, funcionam no Google Colab e recebem matrizes simplificadas em formato txt do processador de dados como base de dados para o aprendizado profundo. Foram criadas duas redes neurais como base de comparação: a rede GAN e a rede NLP. Elas foram treinadas com 12 músicas diferentes de Bach, cada uma transposta de 1 a 12 semitons tanto para cima quanto para baixo, com o intuito de aumentar o vocabulário.

5.2.3.1 Rede GAN

A rede GAN foi construída tomando como base uma rede neural de processamento de imagens, tratando cada nota como um pixel, conforme pode-se ver na imagem abaixo, no formato de matriz simplificada.

```
(4225, 128, 8)
(4225,)
[[65 59 0 ... 0 0 0]
 [65 59 0 ... 0 0 0]
 [79 0 0 ... 0 0 0]
 ...
 [68 41 0 ... 0 0 0]
 [66 44 0 ... 0 0 0]
 [66 44 0 ... 0 0 0]]
```

Figura 9

Abaixo também é possível ver a declaração do gerador e do discriminador da rede:

```

def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(16*8*16, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((16, 1, 128)))
    assert model.output_shape == (None, 16, 1, 128) # Note: None is the batch size

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 32, 2, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(32, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 64, 4, 32)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(16, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 128, 8, 16)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(1, 1), padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 128, 8, 1)

    return model

```

Figura 10

```

def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(16, (5, 5), strides=(2, 2), padding='same', input_shape=[128, 8, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(32, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model

```

Figura 11

Primeiro, o gerador e o discriminador são treinados com as músicas de Bach. Então, o gerador começa a criar músicas falsas para tentar enganar o discriminador, de forma

que ele ache que as músicas são verdadeiras. Esse ciclo se repete ao longo de várias épocas até aprimorar a produção de músicas do gerador.

5.2.3.2 Rede NLP

A rede NLP foi construída tomando como base uma rede de Natural Language Processing, cujo intuito é aprender a escrever textos através da tokenização. O mesmo modelo de matriz simplificada da rede GAN foi usado, de forma que cada linha da matriz (o acorde sendo tocado) seja representado como uma palavra.

```
'50a0a0a0a0a0a0': 2,
```

Figura 12

Como vemos pela figura acima, o token 2 foi atribuído à nota Ré. Pode-se ver abaixo os gráficos de convergência da Rede NLP.

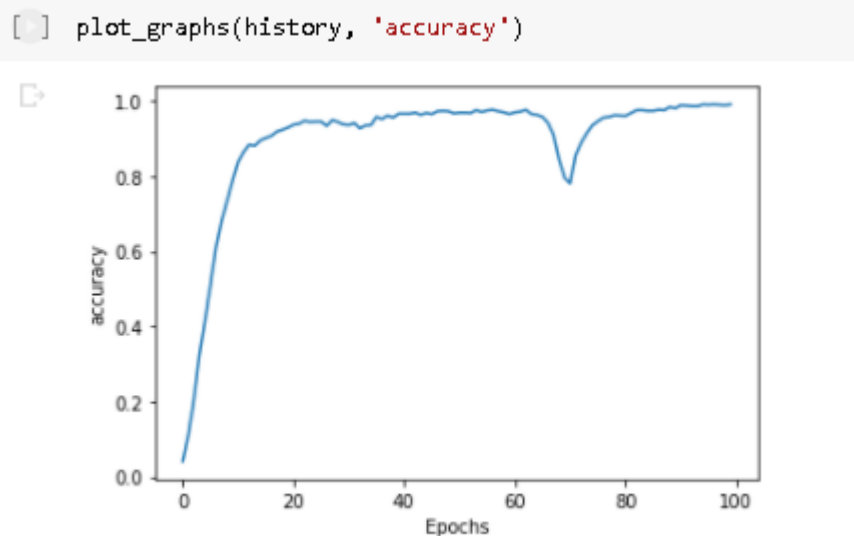


Figura 13

Após o treinamento da rede, é gerada uma frase inicial para ser usada de semente, para que a rede neural faça uma previsão das notas a seguir. Para evitar o overfitting, foi usada uma função aleatória para escolher entre as n notas mais prováveis.

5.2.4 Banco de dados

O banco de dados será o depósito das redes neurais treinadas, assim como das músicas geradas pelas redes neurais e pelo módulo de composição.

O banco de dados online não será implementado até depois da incorporação da rede neural à web application, e será usada a memória local do computador como banco de dados até o término de todas as outras funcionalidades.

5.3 Integração

Nesta seção será descrita a integração dos diferentes módulos descritos acima para compor os três tipos diferentes de composição.

5.3.1 Composição regrada

A composição regrada começa com a coleta das informações do usuário. As informações são coletadas pela interface de usuário e enviadas então para o módulo de escrita. O módulo de escrita então gera uma matriz com os 5 parâmetros fundamentais definidos anteriormente usando as informações coletadas do usuário,

converte a matriz para formato MUSICXML e então envia o arquivo de volta para a interface de usuário para que seja feito o download.

5.3.2 Composição com padrão

A composição com padrão começa com a coleta do arquivo MUSICXML representativo da música que o usuário quer usar como base. O arquivo é coletado pela interface de usuário e enviado para o módulo de leitura, que converte o arquivo para formato matricial. A matriz é então enviada para o módulo de escrita, onde é feita uma análise da música, criando parâmetros para serem usados como base de composição. O módulo de escrita então gera uma matriz baseada nos parâmetros criados, converte a matriz para formato MUSICXML e então envia o arquivo de volta para a interface de usuário para que seja feito o download.

5.3.3 Composição inteligente

A composição inteligente começa com a coleta do banco de dados de música a ser usado para o aprendizado profundo. Os arquivos são coletados pela interface de usuário e enviados para o módulo de leitura. O módulo de leitura converte os arquivos MUSICXML para formato matricial e usa as matrizes para a criação das matrizes simplificadas e as envia para as redes neurais.

A rede neural baseada em processamento de imagens recebe as matrizes simplificadas, converte-as para float e começa o aprendizado profundo. Após o aprendizado, ela produz uma matriz simplificada, cujos valores em float são aproximados para os inteiros mais próximos e envia a matriz para o módulo de escrita. O módulo de escrita então converte a matriz simplificada para o formato matricial

original converte a matriz para formato MUSICXML e então envia o arquivo de volta para a interface de usuário para que seja feito o download.

A rede NLP recebe as matrizes simplificadas, modela cada linha da matriz simplificada como uma palavra e começa o aprendizado profundo. Após o aprendizado, a rede cria uma semente aleatória usada como base para fazer um número arbitrário de previsões de notas sucessivas, até que seja feita a música. A rede então converte as palavras previstas de volta para matriz simplificada, que é enviada para o módulo de escrita. O módulo de escrita então converte a matriz simplificada para o formato matricial original converte a matriz para formato MUSICXML e então envia o arquivo de volta para a interface de usuário para que seja feito o download.

6 ANÁLISE DOS RESULTADOS

Neste capítulo será feita uma análise dos resultados de cada uma das partes do projeto. Como base de comparação para a qualidade das músicas compostas, pode-se usar a música gerada 100% aleatoriamente: “aleatorio.mp3”. É possível escutar todas as músicas produzidas no website

<https://sites.google.com/usp.br/compositor-inteligente>.

6.1 Composição regrada

A composição regrada criou músicas de diferentes estilos, tempos e tons, refrões e sequências de acordes, realizando seu papel como introdução ao compositor inteligente. Pode-se ouvir três exemplos de músicas feitas usando composição regrada na pasta “Músicas do programa”: “dramatica.mp3”, “jingle.mp3” e “flamenca.mp3”, cujos respectivos estilos selecionados estão no nome.

6.2 Composição com padrão

A composição com padrão gerou músicas similares à música de entrada utilizando estatística, sendo capaz de identificar escalas, acordes e sequências de notas e reproduzi-los de forma original. Pode-se ouvir dois exemplos de músicas feitas usando composição com padrão: “musicaprima1.mp3” e “musicaprima2.mp3”, baseadas na música “musicabase.mp3”.

6.3 Composição inteligente

Ambas as redes neurais produziram músicas interessantes, que serão apresentadas a seguir.

6.3.1 GAN

A rede GAN produziu músicas baseadas no banco de dados de músicas de Bach apresentado, mas não produziu músicas agradáveis. O ritmo da música convergiu muito bem, apresentando os padrões esperados (mão direita mais rápida, mão esquerda mais lenta, divisão correta de compassos), como visto na “GAN.mp3”, mas a harmonia nunca chegou a convergir, gerando então músicas ritmicamente corretas mas harmonicamente erradas. Um bom exemplo disso é a música “GAN_overfitting.mp3”, resultado de um overfitting de uma das músicas apresentadas à rede (“originalbach.mp3”), que segue o ritmo da música à risca, mas toca notas que não fazem sentido ser tocadas juntas.

6.3.2 NLP

A rede NLP produziu músicas baseadas no banco de dados de músicas de Bach apresentado, e produziu resultados satisfatórios. O overfitting foi prevenido via o mecanismo de escolha aleatória entre os melhores 4 candidatos, resultando então em uma música um pouco menos exata no ritmo, mas com harmonia muito boa. Pode-se ouvir o resultado da rede NLP na música “BachNLP.mp3”.

6.4 **Aplicação web**

A aplicação web já está funcional sob o endereço compositor.inovax.com.br, e as funções de geração aleatória de música e de composição regrada já podem ser utilizadas de qualquer lugar. Foi possível, portanto, a entrega do produto mínimo viável em conjunto com os estudos de viabilidade de diferentes métodos de composição musical a serem futuramente integrados ao website.

6.5 **Análise conjunta**

Ao analisar os diferentes resultados em conjunto e fazer comparações, é possível identificar algumas das vantagens e desvantagens de cada tipo de composição.

A composição regrada apresentou resultados muito consistentes, mas com pouco grau de liberdade. A composição com padrão apresentou um grau de liberdade grande com boa adaptabilidade, mas teve um pouco de dificuldade com a consistência da harmonia e do ritmo. A rede GAN apresentou excelente consistência no ritmo musical, mas resultados ruins na parte harmônica da música. A rede NLP produziu uma música com alto grau de liberdade e bom sequenciamento harmônico, mas apresentou maiores dificuldades na parte rítmica.

Dados esses resultados, o ideal seria produzir uma música com o grau de liberdade das redes neurais, mantendo o ritmo da rede GAN e a harmonia da rede NLP, fazendo então uma última adaptação a padrões mais convencionais usando os algoritmos de composição regrada.

7 CONCLUSÕES

Neste capítulo serão apresentadas as conclusões dos estudos feitos neste projeto.

7.1 Conclusões atuais

As conclusões serão divididas entre as 3 principais ferramentas de composição de música criadas: a composição regrada, a composição com padrão e a composição inteligente. Então será feita uma comparação entre as três e uma análise para possíveis melhoras.

A composição regrada é a mais simples das três mas já possui grande potencial para a criação de músicas de plano de fundo. Com a escolha do estilo musical, o programa já possui uma grande base de sequências de acordes para serem utilizadas, e as escalas que acompanham esses acordes. Isso cria uma base muito sólida para pessoas que conhecem pouco de música e precisam de um meio simples e eficiente para compor suas músicas.

A composição com padrão tem uma utilidade muito mais específica, mas também importante, que permite que o usuário escolha uma ou algumas músicas já existentes que expressam bem a ideia que ele quer passar com o seu conteúdo, mas que ele não pode usar, e criar uma música de estilo muito parecido mas original.

Por último, a composição inteligente possui o maior potencial de criação e originalidade dos três métodos, sendo o único dos três capaz de criar uma verdadeira obra prima, mas também é a mais difícil de se executar.

Um dos modelos de rede neural utilizado foi baseado em redes neurais de processamento de imagens. Portanto, faz sentido que a parte espacial da música,

como o posicionamento das notas convergiu bem, enquanto que a parte harmônica, em que pequenas diferenças de valores fazem muita diferença, convergiu mal.

O outro modelo de rede neural utilizado foi baseado em redes neurais de processamento de linguagem. Portanto, faz sentido que a parte harmônica da música, ao ser tratada como texto, converge bem, mas a parte espacial da música não converge tão bem.

Como ferramenta individual, a com mais potencial pareceu ser a rede NLP. Com seu alto grau de liberdade e bom encadeamento harmônico, apesar de sua estrutura muito simples (apenas um LSTM), ela está muito próxima de produzir músicas muito boas. Como o encadeamento de notas assemelha-se bastante com o de palavras, apenas o LSTM já é capaz de produzir sequenciamentos muito interessantes, e com mais estudos e testes será possível desenvolver a composição muito mais.

Foi possível notar que a composição regrada compensa bem a falta de refrões e compassos bem definidos da composição com padrão, que por sua vez, complementa bem o acesso da composição regrada a novos estilos não programados inicialmente. Fazer com que as duas ferramentas trabalhem juntas pode melhorar muito a qualidade das músicas produzidas.

Já a rede GAN complementa bem a estrutura da rede NLP, mantendo um ritmo preciso, ao mesmo tempo que a rede NLP produz um encadeamento bonito de notas. É possível que uma versão híbrida das duas redes, em que a parte de processamento de imagens cuide do ritmo, e a parte de processamento linguístico cuide da harmonia, faça músicas de alto nível.

Por último, as ferramentas de composição sem inteligência artificial, apesar de possuírem menos potencial de composição que as redes neurais, complementam bem a exigência das redes por uma grande base de dados, sendo capaz de compor sem música nenhuma. Assim, como um projeto, as quatro partes se complementam muito bem e pavimentam o caminho para um compositor muito poderoso.

7.2 Trabalhos futuros

O trabalho no projeto será continuado mesmo após o término do curso, com o intuito de trazer cada vez mais gente para dentro do universo da composição musical.

Apesar de já compor um produto sólido, existem muitas formas de melhorar o Compositor Inteligente, como a integração das funcionalidades mais avançadas na web application, a concepção de modelos mais completos para representar a música, o estudo mais profundo de redes neurais e formas pelas quais seria possível atingir músicas melhores e a integração das diferentes ferramentas musicais entre si.

A composição regrada ainda aceita vários estilos diferentes, assim como a possibilidade de maiores graus de liberdade e futura integração com os outros módulos para garantir maior consistência. Esta é a principal ferramenta para dar toques finais em músicas já prontas como forma de polimento.

A composição com padrão já é uma forma excelente de procurar novos estilos não presentes na composição regrada, mas ainda parece um estudo melhor nos pesos dos parâmetros utilizados, assim como na definição de novos para procurar produzir músicas cada vez melhores.

Por último, serão feitos estudos sobre as diferentes possibilidades de camadas para o tratamento tanto da harmonia quanto do ritmo da música, para construir redes neurais muito mais completas e adequadas, dado que a rede de NLP possui apenas LSTM e a rede GAN possui apenas algumas convoluções.

Essas funcionalidades serão incorporadas à web application na medida que forem terminadas, para que o Compositor Inteligente se torne um produto completo.

REFERÊNCIAS

[1] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). Generative Adversarial Nets (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[2] Russel, S. and Norvig, P. (2013). Inteligência Artificial. Tradução da 3a edição, Elsevier.

[3] Mitchell, T.M. (1997). Machine Learning. Boston, McGraw-Hill.

[4]

<https://mdm.claretiano.edu.br/perestmus-G02164-2020-01-grad-ead/2020/03/18/ciclo-2-parte-4/>

[5] <https://www.tensorflow.org/tutorials/generative/dcgan>

[6] <https://towardsdatascience.com/build-a-super-simple-gan-in-pytorch-54ba349920e4>

[7] <https://www.youtube.com/watch?v=tPYj3fJGjk>

[8]

https://colab.research.google.com/drive/1F_EWVKa8rbMXi3_fG0w7AtcscFq7Hi7B#forceEdit=true&sandboxMode=true

[9]

<https://colab.research.google.com/drive/1m2cg3D1x3j5vrFc-Cu0gMvc48gWyCOuG#forceEdit=true&sandboxMode=true>

[10]

<https://colab.research.google.com/drive/15Cyy2H7nT40sGR7TBN5wBvgTd57mVKay#forceEdit=true&sandboxMode=true>

[11]

<https://towardsdatascience.com/word-subword-and-character-based-tokenization-know-the-difference-ea0976b64e17>

[12]

<https://colab.research.google.com/github/Imoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%204%20-%20Lesson%202%20-%20Notebook.ipynb#scrollTo=w9vH8Y59ajYL>

[13]

<https://www.youtube.com/watch?v=fNxaJsNG3-s&list=PLQY2H8rRoyvzDbLUZkbudP-MFQZwNmU4S&index=1>