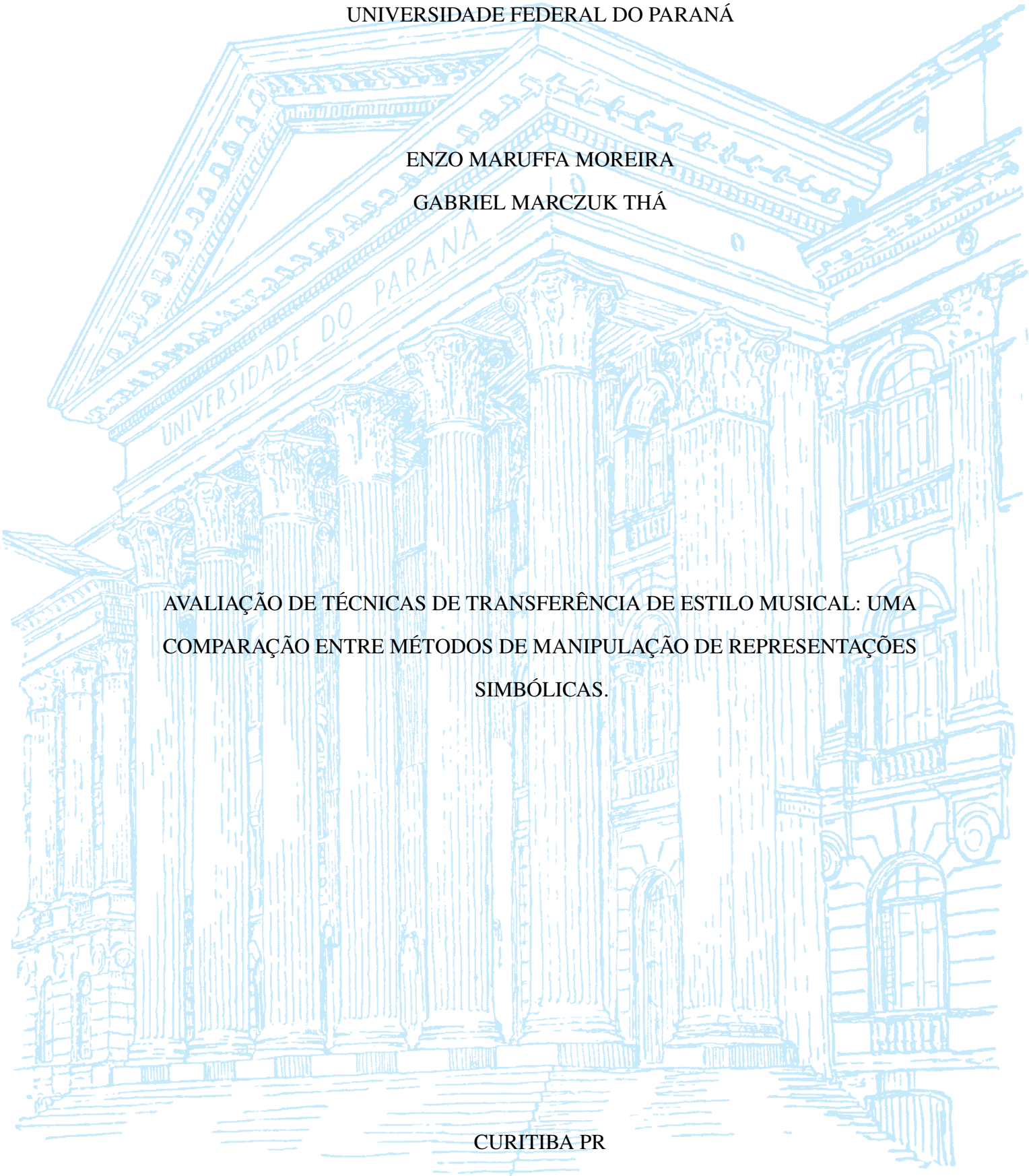
UNIVERSIDADE FEDERAL DO PARANÁ

ENZO MARUFFA MOREIRA

GABRIEL MARCZUK THÁ



AVALIAÇÃO DE TÉCNICAS DE TRANSFERÊNCIA DE ESTILO MUSICAL: UMA COMPARAÇÃO ENTRE MÉTODOS DE MANIPULAÇÃO DE REPRESENTAÇÕES SIMBÓLICAS.

CURITIBA PR

2023

ENZO MARUFFA MOREIRA
GABRIEL MARCZUK THÁ

AVALIAÇÃO DE TÉCNICAS DE TRANSFERÊNCIA DE ESTILO MUSICAL: UMA COMPARAÇÃO ENTRE MÉTODOS DE MANIPULAÇÃO DE REPRESENTAÇÕES SIMBÓLICAS.

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Jaques Spinosa.

CURITIBA PR

2023

AGRADECIMENTOS

Gostaríamos de expressar os mais profundos agradecimentos aos nossos pais, que sempre nos apoiaram em todas as decisões e nunca mediram esforços para nos proporcionar o melhor. Sem o amor, incentivo e suporte incondicional deles, nós não estaríamos aqui hoje.

A todos os nossos professores, que dedicaram seu tempo e conhecimento para ensinar e orientar ao longo desta jornada acadêmica, muito obrigado.

À Universidade Federal do Paraná, que nos proporcionou uma formação de alta qualidade, com um corpo docente e discente de excelência. Estudar nesta instituição foi uma experiência de vida transformadora e enriquecedora.

Ao nosso orientador, Eduardo Jaques Spinosa, que ouviu as mais malucas ideias sempre com paciência e entusiasmo, respondendo-as com valiosos ensinamentos e explicações. Obrigado, professor!

Às nossas namoradas, que aguentaram a nossa constante exaustão e estresse no desenvolvimento de um projeto tão importante para nós, muito obrigado.

Por fim, agradecemos os nossos queridos amigos e também a todas as outras pessoas que, direta ou indiretamente, contribuíram para a nossa formação acadêmica. Esperamos que os resultados aqui apresentados possam contribuir de alguma forma para o avanço da tecnologia.

RESUMO

A transferência de estilo musical é um tópico de pesquisa importante na área de processamento de sinais de áudio. Porém, a ausência de uma abordagem padronizada para avaliar projetos torna difícil analisar o progresso entre diferentes técnicas. Neste Trabalho de Graduação, apresenta-se uma comparação entre dois projetos de transferência de estilo musical baseados em arquivos do tipo "Musical Instrument Digital Interface"(MIDI), utilizando duas métricas de avaliação desenvolvidas pelos presentes autores, sendo uma a distância entre arquivos e a outra a classificação do estilo de uma música. As métricas foram profundamente analisadas entre os projetos, buscando entender os principais fatores por trás de melhores ou piores resultados. Para a comparação, foram gerados 2400 testes, avaliados com base nas métricas, além de uma pontuação resumida. Concluiu-se que os indicadores desenvolvidos foram robustos o bastante para determinar qual projeto apresenta melhores resultados, alertando para possíveis vieses das métricas. Para complementar a análise, uma avaliação subjetiva dos resultados foi realizada, promovendo uma base e flexibilidade necessárias para um tema como a música.

Palavras-chave: Transferência de Estilo Musical. Desenvolvimento de Métricas de Avaliação. Aprendizado de Máquina.

ABSTRACT

Music style transfer is a relevant topic in the signal processing community, albeit the lack of a standardized approach to evaluate projects adds complexity to the measurement of progress between different techniques. This Graduation Thesis presents a comparison between two MIDI-based music-style transfer projects using two evaluation metrics. The metrics were developed by the authors, the first being the distance between files and the second being the style classification of a song. The metrics were thoroughly analyzed between projects, in search of what are the driving factors behind better or worse results. To compare the projects, 2400 tests were generated and evaluated, with the addition of a summary score. The work concludes that the metrics developed were robust enough to determine which project presents the best results, alerting for the possibility of biases. Complementing the analysis, a subjective evaluation of the results was done, adding a certain level of ground truth and flexibility needed for a theme such as music.

Keywords: Music Style Transfer. Development of Evaluation Metrics. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

IA	Inteligência Artificial
PCM	*Pulse Code Modulation*
MIDI	*Musical Instrument Digital Interface*
AM	Aprendizado de Máquina
TE	Transferência de Estilo
AS	Aprendizado Supervisionado
ANS	Aprendizado Não Supervisionado
AR	Aprendizado por Reforço
FK	Função de *Kernel*
SVM	*Support Vector Machines*
SVC	*Support Vector Classification*
HO	Hiperplano Ótimo
RN	Redes Neurais
RNA	Redes Neurais Artificiais
LF	*Loss Function*
CNN	*Convolutional Neural Network*
RNN	*Recurrent Neural Networks*
LM	*Language Model*
CLM	*Chord Language Mode*
GAN	*Generative Adversarial Networks*
LA	*Loss* Adversário
LC	*Loss* Cíclico
TE	Transferência de Estilo
NST	*Neural Style Transfer*
RNC	Redes Neurais Convolucionais
LCT	*Loss* de Conteúdo
LET	*Loss* do Estilo
G2G	Groove2Groove
OS	*One-Shot*
CC	*Chord Chart*
BIAB	*Band-in-a-box*
EC	*Encoder* de Conteúdo
EE	*Encoder* de Estilo
ELU	Unidade Linear Exponencial
SMST	*Symbolic Music Style Transfer*

RF	*Random Forest*
LR	*Logistic Regression*
XGB	*XGBoost Classifier*
LGBM	*LightGBM Classifier*
DTW	*Dynamic Time-Warping*
RQA	*Recurrence Quantification Analysis*

LISTA DE SÍMBOLOS

D_O	Dados originais do conjunto principal
D_P	Dados pré-processados do conjunto principal
D_M	União dos dados originais com os pré-processados
M	Conjunto dos arquivos "origem", "transferência"e "ciclo"produzidos pelos projetos
M_C	Conjunto que possui o par ("origem", "ciclo")
M_T	Conjunto que possui os pares ("origem", "transferência") e ("transferência", "ciclo")
I_{cv2}	A função de similaridade entre espectrogramas
I_{rqa}	A função que calcula o RQA entre dois arquivos de áudio
I_{dtw}	A função que calcula o DTW entre dois arquivos de áudio
CE	Porcentagem estimada pelo classificador de estilo
S	*Score* base de todas as métricas
S^T	*Score* de transferência
S^C	*Score* de ciclo
$S^{T'}$	*Score* de transferência desafiante
S_D	*Score* desafiante resumo de todas as métricas
S^C_{cv2}	*Score* obtido pela similaridade entre espectrogramas no ciclo
S^T_{cv2}	*Score* obtido pela similaridade entre espectrogramas na transferência
S^C_{rqa}	*Score* obtido pelo RQA no ciclo
S^T_{rqa}	*Score* obtido pelo RQA na transferência
S^C_{dtw}	*Score* obtido pelo DTW no ciclo
S^T_{dtw}	*Score* obtido pelo DTW na transferência
S^T_{CE}	*Score* obtido pelo classificador de estilo no ciclo
S^T_{CE}	*Score* obtido pelo classificador de estilo na transferência
$S^{T'}_{cv2}$	*Score* desafiante obtido pela similaridade entre espectrogramas na transferência
$S^{T'}_{rqa}$	*Score* desafiante obtido pelo RQA na transferência
$S^{T'}_{dtw}$	*Score* desafiante obtido pelo DTW na transferência

SUMÁRIO

1 INTRODUÇÃO

A transferência de estilo é uma técnica computacional que possibilita a aplicação do estilo de um artefato B em outro A. O resultado é um novo artefato com a estrutura de A, mas com o estilo de B. Este processo é realizado através de técnicas de Aprendizado de Máquina, principalmente Redes Neurais Artificiais. A prática mais comum é a transferência de estilo entre imagens, campo que apresenta resultados contundentes. De maneira geral, dada a grandiosidade de seus resultados, a manipulação de estilo tem sido uma área da computação que atrai cada vez mais interessados, criando constantemente novas aplicações para tal mecanismo.

Dentro deste domínio existem subdivisões, tal como a transferência de estilo musical, objeto de interesse deste trabalho. Este campo trata da transferência de estilo entre artefatos musicais, aplicando o estilo de uma música em outra ou então inserindo em uma música um estilo pré-estabelecido.

Diversas estratégias de transferência de estilo musical já foram propostas, utilizando diferentes técnicas computacionais, majoritariamente de Aprendizado de Máquina. Além disso, as músicas podem ser simbolizadas de diversas formas, tais como: arquivo de áudio digital, partituras e também representações simbólicas, conhecidas como MIDI. No presente trabalho serão examinadas técnicas de transferência de estilo musical para arquivos de música simbólica, além do estudo e proposição de métricas e métodos de comparação entre tais técnicas.

1.1 MOTIVAÇÃO

A música é um ponto em comum presente no cotidiano dos seres humanos e há quem diga que é impossível viver sem ela. As técnicas de transferência de estilo musical possibilitam a criação de novas músicas originais, além da construção de ferramentas que auxiliam produtores musicais no processo de composição. Sendo assim, a motivação geral deste trabalho, em sua abordagem e desenvolvimento, é uma contribuição científica para estreitar ainda mais a relação entre música e computação, através do estudo de técnicas de transferência de estilo musical.

Durante o processo de análise do estado da arte notou-se que as métricas de comparação e de avaliação dos processos de transferência de estilo musical ainda são muito abstratas e irregulares. Portanto, este trabalho também motiva-se a desenvolver e propor técnicas que possam ser reutilizadas para a melhoria e progresso da manipulação computacional de estilo musical.

1.2 PROPOSTA

Este trabalho propõe a comparação de técnicas de transferência de estilo musical já existentes, em especial duas delas: o Groove2Groove de Cífka et al. (2020) e o Symbolic Music Style Transfer de Brunner et al. (2018). O Groove2Groove utiliza uma técnica de rede neural do tipo *encoder-decoder*, enquanto o Symbolic Music Style Transfer emprega a CycleGAN. Portanto, a proposição da comparação entre ambas é válida, dada a diferenciação dos métodos, definindo-se uma possível melhor abordagem.

Neste processo, ambos os projetos são utilizados de acordo com as especificações dos respectivos autores e então comparados a partir de seus resultados.

A partir disto, é possível enxergar métricas de avaliação dos projetos. Dessa forma, este trabalho também se propõe a estabelecer técnicas justas de comparação entre abordagens de

transferência de estilo musical, sendo elas a identidade entre arquivos e o classificador de estilo para arquivos MIDI.

1.3 OBJETIVOS

O trabalho se divide em objetivos gerais e específicos. Os objetivos gerais são: analisar se o estado da arte atinge resultados satisfatórios e estabelecer métricas de avaliação justas para técnicas de transferência de estilo.

Os objetivos específicos são: estabelecer uma ou mais métricas de avaliação de identidade de arquivos MIDI; avaliar a robustez das métricas desenvolvidas; definir qual dos dois projetos escolhidos (Groove2Groove e Symbolic Music Style Transfer) é melhor com base nas métricas criadas.

1.4 DESAFIOS

Inicialmente, cada método de transferência de estilo musical possui sua própria estratégia e, consequentemente, limitações, tais como: resultados que não correspondem ao estilo desejado, dificuldades de generalização, grande tempo de processamento necessário e a necessidade de conjuntos de dados extensos. Todos esses fatores contribuem para o desafio de avaliação dos projetos.

Além disso, nota-se que na literatura não existem métricas consistentes com o intuito de mensurar a qualidade dos resultados. Isso ocorre porque a percepção da qualidade musical é subjetiva e varia de acordo com o gosto pessoal. Em muitos casos, não há uma resposta "certa" ou "errada" para a transferência de estilo. Portanto, um grande desafio é desenvolver métricas de avaliação justas e que representem, de certa forma, um julgamento semelhante à análise subjetiva.

Por fim, a ausência de um conjunto de dados padronizado, volumoso e amplamente divulgado torna o referencial de comparação consideravelmente mais complexo. Certos projetos podem ainda aplicar transformações nas músicas, tornando ainda maior o desafio de desenvolver estruturas genéricas para a classificação de estilo.

1.5 CONTRIBUIÇÃO

Através do desenvolvimento dos experimentos, no Capítulo 5 e da investigação dos resultados, no Capítulo 6, este trabalho atinge seu propósito da análise de métodos de transferência de estilo musical e do desenvolvimento de métricas de comparação. A partir disso, verifica-se uma contribuição significativa para a área de transferência de estilo musical, considerando o aprofundamento do estudo das técnicas e também as métricas que podem ser reutilizadas em processos futuros.

1.6 ORGANIZAÇÃO

Os capítulos deste trabalho estão organizados da seguinte forma: Capítulo 2, apresenta os conceitos fundamentais de música e computação, essenciais para a compreensão completa do desenvolvimento do estudo; Capítulo 3, mostra o estado da arte da transferência de estilo musical; Capítulo 4, define a metodologia de experimentos e análises que serão desenvolvidas no estudo; Capítulo 5, explicita todos os experimentos realizados no trabalho; Capítulo 6, traz à tona os resultados obtidos a partir da aplicação dos experimentos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é destinado à explicação de conceitos das áreas de música, Inteligência Artificial (IA), Aprendizado de Máquina (AM) e transferência de estilo. Tais conceitos são fundamentais para o completo entendimento das ideias, técnicas e experimentações apresentadas nos capítulos subsequentes.

2.1 SOM

Desde os primórdios da humanidade, o som é um fenômeno importante para o ser humano. “Sabemos que som é onda, que os corpos vibram, que essa vibração se transmite para a atmosfera sob a forma de uma propagação ondulatória, que o nosso ouvido é capaz de captá-la e que o cérebro a interpreta, dando-lhe configurações e sentidos” (Wisnik, 1989).

Fisicamente, o som pode ser entendido como a propagação de uma onda por um meio. A propagação ocorre através de regiões de compressão e rarefação, dando origem ao conceito de onda sonora. Dessa forma, esta movimentação mecânica gera uma vibração na estrutura interna do ouvido humano, o qual tem a capacidade de absorver e, em conjunto com o cérebro, interpretar o que está sendo detectado, como afirmado por Rui e Steffani (2007). O mesmo fenômeno acontece em dispositivos capazes de captar e emitir sons, a exemplo de microfone e alto-falante, sendo o primeiro responsável por identificar as variações e o segundo por criar a perturbação do meio material, a qual será propagada.

Segundo Wisnik (1989), a onda sonora, ainda seguindo o conceito físico, possui duas características principais: a amplitude e a frequência. A amplitude pode ser entendida como intensidade de uma onda sonora. Quanto maior esta for, mais energia a onda carrega. Esse conceito é frequentemente referenciado como volume e padronizadamente medido em decibéis (dD). Já a frequência é definida pelo tempo que a onda sonora necessita para completar um ciclo de crista e vale, determinando assim a altura de um som. Um som alto, isto é, com alta frequência, é considerado agudo, enquanto o inverso, chamado de som baixo, é conhecido como grave. A medida universal de frequência é Hertz (Hz), definida pelo conceito de ciclos por segundo, sendo que o ouvido humano é capaz de captar o intervalo de frequências entre 20 Hz e 20.000 Hz. A Figura 2.1 apresenta duas ondas com frequências diferentes.

Por influência das características físicas, mesmo que dois objetos não idênticos, como instrumentos musicais, consigam gerar ondas sonoras com mesma intensidade e frequência, o padrão vibracional será distinto, permitindo a identificação de diferentes sons que estão presentes na mesma faixa de frequência. Este conceito é conhecido como timbre. “Cada um dos instrumentos vibra também em outras frequências mais rápidas (..) cujo produto reconhecemos como timbre” (Wisnik, 1989). A Figura 2.2 mostra as diferenças que o timbre provoca na onda sonora.

2.2 A MÚSICA

Considerada um dos pilares das manifestações culturais, conceitualmente, uma música é a associação de ritmo e som, de uma maneira que essa combinação seja entendível e agradável ao ouvido humano. Wisnik (1989) discorre sobre o fato de que a música, “em todos os povos, foi vivida como experiência do sagrado, justamente porque nela se trava, a cada vez, a luta cósmica e caótica entre o som e o ruído.” O ritmo pode ser genericamente compreendido como a sucessão

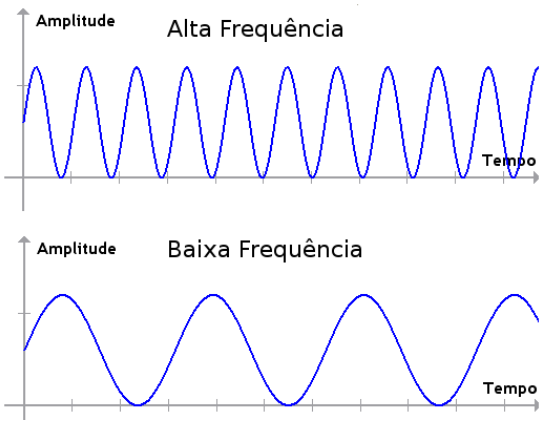


Figura 2.1: Representação de Onda Sonora. Imagem de Soares (2011).

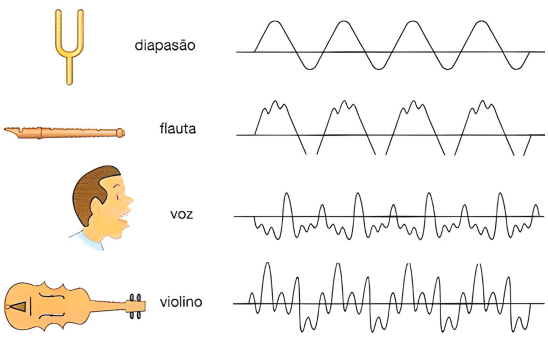


Figura 2.2: Representação de Timbre. Imagem adaptada de Donoso (2015)

contínua de silêncio e períodos que contém som, popularmente conhecidos como batidas ou pulsação. As pulsações ocorrem em intervalos de tempo que respeitam convenções predefinidas, chamadas de divisões rítmicas.

Ainda, existe a divisão sonora entre melodia e harmonia. A melodia é identificada pela produção de uma única nota musical por vez, enquanto a harmonia é a combinação de várias notas tocadas em uma única pulsação do ritmo.

Sendo assim, é entendido em mais alto nível que a música é uma combinação de diferentes instrumentos tocados em conjunto, gerando uma mistura prazerosa ao ouvido humano.

2.2.1 Estilo e Gênero Musical

Comumente os termos estilo e gênero musical são utilizados como sinônimos. Porém, olhando por um viés mais técnico, existem diferenças entre suas definições.

Gênero pode ser considerado uma categoria mais ampla de classificação musical, isto é, o agrupamento com base em características similares, tais como ritmo, melodia, instrumentos, execução, dentre outros.

Por outro lado, estilo é um conceito mais subjetivo, podendo até ser entendido como um subgênero, que se concentra em atributos mais específicos das músicas. A exemplo do uso de um instrumento muito específico ou as características de um artista, além de ser passível definir que cada música pode ter seu próprio estilo, como utilizado por Cífka et al. (2020).

No âmbito deste trabalho, os dois conceitos acima serão utilizados. O projeto da seção 3.1 utiliza o conceito de estilo como definido aqui, enquanto o da seção 3.2 emprega a ideia de estilo como sinônimo de gênero.

2.3 REPRESENTAÇÕES DE MÚSICAS

Ao longo do tempo, com o aprimoramento tecnológico e das técnicas musicais, o modo como as músicas são armazenadas, reproduzidas e representadas também evoluiu. A seguir serão apresentadas algumas das mais comuns representações, bem como as que serão utilizadas nas próximas seções.

2.3.1 Partitura

"A partitura foi o suporte principal para o registro da música ocidental até o início do século XX" (Baia, 2011). Pode ser descrita como um conjunto de notas escritas em sequência, geralmente em um papel, contendo informações de melodia, harmonia e ritmo, além de outros detalhes sobre a maneira de reproduzir, utilizada pelos músicos para executar os sons esperados. A Figura 2.3 apresenta um exemplo de partitura.

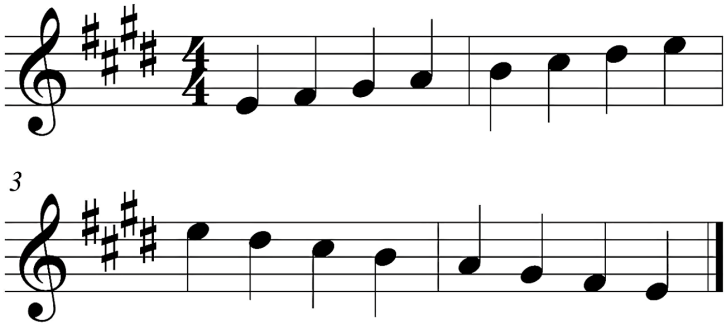


Figura 2.3: Partitura Musical. Imagem de Lopes (2019)

2.3.2 Áudio

Ao passo que o som é uma onda mecânica, por outro lado, o áudio é composto por energia elétrica, sendo esta um sinal digital ou analógico, que quando convertido em energia mecânica, a exemplo de um alto-falante, se torna som. Em outras palavras, o áudio é uma representação elétrica do som.

Com o avanço da tecnologia, surgiram os arquivos de áudio, capazes de armazenar computacionalmente os valores elétricos que representam um som. Dentre os milhares existentes, os dois formatos de arquivo de áudio mais comuns são MP3 e WAV.

2.3.2.1 Arquivos MP3

Fundamentado na especificação de Richardson (2003), um arquivo MP3 é formado pela combinação repetida de blocos de cabeçalho (*header*) e dados (*data*). A Figura 2.4 é a representação estrutural do arquivo.

No cabeçalho são definidas as informações referentes ao conjunto de dados seguintes, tais como taxa de bits e versão do MP3, enquanto os dados representam as informações de frequência e amplitude que deverão ser reproduzidas.

Em sua codificação, um arquivo MP3 passa por um processo de compressão de dados, reduzindo o tamanho do arquivo, com baixa perda de qualidade, o que faz desse formato o mais popular do mundo.

2.3.2.2 Arquivos WAV

O arquivo WAV, abreviação de *Waveform Audio File Format*, é constituído de um único cabeçalho, seguido por diferentes blocos de dados. A Figura 2.5 apresenta a estrutura de um arquivo WAV.

Um WAV é baseado em um método de conversão de som analógico em digital chamado *Pulse Code Modulation* (PCM). Essa técnica não apresenta compressão de dados, isto é, no WAV não há perda de informações. Em contrapartida, seu tamanho é maior, embora proporcione mais

qualidade, sendo o formato mais utilizado para trabalhos profissionais de áudio e, em decorrência disso, é o formato padrão de áudio escolhido para o desenvolvimento deste trabalho.

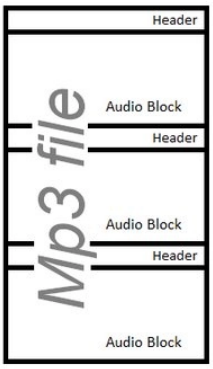


Figura 2.4: Representação dos blocos de um arquivo MP3. Imagem adaptada de Haynie (2018).

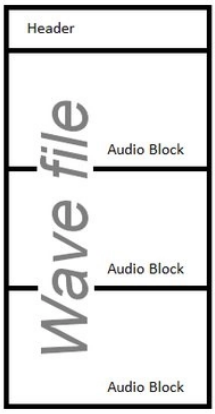


Figura 2.5: Representação dos blocos de um arquivo WAV. Imagem adaptada de Haynie (2018).

2.3.3 Espectrogramas

Um espectrograma é uma forma de representar graficamente uma quantidade de energia. Então, como o som é uma forma de energia, seja ela elétrica ou mecânica, também é possível visualizá-lo em forma de espectrograma.

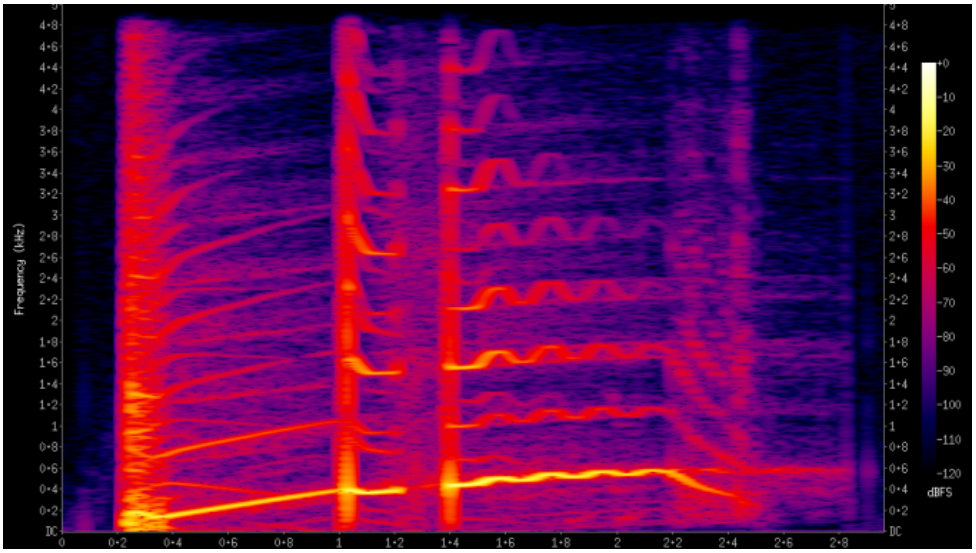


Figura 2.6: Espectrograma de um arquivo de áudio. Imagem de Rossini (2018)

Como é possível observar na Figura 2.6, o espectrograma é um gráfico bidimensional, que também conta com uma representação de cores. O eixo horizontal representa a estrutura temporal, ou seja, um momento específico do som, enquanto o eixo vertical representa as frequências presentes. A gama de cores representa a intensidade de uma frequência em um determinado momento. Cores mais escuras representam baixa intensidade, enquanto as mais claras, definem alta intensidade. A exemplo da Figura 2.6, é notável que no trecho de tempo entre 1.4 e 2.2, existe uma presença intensa das frequências no intervalo entre 0.4kHz e 0.6kHz.

2.3.4 Arquivos MIDI

O arquivo MIDI, abreviação de *Musical Instrument Digital Interface*, é conjunto de dados que formam uma representação simbólica de notas musicais. Estes não possuem nenhum tipo de informação de áudio, ou seja, não produzem som por conta própria, e dessa forma, necessitam estarem combinados com algum outro artefato, como instrumentos virtuais, capazes de interpretá-los e gerarem áudio.

Fundamentado pela especificação criada por MIDI Manufacturers Association (1996), pode-se afirmar que o arquivo MIDI é formado por dados de tempo, métrica, e uma sequência de instruções. Cada instrução representa uma nota musical que será tocada e guarda a informação sobre o momento, intensidade e duração de tal nota. Ainda, mais de uma instrução pode acontecer ao mesmo tempo, concretizando o conceito de harmonia. Sendo assim, quando esse conjunto de instruções é inserido em um dispositivo, este irá reproduzir as notas musicais com as durações e intensidades definidas no MIDI. Um único arquivo MIDI pode conter a combinação de várias sequências MIDI, das quais cada uma pode ser executada ao mesmo tempo por um instrumento diferente, por exemplo.

O arquivo MIDI também pode ser caracterizado abstratamente como uma partitura virtual.

Por fim, este tipo de arquivo é o principal objeto de estudo deste trabalho. Optou-se pelo uso do MIDI por ser um formato simples, leve e bastante poderoso.

2.3.5 *Piano Roll*

O *Piano Roll* é uma ferramenta utilizada para criar, editar e reproduzir arquivos MIDI. Genericamente pode ser entendido como uma representação visual de um arquivo MIDI, tal como na Figura 2.7.

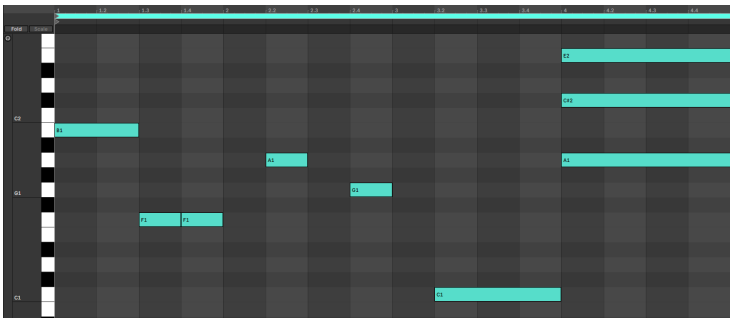


Figura 2.7: Arquivo MIDI representado em um *piano roll*

Tecnicamente, é uma matriz que representa todos os eventos e características de um MIDI. Consequentemente, pela facilidade de se trabalhar com matrizes, o *piano roll* é a mais comum representação de arquivos MIDI para tarefas computacionais.

2.4 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina (AM) é uma área da computação que visa ensinar um algoritmo a realizar tarefas definidas, a exemplo de reconhecimento de padrões, que subjetivamente só seriam possíveis através da inteligência humana. Em 1959, Arthur Samuel, mostrado por Bhavsar et al. (2017), definiu AM como “campo de estudo que dá ao computador a habilidade de aprender sem a necessidade de ser explicitamente programado”, podendo também ser compreendido como a aquisição automática de conhecimento.

De maneira geral, todo aprendizado de máquina tem como eixo principal o processo de treinamento. Tudo se inicia com um modelo teórico proposto, geralmente definido com base em fórmulas numéricas. Tal modelo é exposto a uma base de dados, sendo cada arquivo uma iteração. Dessa maneira, é possível que o modelo aprenda padrões e relações entre os dados apresentados a ele e com isso seja capaz de realizar decisões para uma nova entrada ainda não vista, com base em tudo o que foi aprendido no treinamento.

Algumas das principais aplicações do aprendizado de máquinas são reconhecimento de imagem e fala, processamento de linguagem natural, predição analítica de dados e sistemas de recomendação.

2.4.1 Tipos de Aprendizado de Máquina

Existem diversos tipos de aprendizado de máquina, incluindo Aprendizado Supervisionado (AS), Aprendizado Não Supervisionado (ANS) e Aprendizado por Reforço (AR). Estes três principais serão aqui explicados.

2.4.1.1 Aprendizado Supervisionado

Partindo da ideia apresentada por Rezende et al. (1999), no aprendizado supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido. Em outras palavras, as entradas dadas para o modelo durante o treinamento são sempre constituídas por um par, contendo dados (D) e resultado esperado (R). Sendo assim, o algoritmo deve entender que para o conjunto D, a saída esperada é R. Com base em uma exposição consecutiva aos dados de treino, o modelo entende as relações e adapta seu funcionamento para acertar o maior número de resultados esperados.

Sendo assim, o nome deste tipo de aprendizado de máquina decorre do modelo ser supervisionado pela sua saída esperada durante o treinamento. A contrapartida se dá pela necessidade de todo o banco de dados usado no treinamento passar por um processo de rotulação, indicando qual a saída correta para cada uma das entradas.

2.4.1.2 Aprendizado Não Supervisionado

De acordo com Lorena e de Carvalho (2007), em contraste ao AS, no Aprendizado Não Supervisionado não há a presença de um professor, ou seja, não existem exemplos rotulados. Como nenhum tipo de saída esperada é informada, o modelo precisa aprender os padrões e relações entre os dados por conta própria, sem saber por meio de informações externas se está indo no caminho correto ou não. Alguns dos exemplos mais comuns são algoritmos de clusterização, redução de dimensionalidade, geração de dados similares aos de entrada, dentre outros.

O ANS não necessita da rotulação da base de dados de treinamento, mas por outro lado, medir a qualidade e performance desses algoritmos tende a ser mais complexo, visto que não há nenhum tipo de saída esperada para caracterizar se o algoritmo acertou ou errou em sua predição.

2.4.2 Métodos de Aprendizado de Máquina

Dada a vasta gama de possibilidades de uso do aprendizado de máquina, existem diferentes métodos e algoritmos, supervisionados ou não, que podem ser usados para tais funções. No desenvolvimento deste trabalho, as duas técnicas mais relevantes são as Redes Neurais Artificiais e *Random Forests*, além de outras técnicas, que serão aprofundadas a seguir.

2.4.2.1 *Support Vector Machines*

Fundamentado por Lorena e de Carvalho (2007), as Máquinas de Vetores de Suporte, do inglês *Support Vector Machines* (SVM), são uma técnica de aprendizado de máquina supervisionado embasada pela teoria de aprendizado estatístico, desenvolvida por Vapnik (1998). O objetivo geralmente é obter um classificador por meio do treinamento.

O processo funciona com base na interpretação de um espaço N-dimensional, sendo N o número de características que representam uma determinada entrada. Por exemplo, se uma entrada possui 3 números que a representam, então a SVM irá funcionar em um espaço 3-dimensional.

Inicialmente, todos os objetos de treinamento são espalhados pelo espaço N-dimensional, de forma que cada ponto do espaço corresponda a uma entrada e cada um dos eixos represente uma das características pertencentes aos dados. A partir disso, a finalidade é encontrar um hiperplano que divida e agrupe os dados de uma mesma classe, a partir de várias possíveis hipóteses produzidas.

O Hiperplano Ótimo (HO) é encontrado por meio da maximização da distância entre uma hipótese e os objetos de todas as classes, podendo também ser entendido como um valor intermediário entre as classes. Além disso, os hiperplanos que estão no limite da borda espacial de uma classe são chamados de vetores de suporte, e todos eles estão a uma exata distância M do hiperplano ótimo, conhecida como margem.

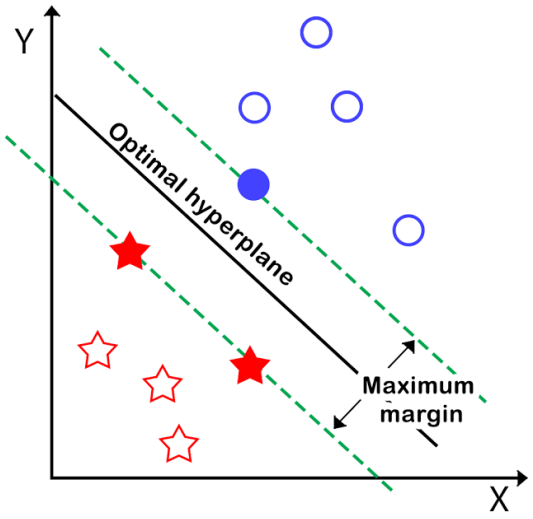


Figura 2.8: Ilustração de uma SVM. Imagem de Dabakoglu (2018)

Na Figura 2.8, as duas linhas tracejadas representam a margem máxima de uma classe, enquanto as estrelas e o círculo preenchidos são os vetores-suporte. A linha contínua denota o hiperplano ótimo, que está situado no ponto médio da distância entre todos os vetores de suporte existentes no espaço.

A partir disso, o hiperplano ótimo é usado como limiar de decisão do classificador, sendo que cada novo objeto de entrada será situado no espaço de acordo com os valores de suas características, e então o HO é usado para definir a qual classe esta nova entrada pertence.

Em um cenário real, os dados de treinamento não são perfeitamente uniformes, podendo não ficar corretamente espalhados pelo espaço ou então com algum tipo de ruído ou discrepância. A SVM possui artifícios para ignorar ruídos de dados, popularmente conhecidos como *outliers*, de forma que não afetem a definição do hiperplano. Mas caso a discrepância espacial seja muito

grande, não é possível apenas ignorá-la. Então, entendemos que os dados não são linearmente separáveis, não sendo possível definir um hiperplano satisfatório o suficiente. A solução para esse problema é a utilização da função de *kernel* (FK).

A função de *kernel* mapeia as entradas para um espaço dimensional ainda maior, possibilitando que um limite linear possa ser encontrado. As FKs mais comuns em Máquinas de Vetores de Suporte são linear, polinomial e radial.

No presente trabalho, usou-se a implementação de uma SVM da biblioteca *scikit-learn* em Python, para a criação de um classificador de arquivos MIDI. Essa implementação é nomeada como *Support Vector Classification* (SVC), sendo equivalente à implementação teórica de uma SVM.

Por fim, com base em Lorena e de Carvalho (2007), vale ressaltar que o desempenho de uma SVM é diretamente afetado pela escolha do *kernel*, pela quantidade de características que representam uma entrada e também pelo tamanho da base de dados do treinamento.

2.4.2.2 Random Forest

O algoritmo *Random Forest*, introduzido por Breiman (2001), é considerado muito poderoso e eficiente, dada a sua capacidade de lidar com dados de muitas dimensões e também de descobrir padrões complexos em dados altamente correlacionados.

Essa técnica se baseia na criação de diversas árvores de decisão, a partir de amostras aleatórias dos dados de treinamento, de forma que os resultados das árvores sejam combinados para obter uma predição ao final. Uma representação da estrutura pode ser vista na Figura 2.9.

A partir de Monard e Baranauskas (2003), uma árvore de decisão é formada por nós de decisão, que contém testes sobre atributos e nós folha, que representam um resultado esperado. A ideia geral é utilizar perguntas simples e binárias para gerar subconjuntos dos dados, até ser possível definir um resultado ou classe para tais dados.

Com isso, o *Random Forest* segue o processo de gerar uma amostragem aleatória nos dados, sendo cada conjunto para uma árvore de decisão diferente, como forma de fazer com que sejam diferentes entre si e capturem relações distintas entre os dados. Após isso, as árvores são construídas de maneira que cada uma faça uma predição única sobre a variável independente, neste caso a classe a qual os dados pertencem. Por fim, para *Random Forest* de classificação, é feita uma votação majoritária dos resultados, isto é, a partir das predições das árvores de decisão, busca-se o valor que mais se repete, sendo esta a predição final do algoritmo.

2.4.2.3 Logistic Regression

A Regressão Logística é uma técnica que visa estimar a probabilidade de uma entrada pertencer a uma classe, utilizando uma função matemática que se adapta de acordo com as características de treinamento. Seguindo a lógica de Ambrosius (2007), é definida como um "modelo estatístico que descreve a relação entre uma variável qualitativa (isto é, variável que possui apenas valores discretos, como a presença de uma doença ou não), e uma variável independente".

Ela é embasada pela utilização de uma regressão linear, mostrada em Best e Wolf (2014), da forma:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{2.1}$$

Neste caso y é a saída, $\{x_1 \ ... \ x_n\}$ correspondem as características de entrada, α é a constante base e $\{\beta_1 \ ... \ \beta_n\}$ são os coeficientes correspondentes a cada característica. Após isso,

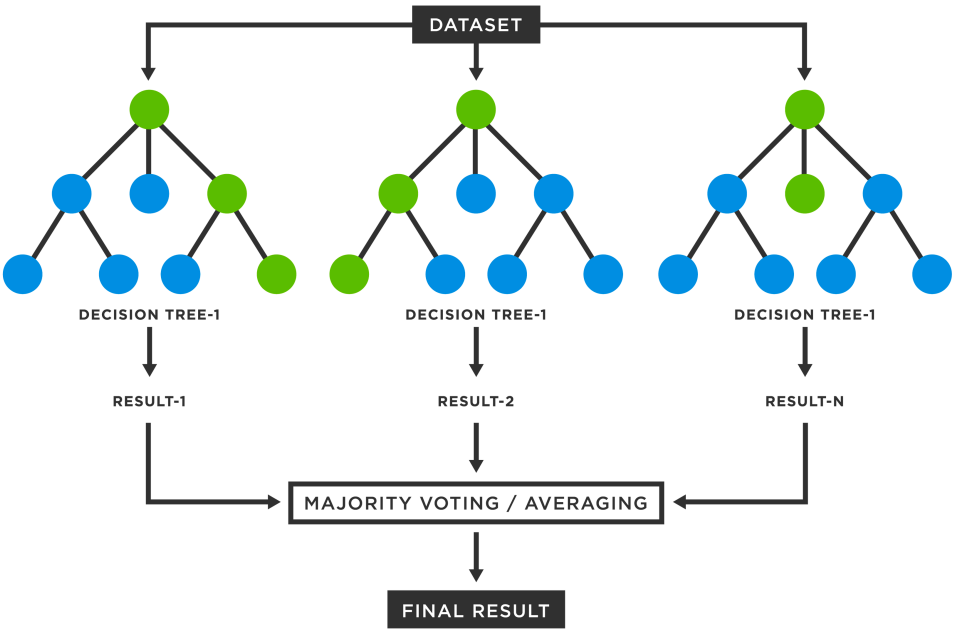


Figura 2.9: Estrutura *Random Forest*. Imagem de Group (2018)

esta função linear é transformada em uma probabilidade com o auxílio da função sigmoide, da forma:

$$p = \frac{1}{1 + e^{-y}} \tag{2.2}$$

Portanto, essa probabilidade é usada para definir se uma entrada pertence ou não a uma determinada classe, configurando assim uma classificação binária.

O processo de treinamento é feito a partir de vários exemplos de entrada, onde a cada iteração os coeficientes $\{\beta_1 \ldots \beta_n\}$ são ajustados para reduzir a função de custo, isto é, acertar mais predições e também diminuir a diferença entre as probabilidades de saída.

Para classificadores de múltiplas classes é comum utilizar várias funções lineares, criando assim um classificador binário para cada classe e então a que possuir a maior probabilidade é dada como saída do algoritmo.

2.4.2.4 XGBoost Classifier

O *eXtreme Gradient Boosting* (XGB), segundo Chen et al. (2015), é um modelo de AM bastante preciso e eficiente. Seu funcionamento se dá pela aplicação de *Gradient Boosting* (GB) em árvores de decisão. Em outras palavras, isso significa que diversas árvores de decisão são formadas iterativamente, de forma que a árvore atual seja construída corrigindo os erros das anteriores.

Realizando uma comparação, na seção 2.4.2.2 várias árvores de decisão são formadas em paralelo para lidar com subconjuntos de dados aleatórios, mas as árvores não possuem interação e troca de informações entre si, sendo apenas combinadas ao final para gerar um resultado. De outra forma, o XGB calcula a cada nova iteração a função de perda da árvore, isto é, qual a diferença entre o resultado obtido e o esperado. Essa função de perda é utilizada para atualizar os nodos de decisão da próxima árvore que será construída. Isso promove um processo contínuo de melhoria até que uma condição de parada seja atingida.

Ao final, assim como em 2.4.2.2, o resultado de todas as árvores de decisão são combinados para atingir uma predição conjunta, como pode ser visto na Figura 2.10.

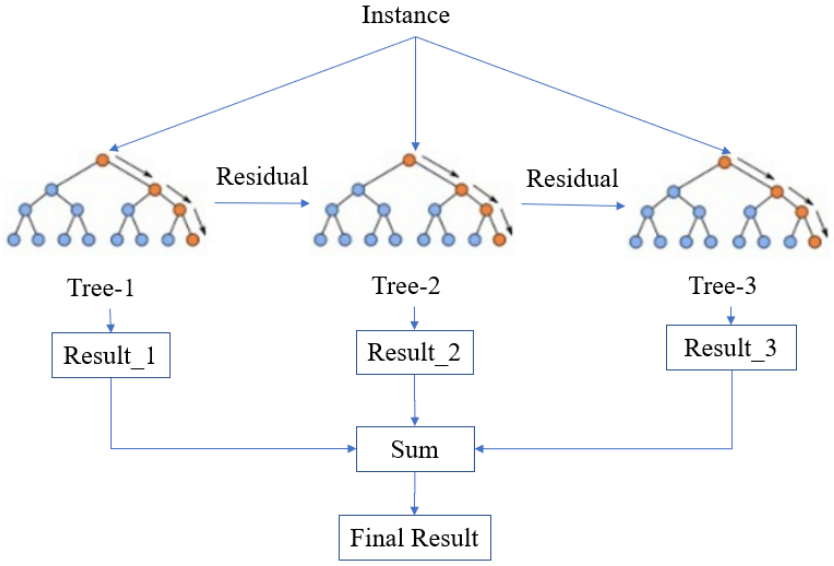


Figura 2.10: Estrutura *XGBoost*. Imagem de Wang et al. (2021)

2.4.2.5 LightGBM Classifier

Estabelecido em Ke et al. (2017), o LightGBM (LGBM), assim como 2.4.2.4, é um algoritmo de *Gradient Boosting* para árvores de decisão. Seu funcionamento é bastante semelhante ao XGB no que diz respeito ao processo iterativo, porém se difere na maneira como a divisão dos nodos e a construção da árvore de decisão é feita.

A divisão de nós é o processo de quebrar os dados em subconjuntos de itens semelhantes, com base nas características de cada um. Isto é feito de forma recursiva, iniciando com um único nó representando todos os dados de treinamento, que então é dividido em dois nós filhos, cada um representando um subconjunto dos dados, repetindo este processo até que uma condição de parada seja atingida.

O XGBoost utiliza uma estratégia de divisão de nós chamada divisão exaustiva, que testa todas as possibilidades de combinações. Por outro lado, o LightGBM utiliza a técnica de histograma, que avalia a quantidade de informação obtida a partir de cada subconjunto, sendo computacionalmente mais eficiente.

Esse procedimento interfere no crescimento e na estrutura final da árvore de decisão. O XGBoost e a maioria dos algoritmos de GB, são definidos pelo crescimento *level-wise*, enquanto o LightGBM utiliza *leaf-wise*.

2.4.3 Redes Neurais Artificiais

As Redes Neurais (RN), tecnicamente conhecidas como Redes Neurais Artificiais (RNA), são atualmente uma das mais populares e efetivas técnicas de AM. Em sua forma mais geral, uma rede neural é um sistema projetado para modelar a maneira como o cérebro realiza uma tarefa particular, assim como explica Fleck et al. (2016).

A estrutura das RNAs é dividia em camadas. Cada camada possui um conjunto de nodos, semelhantes a neurônios humanos, sendo que nodos de camadas adjacentes, mas não de uma mesma camada, são conectados entre si para poderem transferir informações. As camadas geralmente são classificadas como entrada, camadas ocultas e saída, como pode ser visto na Figura 2.11.

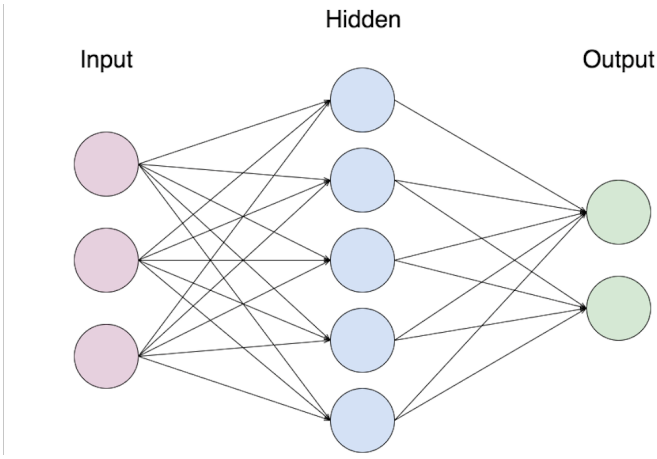


Figura 2.11: Estrutura padrão de uma RN. Imagem de Simoyama (2019)

A quantidade de nodos da camada de entrada usualmente é equivalente ao número de características que um objeto de entrada possui, como por exemplo a quantidade de pixels de uma imagem. As camadas ocultas tem sua estrutura totalmente variável, dependendo da concepção da rede. Já a camada de saída varia de acordo com a funcionalidade da rede, sendo que em uma rede de classificação, por exemplo, pode ter a quantidade de nodos igual a de classes.

Destaca-se que esta estrutura pode variar de acordo com a proposta da rede.

O nodo é a unidade básica de uma rede neural. Sua função é receber um conjunto de entradas, aplicar um cálculo numérico nos valores e retornar um novo valor de saída.

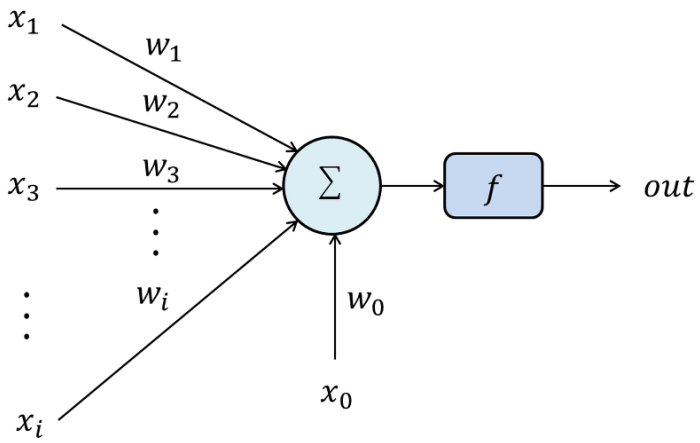


Figura 2.12: Estrutura de um nodo/neurônio. Imagem de Rocha (2017).

Na Figura 2.12, as notações $\{x_1 \ldots x_i\}$ representam os valores de entrada do nodo. Cada um desses valores será multiplicado por um respectivo número $\{w_1 \ldots w_i\}$, conhecido como peso. Seguidamente, todas as multiplicações são somadas, juntamente com um fator de inclinação, representado por w_0. Após isso, uma função de ativação f é aplicada, resultando no valor de saída do nodo, posteriormente transmitido para a próxima camada.

Por fim, os pesos de cada nodo são os fatores que determinantemente apontam, em conjunto, o valor final da saída de uma RN. Sendo assim, a cada iteração do treinamento, esses valores são ajustados pelos seus respectivos neurônios com base em quão errada a saída estava. Esse processo é conhecido como retropropagação, do inglês *back propagation*. O cálculo da taxa de erro é realizado pela função conhecida como *loss function* (LF), e a finalidade teórica

dos geradores é minimizar o valor dessa função, visto que isso significa que estão cada vez mais próximos do resultado esperado.

Como explicado, o modo de funcionamento das RNAs permite que milhares de estruturas e estratégias sejam propostas. A seguir serão mostrados os mais relevantes conceitos, funcionamentos e estruturas para o desenvolvimento deste trabalho.

2.4.3.1 Convolutional Neural Network

As Redes Neurais Convolucionais, do inglês *Convolutional Neural Network* (CNN), são uma forma de rede neural geralmente utilizada em entradas representadas como matrizes, tais como imagens. Assim como as RNAs, elas são compostas por nodos interconectados e divididos em camadas.

Seguindo o raciocínio de Li et al. (2022), existem dois tipos de camadas principais. As camadas de convolução são responsáveis por extrair características das entradas. Para isso, são aplicados filtros ou *kernels* em uma sub-região da matriz, sendo que cada um destes é especializado em detectar um aspecto diferente.

Depois das camadas de convolução existem as de *pooling*, responsáveis por reduzir a dimensão da matriz resultante da convolução, mantendo apenas as informações mais relevantes para a continuidade do processo. Isso é feito para que a rede seja capaz de assimilar tais dados de forma robusta e genérica.

Durante o processo de treinamento, com os exemplos de entrada, a rede aprende os pesos utilizados nos filtros assim como realizar precisamente a representação das características. Com isso, é notável que a CNN não necessite da extração de características prévias. A Figura 2.13 apresenta a estrutura de uma CNN.

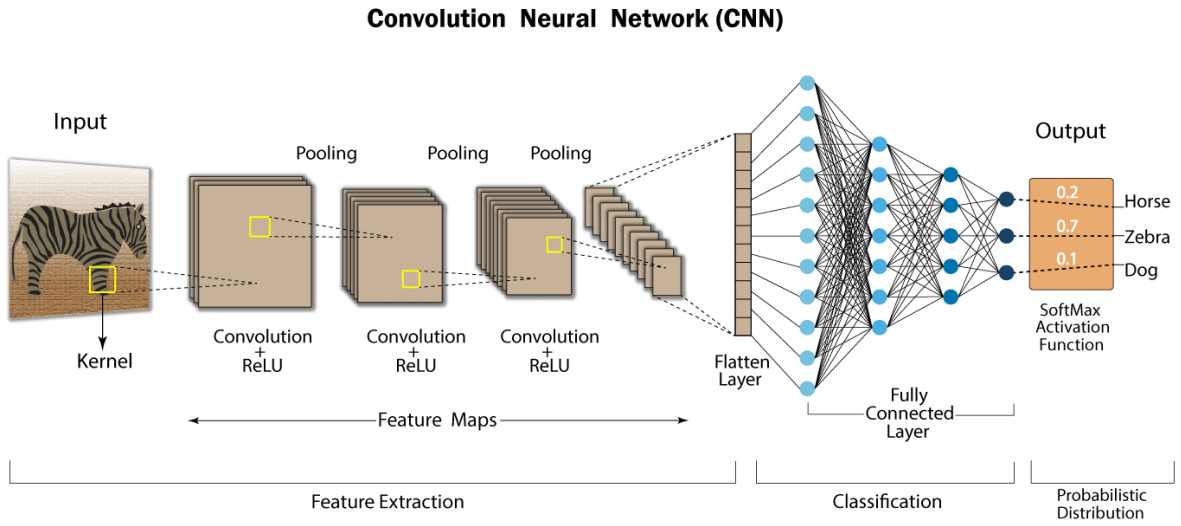


Figura 2.13: Exemplo de CNN. Imagem de Kalita (2022)

2.4.3.2 Recurrent Neural Network

As Redes Neurais Recorrentes, do inglês *Recurrent Neural Networks* (RNN), são um tipo específico de redes neurais especializadas em dados sequenciais, como texto e áudio. Segundo Medsker e Jain (2001), estas são um tipo de rede neural com retroalimentação, feita para aprender

padrões sequenciais ou repetidos em intervalos de tempo. Portanto, seu principal atributo é a capacidade de armazenar e usar informações de um contexto anterior para gerar uma nova saída.

De maneira geral, são compostas por apenas uma camada de neurônios recorrentes e conectados entre si, permitindo que as informações persistam na rede, para que os contextos anteriores sejam utilizados em novas saídas. Isso se difere das redes *feedforward*, como CNN, onde cada entrada é processada de forma individual. A Figura 2.14 apresenta a estrutura de uma RNN.

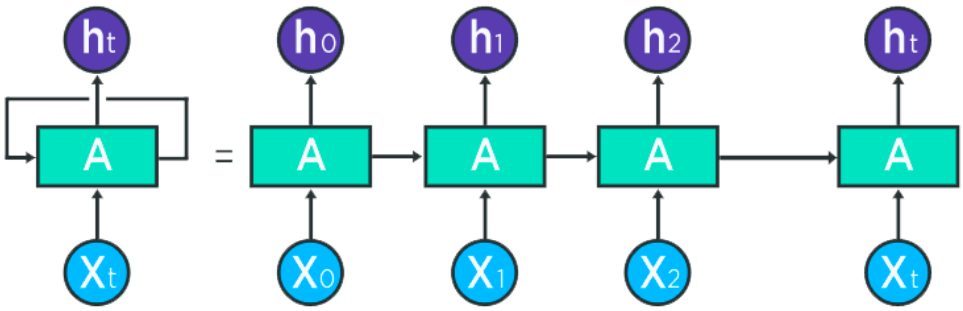


Figura 2.14: Exemplo de RNN. Imagem adaptada de Le (2019)

Dada a sua capacidade de entender e manter o contexto, esse tipo de rede é muito utilizada para geração de texto, reconhecimento de fala, dentre outros. Algumas variantes mais avançadas, como LSTMs (Memória de Curto e Longo Prazos) ou GRUs (Unidade Recorrente Fechada), permitem o reconhecimento de padrões de mais longo prazo.

2.4.3.3 Chord Language Model (CLM)

Um *Language Model* (LM), traduzido como modelo de linguagem, é um modelo matemático utilizado para prever uma sequência de dados. A sua implementação se dá geralmente pelo uso de RNAs e é amplamente utilizado por técnicas de processamento de linguagem natural.

Na utilização normal. o treinamento é feito utilizando grandes bases de dados de textos, fazendo com que o modelo aprenda a probabilidade de padrões existentes. Após isso, quando é dada uma nova entrada, é possível prever o conteúdo que vem em sequência, a partir de todo o conhecimento prévio adquirido no treinamento.

Os modelos de linguagem de acordes, do inglês *Chord Language Model* (CLM), são uma variante dos modelos de linguagem, sendo usados para a predição de acordes a partir de uma sequência dada. Essa técnica é amplamente aprofundada e discutida por Korzeniowski et al. (2018).

Para o seu funcionamento, os acordes são convertidos em *tokens*, análogos a palavras de uma frase. Com isso, a operação passa a ser a predição de *tokens* a partir de uma sequência já existente.

2.4.3.4 Generative Adversarial Networks

Segundo Creswell et al. (2018), as *Generative Adversarial Networks (GANs)*, são uma técnica muito relevante de aprendizado supervisionado e não supervisionado. Esse método foi proposto por Goodfellow et al. (2014), e seu funcionamento é baseado no treinamento em paralelo de duas RNs distintas que competem entre si.

A analogia mais clássica para o entendimento de GANs é o pensamento sobre um falsificador de arte e um perito em falsificação. O falsificador, conhecido tecnicamente como gerador, G, tem como objetivo criar falsificações que sejam as mais reais possíveis, enquanto o perito, conhecido como discriminador, D, recebe obras originais e falsificadas e tem o dever de identificar se cada uma é autêntica ou não. A Figura 2.15 representa a estrutura de uma GAN.

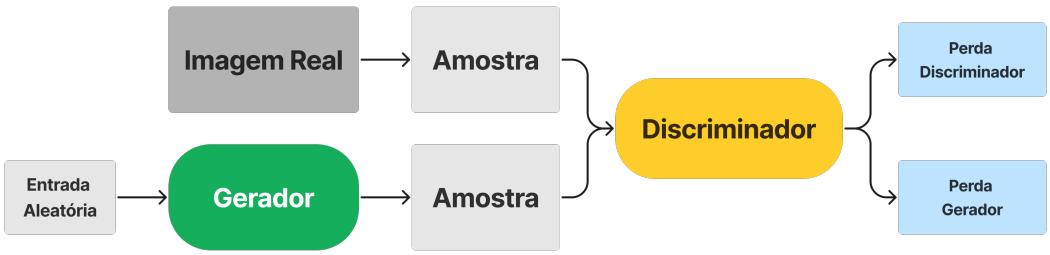


Figura 2.15: Estrutura de funcionamento de uma GAN.

A cada iteração do treinamento o gerador cria uma nova imagem falsa. Em seguida, o discriminador receberá a sua entrada, podendo ser a imagem falsificada de G, ou então uma imagem real da base de dados. Após D gerar um resultado afirmando se a imagem é falsa ou não, ambos ficam sabendo se o resultado foi correto ou não. Se o discriminador atuar sobre uma imagem do gerador e classificá-la verdadeira, significa que G cumpriu seu papel corretamente, enquanto D precisa ser melhorado com base nos seus valores de perda. Por outro lado, se D classificar como falsa uma imagem de G, então D agiu corretamente, fazendo com que G seja melhorado com seus valores de perda, até que seja possível enganar o classificador. É notável que nessa estrutura G e D são treinados ao mesmo tempo durante o processo. Dessa forma, o objetivo final é ter um gerador que crie imagens tão reais quanto as da base de dados usada.

Destaca-se que na explicação acima o conceito de imagem foi usado apenas como ilustração, mas no funcionamento real, o objeto de interesse do treinamento pode ser de diversos tipos.

Cada vez mais as GANs ganham relevância como técnica de AM, e os usos mais comuns são para geração de imagem, vídeo e áudio, aumento de base de dados e sintetização de fala a partir de textos.

2.4.3.5 CycleGAN

Proposta em 2017, e seguindo a definição de Zhu et al. (2017), a CycleGAN é uma técnica de aprendizado de máquina que utiliza como base o funcionamento das GANs, explicadas anteriormente, podendo ser abstratamente entendida como um encadeamento de GANs. Esse método é proposto para transferência de objetos entre domínios, isto é, por exemplo, dadas duas imagens, uma de cavalo e uma de zebra, o objetivo deste procedimento é transformar a imagem de um cavalo em uma zebra e vice-versa.

Para isso, a CycleGAN é formada por dois geradores, G_a e G_b, e dois discriminadores, D_a e D_b. Dados dois domínios de entrada, A e B, o gerador G_a é responsável por converter objetos do domínio B para o domínio A, gerando assim imagens falsas de A, enquanto G_b faz o contrário, gerando imagens falsas de B a partir de imagens de A. Os discriminadores D_a e

D_b definem se os artefatos dados como suas entradas pertencem ou não aos domínios A e B, respectivamente.

Neste ponto, são introduzidos dois conceitos importantes: conteúdo e estilo. O conteúdo se refere a base sólida de um objeto de entrada, isto é, dada uma imagem de entrada, por exemplo, se esta apresentar uma pessoa, então o conteúdo desta entrada é uma pessoa. Por outro lado, o estilo diz respeito a características mais abrangentes e subjetivas, tais como cor e textura. Com isso, por converter objetos de A para B, entende-se modificar o objeto A, mantendo seu conteúdo, mas aplicando o estilo de B.

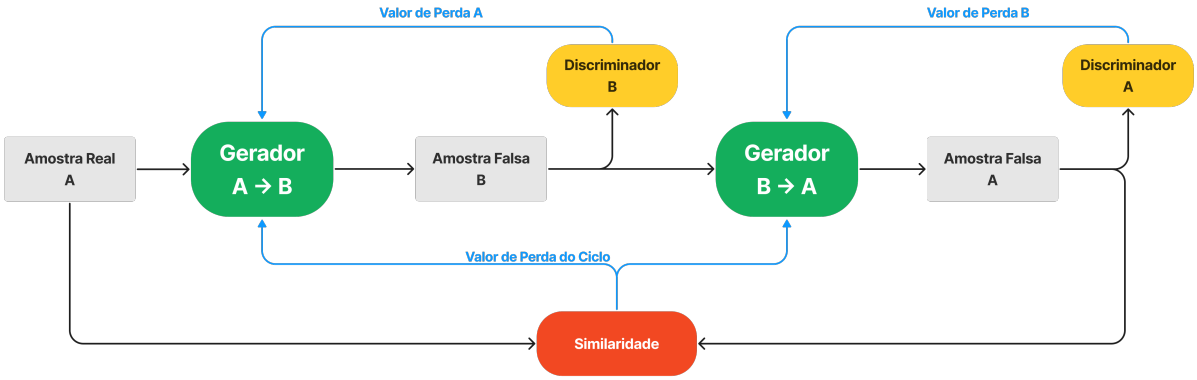


Figura 2.16: Fluxo estrutural e de funcionamento de uma CycleGAN

Seguindo o fluxo da Figura 2.16, percebe-se que para cada iteração do treinamento temos o objeto de entrada A (amostra real), o objeto B gerado por G_b a partir de A, e o segundo objeto A (amostra falsa), gerado por G_a a partir de B. Nota-se também que isso constitui um ciclo, visto que a entrada é do domínio A, sendo convertida para B, e novamente transformada para A.

Os valores de perda comuns, chamados aqui de *Loss* Adversário (LA), são gerados, assim como nas GANs, a partir das decisões tomadas pelos discriminadores, medindo a distância do resultado do gerador para um objeto real. Eles serão usados para ajustar os pesos das redes neurais dos geradores e também dos discriminadores.

Porém, o detalhe principal que representa o funcionamento de uma CycleGAN é o cálculo da diferença entre o objeto A de entrada e A', sendo este último o objeto reconstruído resultante do ciclo de processamento. Essa diferença, aqui chamada de *Loss* Cíclico (LC), é geralmente calculada a partir de uma fórmula de distância, como a distância L1. Esse valor é acrescido ao LA, estimulando o gerador a manter características importantes da imagem de entrada. Dado isso, podemos representar, por exemplo, o valor de perda do gerador G_b da seguinte forma:

$$LossG_b = LA_b + LC \tag{2.3}$$

Dessa forma, o final do treinamento resulta em dois geradores com domínios diferentes, um de A para B e outro com o contrário. E por esse funcionamento e efetividade, as CycleGANs são amplamente utilizadas para aplicações de transferência de estilo.

2.5 TRANSFERÊNCIA DE ESTILO

A Transferência de Estilo (TE) é um procedimento de manipulação de objetos de interesse, onde dadas duas entradas, sendo uma de conteúdo (C) e outra de estilo (E), o objetivo é abstrair as características da entrada E, e aplicá-las na entrada C, de forma que o resultado seja um novo objeto com o conteúdo de C e o estilo de E. Tais objetos podem ser imagens, músicas, vídeos, textos, dentre outros.

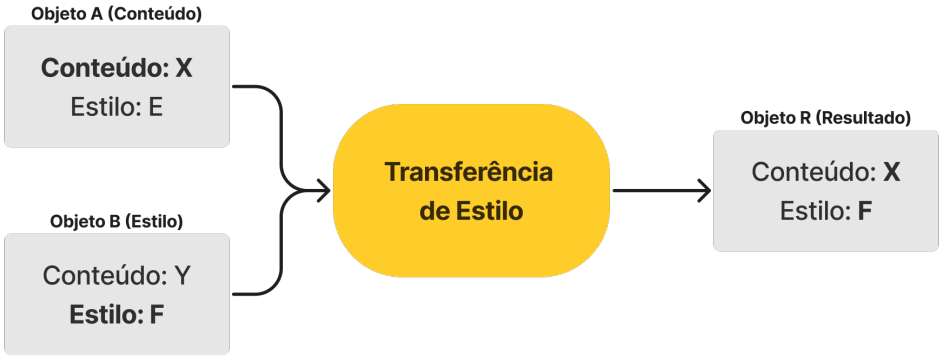


Figura 2.17: Diagrama de Transferência de Estilo

A exemplo da Figura 2.17, nota-se que existem dois objetos de entrada, A e B, sendo estes de conteúdo e estilo, respectivamente. Após a aplicação da transferência de estilo, obtêm-se um objeto de resultado R, contendo o conteúdo de A (X) mas com o estilo de B (F).

Pela versatilidade, a TE possui cada vez mais aplicações. Atualmente as que mais se destacam são criação de obras de arte, sumarização de textos, geração de descrição de imagens e produção sintética de fala com base em texto.

2.5.1 Técnicas de transferência de estilo

Assim como em todas as áreas da computação, na transferência de estilo existem diversas técnicas que buscam o mesmo objetivo. Alguns métodos tendem a ser especializados em um tipo específico de objeto de entrada, como somente imagem, ou então texto. Aqui serão mostradas e brevemente explicadas algumas estratégias que se destacam por seu desempenho e resultado.

2.5.1.1 Neural Style Transfer

Sendo uma das mais clássicas técnicas de TE, a *Neural Style Transfer* (NST), é uma técnica de aprendizado de máquina profundo, que geralmente usa em sua estrutura Redes Neurais Convolucionais (RNC) para transferir o estilo de um objeto de entrada para outro.

Usando uma imagem como exemplo de objeto de interesse, pela definição de Gatys et al. (2015), o processo contém um gerador G, sendo este uma rede neural, e se inicia com uma imagem de ruído (I), uma imagem de conteúdo (C) e uma imagem de estilo (E). I, C e E são passados para uma RNC, como por exemplo a VGG, para que os valores de conteúdo e estilo

sejam extraídos. Após a obtenção de tais valores, o algoritmo os aplica na imagem I da iteração corrente. Em seguida, são calculados os valores de *Loss* do Conteúdo (LCT) e *Loss* do Estilo (LET). LCT é definido como o valor da similaridade existente entre o mapa de características de conteúdo das imagens I e C. LET é definido como o valor da similaridade existente entre a matriz de Gram de características de estilo das imagens I e E. Com isso, os valores LCT e LET são usados para o processo de retropropagação do gerador G, como forma de minimizar tais funções de perda e obter resultados melhores. Então, G gera uma nova imagem, que tomará o lugar de I, e então o processo se repete até a convergência da função de perda.

2.5.1.2 CycleGAN

A transferência de estilo utilizando CycleGAN segue exatamente o mesmo processo de treinamento e utilização descrito em 2.4.3.5. Dados dois domínios de interesse, A e B, com a estrutura de GANs adversárias, ao final do treinamento são obtidos dois geradores, sendo um para objetos do domínio A e outro para B. A competição existente no processo faz com que resultados de boa qualidade sejam gerados, e com isso, a CycleGAN é uma das técnicas de aprendizado de máquina mais utilizada para a transferência de estilo. A Figura 2.18 contém um exemplo de transferência entre cavalos e zebras.

Figura 2.18: Exemplo de TE usando CycleGAN. Imagem de Brownlee (2020)

3 ESTADO DA ARTE

Aqui serão descritos os projetos e pesquisas que embasam e motivam o desenvolvimento deste trabalho.

3.1 GROOVE2GROOVE

Estabelecido por Cífka et al. (2020) em 2020, o projeto Groove2Groove (G2G) consiste em um método de transferência de estilo *One-Shot* (OS) para músicas simbólicas, tal como MIDI. De maneira geral, os algoritmos comuns de transferência de estilo necessitam de um treinamento para cada classe que se deseja transferir. Em contraponto a isso, o conceito *One-Shot* refere-se ao processo de aprender as características de uma classe a partir de um único exemplo de entrada, sem a necessidade de um treinamento completo para a classe de interesse.

O foco de G2G está em TS para os estilos Pop e Jazz, utilizando uma rede neural *encoder-decoder* para realizar a tarefa. Além disso, foi empregada uma técnica de produção sintética de dados para suprir as necessidades do treinamento supervisionado, fato que será explicado mais adiante.

De maneira abstrata, dados dois arquivos MIDI de entrada, A e B, o objetivo é transformar a entrada A de tal maneira que ao final do processo ela mantenha seu conteúdo, mas com a aplicação do estilo de B.

Portanto, a partir do conceito de música e transferência de estilo, é possível oferecer uma nova ferramenta para musicistas e produtores musicais, estimulando a criatividade e o desenvolvimento de novas obras com alta qualidade.

3.1.1 Estilo e Conteúdo

Existem várias plausíveis definições de estilo musical, desde algumas mais conceituais até outras mais técnicas. O mais comum é utilizar estilo como sinônimo de gênero musical, permitindo associar músicas distintas, mas com características semelhantes, em grupos abrangentes. Dessa forma, podemos classificar Pop e Jazz como estilos musicais. Por outro lado, estilo musical também pode ser entendido como os inúmeros aspectos que formam uma música, tais como composição, arranjo, melodia, timbre, ritmo, dentre outros. Com isso, compreende-se abstratamente que cada música ou artista possui seu próprio estilo, criando assim infinitas possibilidades. O Groove2Groove utiliza esta segunda definição.

Sendo os objetos de interesse, músicas dos estilos Pop e Jazz comumente seguem um mapa e sequência de acordes que definem a harmonia e o ritmo base da música. Esse conceito é popularmente conhecido em inglês como *Chord Chart* (CC), e pode ser observado no topo da figura 3.1, na forma $D-^7$, $G-^7$, C^{maj7}, etc.

Durante uma execução, o *Chord Chart* é usado como base para a criação do acompanhamento, sendo este a escolha dos instrumentos, timbres, pequenas modificações melódicas, harmônicas e rítmicas. Partindo disto, o G2G considera o CC como o conteúdo e o estilo como o processo de converter este CC em um acompanhamento. Este ponto necessita de uma subjetividade, visto que assume-se que o *Chord Chart* é independente do estilo, o que nem sempre é verídico. Assim sendo, a tarefa do G2G passa a ser criar um novo acompanhamento para a entrada A, com o estilo de B.

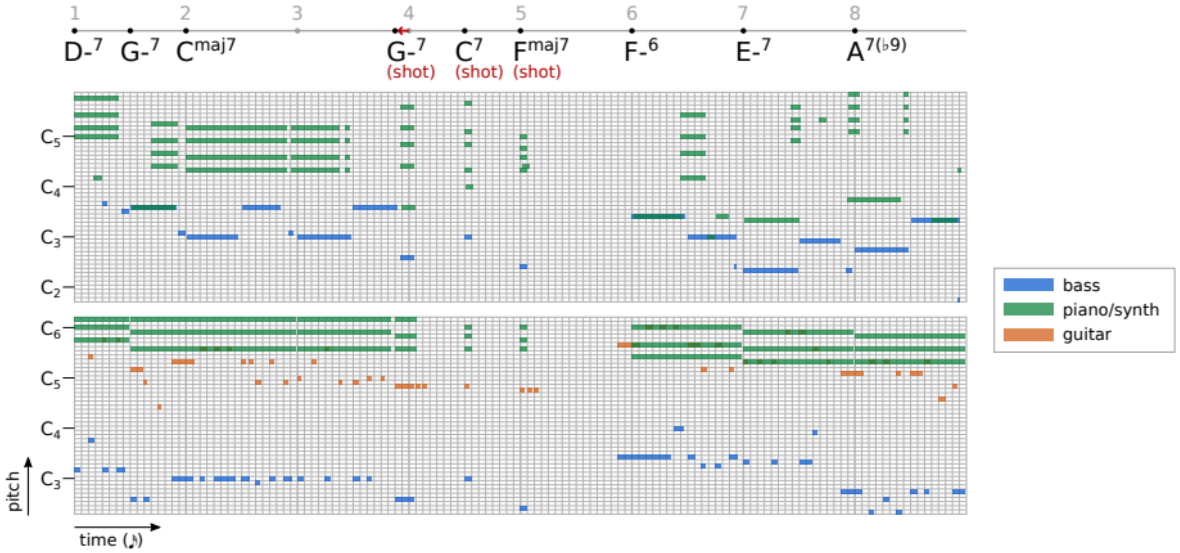


Figura 3.1: Exemplo da representação simbólica de duas músicas, contendo o *Chord Chart* no topo. Imagem de Cífka et al. (2020).

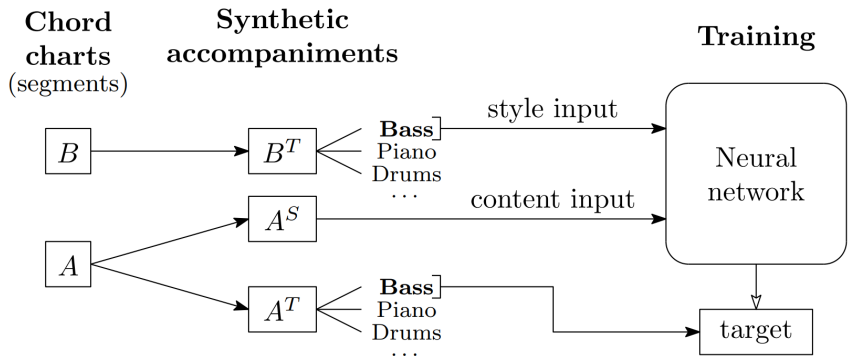


Figura 3.2: A ideia central da abordagem do G2G. Imagem e descrição adaptadas de Cífka et al. (2020).

Na Figura 3.2, nota-se a exemplificação da ideia central do projeto. Neste caso, a partir de tabelas de acordes (representada por A e B), criam-se acompanhamentos sintéticos em diferentes estilos (S e T) usados como dados de treinamento para uma rede neural. A rede recebe a entrada de conteúdo A^S (ou seja, acompanhamento para A no estilo S) e uma única faixa da entrada de estilo B^T, e é treinada para gerar a faixa correspondente do acompanhamento de destino A^T.

3.1.2 Base de Dados

Como dito anteriormente, o G2G funciona com aprendizado de máquina supervisionado. Com isso, a base de dados de treinamento consiste em triplas de fragmentos musicais, da forma (A^S, B^T, A^T), onde A^S é a entrada de conteúdo com o estilo S, B^T é a entrada de estilo, sendo um exemplo do estilo almejado T, e A^T é o alvo do treinamento, ou seja, a saída esperada combinando o conteúdo de A^S com o estilo de B^T.

O processo de obtenção e tratamento dos dados se baseia em dois pilares principais. O primeiro deles é a utilização do *software* musical *Real Band*, juntamente com o pacote *Band-in-a-Box* (BIAB), o qual permite a criação automática de acompanhamentos, aqui entendidos como estilos. O segundo é a criação de CCs a partir de um *chord language model*, explicado em 2.4.3.3,

e treinado a partir do iRb *corpus*, uma biblioteca com milhares de padrões de Jazz, definida em Broze e Shanahan (2013). Ambos os processos serão explicados com mais detalhes a seguir.

3.1.2.1 Obtenção do Mapa de Acordes

Partindo dos mapas de acordes do iRb *corpus*, cada acorde representado no mapa é convertido para um *token*, que contém também as características, como a duração, de tal acorde. Estes *tokens* são usados no treinamento do *chord language model*, que tem a capacidade de gerar um novo *token* a partir dos prévios. Após o treinamento, é gerada uma sequência de *tokens*, que serão convertidos para CCs e então passados pelo BIAB para a adição de variações aleatórias. Esse procedimento é necessário para que o modelo final de transferência de estilo consiga lidar com variações e com a imprevisibilidade dos estilos.

Para os experimentos mostrados em Cífka et al. (2020), foram gerados 1200 *chord charts* para o treinamento, 600 para validação e mais 600 para teste.

3.1.2.2 Obtenção do Acompanhamento

O software BIAB tem a capacidade de gerar acompanhamentos MIDI para *chord charts* em qualquer estilo disponível em sua biblioteca. Nesse âmbito, entende-se estilo como a segunda explicação dada em 3.1.1, ou seja, características únicas de uma ou um conjunto de músicas, e não necessariamente vinculando-as a um gênero musical, tal como Jazz.

Para obter os estilos usados no treinamento, foram obtidos 1,476 estilos do BIAB, criando a Lista de Estilos (LI), sendo 20 separados para validação e mais 20 para testes, de forma que cada um dos acompanhamentos tenha entre 3 e 5 instrumentos. Com isso, dados os CCs gerados em 3.1.2.1, utilizou-se o BIAB para gerar alguns acompanhamentos para cada, dentro dos estilos de LI, de forma a atingir aproximadamente 500 amostras para cada estilo pertencente a LI, sendo em seguida divididos em fragmentos pequenos de 8 compassos. Na Figura 3.1 é possível ver um exemplo de dois acompanhamentos gerados para um único CC, exemplificando em uma escala pequena o que acontece no processo descrito acima.

Ressalta-se que, nesta esfera, estilo e acompanhamento são usados como sinônimos.

3.1.2.3 Agregação dos Dados

Depois dos procedimentos citados em 3.1.2.1 e 3.1.2.2, é necessário transformar os dados nas triplas de formato (A^S, B^T, A^T). Isso é feito por um processo de repetição dentro de todas as possibilidades de pares (A^S, A^T), tal que $S \neq T$, existentes dentro do conjunto gerado com os estilos de LI a partir dos CCs prévios.

Na Figura 3.3(a) estão exemplificados os dados que são usados para gerar cada tripla. Cada linha corresponde a um estilo, cada coluna a uma tabela de acordes e cada ponto é um exemplo existente no conjunto de dados.

3.1.3 Modelo Proposto

O modelo proposto no G2G é uma rede neural do tipo *encoder-decoder*, com sua estrutura básica representada na Figura 3.3 (b) e detalhada na Figura 3.4. Estão presentes dois *encoders*, um para conteúdo e outro para estilo, com consequentemente duas entradas, e um *decoder* que gera a saída.

O *Encoder* de Conteúdo (EC) recebe a entrada de conteúdo C, contendo todas as linhas de MIDI de tal arquivo. O *Encoder* de Estilo (EE) recebe a i-ésima linha de MIDI da entrada de

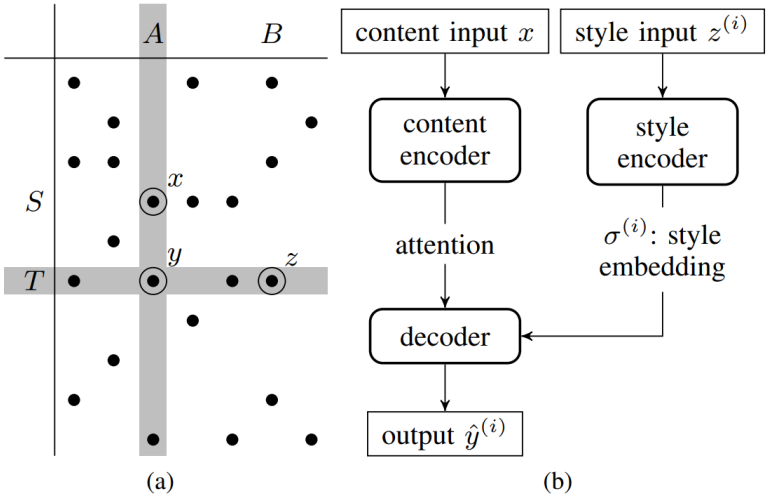


Figura 3.3: (a) Triplas de treinamento (b) Visão de alto nível da arquitetura do modelo. Imagem e descrição adaptadas de Cífka et al. (2020).

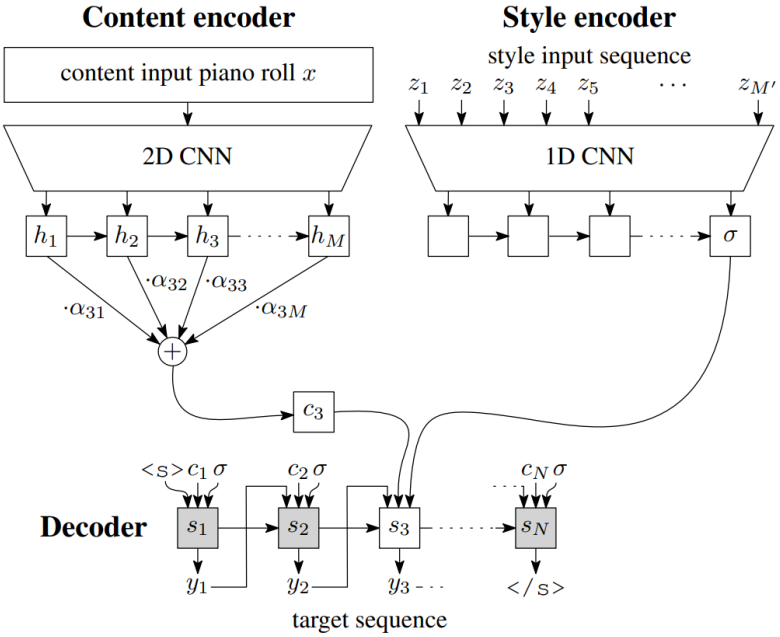


Figura 3.4: Arquitetura detalhada do modelo utilizado no G2G. Imagem e descrição adaptadas de Cífka et al. (2020).

estilo (E), representado como z^i. Então, o decodificador combina as saídas de EC e EE, gerando um resultado correspondente a i-ésima linha de MIDI da saída, denotado como y^i. Destaca-se que nenhuma informação além de z^i é dada explicitamente, com isso é necessário que o modelo descubra apenas a partir disso a maneira correta de chegar na saída esperada. Esse processo é repetido N vezes, sendo N a quantidade de linhas de MIDI presentes em E, fazendo a saída final ser um conjunto de linhas de MIDI, correspondentes aos instrumentos e quantidade existente no acompanhamento E, mas com o conteúdo de C.

Esse modelo permite que os *inputs* sejam genéricos e independentes da quantidade de linhas de MIDI presentes, sendo facilmente adaptáveis a diferentes estruturas e combinações não vistas no treinamento. Porém, faz com que os resultados sejam muito condicionados à entrada de conteúdo.

3.1.3.1 Representação de Dados

As entradas de conteúdo e de estilo são arquivos MIDI, mas são usadas com propósitos diferentes dentro da utilização do modelo proposto. Para isso, a maneira como esses dados são representados é alterada para cada tipo de entrada.

A entrada de estilo e a saída do processo usam uma representação baseada em eventos, como o MIDI. Em outras técnicas, o conceito de tempo é codificado como o intervalo entre eventos, mas nessa abordagem usa-se a codificação a partir da divisão rítmica comum, conhecida como *beat-relative*. Portanto, cada linha de MIDI é codificada como uma sequência de *tokens* de eventos, como *NoteOn*, *NoteOff*, *DrumOn*, *DrumOff*, *SetVelocity*, dentre outras.

A entrada de conteúdo contém diversas linhas que deverão ser processadas juntas. Para isso, é usada uma matriz de *piano roll*, explicada em 2.3.5, com 128 linhas e 4 colunas por batida. Sendo que nessa matriz, cada valor preenchido representa a intensidade de uma determinada nota em um tempo. Com isso, é possível obter uma representação na qual o tamanho depende apenas do tamanho da entrada, fato que auxilia o algoritmo computacionalmente, e diferentemente da baseada em eventos, que varia de acordo com a quantidade de eventos. Destaca-se que o instrumento bateria é descartado do *piano roll*, sendo esta uma limitação do projeto.

3.1.3.2 Detalhes da Arquitetura do Modelo e do Treinamento

Seguindo a estrutura mostrada na Figura 3.4, ambos os codificadores são uma CNN com ativações de Unidade Linear Exponencial (ELU), seguidos por uma RNN contendo uma GRU. As redes convolucionais têm o papel de reduzir a dimensionalidade da entrada, enquanto a RNN integra as informações necessárias durante o treinamento. Porém, a estrutura dos *encoders* é modificada de acordo com a representação dos dados.

O *encoder* de conteúdo aplica consecutivamente duas 2D CNNs na matriz do *piano roll*. Após, a saída tridimensional de características é achatada para uma sequência de vetores de 1024 dimensões, que então será passada para uma GRU, resultando em uma sequência de vetores de 200 dimensões, sendo estes $h_1, ..., h_M$.

O *encoder* de estilo aplica uma sequência de três consecutivas CNNs aos *tokens* de entrada, comprimindo os dados oito vezes. Após, é utilizada uma GRU com 500 unidades, sendo que a última unidade é mantida e usada como representação do vetor de estilo (*style embedding*), representado como σ^i. Salienta-se que o *encoder* tem acesso a apenas uma linha de MIDI, z^i, por iteração, sendo assim a representação de estilo, σ^i, caracteriza apenas a linha z^i.

Já o decodificador segue uma arquitetura *seq2seq* adaptada de Bahdanau et al. (2014), também usando GRU, tentando sempre predizer o próximo *token*, y^i_n, dado um contexto prévio $y^i_1, ... , y^i_{(n-1)}$. Por fim, a saída do *decoder* é uma probabilidade de distribuição *softmax* em y^i_n.

O treinamento de todo o modelo é feito usando Kingma e Ba (2014), com um *batch size* de 64 e com uma taxa de aprendizado exponencial e agressiva, dividindo-a na metade a cada três mil conjuntos de treino, prevenindo assim o *overfitting*.

3.1.4 Avaliação

Aqui serão descritas as métricas e técnicas usadas em Cífka et al. (2020) para avaliar a transferência de estilo.

3.1.4.1 Preservação de Conteúdo

Após a aplicação do modelo descrito em 3.1.3 em uma nova entrada, uma das possibilidades é calcular o quanto o *chord chart* da entrada de conteúdo foi preservado na saída, isto é, o

quanto a sequência, estrutura e harmonia foram transmitidas corretamente pelo modelo. Com essa finalidade, é utilizada a similaridade de cosseno *frame-wise*, que calcula, por amostra, a semelhança entre os vetores de características da entrada e da saída.

Dado que a harmonia pode ser gerada a partir de um conjunto de instrumentos, então esta métrica é aplicada na entrada e saída como um todo, e não em linhas individuais.

3.1.4.2 Correspondência de Estilo

A correspondência de estilo é calculada em com base em 4 diferentes perfis de estilo, medindo o quanto o estilo da entrada é preservado na saída. As métricas representadas na Tabela 3.1 são computadas como uma similaridade de cosseno entre histogramas criados a partir das definições da coluna do meio da tabela. *Start e End* são o tempo de início e fim das notas, respectivamente. *Pitch e Velocity* são respectivamente a nota representada e sua intensidade.

Métricas	**Observações**	**Bins**
pw. time-pitch	(start(b) - start(a), pitch(b) - pitch(a))	24×41
onset-duration	(start(a) mod 4, end(a) - start(a))	24×12
onset-velocity	(start(a) mod 4,velocity(a))	24×8
onset-drum	(start(a) mod 4,pitch(a))	24×128

Tabela 3.1: Métricas de correspondência de estilo. Tabela e descrição adaptadas de Cífka et al. (2020).

O perfil *time-pitch* é calculado a partir de todos os pares de notas a, b existentes no intervalo de 4 batidas. Dados os pares, constrói-se um histograma 2D contendo no eixo x a diferença entre os momentos em que as notas a e b se iniciam, e com o intervalo de notas *(pitch interval)* entre a e b, plotado no eixo y. Após a obtenção de tal histograma, calcula-se a similaridade de cosseno entre este e uma referência confiável daquele estilo alvo.

Os outros perfis, como *onset-duration*, *onset-velocity* e *onset-drum*, seguem a mesma lógica de funcionamento descrita acima, mas utilizando outras medidas para analisar aspectos de duração e intensidade, que não são capturados pelo *time-pitch*.

Portanto, os perfis de estilo são calculados em todas as saídas de teste, medindo a similaridade de cosseno com as referências para dado estilo, sendo descritos os valores de média e desvio padrão, sendo conhecida aqui como macrométrica. Para os conjuntos de dados não sintéticos, ou seja, não gerados pelo processo descrito em 3.1.2, a referência de estilo não está disponível, então as saídas são comparadas com os perfis de entrada de estilo do modelo, sendo conhecida aqui como micrométrica.

3.1.5 Experimentos

Os experimentos são realizados em um conjunto de testes gerado com o processo de 3.1.2 e também na base de dados Bodhidharma, definida em McKay e Fujinaga (2005), com 950 arquivos MIDI, ambos não utilizados em nenhum momento do processo de treinamento. Após um pré-processamento, obteve-se 8.934 fragmentos de MIDI contando com a normalização da intensidade das notas.

Para os dados sintéticos, a triplas (A^S, B^T, A^T) são construídas para os testes, enquanto que para o Bodhidharma, visto que o alvo do treinamento não está disponível, as entradas são da forma (A^S, B^T), sendo B^T aleatoriamente escolhido.

Ademais, são definidos 3 pontos base para a comparação dos resultados do Groove2Groove:

- *CP-CONTENT:* a saída contém uma cópia do conteúdo de entrada, sendo avaliado puramente o estilo; espera-se ter um bom desempenho na preservação de conteúdo mas ruim na correspondência de estilo.
- *CP-STYLE:* a saída contém uma cópia do estilo de entrada, sendo avaliado puramente o conteúdo; espera-se ter um bom desempenho na correspondência de estilo mas ruim na preservação de conteúdo.
- *ORACLE:* a saída contém uma implantação do estilo definido no BIAB; fornece uma avaliação mais realista em todas as métricas.

3.1.5.1 Avaliação dos Resultados

Iniciando com os dados sintéticos, na avaliação da preservação de conteúdo, o Groove2Groove apresenta um resultado excelente, praticamente igual ao *ORACLE*, o qual representa a perfeição. Nos perfis de estilo (macrométrica), o Groove2Groove alcança um resultado maior que o *CP-CONTENT*, mas inferior a *ORACLE* e *CP-STYLE*. Isso significa que a saída está mais próxima do estilo alvo do que a entrada de conteúdo, mostrando que a transferência de estilo funciona parcialmente.

Partindo para a avaliação dos arquivos do Bodhidharma, usando micrométrica, os resultados de preservação de conteúdo e de estilo são significativamente inferiores do que nos dados sintéticos, mostrando que essa base de dados é mais desafiadora para o modelo proposto. Porém, o Groove2Groove ainda supera o valor de CP-CONTENT nas métricas de estilo e de CP-STYLE na preservação de conteúdo.

3.1.5.2 Comparação Subjetiva

A comparação subjetiva é feita ouvindo as saídas do modelo. É dito que, de maneira geral, pela avaliação do trabalho as saídas fazem sentido musicalmente e seguem a harmonia da entrada de uma maneira bem precisa. Também é possível notar que a "sensação" da entrada de estilo, principalmente rítmica e de variações de notas, é mantida. Entretanto, em alguns momentos foi observada falha em reproduzir tais características.

3.2 SYMBOLIC MUSIC STYLE TRANSFER

O Symbolic Music Style Transfer (SMST), traduzido como Transferência de Estilo de Música Simbólica, é uma técnica de transferência de estilo proposta em Brunner et al. (2018), que tem por objetivo transferir um pedaço de uma música de um gênero para outro, fazendo com que esta transferência seja notada, mas também mantendo as características melódicas e estrutura da entrada original.

Esse método não supervisionado adapta a CycleGAN, explicada em 2.4.3.5, para a transferência de música simbólica de um gênero A para outro B, realizando apenas a mudança das notas utilizadas (*pitch changes*).

3.2.1 Arquitetura do Modelo

Seguindo a lógica da CycleGAN, todo o funcionamento se baseia em geradores (G) e discriminadores (D) competindo entre si. G tenta 'enganar' D, enquanto D se concentra em detectar uma amostra original ou gerada por G. A estrutura proposta consiste na adaptação do

encadeamento de duas GANs, assim como uma CycleGAN, contendo dois geradores, mas quatro discriminadores, dois a mais do que o normal.

Tomando por exemplo a transferência de domínio entre A e B, onde domínio corresponde a um gênero musical, como Pop ou Jazz, tem-se um gerador de A para B, $G_{A\to B}$, e um com o contrário, $G_{B\to A}$. Também obtém os discriminadores de amostras falsas de A e B, D_A e D_B, respectivamente. Além disso, para o modelo ser capaz de extrair e interpretar características de mais alto nível, mais dois discriminadores são adicionados, sendo eles $D_{A,m}$ e $D_{B,m}$.

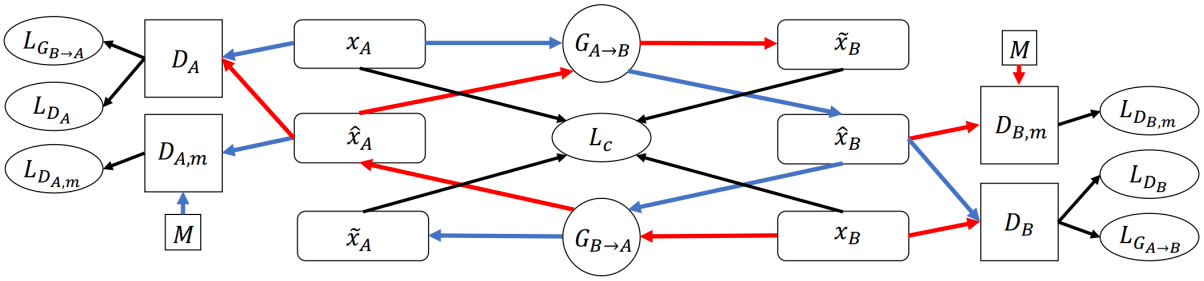


Figura 3.5: Arquitetura do modelo SMST. Imagem e descrição adaptadas de Brunner et al. (2018).

Na Figura 3.5, nota-se a representação da estrutura do modelo proposto pelo SMST. As setas azuis e vermelhas indicam as direções da transferência de estilo, enquanto as pretas indicam as funções de perda. Seguindo as representações azuis, x_A indica uma amostra real do domínio A. $\hat{x}_B$ denota a mesma amostra x_A, mas depois de ser transferida para o domínio B, tal que $\hat{x}_B = G_{A\to B}(x_A)$. Então, $\tilde{x}_A$, expressa $\hat{x}_B$ transferido de volta para o domínio A, ou seja, um ciclo, de forma que $\tilde{x}_A = G_{B\to A}(G_{A\to B}(x_A))$. Por outro lado, as representações vermelhas caracterizam o mesmo processo mas para o outro domínio, isto é, partindo de x_B que realmente pertence ao domínio B. M é a base de dados que contém amostras dos dois domínios, $M = A \cup B$, usados na entrada de amostras verdadeiras dos discriminadores.

As funções de perda são divididas em adversárias e cíclica. As adversárias são calculadas a partir da norma L2, obtendo como resultado:

$$L_{G_{A\to B}} = \| D_B(\hat{x}_B) - 1 \|_2 \tag{3.1}$$

$$L_{G_{B\to A}} = \| D_A(\hat{x}_A) - 1 \|_2 \tag{3.2}$$

Contudo, o *loss* cíclico, explicado em 2.4.3.5, se baseia no termo L1, sendo calculado como:

$$L_C = \| \tilde{x}_A - x_A \|_1 + \| \tilde{x}_B - x_B \|_1 \tag{3.3}$$

Ainda seguindo o raciocínio de Brunner et al. (2018) e enxergando de uma maneira mais abrangente a tarefa de um gerador (G) de aprender a transformar de um gênero de origem para um alvo, é necessário que G saiba as características de ambos os gêneros, para que o discriminador possa ser enganado. Porém, isso pode enviesar o gerador a criar objetos que enganem o discriminador (D), mas que não necessariamente soem como uma música real. Para resolver este problema, dois novos discriminadores (D_m) são adicionados como forma de gerar diferentes valores de perda e fazer com que o gerador aprenda outras características de alto nível não assimiladas com apenas dois Ds. Nessa proposta, a principal diferença entre D e D_m é que D se concentra em distinguir uma amostra falsa de uma verdadeira de um único gênero, enquanto D_m funciona distinguindo-a de múltiplos gêneros pertencentes a M. O resultado desse processo

faz com que o gerador mantenha mais características da entrada original e gere saídas que fazem sentido musicalmente.

3.2.2 Base de Dados e Pré-processamento

A base de dados utilizada no treinamento é formada pelos gêneros musicais Jazz, Clássico e Pop, com amostras que foram coletadas de diversas fontes. Essas amostras são arquivos MIDI, representados por meio de um *piano roll*.

Como dito anteriormente, o *piano roll* é uma representação em uma matriz de um arquivo MIDI, de tamanho definido pelo número de notas que podem ser representadas, geralmente 127, multiplicado pela quantidade de divisões rítmicas utilizadas. Para esse método, o compasso foi dividido em 16 vezes. Além disso, escolheu-se reduzir o número de notas, considerando apenas o intervalo entre as notas C1 e C8, resultando em 84 possíveis notas. Portanto, a matriz possuirá 16 x 84 posições por compasso. Também optou-se por utilizar uma frase de 4 compassos para a entrada do treinamento, resultando em uma matriz de tamanho 64 x 84.

Os arquivos MIDI podem conter internamente diversas linhas individuais, cada uma para um instrumento separado. Porém, o modelo proposto se concentra em fazer a transferência de estilo apenas com a mudança das notas. Partindo disso, a primeira fase do pré-processamento é fundir todas as linhas de MIDI em uma só. Isso é feito com a justificativa de manter juntas a maior quantidade possível de informações da entrada, mesmo que cause uma confusão sonora. Além disso, as linhas de bateria, se existirem, são omitidas. Na sequência, todas as amostras que não possuem a assinatura de tempo 4/4 são descartadas, e as restantes são divididas em fragmentos de 4 compassos. Isso resulta em 12.341 amostras de Jazz, 16.545 de Clássico e 20.780 de Pop.

3.2.3 Parâmetros e Treinamento

Para o treinamento, antes de ser passada para a rede neural, a amostra passa por uma normalização de *pitch*, fazendo com que todas as representações de notas fiquem no intervalo entre 0 e 1.

Input: ($batchsize \times 64 \times 84 \times 1$)					
layer	filter	stride	channel	instance norm	activation
conv	4×4	2×2	64	False	LReLu
conv	4×4	2×2	256	True	LReLu
conv	1×1	1×1	1	False	None
Output: ($batchsize \times 16 \times 21 \times 1$)					

Tabela 3.2: Estrutura do discriminador do jMIR

A estrutura da rede do discriminador e do gerador podem ser vistas nas Tabelas 3.2 e 3.3, respectivamente. Destaca-se que é usado um otimizador Adam, que considera os valores das funções de perda para atualizar os pesos da rede.

3.2.4 Experimentos

Para a avaliação dos experimentos, os autores testaram diferentes técnicas e métricas que serão explicadas a seguir, e combinadas ao final para a avaliação dos resultados.

Input: ($batchsize \times 64 \times 84 \times 1$)					
layer	filter	stride	channel	instance norm	activation
conv	7×7	1×1	64	True	ReLu
conv	3×3	2×2	128	True	ReLu
conv	3×3	2×2	256	True	ReLu
$10 \times$ ResNet	3×3	1×1	256	True	ReLu
	3×3	1×1	256	True	ReLu
deconv	3×3	2×2	128	True	ReLu
deconv	3×3	2×2	64	True	ReLu
deconv	7×7	1×1	1	False	Sigmoid
Output: ($batchsize \times 16 \times 21 \times 1$)					

Tabela 3.3: Estrutura do gerador do jMIR

3.2.4.1 Classificador de Gênero

Para avaliar parametricamente a transferência, foi construído um classificador (C) de estilo $C_{A,B}$, que retorna a probabilidade de uma amostra de entrada pertencer ao estilo A (P_A) ou B (P_B). A arquitetura da rede neural deste classificador pode ser vista na Tabela 3.4.

Input: ($batchsize \times 64 \times 84 \times 1$)					
layer	filter	stride	channel	instance norm	activation
conv	1×12	1×12	64	False	LReLu
conv	4×1	4×1	128	True	LReLu
conv	2×1	2×1	256	True	LReLu
conv	8×1	8×1	512	True	LReLu
conv	1×7	1×7	2	False	Softmax
Output: ($batchsize \times 2$)					

Tabela 3.4: Estrutura do classificador do jMIR

Os dados de treinamento do classificador são reais e os mesmos do treinamento do modelo em 3.2.1, com uma divisão 90/10 para dados treino e teste. Os resultados de acurácia de treinamento dos classificadores são os seguintes:

- **Jazz x Classic:** 88,89 %
- **Classic x Pop:** 84,66 %
- **Jazz x Pop:** 67,07 %

É notável que para os gêneros *Jazz* e *Pop* o classificador apresenta um resultado abaixo dos demais, indicando que C entende-os como gêneros semelhantes, levando em consideração apenas as notas utilizadas.

Na sua utilização, o classificador é avaliado apenas com dados falsos, ou seja, gerados pelo modelo. Partindo disso, uma transferência de estilo é aqui considerada como efetiva quando $P_A(x_A) > 0{,}5$ e $P_A(\hat{x}_B) < 0{,}5$, ou seja, se a amostra transferida está mais longe do gênero de origem do que a amostra de origem.

Para medir a força e qualidade da transferência de estilo uma equação foi desenvolvida com base nos valores de probabilidade resultantes do classificador. A exemplo da transferência ($A \rightarrow B \rightarrow A$) tem-se:

$$S^D_{A\rightarrow B} = \frac{1}{2}(P(A|x_A) - P(A|\hat{x}_B) + P(A|\tilde{x}_A) - P(A|\hat{x}_B)) \tag{3.4}$$

Partindo disso, também é possível definir um valor para o modelo completo, considerando os dois sentidos da transferência de estilo:

$$S^D_{tot} = \frac{1}{2}(S^D_{A\rightarrow B} + S^D_{B\rightarrow A}) \tag{3.5}$$

3.2.4.2 Ruído de Entrada e Modificações do Discriminador

Para fazer com que o gerador e o discriminador aprendam mais e melhores características dos gêneros, adicionou-se um ruído Gaussiano nas entradas reais e falsas do discriminador. Isso afeta os valores das funções de perda, e consequentemente será propagado também para os geradores. Com isso, foram criados diferentes modelos, cada um utilizando uma das seguintes intensidades de ruído: 0; 0,01; 0,1; 1; 3 e 5.

Além disso, também foram criadas modificações na estrutura da rede criando 3 novas redes. A primeira delas, modelo *base*, não conta com os discriminadores extras $D_{A,m}$ e $D_{B,m}$. O modelo *partial* apresenta exatamente a estrutura explicada em 3.2.1, contando com $D_{A,m}$ e $D_{B,m}$, tal que $M = A \cup B$. Já o modelo *full*, também conta com a estrutura completa, mas com $M = A \cup B \cup C$, sendo C o gênero que não está envolvido no modelo em questão.

σ_D	0	0.01	0.1	**1**	3	5
L_c	0.37	0.98	0.20	**0.00**	0.29	0.87
L_G	1.20	1.87	1.00	**0.52**	0.80	1.56
L_D	0.36	0.27	0.41	**0.49**	0.50	0.44

Tabela 3.5: Valores de perda cíclica (L_c), do gerador (L_G) e do discriminador (L_D) para o modelo *base*. Tabela adaptada de Brunner et al. (2018)

Analisando brevemente a Tabela 3.5, percebe-se que para M_{base}, o valor de perda do ciclo converge para 0 e L_G e L_D permanecem balanceados apenas com o ruído $\sigma_D = 1$. Segundo Brunner et al. (2018), isso indica que com este valor de ruído o modelo convergiu para um estado ótimo.

Por fim, mesclando todas as possibilidades de modificação de modelo e ruído citadas, ao final tem-se 54 diferentes modelos criados.

3.2.4.3 Análise dos Resultados da Transferência de Estilo

Considerando todos os 54 modelos citados acima, foram analisados os mais relevantes deles levando em consideração a combinação de discriminadores e nível de ruído.

Verificando o S^D_{tot} na Tabela 3.6b, é notável que o M_{full} com ruído = 1, apresenta um resultado melhor que os outros modelos, mas o mesmo não se reprete nas Tabelas 3.6a e 3.6c. Porém, o fato de $M_{partial}$ ser melhor do que o M_{base} está presente nas três tabelas. Além disso, a análise da estutura do MIDI de resultado mostra que $M_{partial}$ produz saídas musicalmente melhores do que M_{base}, fato que corrobora ainda mais a utilização de dois discriminadores extras para manter características relevantes da amostra original.

	M_{base} $\sigma_D = 1$	$M_{partial}$ $\sigma_D = 0$	M_{full} $\sigma_D = 0.01$
A	88.09%	88.09%	88.09%
A→B	20.62%	9.87%	20.00%
A→B→A	88.18%	87.73%	85.16%
B	92.53%	92.53%	92.53%
B→A	20.71%	25.87%	20.09%
B→A→B	92.53%	89.51%	90.49%
S^D_{tot}	69.7%	**71.6%**	69.0%

a Transferência *Jazz* e *Classic*. A: *JAZZ*, B: *CLASSIC*

	M_{base} $\sigma_D = 0.1$	$M_{partial}$ $\sigma_D = 1$	M_{full} $\sigma_D = 1$
A	86.91%	86.91%	86.91%
A→B	45.12%	38.57%	26.61%
A→B→A	83.26%	87.39%	87.18%
B	80.04%	80.04%	80.04%
B→A	49.14%	23.13%	24.30%
B→A→B	72.48%	79.45%	80.15%
S^D_{tot}	33.6%	52.6%	**58.1%**

b Transferência *Classic* e *Pop*. A: *CLASSIC*, B: *POP*

	M_{base} $\sigma_D = 0.01$	$M_{partial}$ $\sigma_D = 0.01$	M_{full} $\sigma_D = 0$
A	60.53%	60.53%	60.53%
A→B	21.60%	14.84%	23.64%
A→B→A	57.51%	58.84%	60.53%
B	73.60%	73.60%	73.60%
B→A	40.62%	43.11%	49.24%
B→A→B	74.40%	72.09%	73.06%
S^D_{tot}	35.4%	**37.3%**	30.5%

c Transferência *Jazz* e *Pop*. A: *JAZZ*, B: *POP*

Tabela 3.6: Resultado da transferência de estilo para combinações de modelo e ruído.

Por fim, os autores ressaltam que em termos gerais o modelo que melhor produz resultados sonoramente bons é o M_{full}, mesmo sendo uma análise subjetiva. Além disso, destacam que a utilização da CycleGAN para a transferência de estilo musical é promissora, mas ainda demanda aprimoramento para produzir melhores resultados.

3.3 CONSIDERAÇÕES FINAIS

Nesta seção, foram apresentadas detalhadamente duas técnicas, Groove2Groove e Symbolic Music Style Transfer, que fazem parte do estado da arte da transferência de estilo musical utilizando representação simbólica de músicas. Portanto, a partir desta base teórica é

possível prosseguir na avaliação dos dois projetos, bem como utilizar métricas de comparação entre seus resultados, sendo este o objeto de interesse deste trabalho.

4 METODOLOGIA

Tendo como objetivo final a comparação de diferentes técnicas, optou-se, dentre outros avaliados, pela escolha dos dois projetos presentes na seção 3, Estado da Arte. Além disso, duas métricas foram desenvolvidas para tornar os artefatos produzidos pelos projetos avaliáveis e comparáveis. Também, um conjunto de dados apropriado para a comparação foi escolhido, além do desenvolvimento de um classificador usando técnicas de Aprendizado de Máquina para permitir a utilização de uma das métricas.

4.1 ESCOLHA DOS PROJETOS

O presente trabalho foca em comparar de forma justa os projetos Groove2Groove (apresentado na seção 3.1) e Symbolic Music Style Transfer (apresentado na seção 3.2). Outros projetos como MuseGAN e MelGAN, Pasini (2019), foram avaliados mas não incluídos neste trabalho. No caso deste último, a natureza dos artefatos produzidos se torna um empecilho: arquivos WAV, e não arquivos MIDI. A utilização de arquivos de áudio excluiria a possibilidade de métricas baseadas em arquivos MIDI, uma vez que a relação entre representações simbólicas e áudio não é uma bijeção simples tornando difícil o processo de conversão do arquivo WAV em MIDI.

Ambos os projetos foram desenvolvidos em um espaço de tempo próximo e utilizam técnicas bastante diferentes, enriquecendo as possibilidades de análise. Além disso, os projetos também tiveram seus códigos-fonte disponibilizado pelos autores, permitindo que o uso de ambos esteja equiparado com a implementação original e com as métricas de referência de cada.

Uma distinção interessante entre os projetos está nas entradas e saídas: como pode-se observar nas Figuras 4.1 e 4.2, o projeto Groove2Groove recebe um arquivo de conteúdo e um arquivo de estilo, enquanto o projeto Symbolic Music Style Transfer recebe apenas o arquivo de conteúdo, sendo a definição de estilo de entrada e saída feita com base no modelo utilizado.

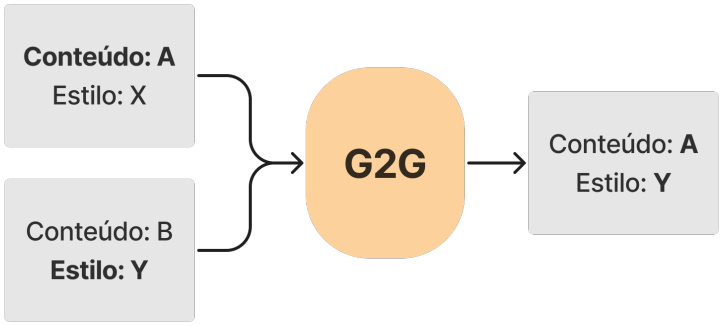


Figura 4.1: Fluxo de entrada e saída do Groove2Groove.

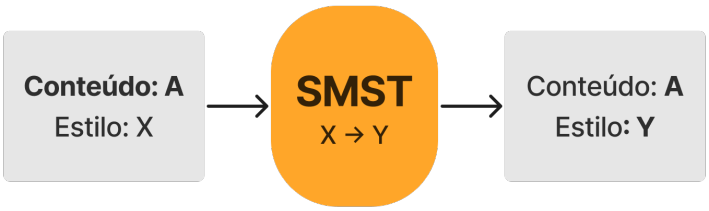


Figura 4.2: Fluxo de entrada e saída do Symbolic Music Style Transfer.

Outro fator que influencia a comparação está no pré-processamento de arquivos. O projeto Symbolic Music Style Transfer realiza um intenso processamento, detalhado em 3.2.2. O processo acaba removendo características bastante relevantes do arquivo, alterando a música percebida completamente. Por outro lado, o projeto Groove2Groove trabalha com os MIDIs sem alterações. A percepção dos efeitos do pré-processamento é essencial para o entendimento dos resultados das métricas, além de ser um fator chave para decisões de projeto.

4.2 CONJUNTO DE DADOS

Para uma comparação justa entre os projetos, fez-se necessário a escolha de um conjunto de dados apropriado. Percebeu-se também que existe uma baixa quantidade de acervos de arquivos MIDI previamente classificados por estilo - sendo o conjunto de dados mais utilizado em outras pesquisas o Bodhidharma, criado por Cory (McKay, 2004).

O Bodhidharma possui um total de 950 arquivos MIDI, pertencendo a 38 estilos diferentes (sendo 25 músicas por estilo). O propósito dos autores ao realizar a curadoria foi o de classificação automática de estilo, e não de transferência de estilo (McKay, 2004).

Outro conjunto de dados possível é o do projeto Symbolic Music Style Transfer, que possui um total de 4566 arquivos MIDI, divididos em três estilos: Clássico, Pop e Jazz (Brunner et al., 2018). A divisão é desbalanceada entre os estilos, sendo, respectivamente, 2637, 1253 e 676 arquivos por estilo.

Uma outra alternativa percebida é utilizada pelo projeto de Teng et al. (2017), apelidado de *Composing.ai*. O conjunto possui 77153 arquivos, após a remoção de duplicatas. Entretanto, não está separado por estilo, tornado o uso impossível sem um grande trabalho de curadoria de estilos.

Nome do conjunto	*# de arquivos MIDI*	*Separado por estilo?*	*# de estilos*	*Menor # de arquivos por estilo*
Bodhidharma	950	Sim	38	25
Symbolic Music Style Transfer	4566	Sim	3	676
Composing.ai	77153	Não	-	-

Tabela 4.1: Comparação entre conjuntos de dados

Após testes usando o Bodhidharma, foi observado que o baixo número de elementos por estilo impediu que o classificador, descrito em Classificador 4.3.1, aprendesse as nuances entre os estilos. Além disso, o uso do Bodhidharma para treinamento dos modelos do Symbolic

Music Style Transfer também não resultou em modelos robustos, uma vez que a amostra era pequena. Portanto, avaliando todos os conjuntos de dados, optou-se por utilizar apenas o do projeto Symbolic Music Style Transfer.

O uso do conjunto de dados do Symbolic Music Style Transfer provou-se bastante prático. Por possuir apenas três estilos, foram treinados três modelos, sendo um para cada par de estilos. Além disso, utilizou-se os mesmos parâmetros que na implementação original descrita por Brunner et al. (2018), tornando justa a comparação realizada. O tamanho do conjunto também permitiu que, na realização dos testes, houvesse grande diversidade de músicas e peculiaridades, amortizando diferenças provocadas por diferentes músicas.

É importante a percepção de que o projeto Groove2Groove não pode ter os modelos treinados pelo conjunto de dados do Symbolic Music Style Transfer, uma vez que exige um conjunto de dados para aprendizado supervisionado (ou seja, o resultado ideal da transferência de estilo existindo previamente). Entretanto, o projeto ainda pode ser avaliado utilizando-se de qualquer outro conjunto, uma vez que a proposta é extrair o estilo de uma das entradas para aplicar em outra.

4.3 TÉCNICAS DE AVALIAÇÃO DA TRANSFERÊNCIA DE ESTILO

Após a aplicação dos métodos de transferência de estilo, é necessário mensurar a qualidade dos resultados e se a transferência ocorreu com sucesso. A seguir serão explicitadas as técnicas de avaliação propostas e utilizadas no presente trabalho, também ilustradas em 4.3.

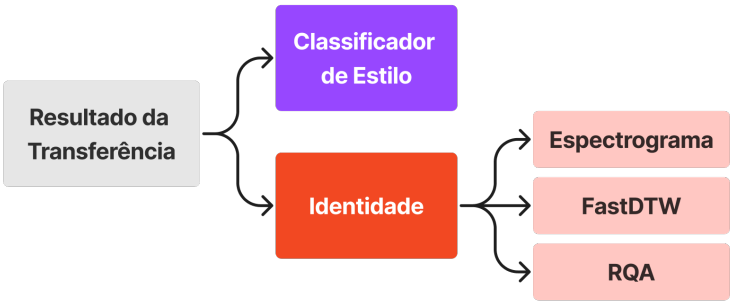


Figura 4.3: Representação das métricas de avaliação.

4.3.1 Classificador de Estilo

No intuito de empregar um agente externo aos projetos para dar parecer sobre os resultados, foi proposta a criação de um classificador de estilo para arquivos MIDI. Seu funcionamento se dá a partir de um arquivo de entrada, resultando na porcentagem de confiança para cada classe da classificação e também em uma predição final, como ilustrado na Figura 4.4. Diversas técnicas e estruturas foram testadas, mas antes disso é necessário definir o conjunto de dados de treinamento e extrair as características de tais dados.

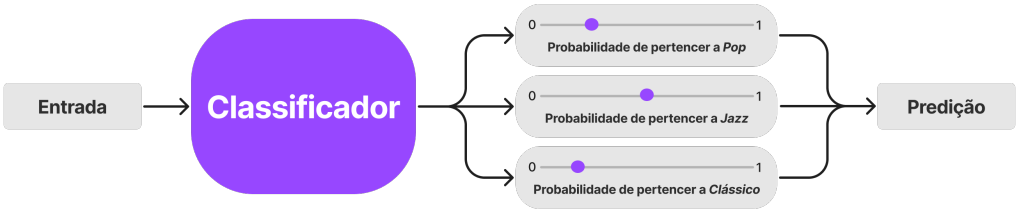


Figura 4.4: Funcionamento do Classificador de Estilo proposto.

4.3.1.1 Dados de Treinamento

Como citado anteriormente, optou-se por utilizar no projeto o conjunto de dados D de Symbolic Music Style Transfer, que possui 4359 arquivos MIDI e 3 diferentes classes: *Classic*, *Jazz* e *Pop*. O conjunto foi dividido em 80% para treinamento, 5% para validação e 15% para teste. Ressalta-se que essa base não possui uma quantidade equilibrada de classes, de forma que a classe *Classic* corresponde a aproximadamente metade dos dados totais. Dessa forma, é necessário realizar um processo de balanceamento que será aprofundado em 4.3.1.4.

Os dois projetos escolhidos, Groove2Groove e SMST, possuem maneiras diferentes de tratar os dados internamente e consequentemente geram saídas com características distintas. O Groove2Groove proporciona uma saída em MIDI que possui todas as linhas de MIDI da entrada, ou seja, mantém os instrumentos durante todo o processo. Por outro lado, o SMST, durante o pré-processamento explicado em 3.2.2, combina todas as linhas de MIDI do arquivo para criar uma única e como efeito tem-se a saída também com uma única linha de MIDI. Sendo assim, considerar para o classificador apenas os dados originais ou então só os dados pós-processados torna a classificação injusta. Para resolver este problema foram criadas 3 novas divisões do conjunto de dados inicial:

- D_O : Possui exatamente os mesmos dados do conjunto original S; Na teoria, favorece o Groove2Groove pelo estilo da saída.
- D_P : Possui os dados de D mas com a aplicação do pré-processamento do SMST. Na teoria, favorece o SMST pelo estilo da saída.
- D_M : Possui a união de D_O e D_P, ou seja, dados originais e também pré-processados. Na teoria, oferece uma comparação mais justa.

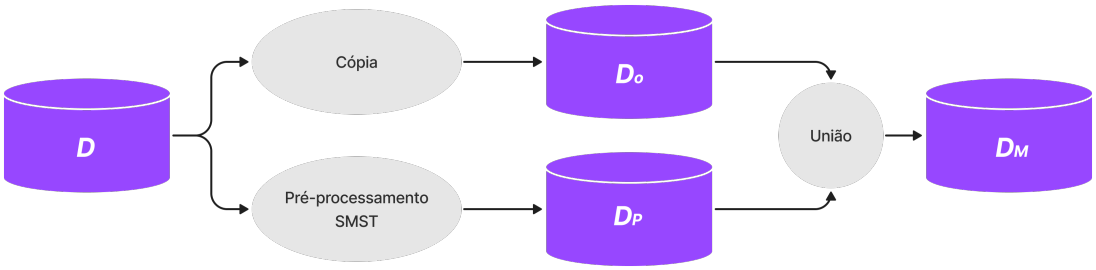


Figura 4.5: Representação dos conjuntos de dados utilizados no Classificador de Estilo

4.3.1.2 *Extração de Características*

Em essência, o treinamento e utilização de um classificador exigem a obtenção de características que representam os dados que serão classificados. Portanto, tendo a base de dados inteira em arquivos MIDI, é necessário extrair características dos mesmos. Para isso, optou-se pela utilização de duas diferentes ferramentas: O *jMIR* e o *PrettyMIDI*.

O jMIR é um *software* de código aberto que, acompanhando a definição de Mckay e Fujinaga (2010), disponibiliza diversas ferramentas que auxiliam em tarefas que envolvam os âmbitos de música e computação. Algumas dessas ferramentas são: jSymbolic, jAudio, jLyrics, dentre outras. Este projeto interessa-se pela utilização do jSymbolic 2.2, uma ferramenta de extração de características e dados estatísticos de arquivos simbólicos de música, tais como MIDI, definida em McKay et al. (2018).

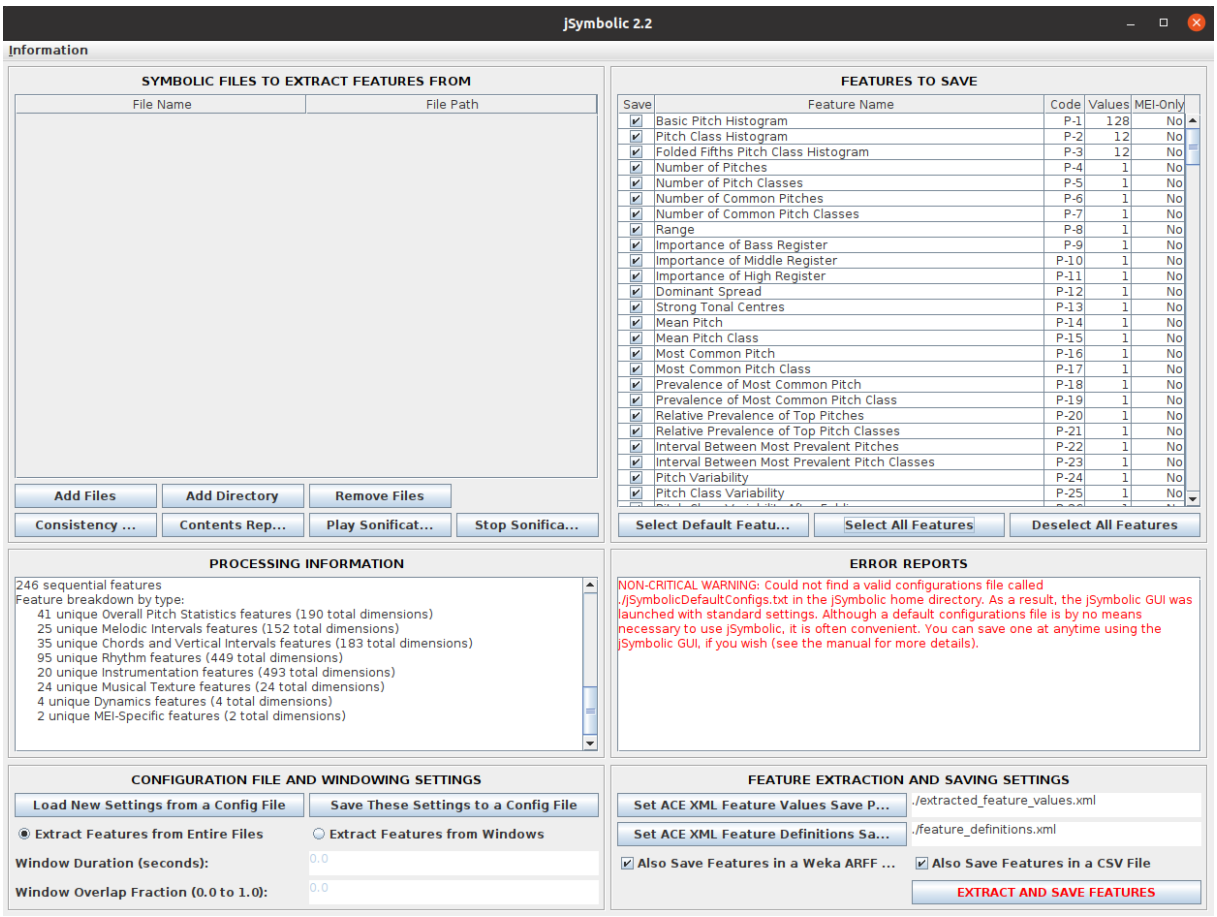


Figura 4.6: Interface de usuário do jSymbolic 2.2

O jSymbolic 2.2 oferece 246 características[1] possíveis de serem extraídas de um MIDI, desenvolvidas com intensa pesquisa na área musical. Segundo os autores, tais características podem ser divididas, no mais alto nível, em 7 grupos distintos:

- **Estatísticas de *Pitch*:** Quão comuns são as diversas notas relativamente umas às outras? Qual é sua amplitude? Quanta variedade de notas existe?
- **Melodias e Intervalos Horizontais:** Que tipos de intervalos melódicos estão presentes? Quanta variação melódica existe? Que tipos de frases musicais são usadas e com que frequência são repetidas?

[1] A explicação de todas as características é disponibilizada por McKay (2018)

- **Acordes e Intervalos Verticais:** Que intervalos verticais estão presentes? Que tipos de acordes eles representam?
- **Ritmo:** As características são calculadas com base nos intervalos de tempo entre as notas e as durações de notas individuais. Que andamento e quais padrões rítmicos estão presentes?
- **Instrumentação:** Quais instrumentos estão presentes e quais são enfatizados em relação aos outros?
- **Textura:** Quantas vozes independentes há e como elas interagem (por exemplo, polifônicas ou homofônicas)?
- **Dinâmica:** Quão intensas são as notas e quais tipos de variações de dinâmica ocorrem?

Para a utilização nas análises deste projeto, todas as características disponíveis no jSymbolic 2.2 foram utilizadas, com exceção dos múltiplos histogramas existentes. Os histogramas apresentam uma distribuição e não um valor único de característica, por isso foram desconsiderados, resultando em 232 características. Portanto, após ser processado pela ferramenta, cada arquivo MIDI passa a ser representado por um vetor de 232 posições. Para facilitar o teste e treinamento das diferentes estruturas mostradas em 4.3.1.3, foram extraídas previamente todas as características dos conjuntos D_O, D_P e D_M, gerando um arquivo para cada um, sendo estes reutilizados durante o processo, sem a necessidade de uma nova extração.

O PrettyMIDI é uma biblioteca de Python usada para criar, manipular, sintetizar e extrair informações de arquivos MIDI. Ela oferece funções para coletar dados dos arquivos, tais como notas utilizadas, duração e intensidade das notas, divisão rítmica e temporal do arquivo e algumas informações de variações musicais.

Como é notável, a qualidade e a quantidade de características são bem menores em comparação ao jMIR. Realizando alguns testes preliminares com o classificador SVM, notou-se que pela pequena quantidade de dados o seu desempenho e acurácia da classificação são bem menores do que o mesmo classificador treinado com os dados do jMIR.

Portanto, decidiu-se descartar o uso desta biblioteca e então aplicar apenas o jMIR para todos os testes e resultados obtidos a partir deste ponto.

4.3.1.3 Estrutura do Classificador

Com a representação das entradas por meio de vetores de características, o próximo passo é utilizar tais dados para construir classificadores e então estimar novas entradas. Aqui foi proposta a utilização e comparação de diferentes técnicas estatísticas e de aprendizado de máquina, possibilitando a aplicação de múltiplas abordagens sobre o mesmo conjunto de dados, a título de definir qual técnica melhor se adapta ao objetivo. Embora diversos procedimentos tenham sido testados, aqui serão mostrados os mais relevantes junto com uma breve explicação de seus parâmetros e implementação em Python 3.

- ***Support Vector Classification* (SVC):** utilização da técnica 2.4.2.1 implementada com a biblioteca *sklearn*.

```
1  from sklearn.svm import SVC
2  classifier = SVC(random_state=42, probability=True)
```

O parâmetro *random state* permite a padronização do estado aleatório do classificador, fazendo com que todas as iterações de treinamento sejam uniformes. Além disso, *propability* permite que os níveis de confiança por classe também sejam disponibilizadas como saída de uma predição.

- ***Random Forest* (RF):** utilização da técnica 2.4.2.2 implementada com a biblioteca *sklearn*.

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=50,
max_features="auto", random_state=44)
```

O *random state* funciona de maneira análoga ao SVC. Além disso, *n_estimators* define o número de "árvores" presentes na floresta, neste caso 50. Por fim, *max_features* determina a quantidade de características que devem ser consideradas antes da divisão, sendo que "auto" representa: max_features = sqrt(n_features).

- ***Logistic Regression* (LR):** utilização da técnica 2.4.2.3 implementada com a biblioteca *sklearn*.

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
```

Utiliza todos os parâmetros padrão da implementação da biblioteca.

- ***XGBoost Classifier* (XGB):** utilização da técnica 2.4.2.4 implementada com a biblioteca *xgboost*.

```
import xgboost as xgb
classifier = xgb.XGBClassifier(objective="multi:softprob",
random_state=42)
```

O *random state* funciona de maneira análoga ao SVC. Além disso, o *objective* define qual é a tarefa e o objetivo do treinamento.

- ***LightGBM Classifier* (LGBM):** utilização da técnica 2.4.2.5 implementada com a biblioteca *lightgbm*.

```
import lightgbm as lgb
classifier = lgb.LGBMClassifier()
```

Utiliza todos os parâmetros padrão da implementação da biblioteca.

Como forma de melhorar o resultado do treinamento, para todos os classificadores citados acima, exceto o *Random Forest*, realiza-se um pré-processamento em formato de *pipeline*, adaptado de Pandala (2019), para organização dos dados.

Listing 4.1: Some Java code

```
numeric_transformer = Pipeline(steps=[("imputer",
SimpleImputer(strategy="mean")), ("scaler", StandardScaler())])
numeric_features = X_train.select_dtypes(include=[np.number]).columns
preprocessor = ColumnTransformer(transformers=[("numeric",
numeric_transformer, numeric_features)])
pipe = Pipeline(steps=[("preprocessor", preprocessor),("classifier",
lgb.LGBMClassifier()),])
```

Este *pipeline*, representado em 4.1, tem o intuito de tratar, limpar e normalizar o conjunto de dados antes de ser utilizado no treinamento. O primeiro passo é criar um transformador utilizando a função *SimpleImputer(strategy="mean")* e *StandardScaler()*.

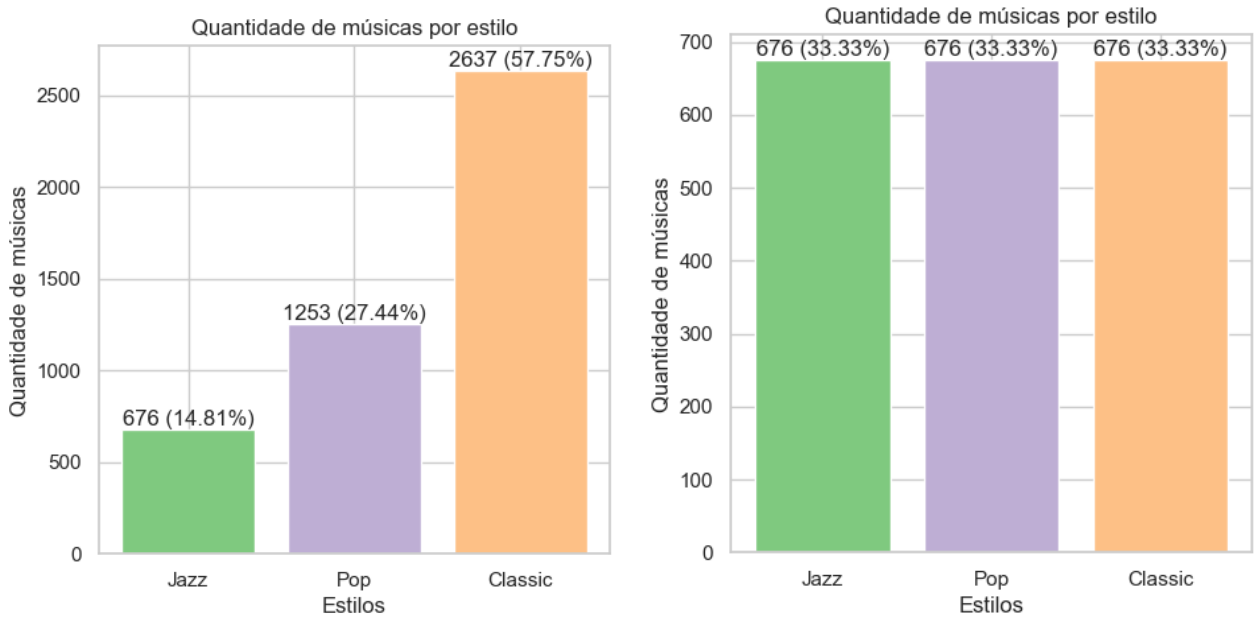
A *SimpleImputer(strategy="mean")* é utilizada para preencher valores indefinidos no conjunto de dados de acordo com a estratégia definida, neste caso sendo a média dos outros valores. Já o *StandardScaler()* realiza um tipo de normalização dos dados e é muito utilizado quando as características tem escalas diferentes e então deseja-se padronizá-las para que todas tenham um impacto similar no modelo que está sendo treinado.

Em sequência, todos os dados numéricos são coletados na variável *numeric_features* e então a função *ColumnTransformer()* é utilizada para aplicar as transformações citadas acima em todos os dados e então dá-se origem ao pré-processador. Por fim, define-se o último fluxo, aplicando o pré-processador e em seguida passando os dados para a estrutura do classificador definida anteriormente.

4.3.1.4 Treinamento

Tendo coletado toda a base de dados, extraído as características e então definido a estrutura do classificador, agora é necessário treiná-lo para então verificar os resultados obtidos.

Seguindo a explicação feita em 4.3.1.1, os conjuntos de dados estabelecidos não possuem um número uniforme de elementos por classe. Dessa forma, após a divisão dos conjuntos em dados de treinamento (80%), validação (5%) e teste (15%), é necessário aplicar o balanceamento.



(a) Partição de treinamento de um conjunto D_M sem balanceamento de classes

(b) Partição de treinamento de um conjunto D_M com balanceamento de classes

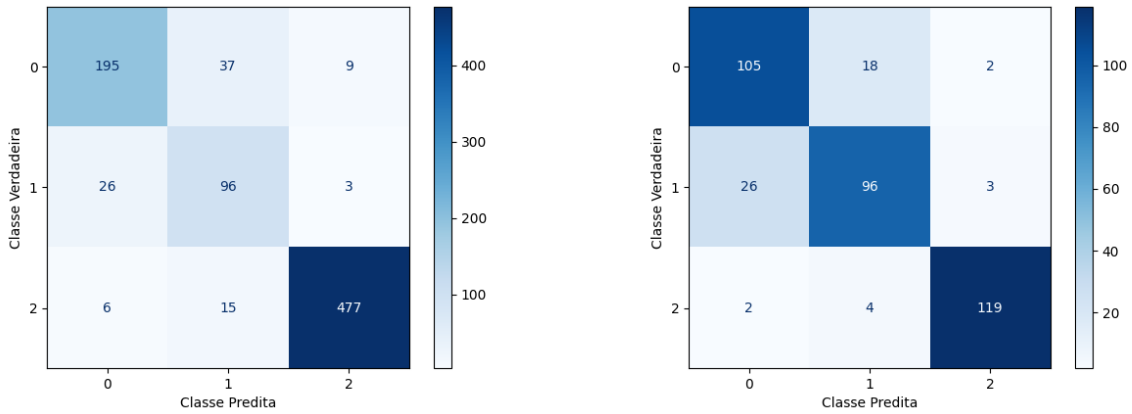
Figura 4.7: Comparação entre balanceamento e desbalanceamento de dados

Na Figura 4.7(a) está representada a partição de treinamento de um conjunto D_M. Essa partição possui 4566 dados e é notável que a classe *Classic* corresponda a mais da metade do total. Então, realizar o treinamento com essa divisão pode resultar em um *overfitting*, ou seja, fazer com que o classificador fique especializado em detectar apenas a classe *Classic*.

Para resolver isso, é aplicado o balanceamento de dados. Esse processo consiste em analisar a quantidade de instâncias em cada classe, identificar o menor valor e então limitar as

outras classes para terem o mesmo número de objetos que a de menor tamanho. Tomando como exemplo a Figura 4.7(a), vemos que a classe de menor tamanho é a *Jazz* com 676 entradas. Então, no balanceamento as outras classes, *Pop* e *Classic*, serão limitadas a 676 objetos. O resultado pode ser observado na Figura 4.7(b). Embora implique em um significativo descarte de dados, esta técnica proporciona um treinamento justo e com mais precisão.

Adicionando uma análise paralela, vale a dúvida se as partições de validação e teste também necessitam do processo de balanceamento. É perceptível que dado um classificador C eficiente em identificar a classe A mas não a classe B, caso o conjunto de testes possua mais dados de A, então o valor de acurácia será enviesado, isto é, a taxa de acerto será tendenciosamente maior, visto que C tem facilidade para acertar A. Mas é visível que a acurácia enxerga apenas o valor total de acertos. Por outro lado, olhando para a taxa de acerto individual de cada classe, é possível fazer uma análise mais profunda sobre o real impacto do balanceamento.



(a) Predição da partição de testes de um conjunto D_M sem balanceamento de classes

(b) Predição da partição de testes de um conjunto D_M com balanceamento de classes

Figura 4.8: Matrizes de confusão para dados balanceados e não balanceados

Na Figura 4.8 estão ilustradas duas matrizes de confusão das predições de uma partição de testes. As classes são identificadas pelos índices 0, 1 e 2, representando *Pop*, *Jazz* e *Classic*, respectivamente. Entende-se que cada linha e coluna representam uma classe, de forma que se a classe verdadeira for X e a predita também for X, considera-se um acerto e, por consequência, o interesse está nos valores da diagonal principal da matriz.

Analisando a Figura 4.8(a), percebe-se o total de 834 objetos, divididos em 498 *Classic*, 125 *Jazz* e 241 *Pop*. Então, considerando os valores das predições, tem-se as seguintes taxas de acerto:

- ***Classic*:** 95,78% (477/498)
- ***Jazz*:** 76,8% (96/125)
- ***Pop*:** 80,91% (195/241)

Por outro lado, a Figura 4.8(b) apresenta os mesmos dados, mas com o balanceamento aplicado a partir da classe de menor tamanho, sendo *Jazz* (125). Dessa forma, tem-se 375 objetos totais, sendo *Classic*, *Jazz* e *Pop* igualmente divididos com 125 cada. Verificando as predições, apresentam-se as taxas abaixo:

- ***Classic*:** 95,2% (119/125)

- ***Jazz*:** 76,8% (96/125)
- ***Pop*:** 84% (105/125)

Comparando as taxas de acerto com e sem balanceamento, nota-se que, embora haja diferença, permanecem com o mesmo grau de grandeza. Isso reflete que a falta de balanceamento não está alterando as taxas de acerto por classe. Um detalhe que se destaca é a redução em mais de 50% da quantidade total de objetos após o balanceamento, sendo, mais precisamente, 459 descartados. O mesmo fenômeno acontece na partição de validação. Logo, optou-se por aplicar o balanceamento apenas nos dados de treinamento, mantendo validação e teste intactos, para preservar o número de amostras e consequentemente validar com mais robustez os resultados.

Conjunto	Treino	Validação	Teste
D_O	1245	211	610
D_P	783	80	246
D_M	2028	293	864

Tabela 4.2: Tamanho dos conjuntos após o balanceamento

Na Tabela 4.2 é possível ver a quantidade de arquivos MIDI utilizados em treino, validação e teste para cada um dos agrupamentos definidos.

Por fim, a combinação dos 3 conjuntos de dados, D_O, D_P e D_M, com as 5 técnicas de classificadores, resulta em 15 diferentes modelos. Os resultados obtidos serão discutidos nas próximas seções.

4.3.2 Identidade

Como explicado, o desenvolvimento das métricas ocorreu para uma comparação justa entre os projetos. Os resultados do projeto Groove2Groove apresentam grande complexidade musical, com múltiplos instrumentos e melodias. Então, criou-se a hipótese que, em uma comparação de estilos, os artefatos produzidos pelo Groove2Groove seriam mais precisamente apontados por um classificador de MIDI, tal qual 4.3.1, do que os produzidos pelo Symbolic Music Style Transfer, por possuírem mais elementos musicais. Em contrapartida, seguindo a ideia de Brunner et al. (2018), é desejável que as técnicas de transferência de estilo musical sejam isomorfas em ambos os sentidos, isto é, ao partir de um estilo A, transferir para B e após voltar para A', idealmente, A e A' devem ser o mais próximo possível. Então, optou-se por também mensurar a identidade, ou seja, comparar o arquivo original de entrada com a transferência e com o ciclo, assim como descrito por Brunner et al. (2018). Para isso, é necessário ser capaz de estimar a semelhança entre dois arquivos MIDI, processo que está representado na Figura 4.9 e será detalhado na seção 4.3.2.

Figura 4.9: Funcionamento da Identidade proposta.

Como supracitado, o projeto Symbolic Music Style Transfer também implementou uma função de identidade, uma vez que a técnica prioriza a *consistência de ciclo*. Entretanto, a

comparação de arquivos MIDI é um tópico bastante complexo. Ao analisar o Symbolic Music Style Transfer, observa-se que o pré-processamento dos arquivos e a redução para apenas um único instrumento torna o formato de representação bastante conveniente para que a diferença entre a entrada da rede e o ciclo seja calculada como uma distância de matrizes. Entretanto, o mesmo não pode ser aplicado para arquivos MIDI que fogem desse formato. A exemplo, arquivos MIDI podem possuir internamente diversas linhas de eventos, então, pela diversidade de instrumentos e até mesmo pela ordem de tais linhas, o mapeamento e o cálculo da distância de matrizes torna-se um processo impreciso e complexo, dificultando ainda mais a busca por uma função apropriada.

Mitzenmacher e Owen (2001) abordaram o problema com um processo de *fingerprinting*, resultando em uma *hash* que representa o arquivo. Observa-se que na técnica empregada, o valor de semelhança encontrado é reduzido consideravelmente com apenas 6% das notas do arquivo original alteradas. Empiricamente, pode-se concluir que as técnicas aplicadas pelo Symbolic Music Style Transfer e Groove2Groove tendem a produzir arquivos MIDI que, quando transformados em áudio, soam parecidos, mas apresentam uma significativa quantidade de diferenças no arquivo original. Concluiu-se, portanto, que a técnica não apresentaria capacidade de apresentar valores razoáveis para a comparação dos arquivos originais e dos arquivos de ciclo.

Outras abordagens como as dos projetos de Raffel e Ellis (2015) e Goródscy et al. (2018) aplicam-se na comparação de arquivos MIDI e arquivos de áudio. Observa-se que ambos baseiam-se na aplicação do *Dynamic Time-Warping* (DTW), que tem o objetivo de alinhar duas séries temporais cuja velocidade pode variar. A aplicação de algoritmos é bastante popular em diversos outros domínios, como o alinhamento de informação de diversos sensores. O principal desafio para a aplicação do algoritmo diretamente nos arquivos MIDI está na existência de múltiplos instrumentos, uma vez que cada arquivo apresenta múltiplas sequências. Entretanto, Raffel e Ellis (2015) e Goródscy et al. (2018) aplicam o DTW na comparação de dois arquivos de áudio, realizando a renderização do arquivo MIDI para um arquivo de áudio.

Finalmente, Xia et al. (2013) apresentam técnicas com bastante potencial para a construção de uma ferramenta buscadora de arquivos MIDI. Porém, a técnica possui uma implementação complexa mas não disponibilizada, o que tornou seu uso impraticável para o escopo deste trabalho.

Na ausência de uma técnica padronizada e consistente para a comparação de arquivos MIDI, a comparação entre arquivos de áudio e espectrogramas derivados dos arquivos MIDI tornou-se necessária. Neste contexto, três métricas de identidade foram definidas, sendo todas descritas na sequência:

- O *score Recurrence Quantification Analysis* (RQA) entre os dois arquivos de áudio, apresentada por Serrà et al. (2009)
- O *score* do menor caminho aproximado do *Fast Dynamic Time Warping* (Salvador e Chan, 2004)
- O *score* da semelhança entre os espectrogramas de ambas as produções.

4.3.2.1 RQA

Bastante aprofundado em Webber e Marwan (2015), o *Recurrence Quantification Analysis* (RQA) é um método matemático usado para usado para estudar relações e dinâmicas em séries temporais. Ele é formado a partir da concepção de recorrência. Dessa forma, dada uma série temporal de dados, um estado é considerado recorrente se for semelhante a um estado

anterior. Com esta técnica, o RQA permite a obtenção de várias métricas, como quantidade de recorrências, duração média das recorrências, dentre outras. Estas medidas podem ser utilizadas para comparar séries temporais de dados, tais como áudios.

A implementação do RQA utilizada neste trabalho foi a da biblioteca `Librosa`, desenvolvida por McFee et al. (2015). Como esta implementação foi criada para a detecção de *covers* de músicas (ou seja, interpretações da mesma música por outros artistas), considerou-se apropriada para avaliar a semelhança entre as músicas.

Para o RQA, utilizou-se a métrica de conectividade com cálculo da distância euclidiana, resultando em uma matriz de alinhamentos e um *score* de similaridade como saída.

Vale ressaltar que esta técnica utiliza áudio para o cálculo do alinhamento, sendo assim, é necessário converter os arquivos MIDI para áudio.

4.3.2.2 FastDTW

O algoritmo *Dynamic Time Warping* (DTW) é usado para mensurar a similaridade entre duas sequências temporais. Segundo Müller (2007), foi proposto originalmente para comparar padrões de fala em ferramentas automáticas de reconhecimento de fala.

Seu funcionamento se dá pela comparação de duas sequências de valores, encontrando o alinhamento ótimo entre elas. Para isso, calcula-se a distância entre todos os pares de pontos existentes nas sequências e então encontra o caminho mais curto entre cada par, o qual corresponde ao alinhamento.

Dado o número de operações e comparações, o DTW possui complexidade de $O(n^2)$, crescendo quadraticamente. Como pretendia-se realizar uma grande bateria de testes, além do fato de que as músicas utilizadas são "completas" e portanto tendem a ter pelo menos alguns minutos de execução, optou-se pelo uso do *Fast DTW*, apresentado por Salvador e Chan (2004). Esta é uma variação otimizada do DTW que não realiza a comparação de todos os possíveis pares de pontos, mas sim naqueles que estão em uma região próxima da sequência, chamada de janela de busca. Portanto, o resultado final é uma aproximação do valor real do DTW.

Embora exista discussão sobre uso do Fast DTW Wu e Keogh (2020), observa-se que as aproximações estavam extremamente próximas do ótimo do DTW, porém a implementação do Fast DTW pela biblioteca *fastdtw* (2015) mostrou ser mais eficiente do que a apresentada por Wu e Keogh (2020).

Para o Fast DTW, utilizou-se a distância euclidiana, resultando em um score de saída que representa a distância entre os dois arquivos.

Por fim, assim como o RQA, o FastDTW funciona com arquivos de áudio, logo, também foi necessário realizar a conversão de MIDI para áudio.

4.3.2.3 Semelhança entre imagens

Assim como mostrado em 2.3, existem diversas formas de representar uma mesma música. Sendo assim, é possível converter um arquivo MIDI em áudio e em seguida em um espectrograma, tendo como resultado final uma imagem. Partindo disso, dados dois espectrogramas, é possível mensurar a diferença entre as músicas ali representadas a partir da distância entre as imagens.

A geração dos espectrogramas baseou-se na biblioteca `Librosa` e o cálculo da diferença entre imagens é feito com a biblioteca `OpenCV`, por Bradski (2000), através da diferença absoluta entre imagens.

```
1 import cv2
2 result = cv2.absdiff(spec1, spec2)
```

O algoritmo para cálculo baseia-se na porcentagem de pixels que diferem da imagem de referência.

4.3.3 *Score*

Com as métricas acima definidas, foi criado um *score* para resumir os resultados. O conceito de um *score* tornou-se especialmente útil para sumarizar os resultados das diversas métricas, obtendo um valor mais direto de qual projeto havia performado melhor. Para melhor entendimento do uso do *score*, algumas percepções a respeito da forma das métricas são necessárias.

4.3.3.1 Considerações sobre a identidade

A identidade foi calculada com o uso de três métricas diferentes, a fim de capturar as nuances de cada formato de representação e algoritmo. Entretanto, observa-se que dependendo dos elementos comparados o valor ideal da métrica é diferente. Ao comparar um arquivo de uma música original com sua contraparte estilizada, deseja-se que ambos não sejam iguais mas também que não sejam completamente diferentes. Dessa forma, conclui-se que um valor de semelhança integral ou nulo não é ideal. Após a normalização dos valores em um intervalo de $[0, 1]$, não é trivial identificar qual o valor ideal para as funções de identidade.

Além disso, partindo de técnicas como a CycleGAN em que se produz um arquivo de "volta" (ou seja, o conteúdo estilizado para o domínio alvo e então estilizado novamente para o domínio de origem), observa-se que neste caso as métricas de identidade entre o arquivo de "origem" e o arquivo de "ciclo" idealmente terão valor integral - sendo assim iguais. Conclui-se, portanto, que o *score* base deve considerar essa nuance e os diferentes objetivos de cada transferência.

4.3.3.2 Considerações sobre o classificador de estilo

O classificador de estilos é um ferramental desenvolvido com base em técnicas estatísticas. Sendo assim, embora possa apresentar bons resultados em conjuntos de treino, validação e teste, os exemplos produzidos pela transferência de estilo podem apresentar padrões musicais e "falhas" que não estavam presentes nos conjuntos de treinamento. Quando se considera o pré-processamento aplicado pelo projeto Symbolic Music Style Transfer, as mudanças podem se tornar ainda maiores e os resultados mais ambíguos. Conclui-se que, embora o classificador apresente resultados sólidos em exemplos "normais", as produções da transferência podem introduzir certas confusões. Porém, o objetivo das técnicas de transferência de estilo é produzir músicas que sejam tecnicamente boas e também apresentem consistência musical. Assim, as dificuldades da transferência de estilo foram consideradas como um problema da técnica, e não do classificador.

Outra consideração está no formato dos resultados do classificador. Pela maneira que foi construído, o classificador tem como saída a probabilidade que a entrada corresponda a um estilo de música. Ou seja, três valores dentro do intervalo $[0, 1]$, sendo correspondentes aos estilos presentes no conjunto de dados escolhido. A soma das probabilidades é sempre 1. Assim, existe uma subjetividade no acerto do classificador: caso o classificador classifique uma entrada com 40% de confiança, tal classificação deve ter o mesmo peso que um palpite certeiro de 99.9% de confiança? Optou-se por explorar ambas as possibilidades, com o uso padrão sendo das porcentagens de confiança e não da noção binária de acerto/erro.

4.3.3.3 Hipóteses de Avaliação

A partir das técnicas de avaliação, aliadas às considerações apresentadas em 4.3.3.1 e 4.3.3.2, foram definidas hipóteses sobre os resultados para oferecer uma base para análise futura:

1. O projeto Symbolic Music Style Transfer terá melhor desempenho que o Groove2Groove nas métricas de identidade

    - Considerando que o projeto Symbolic Music Style Transfer utiliza a CycleGAN como base, metodologia que utiliza como guia a consistência de ciclo, espera-se que os artefatos de ciclo produzidos pelo Symbolic Music Style Transfer sejam mais semelhantes entre si. Além disso, o efeito do pré-processamento torna a música consideravelmente mais simples, reduzindo o tamanho do conjunto de frequências, notas e produções possíveis.

2. O projeto Groove2Groove terá melhor desempenho que o Symbolic Music Style Transfer nas métricas de classificação de estilo

    - A aplicação do pré-processamento do SMST não garante que todos os elementos de entrada estarão no conjunto pré-processado, uma vez que somente arquivos MIDI com certas características são aceitos. Sendo assim, o conjunto de músicas pré-processadas apresentado ao classificador é menor do que o conjunto original. Ou seja, o classificador terá mais informação a respeito das músicas na forma original. Além disso, um dos efeitos do pré-processamento é a simplificação de diversos aspectos da música. A simplificação excessiva torna as características do estilo da música bastante deturpadas, deixando as bordas entre os estilos, já tênues, ainda menos explícitas. Uma visualização clara deste fenômeno pode ser encontrada no Apêndice A.

3. A performance das métricas oscilará dependendo dos estilos na direção da transferência

    - Certos estilos apresentam características mais específicas do que outros, como o Clássico. Como ilustrado no Apêndice A, não existe uma diferença trivial entre *Pop* e *Jazz*. Sendo assim, transferências entre estes estilos podem performar de maneira diferente das direções, incluindo o estilo Clássico na classificação de estilo. Tal diferença pode se tornar ainda maior nas transferências do Symbolic Music Style Transfer, devido ao pré-processamento e as consequências supracitadas.

4. O projeto Groove2Groove terá resultados musicalmente melhores do que o Symbolic Music Style Transfer

    - Como já mencionado abundantemente, o pré-processamento do SMST descarta características importantes dos arquivos, tal como: a quantidade de instrumentos e linhas MIDI existentes, combinando todas em uma única. Consequentemente, o resultado também será uma única linha que representa um único instrumento, afetando significativamente o resultado sonoro. Por outro lado, o G2G mantém o arquivo de MIDI com sua estrutura original durante todo o processo, favorecendo um resultado musicalmente melhor.

5. O projeto Groove2Groove será beneficiado pelo classificador treinado com D_O, enquanto Symbolic Music Style Transfer será favorecido pelo treinado com D_P

- Os elementos presentes no conjunto D_O são muito semelhantes aos dados de entrada e saída do projeto Groove2Groove. Portanto, o G2G tende a ter melhores resultados com o classificador treinado neste mesmo conjunto de dados. A mesma justificativa é valida para o projeto Symbolic Music Style Transfer em relação ao conjunto D_P.

4.3.4 Considerações Finais

As técnicas de avaliação de transferência de estilo são fundamentais para o desenvolvimento deste trabalho. Uma grande quantidade de tempo foi empregada com o intuito de encontrar e formular técnicas justas de avaliação para poder comparar corretamente os projetos Groove2Groove e Symbolic Music Style Transfer. Com isso, definiram-se dois pilares de avaliação: o Classificador de Estilo e a Identidade, junto com suas derivativas. Por fim, as hipóteses formuladas auxiliam a identificar se os resultados obtidos são coerentes com os esperados. Todas as informações aqui apresentadas serão utilizadas e examinadas nas seções seguintes.

5 EXPERIMENTOS

5.1 ESCOLHA DO CLASSIFICADOR DE ESTILO

Como exposto em 4.3.1, a combinação das 5 técnicas de classificação com os 3 conjuntos de dados resulta em 15 diferentes classificadores. Partindo daí, é necessário compará-los para decidir pela utilização ou não de cada um deles no processo de avaliação dos projetos.

O primeiro passo é treinar os classificadores com os conjuntos D_O, D_P e D_M. A quantidade de dados utilizada está representada na Tabela 4.2. As implementações escolhidas para os modelos são otimizadas, tornando rápido o processo de treinamento, com aproximadamente 100 segundos para cada um.

No universo de aprendizado de máquina, a métrica mais comum para comparação de modelos é a acurácia. Dessa forma, o principal objeto de análise desta etapa é a acurácia dos classificadores. Além disso, também é feita a divisão entre acurácia de validação (A_c) e teste (A_t), sendo A_t a mais relevante pela quantidade de entradas existentes, vide Tabela 4.2.

Porém, a acurácia é uma medida que pode ser enviesada por alguns fatores, como por exemplo o desbalanceamento de classes na subdivisão de teste. Sendo assim, para os modelos que apresentam os melhores valores de acurácia também são analisadas as taxas de acerto por classe, como forma de garantir que o classificador realmente é efetivo e que pode ser usado com confiança na comparação dos projetos.

5.2 PREPARAÇÃO DOS PROJETOS

Antes da realização dos experimentos, é necessário realizar o treinamento e preparação dos modelos dos projetos escolhidos. Optou-se por manter os parâmetros de referência dos artigos originais, prezando a fidelidade dos resultados apresentados pelos seus autores.

Para o projeto Symbolic Music Style Transfer, Brunner et al. (2018) apresentam diversos resultados para variados hiperparâmetros, como mencionado na seção 3.2.4.3. Neste sentido, os autores descrevem que o ruído $\sigma_D = 1$ tende a levar o modelo para um resultado ótimo, minimizando o valor de perda do resultado cíclico e equilibrando a perda de gerador e discriminador. Além disso, pelos valores de comparação observados, a utilização de dois discriminadores a mais ($M_{partial}$) se mostra mais estável do que as versões M_{base} e M_{full}, que possuem o formato padrão da *CycleGAN* e a extensão das classes de dados do conjunto de treinamento M do discriminador, respectivamente.

Sendo assim, optou-se pela combinação das configurações que geram uma estrutura consistente do projeto, a partir da análise dos resultados apresentados por Brunner et al. (2018). Então, utilizou-se $\sigma_D = 1$ e $M_{partial}$.

Já no projeto Groove2Groove, segundo Cífka et al. (2020), os melhores resultados preliminares foram encontrados nos modelos que utilizam o hiperparâmetro $\tau = 0.6$. Embora os resultados apresentados sejam baseados no modelo $v01_{drums\_vel}$, que considera tanto a presença da percussão quanto da velocidade, Cífka (2020) menciona que o modelo $v01_{drums}$, modelando apenas percussão e não velocidade apresenta resultados **significativamente** melhores em termos subjetivos. Com base nos comentários dos autores, optou-se pela escolha do $v01_{drums}$ com $\tau = 0.6$.

5.3 CASOS DE TESTE

Para a avaliação das técnicas, criou-se a definição de um caso de teste, que é gerado a partir da definição de dois estilos de música: origem e alvo (por exemplo, *Pop* -> *Jazz*). Chama-se a sequência de estilos "*Pop* -> *Jazz*" de uma direção, e a sequência oposta "*Jazz* -> *Pop*" de "direção oposta".

5.3.1 Geração dos artefatos

Com base num caso de teste, dois arquivos MIDI aleatórios do conjunto de teste são escolhidos - um para cada estilo. Chama-se o arquivo pertencente ao estilo de origem de "conteúdo", e o arquivo pertence ao estilo alvo de "estilo". Com os arquivos definidos, são gerados os artefatos para cada projeto.

Para o Symbolic Music Style Transfer, realizam-se as seguintes operações:

1. Realizar o pré-processamento no arquivo de conteúdo
2. Escolher o modelo que realiza a transferência dos estilos escolhidos
3. Realizar a transferência
4. Agrupar as frações de MIDI produzidas
    - Devido ao pré-processamento, o Symbolic Music Style Transfer quebra os arquivos de entrada e saída em pequenos pedaços de mesmo tamanho
    - O Symbolic Music Style Transfer produz como saída tanto o arquivo de "conteúdo" pré-processado (chamado de "origem"), o produto da transferência (chamado de "transferência") e o arquivo de "ciclo" (ou seja, o arquivo "transferência" já processado pelo modelo que realiza a conversão na direção oposta)

Já para o Groove2Groove, realizam-se as seguintes operações:

1. Realizar a conversão com o arquivo "conteúdo" como conteúdo e "estilo" como estilo. O arquivo "conteúdo" será chamado posteriormente também de "origem".
2. A saída do processo é um novo arquivo, resultado da aplicação do arquivo "estilo" em "conteúdo". A saída é o arquivo de "transferência".
3. Realizar a conversão com o arquivo "transferência" como conteúdo e "conteúdo" como estilo.
4. A saída do processo é um novo arquivo, que, teoricamente, espera-se que seja igual ao arquivo "conteúdo". Chama-se este arquivo de "ciclo".

5.3.2 Processo de avaliação dos artefatos

Uma vez que o processo descrito na seção 5.3.1 foi realizado, pode-se avaliar os resultados com base nas métricas descritas na seção 4.3.

As métricas de **identidade** são aplicadas para os seguintes pares de artefatos:

- Symbolic Music Style Transfer

1. Arquivo “origem” do SMST (pré-processado) e o arquivo “transferência” do SMST. Vide Identidade 1 na Figura 5.1.
2. Arquivo “origem” do SMST (pré-processado) e o arquivo “ciclo” do SMST. Vide Identidade 3 na Figura 5.1.
3. Arquivo “transferência” do SMST e o arquivo “ciclo” do SMST. Vide Identidade 2 na Figura 5.1.

- Groove2Groove

1. Arquivo “origem” do G2G e o arquivo “transferência” do G2G. Vide Identidade 1 na Figura 5.2.
2. Arquivo “origem” do G2G e o arquivo “ciclo” do G2G. Vide Identidade 3 na Figura 5.2.
3. Arquivo “transferência” do G2G e o arquivo “ciclo” do G2G. Vide Identidade 2 na Figura 5.2.

Já as métricas de **classificação de estilo** são aplicadas para todos os artefatos individualmente, sendo estes:

- Symbolic Music Style Transfer

1. Arquivo “transferência” do Symbolic Music Style Transfer. Vide Classificador 1 na Figura 5.1.
2. Arquivo “ciclo” do Symbolic Music Style Transfer. Vide Classificador 2 na Figura 5.1.

- Groove2Groove

1. Arquivo “transferência” do Groove2Groove. Vide Classificador 1 na Figura 5.2.
2. Arquivo “ciclo” do Groove2Groove. Vide Classificador 2 na Figura 5.2.

5.4 TESTES REALIZADOS

Ao todo, foram realizados 2400 testes. A Tabela 5.1 apresenta a quantidade de testes gerados para cada direção.

Estilo de Origem	Estilo Alvo	Quantidade de Testes
Classic	*Jazz*	400
Classic	*Pop*	400
Jazz	*Classic*	400
Jazz	*Pop*	400
Pop	*Classic*	400
Pop	*Jazz*	400

Tabela 5.1: Casos de teste da avaliação

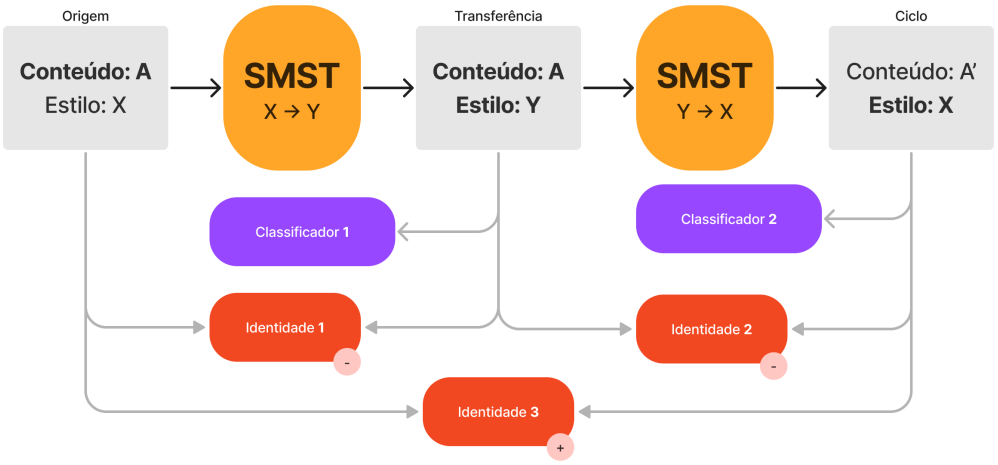


Figura 5.1: Métricas de identidade e classificação para o Symbolic Music Style Transfer.

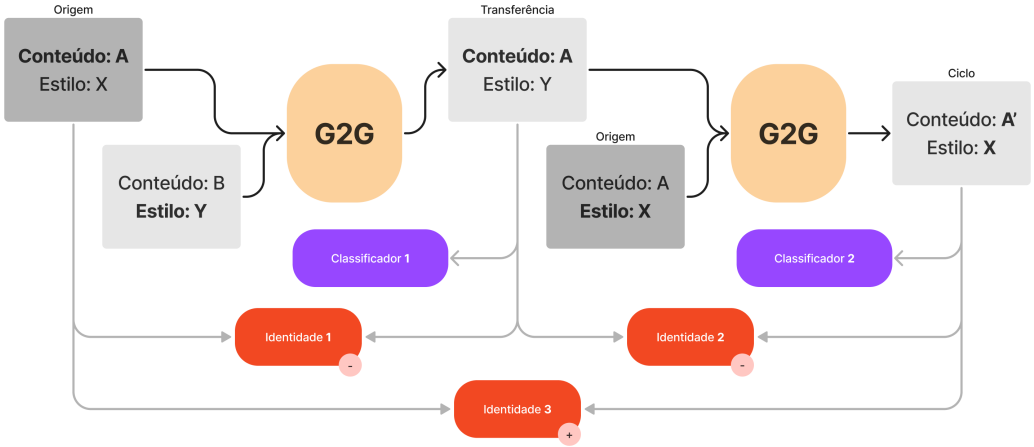


Figura 5.2: Métricas de identidade e classificação para o Groove2Groove.

Observa-se que para cada direção existem 400 testes. Cada caso de teste possui 7 arquivos MIDI, sendo que 6 serão utilizados para avaliação (o arquivo de estilo não é utilizado em nenhuma das métricas apresentadas na Seção 4.3). Além disso, para cada um dos 6 arquivos são gerados também arquivos de áudio no formato `WAV` e imagens dos espectrogramas, no formato `PNG`. Ao todo, 43.200 artefatos foram gerados.

5.5 AMBIENTE DE EXPERIMENTAÇÃO E ANÁLISE

Todo o desenvolvimento do projeto (incluindo o treinamento dos modelos do Symbolic Music Style Transfer e Groove2Groove, além da geração e avaliação dos casos de teste) ocorreu em um mesmo dispositivo, que possuía como sistema operacional o Linux Mint 20.3, com processador AMD Ryzen 5 3600 (com 6 núcleos físicos e 12 núcleos lógicos) e 32GB de RAM DDR4. Além disso, a placa gráfica (GPU) também utilizada para treino e geração dos testes foi a NVIDIA GeForce RTX 2070 SUPER. O ambiente utilizou Python 3.6.8 (Team, 2016), além de Tensorflow 12.0.

A etapa de cálculo das métricas fez uso somente da CPU, uma vez que as bibliotecas utilizadas não fazem uso da aceleração da GPU. Para tal, optou-se por realizar a paralelização em 10 processos diferentes. Não foram utilizados 12 processos (total de núcleos lógicos) pois em testes empíricos iniciais observou-se que o volume de dados era maior do que as memórias *cache* do processador suportavam, causando constantes *cache misses* e consequentemente tornando a avaliação mais lenta.

Com todos os casos de testes gerados e avaliados, uma análise das métricas foi realizada e descrita com propriedade no Capítulo 6. Para esta, as bibliotecas *Pandas* e *Matplotlib* foram utilizadas, além do ambiente *Jupyter*.

5.6 EXPERIMENTOS DO *SCORE* BASE

Baseado nas considerações das métricas, discutidas na seção 4.3.3, buscou-se criar fórmulas para o processo de *scoring*, visando auxiliar a análise de tais valores.

De início, foram experimentadas algumas combinações, tal como: considerar todos os valores das métricas em uma única fórmula. Mas estas abordagens genéricas se mostraram mais difíceis de avaliar e também mais enviesáveis, dados os tipos e comportamentos das métricas.

Examinando as métricas de identidade, por exemplo, existe uma ambiguidade a respeito do valor ótimo a partir do tipo de entrada, assim como discutido anteriormente na seção 4.3.3.1. Esta diferença propicia a criação de dois tipos de *scores*: o *score* de transferência, ou seja, voltado para as etapas em que se compara uma única etapa da mudança de domínio, indo de um estilo de origem para um alvo e o *score* de ciclo, isto é, duas etapas de transferência, partindo de uma origem, indo para o estilo alvo e em seguida voltando para o estilo da origem.

No primeiro caso, trabalha-se com a incerteza do valor ideal de similaridade e, no presente trabalho, optou-se por utilizar o valor ambíguo ideal como 0, uma vez que a descoberta do limiar foge do escopo da análise presente. Pode-se visualizar o resultado da decisão na Figura 5.3. No segundo, sabe-se que o ideal é a similaridade integral, mantendo o valor igual a métrica como pode ser visto na figura 5.4. Apesar da distinção supracitada, os *scores* da métrica de classificação de estilo operam da mesma maneira em ambos os casos.

Destaca-se que todas as equações consideram que os valores das métricas de identidade estão no intervalo $[0, 1]$, sendo 1 o valor máximo de semelhança e 0 o mínimo. Sendo assim, dada uma métrica X e duas entradas A e B, se X resultar em um valor próximo de 1 significa que as entradas A e B são semelhantes, caso contrário, não.

Portanto, pode-se estabelecer as equações abaixo para a composição dos *scores*, que utilizam as seguintes notações:

- M_C: Conjunto que possui apenas o par (“origem”, “ciclo”), cuja similaridade ideal é 1.

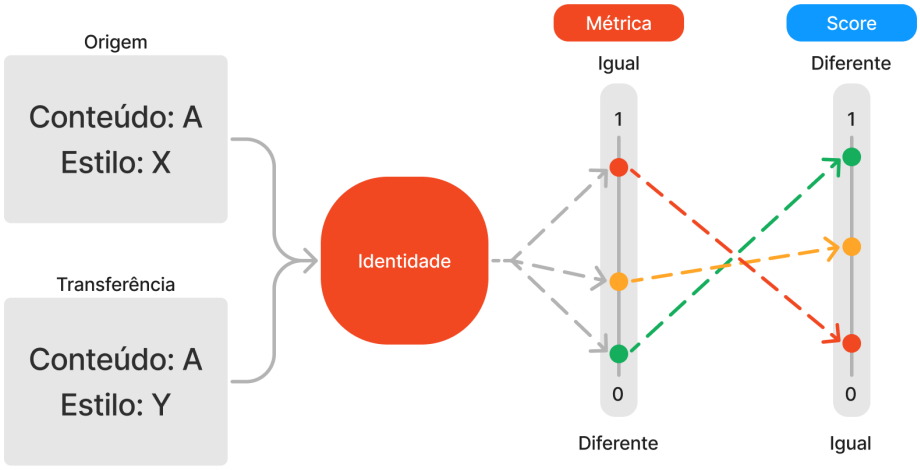


Figura 5.3: Visualização do *score* de identidade (transferência)

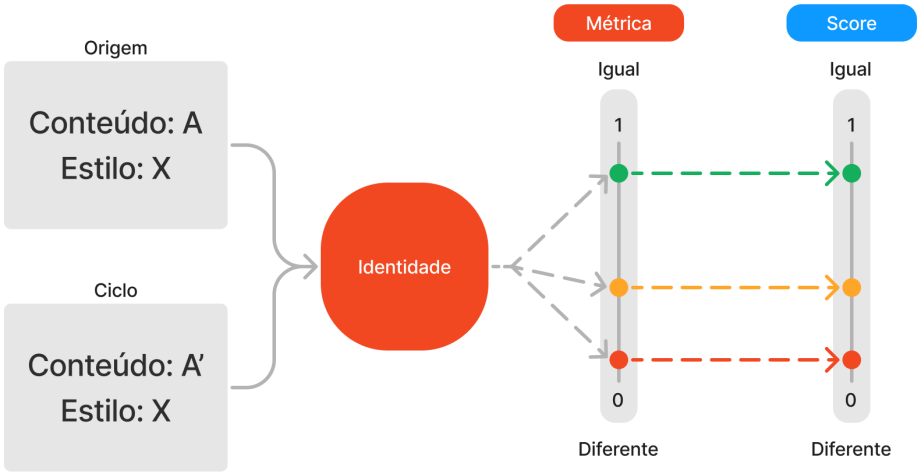


Figura 5.4: Visualização do *score* de identidade (ciclo)

- M_T: Conjunto que possui os pares ("origem", "transferência") e ("transferência", "ciclo"), cuja similaridade ideal é imprecisa - deseja-se que os arquivos não sejam nem completamente diferentes e nem iguais.

- M: o conjunto dos três arquivos MIDI produzidos pelo projeto: "origem", "transferência" e "ciclo".
- I_{cv2}: A função de similaridade entre espectrogramas (*OpenCV*), retornando 1 em caso de similaridade máxima e 0 em caso de mínima.
- I_{rqa}: A função que calcula o *score* RQA entre dois arquivos de áudio, retornando 1 em caso de similaridade máxima e 0 em caso de mínima.
- I_{dtw}: A função que calcula o *score* FastDTW entre dois arquivos de áudio, retornando 1 em caso de similaridade máxima e 0 em caso de mínima. O cálculo aplicado é a divisão da distância entre os arquivos de áudio e o comprimento do menor arquivo, obtendo a proporção da distância. Então, subtrai-se de 1 o valor, invertendo a relação e obtendo uma relação de semelhança.
- CE: A porcentagem estimada pelo classificador de estilo de que um arquivo MIDI pertence ao estilo esperado.
- S^C_{cv2}: O *score* de ciclo obtido somente pela similaridade entre espectrogramas, premiando a similaridade entre os pares.
- S^T_{cv2}: O *score* de transferência obtido somente pela similaridade entre espectrogramas, premiando a diferença entre os pares.
- S^C_{rqa}: O *score* de ciclo obtido somente pelo *score* RQA, premiando a similaridade entre os pares.
- S^T_{rqa}: O *score* de transferência obtido somente pelo *score* RQA, premiando a diferença entre os pares.
- S^C_{dtw}: O *score* de ciclo obtido somente pelo *score* DTW, premiando a similaridade entre os pares.
- S^T_{dtw}: O *score* de transferência obtido somente pelo *score* DTW, premiando a diferença entre os pares.
- S^C_{CE}: O *score* obtido somente pelo classificador de estilo no ciclo.
- S^T_{CE}: O *score* obtido somente pelo classificador de estilo na transferência.
- S^T: O *score* de transferência.
- S^C: O *score* de ciclo.
- S: O *score* base de todas as métricas.

O *score* de ciclo, portanto, será modelado pelas seguintes equações:

$$S^C_{cv2} = \frac{\sum_m^{M_C} I_{cv2}(m)}{|M_C|} \qquad (5.1) \qquad\qquad S^C_{rqa} = \frac{\sum_m^{M_C} I_{rqa}(m)}{|M_C|} \qquad (5.2)$$

$$S^C_{dtw} = \frac{\sum_m^{M_C} I_{dtw}(m)}{|M_C|} \quad (5.3) \qquad S^C_{CE} = \frac{\sum_m^{M_C} CE(m)}{|M_C|} \quad (5.4)$$

$$S^C = \frac{S^C_{cv2} + S^C_{rqa} + S^C_{dtw} + 2S^C_{CE}}{5} \quad (5.5)$$

O *score* de transferência, por outro lado, será modelado pelas seguintes equações:

$$S^T_{cv2} = 1 - \frac{\sum_m^{M_T} I_{cv2}(m)}{|M_T|} \quad (5.6) \qquad S^T_{rqa} = 1 - \frac{\sum_m^{M_T} I_{rqa}(m)}{|M_T|} \quad (5.7)$$

$$S^T_{dtw} = 1 - \frac{\sum_m^{M_T} I_{dtw}(m)}{|M_T|} \quad (5.8) \qquad S^T_{CE} = \frac{\sum_m^{M_T} CE(m)}{|M_T|} \quad (5.9)$$

$$S^T = \frac{S^T_{cv2} + S^T_{rqa} + S^T_{dtw} + 2S^T_{CE}}{5} \quad (5.10)$$

De maneira simplificada, as funções de pontuação mostradas acima são formadas pela média dos valores das técnicas para todos os resultados que estão dentro do conjunto de dados em questão. A exemplo da Equação 5.1, estão sendo analisados os valores de identidade do espectrograma para os pares (“origem”, “ciclo”) pertencentes a M_C. Neste caso, o valor da pontuação deve ser alto se os pares forem semelhantes. Para isso, soma-se todos os resultados da métrica I_{cv2} e divide-se pela quantidade de pares para se obter a média. O raciocínio é análogo para as outras equações de ciclo.

Para a identidade o raciocínio é semelhante, mas por outro lado, a pontuação deve ser alta se os pares forem suficientemente distantes entre si. Analisando mais profundadamente a Fórmula 5.6, pares distantes irão resultar em uma média baixa de valores da função I_{cv2}. Embora isso seja o valor buscado, se o *score* levar em consideração apenas a média, terá, consequentemente, um valor baixo. Sendo assim, é necessário que a média seja subtraída de 1 para formar o valor da pontuação. Dessa forma, para pares distantes entre si o valor do *score* será alto e próximo de 1.

Com ambos os *scores* de ciclo e de transferência definidos, pode-se formular um *score* base do projeto inteiro. Uma abordagem bastante direta é a média:

$$S = \frac{S^C + S^T}{2} \quad (5.11)$$

A composição dos *scores* a partir das métricas pode ser vista na Figura 5.5

Uma consideração interessante a respeito de S está na possibilidade de enviesamento a partir de S^C e S^T. Supondo que as transferências de estilo apresentem pouca similaridade, espera-se um valor muito baixo nos resultados das funções de identidade. Consequentemente, pode-se esperar que S^C tenha valor próximo de 0, enquanto S^T tenha valor próximo de 1. Ressalta-se que S^T próximo de 1 demonstra que a transferência ocorreu com sucesso, ou seja, os artefatos são pouco similares. Entretanto, como já existe pouca similaridade entre todos, este valor é apenas um reflexo do cálculo da métrica e não uma representação de sucesso. Então, desconsiderando S_{CE}, o valor médio de S será enviesado e próximo de 0,5. Tal enviesamento pode tornar a diferença entre os projetos menos visível, especialmente na análise dos scores de identidade. O problema pode ser visto visualmente na Figura 5.6. Os pontos vermelhos

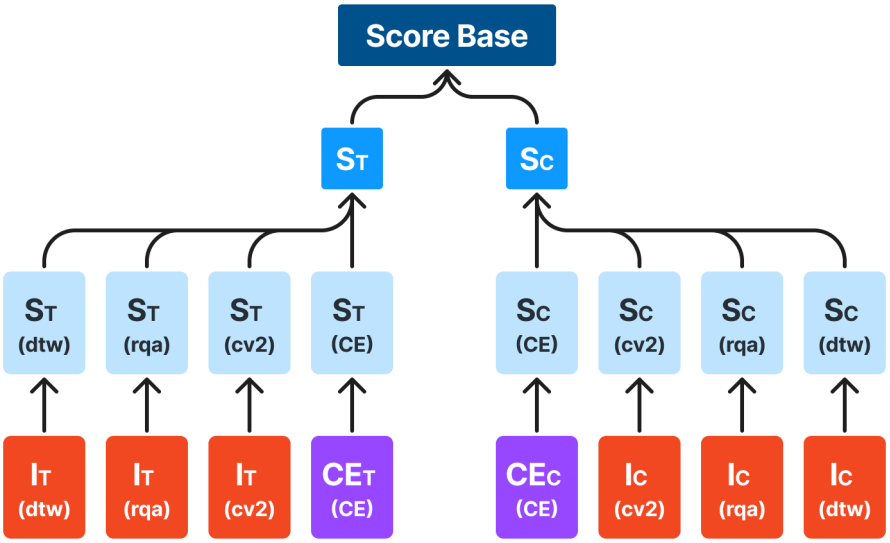


Figura 5.5: Composição do *score* base

representam as métricas de identidade, enquanto os azuis representam os *scores* (seguindo a representação da Figura 5.5. Observa-se que ao final o resultado é 0,5, com todas as entradas tendo valor 0.

Como proposta alternativa, apresenta-se S_D, o *score* desafiante. As principais diferenças de S_D estão na existência de um peso dinâmico aplicado nas funções de identidade do *score* de transferência.

Adiciona-se a seguinte notação:

- $S^{T'}_{cv2}$: O *score* desafiante obtido somente pela similaridade entre espectrogramas, premiando a diferença entre os pares.
- $S^{T'}_{rqa}$: O *score* desafiante obtido somente pelo *score* RQA, premiando a diferença entre os pares.
- $S^{T'}_{dtw}$: O *score* desafiante obtido somente pelo *score* DTW, premiando a diferença entre os pares.
- $S^{T'}$: O *score* de transferência desafiante.
- S_D: O *score* desafiante resumo de todas as métricas.

O *score* de transferência desafiante, portanto, será modelado pelas seguintes equações:

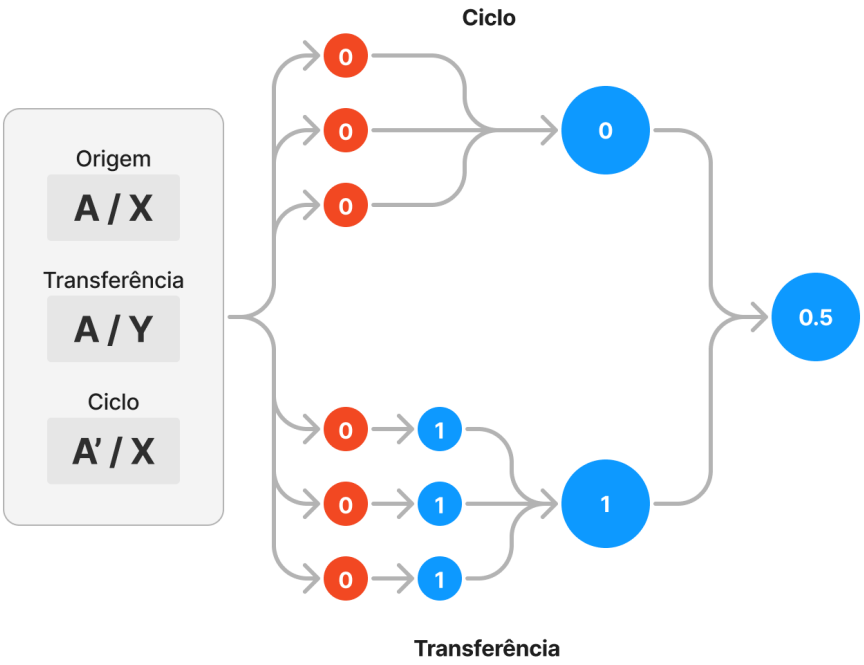


Figura 5.6: Visualização do problema do enviesamento no *score* base

$$S^{T'}_{cv2} = S^{C}_{cv2}(1 - \frac{\sum_m^{M_T} I_{cv2}(m)}{|M_T|}) \quad (5.12) \qquad S^{T'}_{rqa} = S^{C}_{rqa}(1 - \frac{\sum_m^{M_T} I_{rqa}(m)}{|M_T|}) \quad (5.13)$$

$$S^{T'}_{dtw} = S^{C}_{dtw}(1 - \frac{\sum_m^{M_T} I_{dtw}(m)}{|M_T|}) \quad (5.14) \qquad S^{T'} = \frac{S^{T'}_{cv2} + S^{T'}_{rqa} + S^{T'}_{dtw} + 2S^{T}_{CE}}{5} \quad (5.15)$$

Com as mudanças no score de transferência, pode-se modelar o *score* desafiante do projeto inteiro também como uma média:

$$S_D = \frac{S^C + S^{T'}}{2} \quad (5.16)$$

Observa-se que a principal mudança está na introdução de um peso no cálculo das funções de identidade, sendo este peso dinâmico de acordo com o valor da similaridade (do ciclo) da correspondente. Como citado anteriormente, existem casos onde todos os artefatos envolvidos no processo possuem similaridade baixa entre si, ou seja, o produto dos modelos é, de certo modo, "aleatório". Estes casos promovem uma supervalorização do valor S^T, causando o enviesamento de S. Então, o *Score* Desafiante tem como objetivo normalizar tais casos evitando

o enviesamento. Para isso, caso os pares, cuja similaridade deve ser maximizada, tenham valores baixos, o peso atribuído às funções de minimização também diminui.

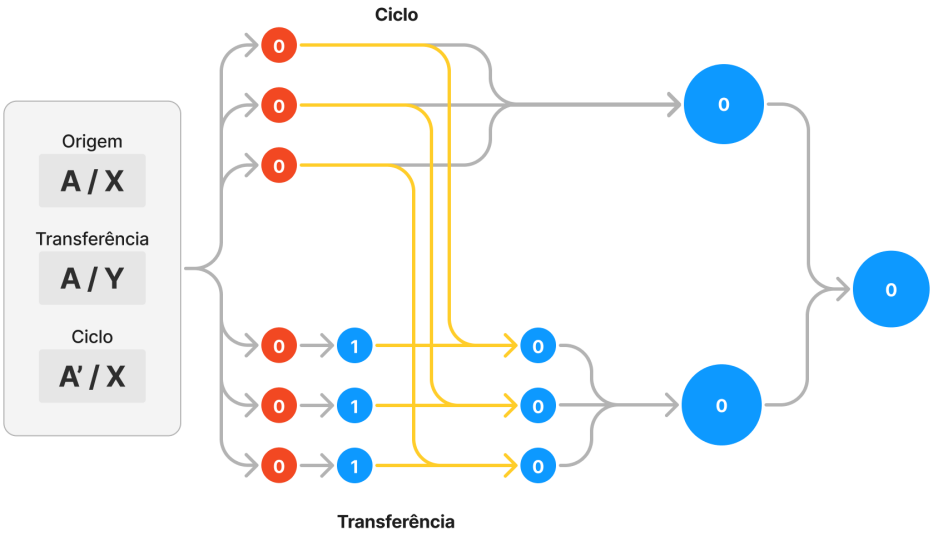


Figura 5.7: Visualização da solução proposta para o problema do enviesamento no *score* base

A figura 5.7 apresenta a solução proposta - os *scores* de ciclo, no topo, multiplicam os respectivos *scores* de transferência. Assim, busca-se a redução no enviesamento em casos favorecidos pela aleatoriedade.

5.7 OBSERVAÇÃO SUBJETIVA

As métricas definidas até este momento levam em consideração aspectos técnicos dos arquivos MIDI. Ponderando do ponto de vista musical, o fator mais importante é ter um resultado agradável aos ouvidos humanos e que demonstre que a transferência de estilo ocorreu com sucesso. Essa é uma métrica subjetiva, visto que não é possível ter uma avaliação completamente coesa a partir da observação humana. Ainda assim, realizou-se a avaliação sonora de alguns resultados gerados pelos projetos Groove2Groove e Symbolic Music Style Transfer, como forma de averiguar se a transferência de estilo de ambos é expressiva.

5.8 CONSIDERAÇÕES FINAIS

O processo de experimentação mostrou-se mais complexo do que o esperado. Embora os projetos Symbolic Music Style Transfer e Groove2Groove tiveram implementações divulgadas pelos autores, foi necessário o desenvolvimento de um ferramental para a orquestração de ambos de maneira flexível e simples. Além disso, ambos os projetos apresentavam erros e problemas na cadeia de dependências documentada, tornando-se necessária a manutenção da implementação original. Outro desafio encontrado foi o tempo de execução dos experimentos. O treino inicial

dos modelos dos projetos demorou cerca de 24h, enquanto a geração dos 2400 testes demorou cerca de 30 horas. O cálculo de todas as métricas demorou cerca de 48h, sendo a computação do valor RQA a mais lenta, dada sua complexidade computacional $O(N^2)$, sendo N o tamanho da série temporal que está sendo processada.

6 RESULTADOS

Na seção de Resultados, serão apresentadas as diversas análises realizadas e os resultados obtidos com os *scores* e métricas. Grande parte dos *scores* serão aprofundados até o nível de estilo "destino" da transferência, a fim de oferecer maior profundidade e criteriosidade. Além disso, certas hipóteses estabelecidas previamente estão diretamente associadas a análise dos resultados por estilo.

6.1 ESCOLHA DO CLASSIFICADOR DE GÊNERO

Nesta seção serão mostrados os resultados do treinamento e também discutidos os aspectos de vantagens e desvantagens para cada um dos classificadores propostos.

Para generalizar e facilitar as análises, o principal valor examinado é a acurácia total de cada um dos modelos na validação e principalmente nos testes. As Tabelas 6.1, 6.2 e 6.3 apresentam todos os valores para cada respectivo modelo de classificador e conjunto de dados.

Técnica	Validação	Teste
SVC	94%	90%
RF	96%	91%
LR	93%	82%
XGB	**98**%	**92**%
LGBM	97%	92%

Tabela 6.1: Acurácia para o conjunto D_O

Técnica	Validação	Teste
SVC	84%	81%
RF	82%	81%
LR	74%	77%
XGB	**88**%	**83**%
LGBM	86%	81%

Tabela 6.2: Acurácia para o conjunto D_P

Técnica	Validação	Teste
SVC	93%	89%
RF	92%	87%
LR	88%	86%
XGB	**94**%	**88**%
LGBM	92%	89%

Tabela 6.3: Acurácia para o conjunto D_M

Analisando as tabelas, nota-se que as técnicas XGB e LGBM apresentam resultados muito semelhantes. Isso pode ser explicado pela semelhança geral existente entre os algoritmos, visto que ambos trabalham com *gradient boosting* de árvores de decisão. Todavia, para os três conjuntos de dados, o XGB apresenta o melhor resultado para validação e teste. Esse fato é decorrente da habilidade que este algoritmo tem de lidar com dados de grandes dimensões, sendo 232 neste caso, além de possuir mecanismos de regularização e normalização que evitam o *overfitting*.

Um fato interessante é que os valores de acurácia para todas as técnicas no conjunto D_P são consideravelmente menores que em D_O e D_M. Isso mostra que o pré-processamento do projeto Symbolic Music Style Transfer modifica notavelmente os dados a ponto de confundir o classificador e reduzir a sua taxa de acerto total. Além disso, as acurácias no conjunto D_M também são afetadas e menores do que em D_O, visto que D_P está contido em D_M. Mais detalhes e a visualização desta análise podem ser encontrados no Apêndice A.

Com isso em mente, é necessário avaliar a taxa de acerto por classe dos classificadores XGB, verificando se são estáveis para cada uma das classes e consequentemente confiáveis para serem utilizados na avaliação dos projetos.

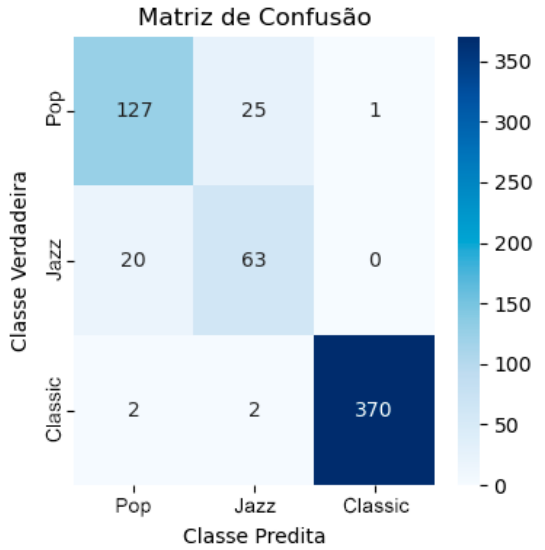


Figura 6.1: Matriz de Confusão para D_O

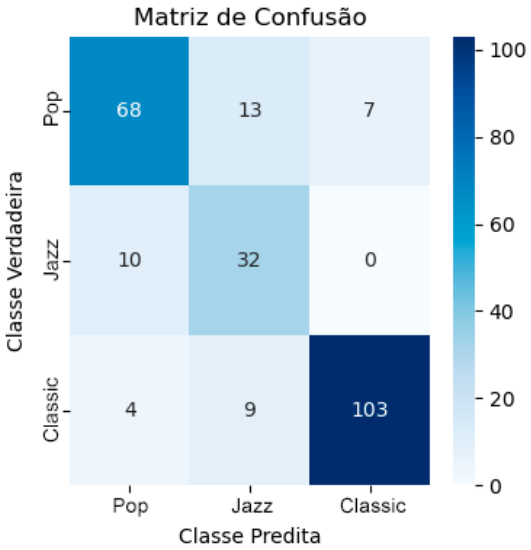


Figura 6.2: Matriz de Confusão para D_P

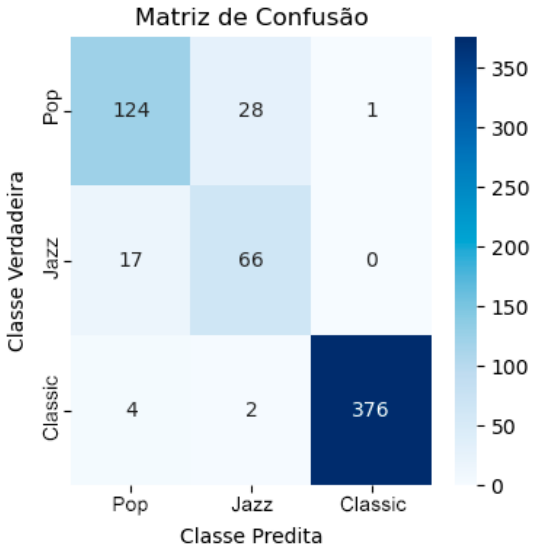


Figura 6.3: Matriz de Confusão para D_M

As Figuras 6.1, 6.2 e 6.3 representam as matrizes de confusão para as predições do classificador XGB no subconjunto de testes dos respectivos conjuntos D_O, D_P e D_M.

Analisando a taxa de acerto por classe tem-se:

- D_O: *Pop:* 83%; *Jazz:* 75,9%; *Classic:* 98,9%
- D_P: *Pop:* 77%; *Jazz:* 76,9%; *Classic:* 88,79%
- D_M: *Pop:* 81%; *Jazz:* 79,5%; *Classic:* 98,4%

Como explicado em 4.3.1.4, o subconjunto de testes não passa por nenhum tipo de balanceamento de dados, isto é, continua contendo mais dados da classe *Classic* do que das demais classes. Portanto, este fato ajuda a justificar a maior quantidade de acertos nesta classe.

Mesmo com uma considerável variação entre as taxas de acerto das diferentes classes, os níveis de sucesso do classificador XGB foram considerados suficientemente bons para o propósito da sua utilização neste trabalho.

Então, escolheu-se como técnica principal do classificador o algoritmo XGB. Com isso, para a avaliação dos projetos tem-se os 3 classificadores XGB, sendo que cada um foi treinado com um conjunto de dados diferente dentre D_O, D_P e D_M.

6.2 AVALIAÇÃO DO *SCORE* DE TRANSFERÊNCIA (S^T)

Como definido na seção 5.6, o *score* de Transferência refere-se à avaliação de um par de arquivos MIDI que pertence a domínios diferentes, mais especificamente, à comparação de um arquivo base e o resultado após uma única transferência de estilo.

6.2.1 *Score* do Classificador de Estilo (S^T_{CE})

Com base nos três classificadores (um para cada conjunto de dados, como definido na seção 6.1), todos os arquivos de "transferência" foram avaliados pelos três classificadores. Considerando que o classificador emite a probabilidade para cada estilo de música, pode-se ordenar o valor das classificações. Assim, caso o primeiro palpite seja o correto, considera-se um acerto. Analogamente, caso a segunda classificação seja a correta, considera-se um "Acerto com o 2º palpite", e o mesmo se aplica para a terceira e última classificação. A Tabela 6.4 apresenta os resultados obtidos a partir de cada classificador para o projeto Groove2Groove, enquanto a Tabela 6.5 apresenta os resultados para o projeto Symbolic Music Style Transfer.

Conjunto de dados	**Acertos**	Acertos com o 2º palpite	Acertos com o 3º palpite
D_O	**1439**	885	76
D_P	**612**	742	1046
D_M	**904**	1101	395

Tabela 6.4: Acertos do projeto Groove2Groove em arquivos de "transferência"

Conjunto de dados	**Acertos**	Acertos com 2º palpite	Acertos com o 3º palpite
D_O	**800**	800	800
D_P	**1166**	710	524
D_M	**800**	800	800

Tabela 6.5: Acertos do projeto Symbolic Music Style Transfer em arquivos de "transferência"

Sumarizando os resultados apresentados, observa-se que o projeto Groove2Groove apresenta performance melhor com o classificador baseado em D_O. Sendo assim, os resultados dele serão utilizados para o cálculo dos *scores* deste projeto. Na Figura 6.4, pode-se perceber a maior facilidade do classificador em identificar o estilo *Classic*. Além disso, o classificador apresenta certa confusão a respeito do *Jazz* e *Pop*, com performance ruim ao diferenciar ambos os estilos. Comprovando a mesma ideia, a Figura 6.5 apresenta a distribuição da porcentagem de confiança do classificador na resposta correta (em uma escala de 0 até 1, onde 1 é confiança total). Observa-se que a confiança do classificador é mais consistente para os casos de música clássica.

A partir dos apontamentos acima, pode-se concluir que o projeto Groove2Groove apresenta maior dificuldade em transferir arquivos para os estilos *Pop* e *Jazz* mantendo características significativas para futura análise.

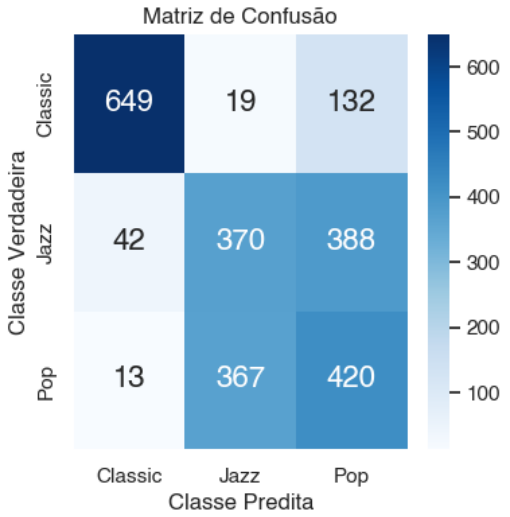


Figura 6.4: Matriz de confusão dos resultados da avaliação do projeto Groove2Groove pelo classificador de Estilos

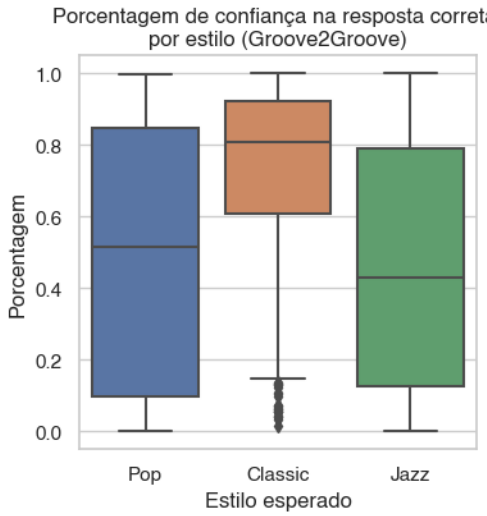


Figura 6.5: Distribuição dos resultados da avaliação do projeto Groove2Groove pelo classificador de Estilos

No caso do projeto Symbolic Music Style Transfer, é notável que o classificador baseado no conjunto D_P apresente melhor performance, o qual será usado para todas as análises futuras. Com a matriz de confusão apresentada na Figura 6.6, destaca-se o fato de que o classificador acertou, sem exceção, todas as transferências para o estilo *Jazz*. Além disso, o classificador erroneamente atribuiu o estilo *Jazz* para quase todos os outros casos. A confusão pode ser atribuída aos efeitos do pré-processamento, tornando as divisões entre os estilos consideravelmente mais tênues. Da mesma forma, o *Jazz* apresenta características mais dinâmicas que podem ter sido confundidas pelas imperfeições nas transferências realizadas pelo Symbolic Music Style Transfer.

Na Figura 6.7 observa-se a distribuição da confiança do classificador para cada estilo. Observa-se que as porcentagens atribuídas para o *Jazz* estão divididas em uma amplitude minúscula, mostrando que o classificador apresenta dificuldades em ter palpites mais acertivos. Além disso, a baixa confiança no estilo *Pop* e *Classic* estão de acordo com os resultados, traduzindo-se em poucos "palpites nº 1".

Nota-se que a mediana do *Jazz*, diferente dos resultados do Groove2Groove, é bem próxima de 0,6 (em uma escala de 0 até 1, onde 1 é confiança total), ou seja, apresentou menor certeza nos palpites para o projeto Symbolic Music Style Transfer do que Groove2Groove.

Embora o classificador baseado em D_P já tenha sido escolhido para o projeto Symbolic Music Style Transfer, ainda observa-se um resultado interessante na análise dos outros classificadores vistos na Tabela 6.5: os classificadores baseados em D_M e D_O acertaram 800 testes nos 1º, 2º e 3º palpites, individualmente. Investigações mais aprofundadas mostraram que os classificadores apresentaram apenas um único estilo como resposta para todos os testes, acertando exatamente 1/3 dos testes.

6.2.2 Identidade

As métricas de Identidade apresentam um processo de análise parcialmente diferente, uma vez que não se busca promover uma função "campeã" como se faz no Classificador de Estilo.

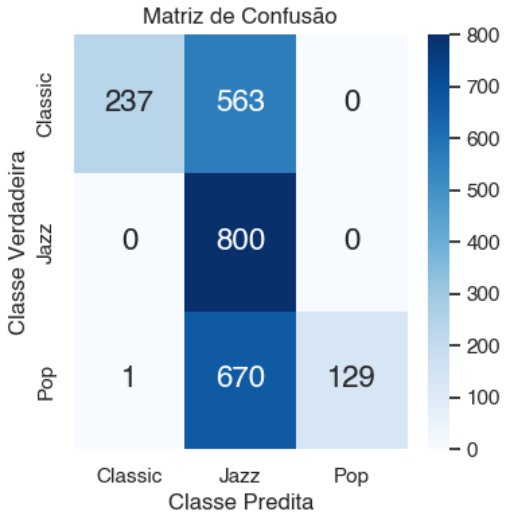


Figura 6.6: Matriz de confusão dos resultados da avaliação do projeto Symbolic Music Style Transfer pelo classificador de Estilos

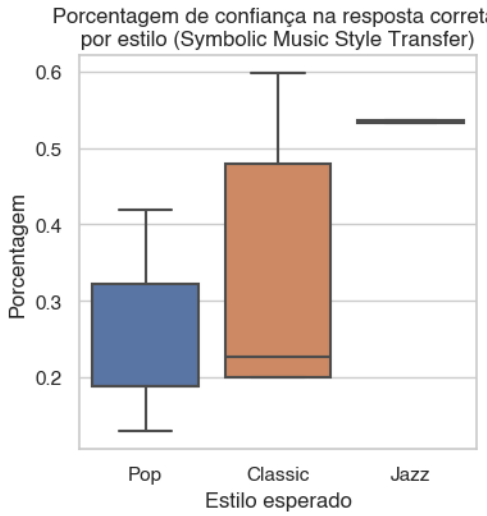


Figura 6.7: Distribuição dos resultados da avaliação do projeto Symbolic Music Style Transfer pelo Classificador de Estilos

Cada métrica será analisada comparando os resultados de cada projeto e suas distribuições. Além disso, na presente seção, serão comparados os pares do conjunto M_T, definidos na seção 5.6 - ou seja, elementos cujo valor ideal é "ambíguo", não sendo a similaridade e distância completas. A apresentação considerará o resultado das métricas já transformado nos *scores*.

6.2.2.1 Score de Similaridade de Espectrogramas (S^T_{cv2})

A Similaridade de Espectrogramas é calculada a partir das imagens geradas pelos arquivos de áudio. A Tabela 6.6 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino". Além disso, a Figura 6.8 apresenta uma comparação da distribuição de ambos os projetos.

Projeto	**Estilo "Destino"**	**Média**	**Desvio Padrão**	S^T_{cv2}
Groove2Groove	*Classic*	0,886	±0,069	**0,908**
	Jazz	0,933	±0,049	
	Pop	0,939	±0,044	
Symbolic Music Style Transfer	*Classic*	0,777	±0,008	**0,776**
	Jazz	0,773	±0,007	
	Pop	0,78	±0,001	

Tabela 6.6: Média e desvio padrão dos resultados da comparação de espectrogramas, agrupada por projeto e estilo "destino"

Pode-se observar na Tabela 6.6 que o projeto Symbolic Music Style Transfer apresenta resultados significativamente mais próximos (ou seja, *scores* menores) com um desvio padrão menor. Por outro lado, também se percebe que o projeto Groove2Groove apresenta resultados menos diferentes no estilo *Classic*. Comparando o pré-processamento realizado pelo Symbolic Music Style Transfer e a simplificação que ocorre na música com a performance do Groove2Groove, no estilo *Classic*, conclui-se que pelo menos parte dos resultados mais próximos do Symbolic Music Style Transfer ocorreram como produto da simplificação na música.

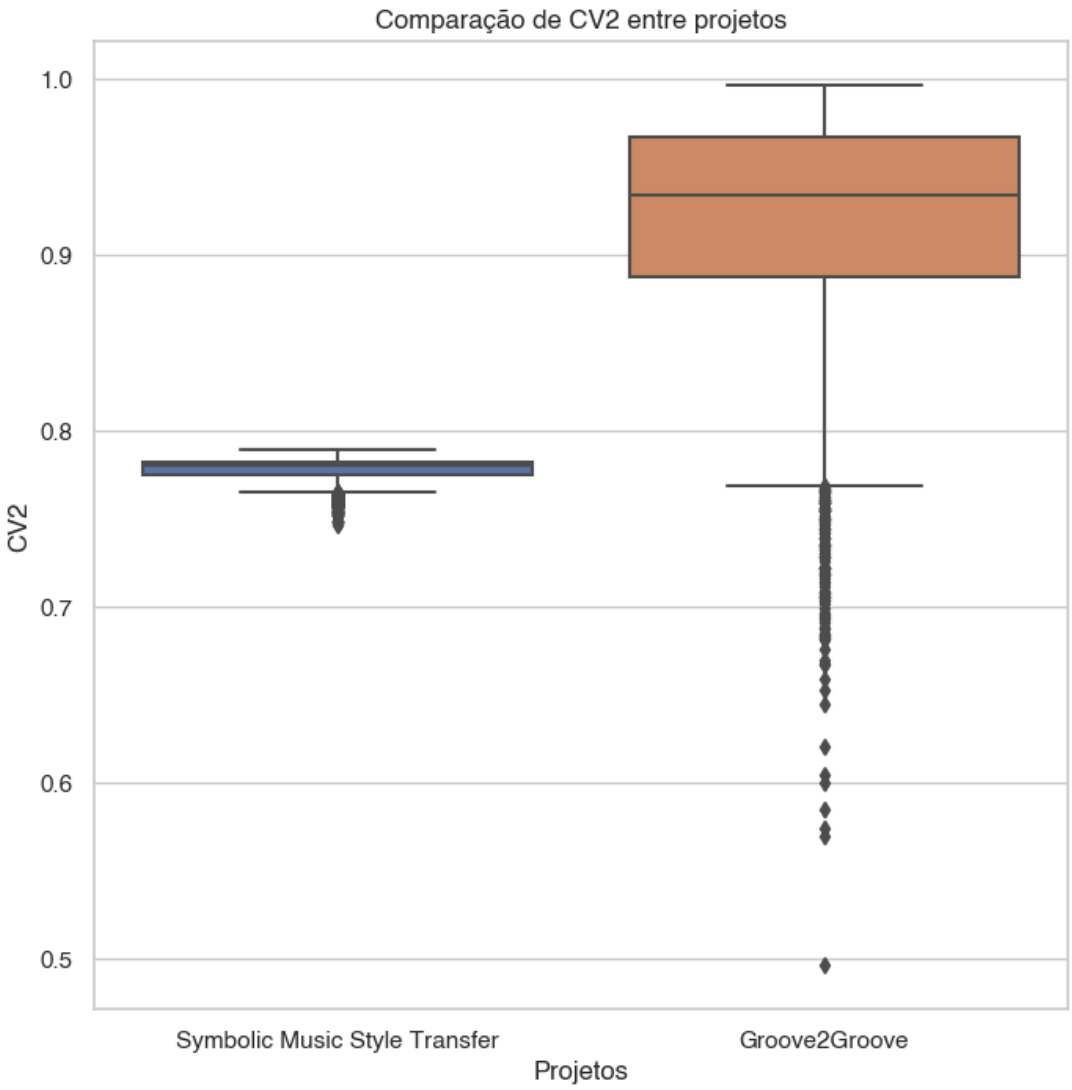


Figura 6.8: Distribuição do *score* de similaridade de espectrogramas entre cada projeto

Na Figura 6.8, observa-se que os valores de semelhança dos espectrogramas para o projeto Symbolic Music Style Transfer estão concentrados, sendo que o melhor resultado não ultrapassa os 0,8 de *score*. Por outro lado, embora o projeto Groove2Groove apresente a maioria dos resultados como melhores do que o Symbolic Music Style Transfer, percebe-se que uma pequena parcela dos casos de teste tiveram performance inferior, com menos que 0,5 do *score* total. Concluiu-se que parte da performance do Groove2Groove está associada a maior complexidade das músicas e, consequentemente, maior possibilidades de espectrogramas.

6.2.2.2 Score RQA (S^T_{rqa})

O *score* RQA é calculado a partir dos arquivos de áudio. A Tabela 6.7 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino". Além disso, a Figura 6.9 apresenta uma comparação da distribuição do *score* de ambos os projetos.

Pode-se observar na Tabela 6.7 que, novamente, o Symbolic Music Style Transfer apresenta uma média menor no valor dos *scores* de identidade, ou seja, maior semelhança entre os projetos. Entretanto, percebe-se também que o desvio padrão em ambos os projetos é consideravelmente alto - maior do que a média em todos os casos.

Projeto	Estilo "Destino"	Média	Desvio Padrão	S^T_{rqa}
Groove2Groove	*Classic*	0,988	±0,018	**0,983**
	Jazz	0,979	±0,031	
	Pop	0,981	±0,025	
Symbolic Music Style Transfer	*Classic*	0,959	±0,059	**0,910**
	Jazz	0,881	±0,159	
	Pop	0,889	±0,144	

Tabela 6.7: Média e desvio padrão dos *scores* RQA, agrupada por projeto e estilo "destino"

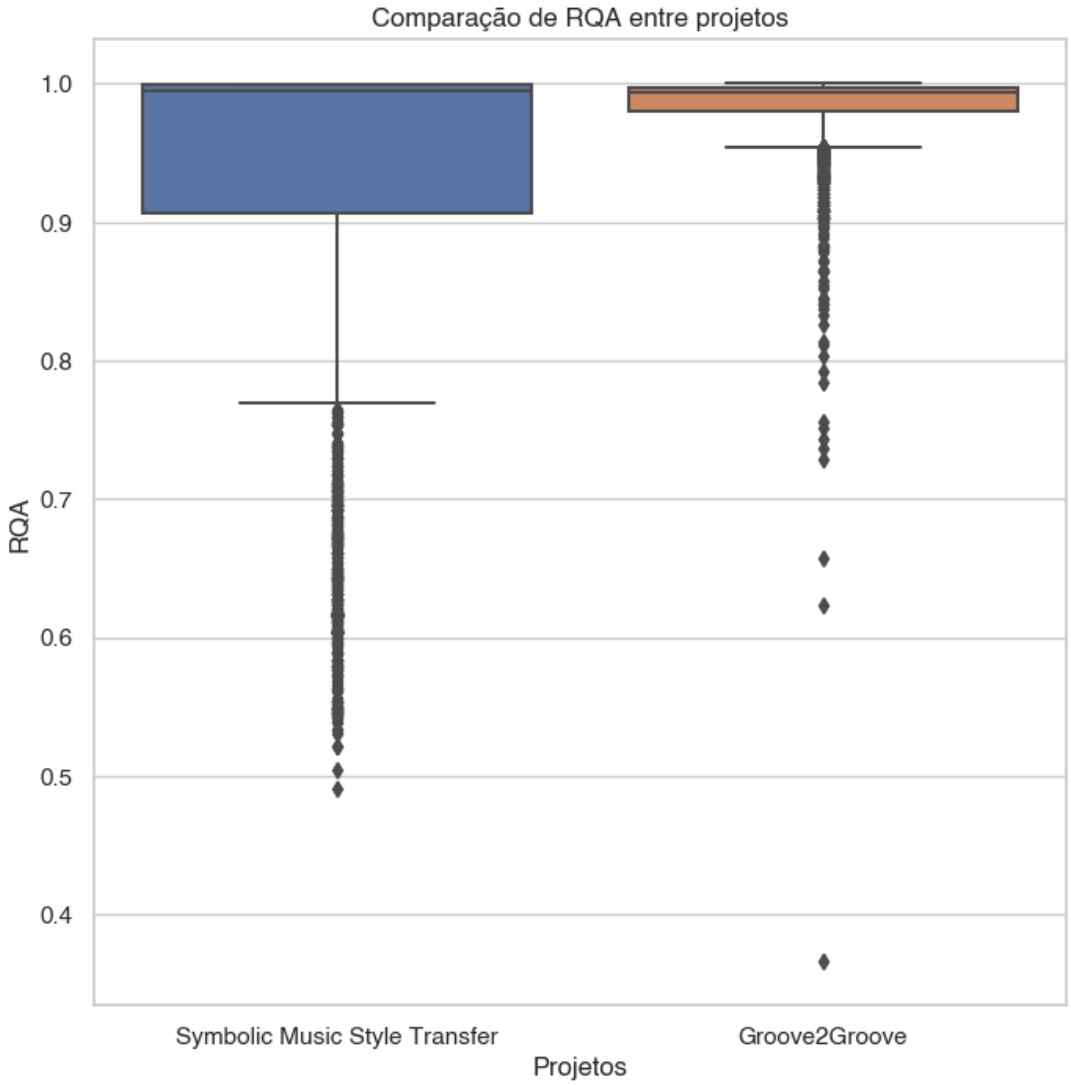


Figura 6.9: Distribuição do *score* RQA entre cada projeto

Observando a Figura 6.9, nota-se que os projetos possuem mais da metade dos casos de teste em um intervalo extremamente pequeno de valores. Entretanto, o projeto Symbolic Music Style Transfer atinge um valor menor de *score*, sendo mais de um quarto dos casos entre 91% e 77%. Por outro lado, o Groove2Groove apresenta mais da metade dos valores entre 1 e 0.99. Curiosamente, o Groove2Groove apresenta o caso de teste com o menor valor de *score* RQA, estando abaixo de 0,4.

6.2.2.3 *Score FastDTW* (S^T_{dtw})

O *score* FastDTW é calculado a partir dos arquivos de áudio. A Tabela 6.8 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino". Além disso, a Figura 6.10 apresenta uma comparação da distribuição do *score* de ambos os projetos.

Projeto	**Estilo "Destino"**	**Média**	**Desvio Padrão**	S^T_{dtw}
Groove2Groove	*Classic*	0,121	±0,022	**0,128**
	Jazz	0,131	±0,025	
	Pop	0,132	±0,025	
Symbolic Music Style Transfer	*Classic*	0,093	±0,008	**0,094**
	Jazz	0,093	±0,013	
	Pop	0,096	±0,005	

Tabela 6.8: Média e desvio padrão dos *scores* FastDTW, agrupada por projeto e estilo "destino"

A Tabela 6.8 apresenta resultados bastante consistentes com os anteriores, nos quais o projeto Symbolic Music Style Transfer apresenta *scores* de similaridade menores do que o Groove2Groove. Do mesmo modo, observando a Figura 6.10, nota-se que, novamente, o projeto Groove2Groove apresenta casos de teste específicos com os menores valores de similaridade, embora tenha mais da metade dos valores maiores do que o maior valor encontrado nas "transferências" do Symbolic Music Style Transfer.

6.2.3 Análise do *score* de Transferência (S^T)

A partir das métricas acima, é possível calcular o *score* de Transferência de cada projeto. A Tabela 6.9 apresenta a média e o desvio padrão de cada *score* para cada projeto.

Score	**Projeto**	**Média**	**Desvio Padrão**
Score Espectrogramas	Symbolic Music Style Transfer	0,776	±0,003
	Groove2Groove	0,908	±0,050
Score RQA	Symbolic Music Style Transfer	0,910	±0,070
	Groove2Groove	0,983	±0,020
Score FastDTW	Symbolic Music Style Transfer	0,094	±0,003
	Groove2Groove	0,128	±0,022
Score CE	Symbolic Music Style Transfer	0,368	±0,151
	Groove2Groove	0,56	±0,344
Score Transferência	Symbolic Music Style Transfer	**0,503**	±0,058
	Groove2Groove	**0,628**	±0,137

Tabela 6.9: Média e desvio padrão dos *scores* de Transferência agrupados por projeto e tipo de *score*

Pode-se observar que o Groove2Groove apresenta performance superior ao Symbolic Music Style Transfer em todos os *scores*. Destacam-se dois fatores para tal fato: 1) As funções de identidade foram avaliadas com 0 como alvo, ou seja, distância completa, fator negativo para o projeto Symbolic Music Style Transfer que aparenta reter melhor características; 2) O projeto Groove2Groove apresentou performance consideravelmente superior no processo de classificação de estilo, fator que contribuiu para a melhor performance.

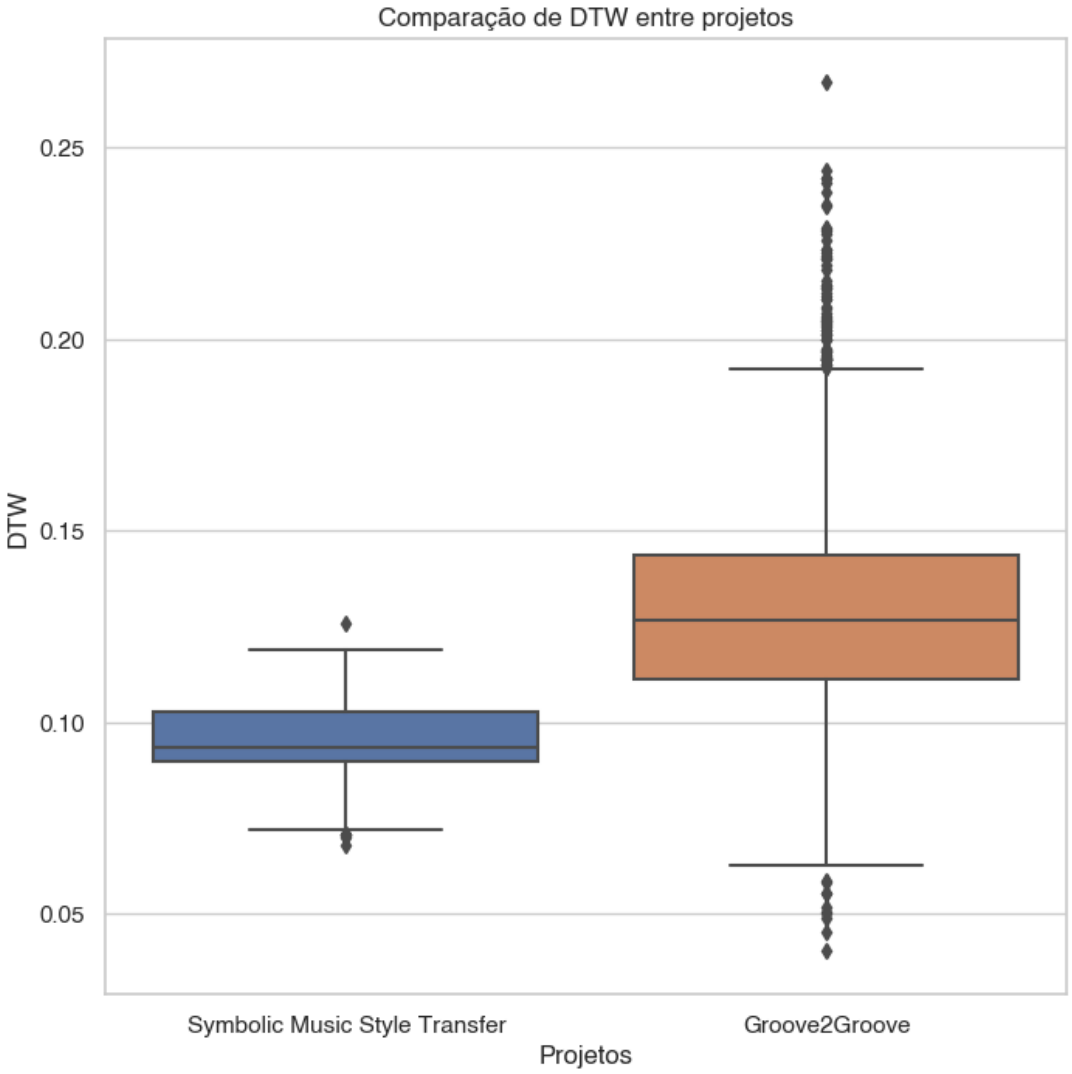


Figura 6.10: Distribuição do *score* FastDTW entre cada projeto

Analisando a Figura 6.11, pode-se visualizar as distribuições dos diferentes tipos de *scores*. Nota-se que na maior parte das métricas o projeto Groove2Groove resulta em uma amplitude de valores maior, causada pela maior diversidade e complexidade dos artefatos. Além disso, considera-se o Groove2Groove o melhor em termos de *Score* de Transferência, por apresentar a maior média de *scores* como pode ser visto na Tabela 6.9.

6.3 AVALIAÇÃO DO *SCORE* DE CICLO (S^C)

Como definido na seção 5.6, o *score* de Ciclo refere-se à avaliação de um par de arquivos MIDI que pertencem ao mesmo domínio, mais especificamente, à comparação de um arquivo base e o resultado após duas transferências de estilo, retornado ao domínio original.

6.3.1 *Score* do Classificador de Estilo (S^C_{CE})

Analogamente a seção 6.2.1, os arquivos do tipo "ciclo" foram avaliados pelos três classificadores, com a mesma ordenação e consideração dos três palpites. A Tabela 6.10 apresenta

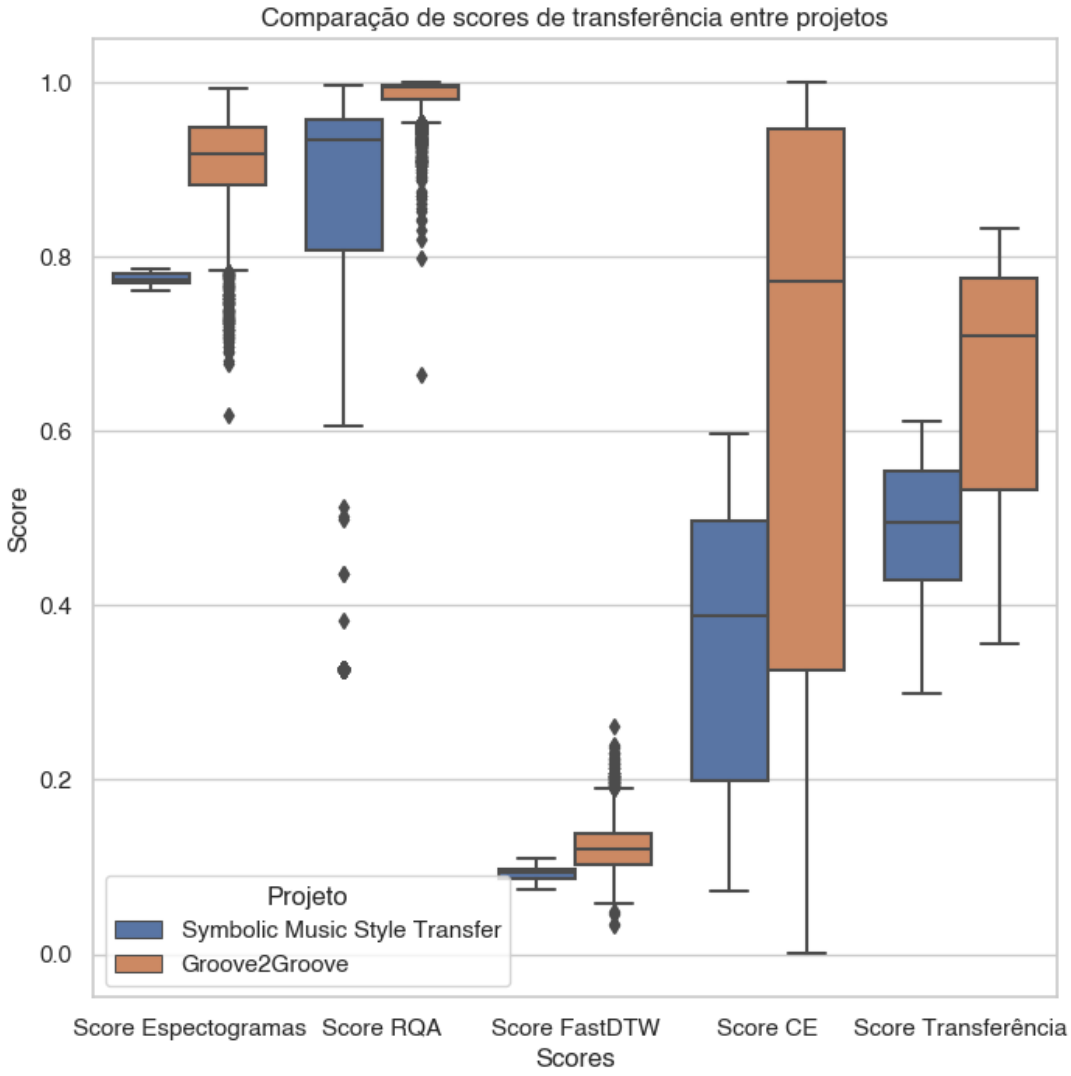


Figura 6.11: Distribuição dos *scores* de Transferência entre cada projeto

os resultados obtidos por cada classificador para o projeto Groove2Groove, enquanto a Tabela 6.11 apresenta os resultados para o projeto Symbolic Music Style Transfer.

Conjunto de dados	**Acertos**	Acertos com o 2^o palpite	Acertos com o 3^o palpite
D_O	**1813**	580	7
D_P	**1419**	568	413
D_M	**1793**	581	26

Tabela 6.10: Acertos do projeto Groove2Groove em arquivos de "ciclo"

Analisando os resultados apresentados nas Tabelas 6.10 e 6.11, observa-se mais uma vez que o Groove2Groove apresenta performance melhor com o classificador baseado em D_O, assim como o projeto Symbolic Music Style Transfer atinge resultados melhores com o treinado em D_P. Novamente, ambos os classificadores serão os utilizados para as próximas análises do *score* de ciclo.

Na Figura 6.12, percebe-se a maior facilidade do Classificador de identificar o estilo *Clássico*. Além disso, quando comparado a matriz de confusão de transferência, vista na Figura 6.4, observa-se que o Classificador apresenta melhor desempenho na classificação dos estilos *Jazz*

Conjunto de dados	**Acertos**	Acertos com 2^o palpite	Acertos com o 3^o palpite
D_O	**800**	800	800
D_P	**1008**	608	784
D_M	**400**	1200	800

Tabela 6.11: Acertos do projeto Symbolic Music Style Transfer em arquivos de "ciclo"

e *Pop*, em relação à transferência. A Figura 6.13 também apresenta resultados da distribuição de confiança nos palpites consideravelmente mais estáveis em relação aos casos de transferência, vistos na Figura 6.4.

Com base nos apontamentos acima, conclui-se que o projeto Groove2Groove produz versões de ciclo mais consistentes que as de transferência, sendo capaz de trazer características do estilo original a ponto do estilo ser identificado apropriadamente.

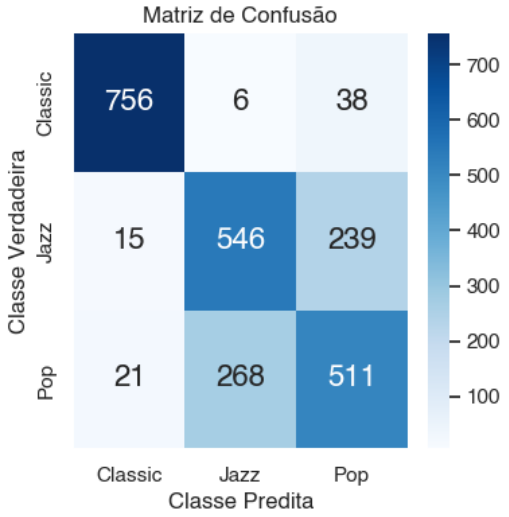


Figura 6.12: Matriz de confusão dos resultados da avaliação do projeto Groove2Groove pelo Classificador de Estilos

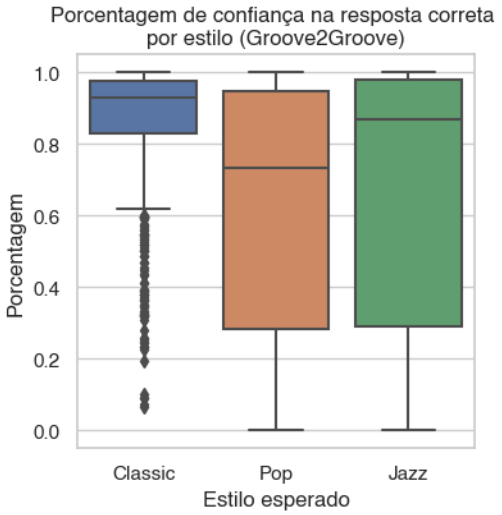


Figura 6.13: Distribuição dos resultados da avaliação do projeto Groove2Groove pelo Classificador de Estilos

Já no caso do projeto Symbolic Music Style Transfer encontram-se resultados enviesados para o estilo *Jazz*. Destaca-se que o Classificador acertou, sem exceção, todas as transferências para o estilo *Jazz*. Além disso, o Classificador novamente atribuiu erroneamente o estilo *Jazz* para quase todos os outros casos, especialmente do estilo *Pop*. Atribui-se os resultados enviesados para o *Jazz* pelo mesmo motivo que mencionado na Figura 6.14 - o pré-processamento do Symbolic Music Style Transfer. Também nota-se que, no caso do ciclo, o projeto Symbolic Music Style Transfer foi capaz de acertar pelo menos 25% dos casos de estilo *Classic*, diferentemente dos resultados anteriores.

A análise da distribuição de confiança também apresenta resultados parecidos, nos quais o estilo *Pop* apresenta pouquíssimos exemplos cujo palpite correto teve confiança maior que 33%.

Finalmente, observa-se o mesmo padrão encontrado na análise do Classificador de Estilos do *score* de transferência, em que os classificadores acertam números específicos de casos de teste do projeto Symbolic Music Style Transfer. Uma investigação aprofundada mostrou resultados parecidos, nos quais os classificadores baseados em D_O e D_M, em resumo, respondem apenas um estilo. A única diferença é que D_M, neste caso, apresentou desempenho ainda pior.

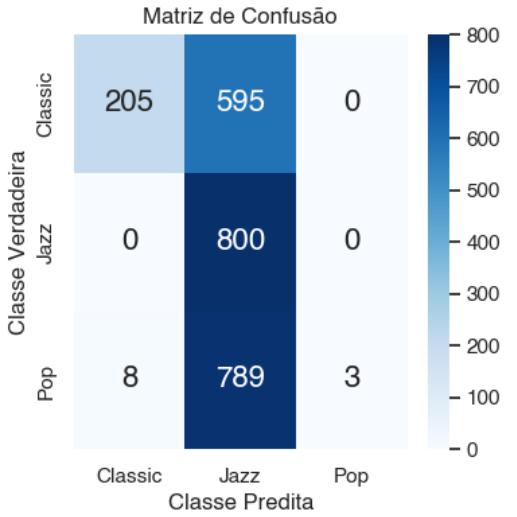


Figura 6.14: Matriz de confusão dos resultados da avaliação do projeto Symbolic Music Style Transfer pelo Classificador de Estilos

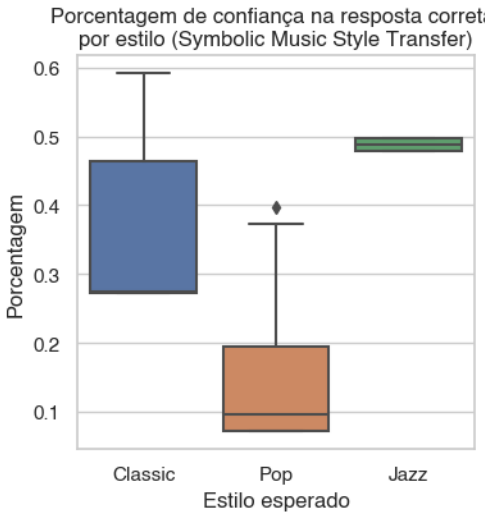


Figura 6.15: Distribuição dos resultados da avaliação do projeto Symbolic Music Style Transfer pelo Classificador de Estilos

6.3.2 Identidade

Novamente, as métricas de Identidade apresentam um processo de análise parcialmente diferente, uma vez que não se busca promover uma função campeã como se faz no Classificador de Estilo. Cada métrica será analisada comparando os resultados de cada projeto e suas distribuições. Diferente da análise anterior, serão comparados os pares do conjunto M_C, definido na seção 5.6 - ou seja, elementos cujo valor ideal é 1. A apresentação ira considerar o resultado das métricas já transformado nos *scores*.

6.3.2.1 Score de Similaridade de Espectrogramas (S^C_{cv2})

A similaridade de espectrogramas é calculada a partir das imagens geradas a partir dos arquivos de áudio. A Tabela 6.12 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino". Além disso, a Figura 6.16 apresenta uma comparação da distribuição de ambos os projetos.

Projeto	**Estilo "Destino"**	**Média**	**Desvio Padrão**	S^C_{cv2}
Groove2Groove	Classic	0,072	±0,047	**0,089**
	Jazz	0,1	±0,065	
	Pop	0,095	±0,062	
Symbolic Music Style Transfer	Classic	0,228	±0,009	**0,227**
	Jazz	0,222	±0,006	
	Pop	0,233	±0,001	

Tabela 6.12: Média e desvio padrão dos resultados da comparação de espectrogramas, agrupada por projeto e estilo "destino"

Observa-se na Tabela 6.12 que o projeto Symbolic Music Style Transfer apresenta resultados mais semelhantes do que o projeto Groove2Groove. Além disso, o Symbolic Music Style Transfer apresenta desvio padrão menor, sendo bem mais robusto.

Na Figura 6.16, observa-se que os valores de semelhança dos espectrogramas para o Symbolic Music Style Transfer estão todos concentrados, concordando com os resultados

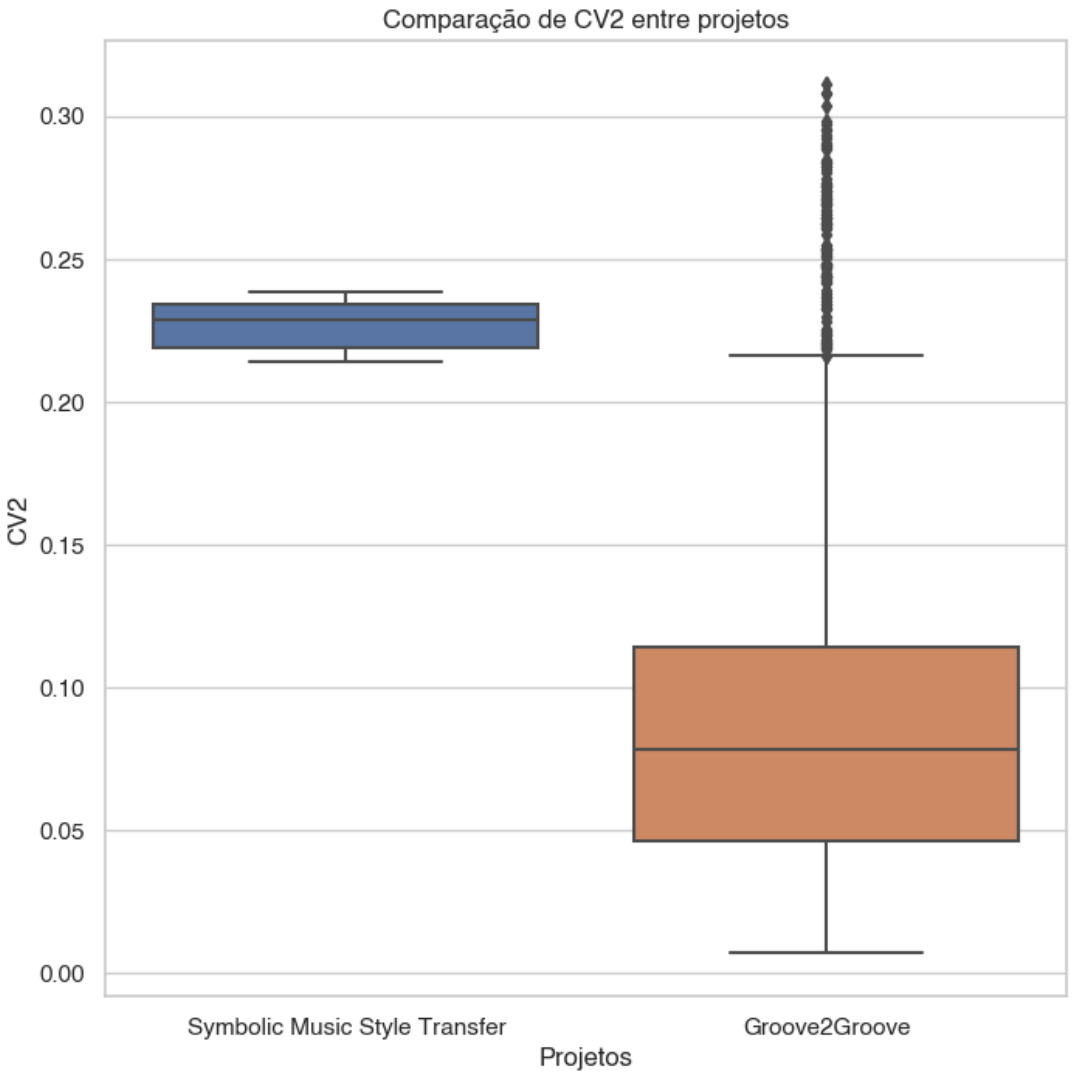


Figura 6.16: Distribuição do *score* de similaridade de espectrogramas entre cada projeto

apresentados na Tabela 6.12. Novamente, o projeto Groove2Groove apresenta maior amplitude de valores, contando com os melhores e piores resultados. Conclui-se que o projeto Symbolic Music Style Transfer domina a comparação, apresentando resultados estáveis e consideravelmente melhores.

6.3.2.2 Score RQA (S^C_{rqa})

O *score* RQA é calculado a partir dos arquivos de áudio. A Tabela 6.13 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino". Além disso, a Figura 6.17 apresenta uma comparação da distribuição do *score* de ambos os projetos.

A Tabela 6.13 apresenta uma performance consideravelmente melhor do projeto Symbolic Music Style Transfer em relação ao Groove2Groove. Especialmente para o estilo *Jazz*, o projeto Symbolic Music Style Transfer mantém a média em 0,363 - 36% de similaridade. Por outro lado, o projeto Groove2Groove atinge valores extremamente baixos, com uma média de 0,3% de similaridade no estilo *Classic*.

Analisando a Figura 6.17 continuam as afirmações anteriores, com o Symbolic Music Style Transfer estando consideravelmente melhor. Os *outliers* do projeto atingem quase 65%

Projeto	Estilo "Destino"	Média	Desvio Padrão	S^C_{rqa}
Groove2Groove	*Classic*	0,003	±0,005	**0,015**
	Jazz	0,021	±0,025	
	Pop	0,020	±0,024	
Symbolic Music Style Transfer	*Classic*	0,119	±0,125	**0,206**
	Jazz	0,363	±0,309	
	Pop	0,134	±0,078	

Tabela 6.13: Média e desvio padrão dos *scores* RQA, agrupada por projeto e estilo "destino"

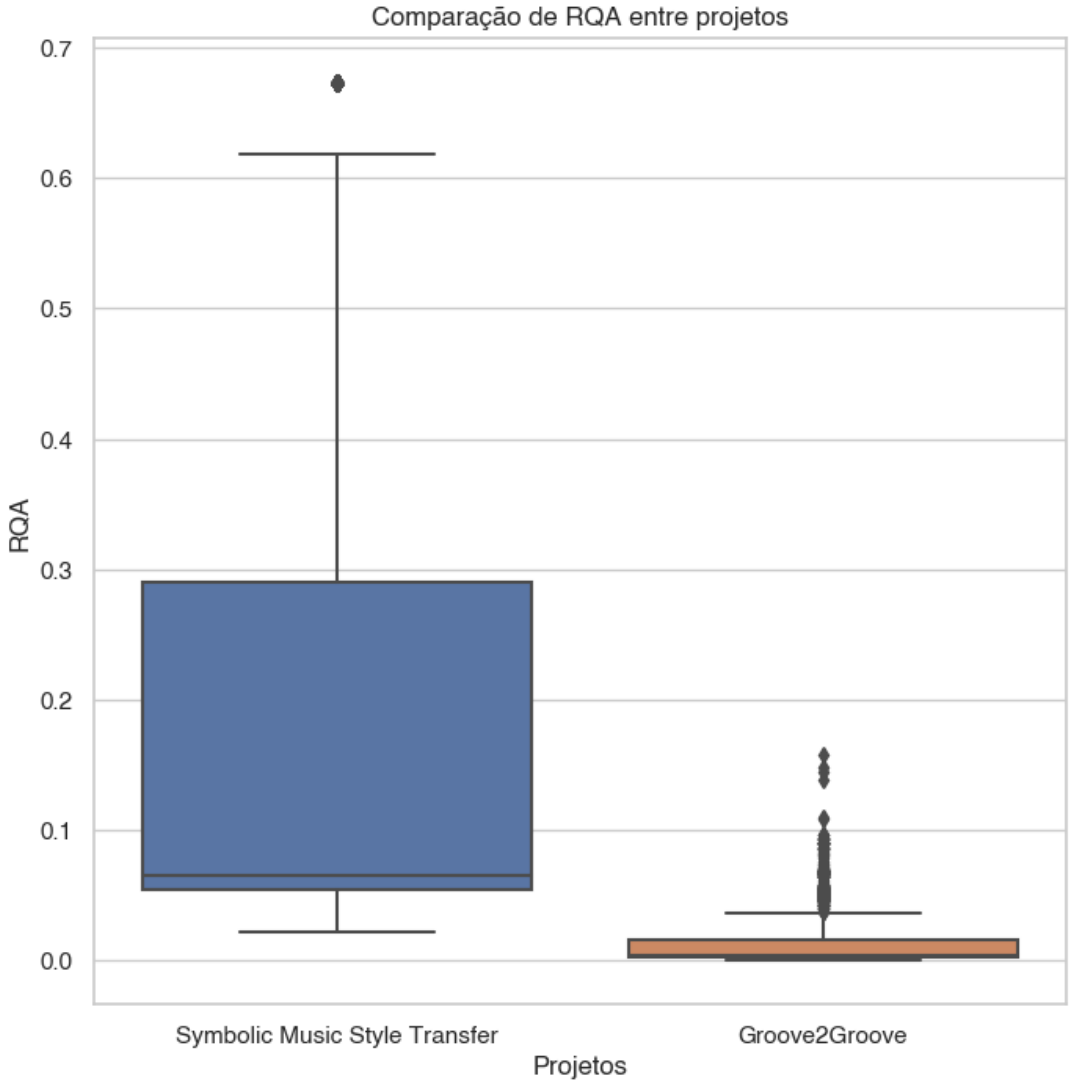


Figura 6.17: Distribuição do *score* RQA entre cada projeto

de similaridade, valor elevadíssimo em relação aos resultados do Groove2Groove. Nota-se também que projeto Symbolic Music Style Transfer apresenta agora maior amplitude de valores, analogamente ao Groove2Groove nas métricas do *Score* de Transferência.

6.3.2.3 Score FastDTW (S^C_{dtw})

O *score* FastDTW é calculado a partir dos arquivos de áudio. A Tabela 6.14 apresenta uma comparação entre a média e o desvio padrão de cada projeto para cada estilo "destino".

Além disso, a Figura 6.18 apresenta uma comparação da distribuição do *score* de ambos os projetos.

Projeto	Estilo "Destino"	Média	Desvio Padrão	S^C_{dtw}
Groove2Groove	*Classic*	0,883	±0,025	**0,884**
	Jazz	0,883	±0,028	
	Pop	0,886	±0,029	
Symbolic Music Style Transfer	*Classic*	0,915	±0,01	**0,913**
	Jazz	0,906	±0,003	
	Pop	0,917	±0,002	

Tabela 6.14: Média e desvio padrão dos *scores* FastDTW, agrupada por projeto e estilo "destino"

Diferente dos *scores* anteriores, ambos os projetos apresentam um resultado bastante elevado. Ainda assim, o projeto Symbolic Music Style Transfer apresenta média superior em todos os estilos, além de um desvio padrão menor e mais robusto.

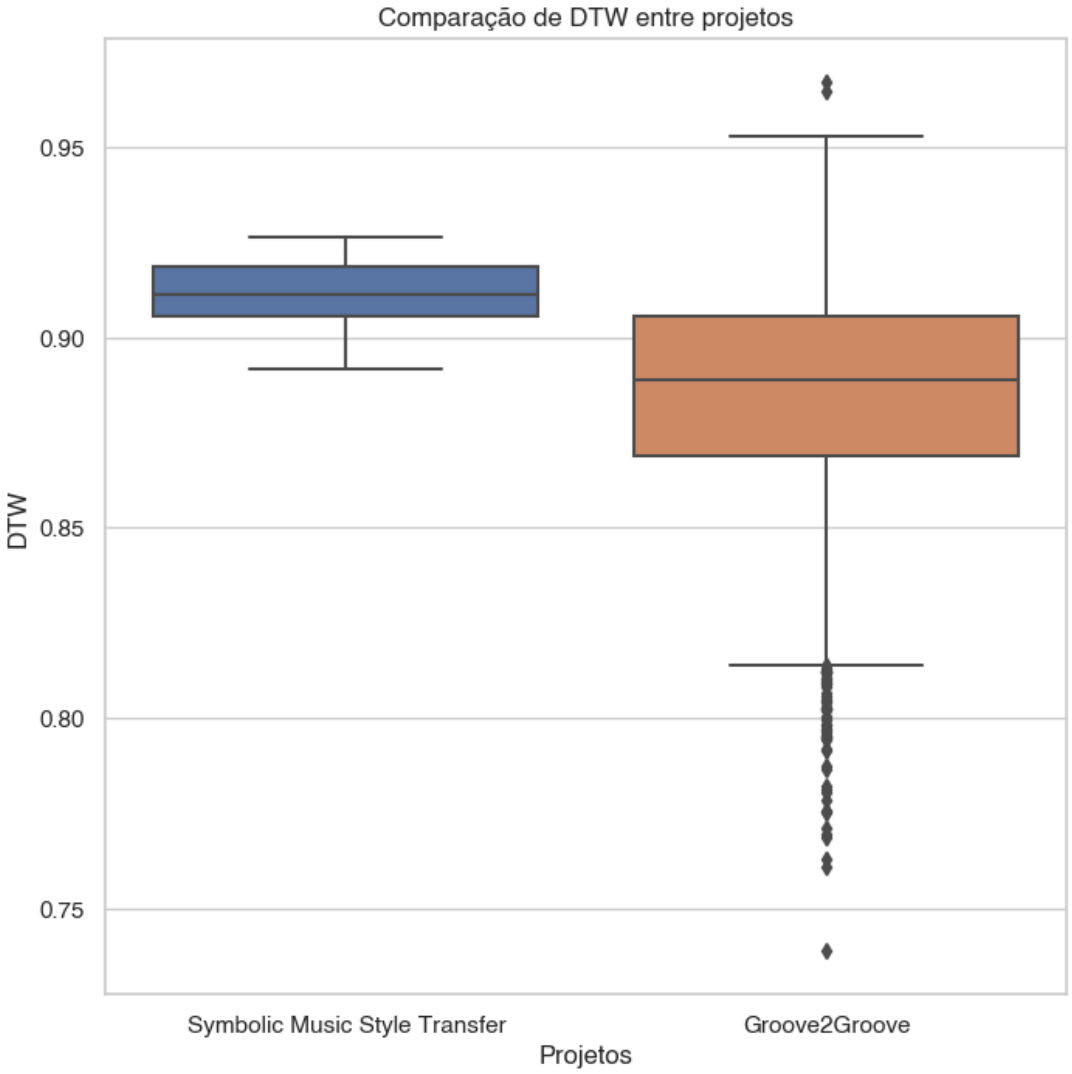


Figura 6.18: Distribuição do *score* FastDTW entre cada projeto

Analisando a Figura 6.18, observa-se uma distribuição que corrobora com as afirmações acima. Entretanto, nota-se a presença de *outliers* nos resultados do Groove2Groove, sendo o projeto com os melhores e piores resultados individuais.

6.3.3 Análise do *score* de Ciclo (S^C)

A partir das métricas acima, é possível calcular o *score* de ciclo de cada projeto. A Tabela 6.15 apresenta a média e o desvio padrão de cada *score* para cada um.

Score	**Projeto**	**Média**	**Desvio Padrão**
Score Espectrogramas	Symbolic Music Style Transfer	0,227	±0,008
	Groove2Groove	0,089	±0,060
Score RQA	Symbolic Music Style Transfer	0,206	±0,227
	Groove2Groove	0,015	±0,022
Score FastDTW	Symbolic Music Style Transfer	0,913	±0,008
	Groove2Groove	0,884	±0,028
Score CE	Symbolic Music Style Transfer	0,401	±0,072
	Groove2Groove	0,713	±0,330
Score Ciclo	Symbolic Music Style Transfer	**0,401**	±0,072
	Groove2Groove	**0,483**	±0,136

Tabela 6.15: Média e desvio padrão dos *scores* de Ciclo agrupados por projeto e tipo de *score*

Observa-se na Tabela 6.15 que o projeto Symbolic Music Style Transfer apresenta resultados superiores em todas as métricas de Identidade, enquanto o projeto Groove2Groove apresenta resultados melhores na Classificação de Estilo. Embora o Symbolic Music Style Transfer tenha mais métricas com performance melhor, o Groove2Groove obteve um *score* final superior, destacando-se como o melhor projeto em termos de ciclo também. Para a justificativa de tais resultados, destacam-se os fatores de:

- A diferença dos projetos no *Score* FastDTW é bastante pequena.
- O *Score* Ciclo atribui um peso de 2 para o *Score* CE, que é bastante a favor do projeto Groove2Groove

Analisando a Figura 6.19, pode-se visualizar as distribuições dos diferentes tipos de *scores*. Observa-se que o projeto Symbolic Music Style Transfer apresentou no *score* RQA resultados elevados nos dois últimos quartis de valores, sendo este um dos principais fatores que provocaram a grande diferença em relação ao projeto Groove2Groove. Por outro lado, pode-se perceber a superioridade do projeto Groove2Groove no *score* da classificação de estilo, com uma grande quantidade dos valores acima de 0,9. Finalmente, a distribuição do *score* de ciclo mostra que o projeto Groove2Groove possui alguns exemplos com performance pior ao Symbolic Music Style Transfer, mas na vasta maioria melhor.

6.4 AVALIAÇÃO DO *SCORE* BASE (S)

Apresentados os resultados do *Score* de Transferência e o *Score* de Ciclo, uma análise do *Score* Base pode ser feita com fundamentações melhores. O *Score* Base é um resumo de ambos os *scores*, calculado com uma média simples entre ambos. Na Tabela 6.16, observa-se que o projeto Groove2Groove apresenta uma média de Score Base melhor, com 0,1 de vantagem. Além

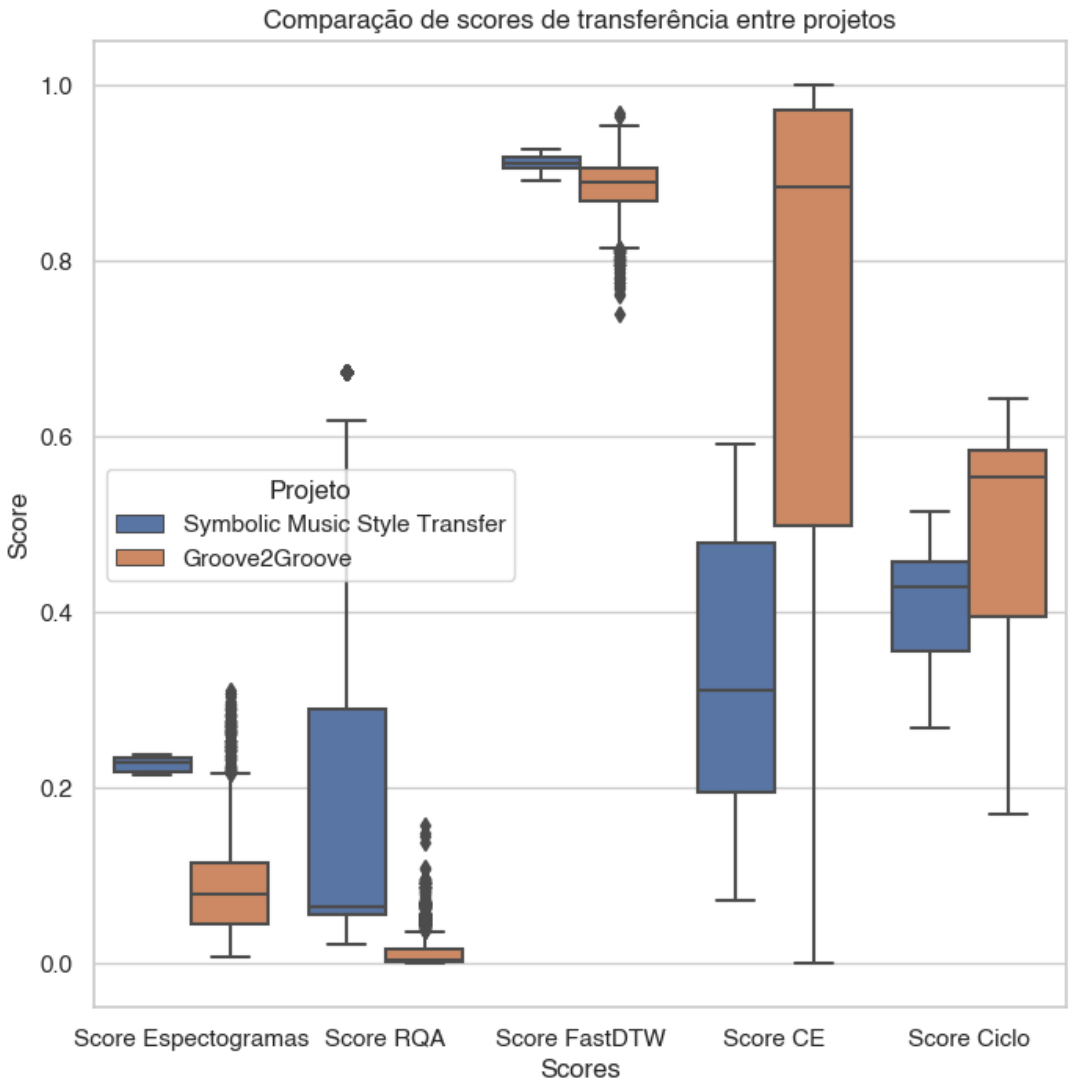


Figura 6.19: Distribuição dos *scores* de Ciclo entre cada projeto

disso, o Groove2Groove resulta em uma performance melhor tanto no Score de Transferência quanto no Score de Ciclo.

A Figura 6.20 apresenta os mesmos resultados, com a distribuição dos *scores* entre os exemplos. Pode-se observar que, de fato, o projeto Groove2Groove apresenta melhor performance em todos os aspectos.

Além disso, a hipótese anterior do enviesamento da aleatoriedade nos *Scores* de Transferência se provou correta. Percebe-se que ambos os projetos obtiveram *score* bastante elevado na comparação de espectrogramas e métrica *RQA*, o que elevou consideravelmente o valor do *Score* Base. Como discutido previamente na seção 5.6, um dos motivos é o fato da distância entre dois arquivos MIDI ser grande. Como tentativa de solução, foi proposto o *Score Desafiante*, cujos resultados serão apresentados na seção 6.5. Em síntese, pela avaliação do *Score* Base, considera-se o projeto Groove2Groove superior ao Symbolic Music Style Transfer.

Projeto	Score	Média	Desvio Padrão
Groove2Groove	Transferência	0,628	±0,137
	Ciclo	0,483	±0,136
	Base	**0,555**	±0,087
Symbolic Music Style Transfer	Transferência	0,503	±0,058
	Ciclo	0,401	±0,072
	Base	**0,452**	±0,028

Tabela 6.16: Média e desvio padrão dos *scores* de Transferência, Ciclo e Base, agrupado por projeto

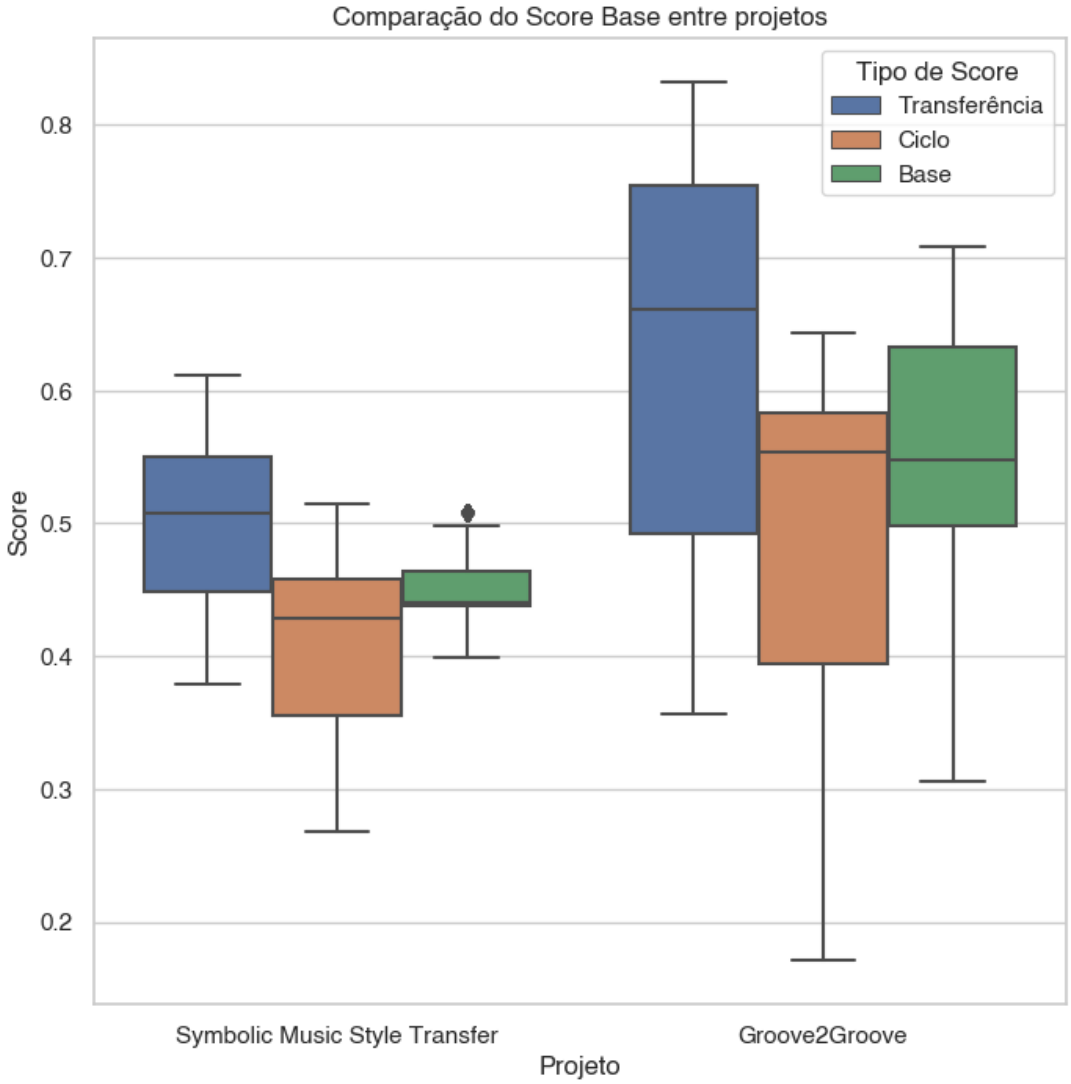


Figura 6.20: Distribuição dos *scores* Base entre cada projeto

6.5 AVALIAÇÃO DO *SCORE* DESAFIANTE (S_D)

Para a avaliação do *Score* Desafiante, serão inicialmente apresentados os resultados das mudanças realizadas no *Score* de Transferência ($S^{T'}$), comparando o *Score* Desafiante com o Score *Base* e mostrando qual apresenta maior robustez para análise.

6.5.1 Score de Transferência Desafiante ($S^{T'}$)

Os resultados das mudanças aplicadas nos *scores* podem ser encontradas na Tabela 6.17. Além disso, a distribuição dos *scores* pode ser vista na Figura 6.21.

Score	**Projeto**	**Média**	**Desvio Padrão**
Score Espectrogramas	Symbolic Music Style Transfer	0,176	±0,005
	Groove2Groove	0,078	±0,048
Score RQA	Symbolic Music Style Transfer	0,174	±0,181
	Groove2Groove	0,014	±0,02
Score FastDTW	Symbolic Music Style Transfer	0,086	±0,003
	Groove2Groove	0,113	±0,018
Score CE	Symbolic Music Style Transfer	0,368	±0,151
	Groove2Groove	0,56	±0,344
Score Transferência (desafiante)	Symbolic Music Style Transfer	**0,235**	±0,084
	Groove2Groove	**0,265**	±0,137

Tabela 6.17: Média e desvio padrão dos *scores* de Transferência (desafiante) agrupados por projeto e tipo de *score*

Percebe-se que, com a mudança realizada, os valores de *score* (com exceção do *Score* da Classificação de Estilo) diminuem consideravelmente. Além disso, o projeto Symbolic Music Style Transfer apresenta performance superior ao Groove2Groove nos *scores* de Espectrogramas e no *score* RQA, diferente do *score* de Transferência anterior apresentado na Tabela 6.9.

Portanto, pode-se concluir que a adição de um peso auxiliou na supervalorização dos valores de distância. Como o projeto Symbolic Music Style Transfer apresenta estabilidade na maioria das métricas de ciclo, uma valorização média das métricas do *Score* de Transferência acaba ocorrendo em todos os casos. Sendo assim, as mudanças acabam punindo o projeto Groove2Groove.

Analisando a Figura 6.21, pode-se observar que a distribuição do *Score* de Transferência entre os projetos é consideravelmente mais próximo do que na Figura 6.11 e também ficam visíveis os efeitos da punição aplicada no cálculo dos *scores*. Além disso, pode-se observar que embora o projeto Groove2Groove apresente a mediana maior do que o projeto Symbolic Music Style Transfer, a média do Symbolic Music Style Transfer é maior.

6.5.2 Score Desafiante (S_D)

Feita a análise das mudanças no *Score* de Transferência, pode-se verificar os resultados do *Score* Desafiante e os elementos que o compõe. A Tabela 6.18 apresenta os resultados para cada projeto, enquanto a Figura 6.22 apresenta a distribuição dos resultados.

Projeto	**Score**	**Média**	**Desvio Padrão**
Groove2Groove	Transferência (desafiante)	0,265	±0,137
	Ciclo	0,483	±0,136
	Desafiante	**0,374**	±0,089
Symbolic Music Style Transfer	Transferência (desafiante)	0,235	±0,084
	Ciclo	0,401	±0,072
	Desafiante	**0,318**	±0,050

Tabela 6.18: Média e desvio padrão dos *scores* de Transferência (desafiante), ciclo e Desafiante, agrupado por projeto

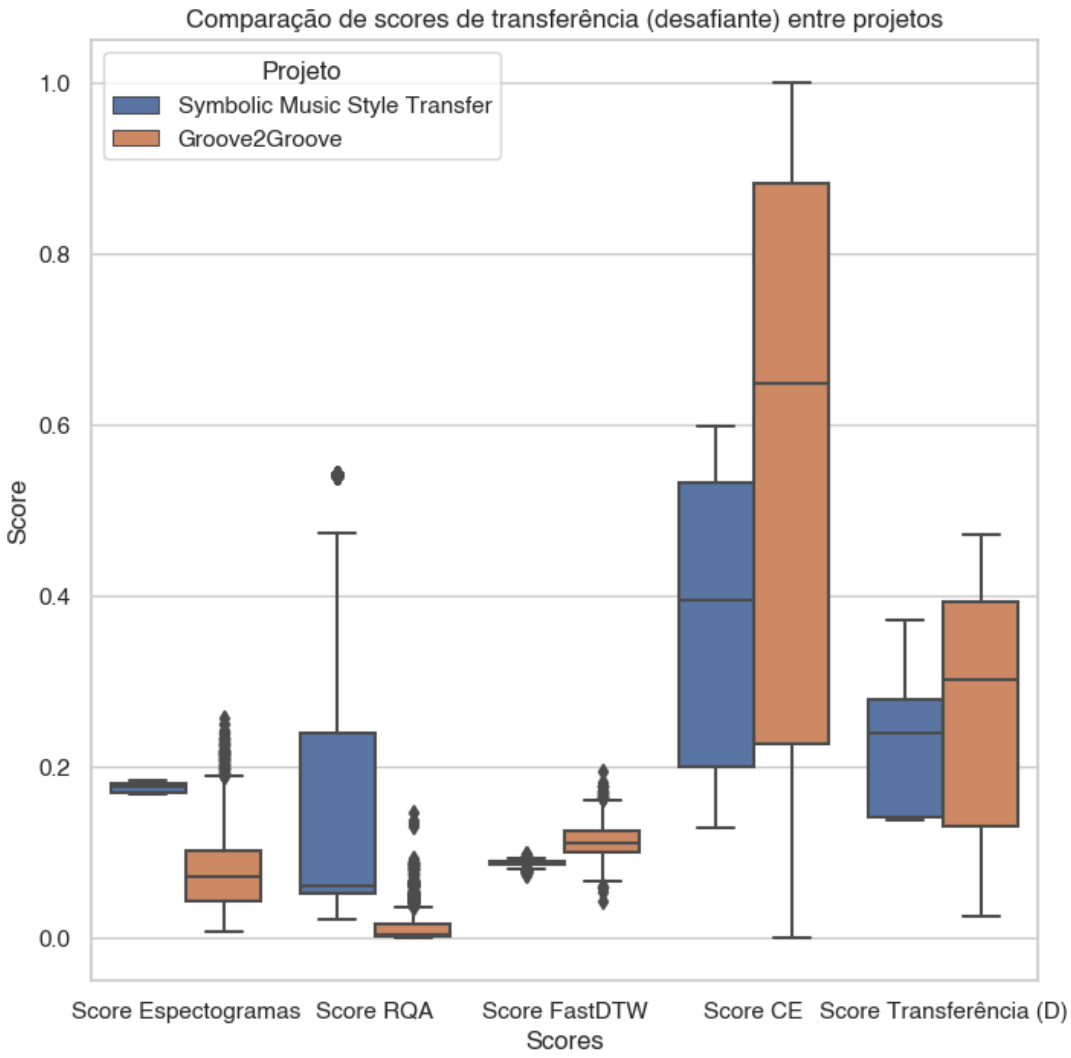


Figura 6.21: Distribuição dos *scores* de Transferência (desafiante) entre cada projeto

Pode-se observar que o projeto Groove2Groove continua tendo a melhor performance entre os dois projetos. Entretanto, a diferença entre os resultados é consideravelmente menor, além de um aumento no valor da relação da média e do desvio padrão (ou seja, o desvio padrão é proporcionalmente maior). Percebe-se que o *Score* de Ciclo é consideravelmente melhor para o projeto Groove2Groove, sendo o principal responsável pelo resultado.

6.5.3 Discussão

Com ambos os *scores* analisados, algumas observações podem ser realizadas:

- O projeto Groove2Groove apresenta os melhores resultados em ambos os casos.
- As mudanças do *score* Desafiante alteram drasticamente os resultados do score de Transferência, beneficiando o projeto Symbolic Music Style Transfer.
- De fato ocorre o enviesamento do *score* de Transferência pela distância entre os pares de arquivos comparados ser naturalmente alta.

Sendo assim, com base nos resultados encontrados, considera-se o projeto Groove2Groove como tecnicamente o melhor, apresentando mais robustez nas métricas desenvolvidas.

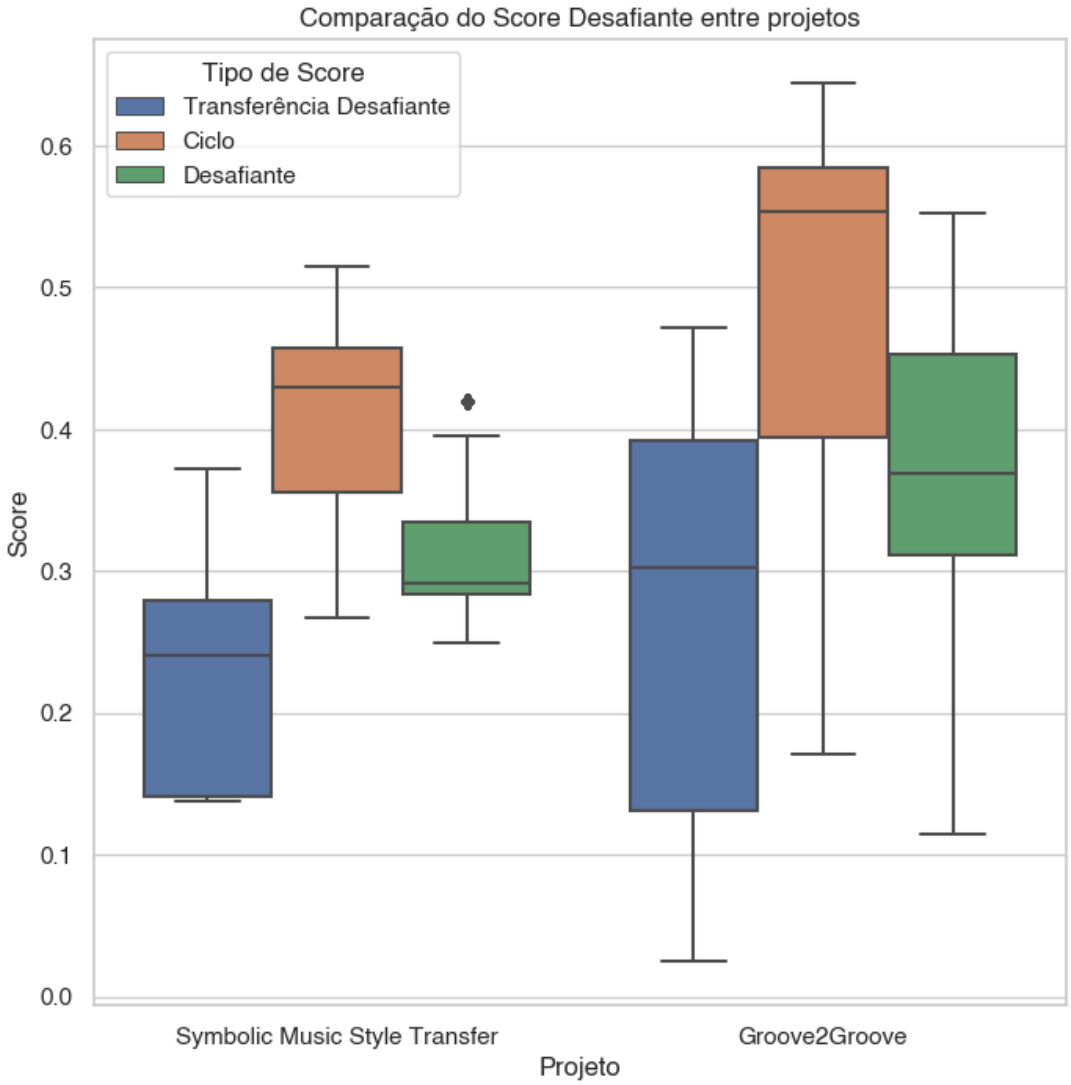


Figura 6.22: Distribuição dos *scores* de Transferência (desafiante), Ciclo e Desafiante entre cada projeto

6.6 VISUALIZAÇÃO DOS RESULTADOS DOS PROJETOS

Os projetos Symbolic Music Style Transfer e Groove2Groove, como supracitado, geram arquivos MIDI como resultado. Esta seção é destinada a visualização dos *piano rolls* que representam as saídas, bem como a disponibilização do áudio de tais arquivos.

Dentre as diversas saídas geradas pelos testes, escolheu-se um par específico contendo as músicas *Adele - Hello*, disponibilizada por Laurie (2015), e *Cheryl Lynn - Got To Be Real*, por Lynn (2016). A utilização deste par deu-se pela características das músicas, visto que ambas são conhecidas, com diversos instrumentos, variações melódicas e diferentes intensidades. Portanto, representa um caso real de transferência de estilo musical.

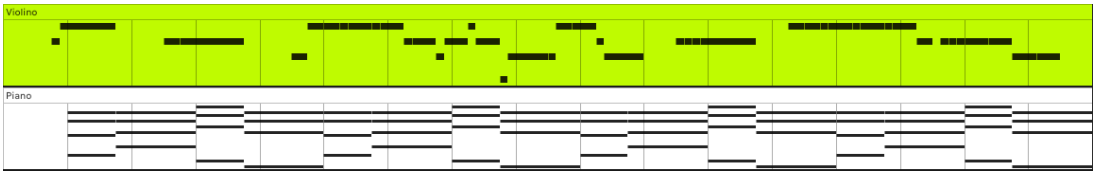


Figura 6.23: 16 primeiros compassos da representação MIDI da música *Adele - Hello*.

A música *Adele - Hello* está classificada na base de dados como uma música *Pop*. Na Figura 6.23 nota-se que o arquivo MIDI possui uma linha de violino e outra de piano, em sequência. Além disso, o áudio gerado por este MIDI, disponibilizado por Maruffa e Tha (2022a), é fiel à música original.

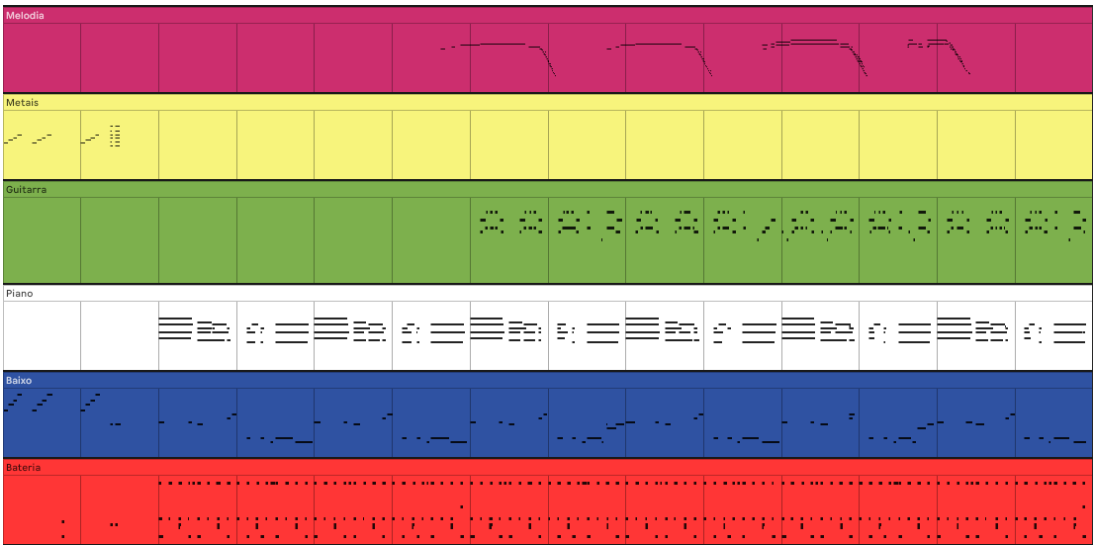


Figura 6.24: 14 primeiros compassos da representação MIDI da música *Cheryl Lynn - Got To Be Real*.

O MIDI da música *Cheryl Lynn - Got To Be Real*, mostrado na Figura 6.24, conta com 6 linhas: Melodia, Metais, Guitarra, Piano, Baixo e Bateria. Na base de dados do SMST esta música é classificada como *Jazz*. Assim como na música anterior, o áudio gerado a partir do MIDI, por Maruffa e Tha (2022b), é fiel à música original.

Utilizando estas duas músicas é possível visualizar os resultados de ambos os projetos testados.

6.6.1 Visualização do Groove2Groove

Partindo da explicação dada na seção 4.1, o projeto G2G necessita de uma entrada de conteúdo e uma de estilo. Neste caso, utiliza-se a música *Adele - Hello* como conteúdo, denotada a partir de agora como C, e *Cheryl Lynn - Got To Be Real* como estilo, referenciada como E.

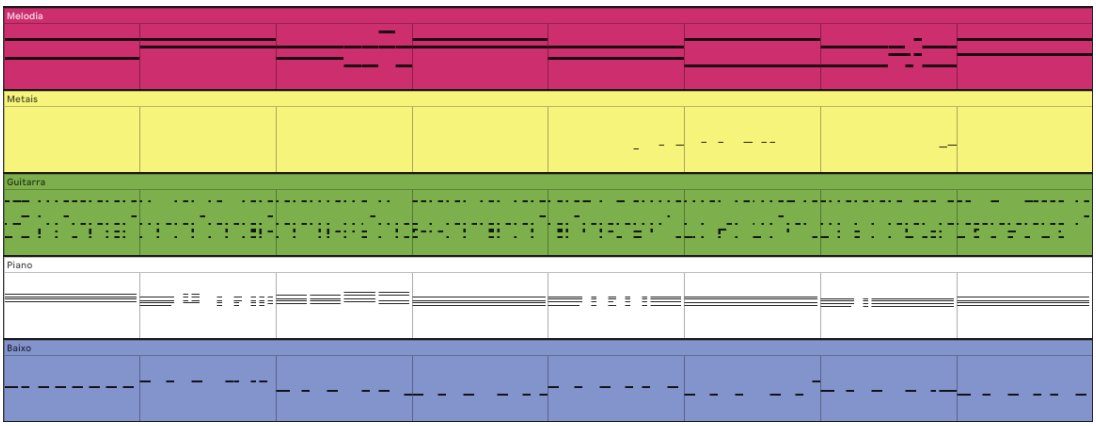


Figura 6.25: 8 primeiros compassos da representação MIDI da transferência de estilo gerada pelo G2G.

A Figura 6.25 mostra o resultado da transferência de estilo da entrada C com o estilo de E. Ressalta-se que a saída ilustrada contém todas as linhas de MIDI da entrada de estilo, com

exceção da bateria, visto que o G2G descarta a bateria em seu funcionamento. Este fato mostra que os instrumentos do acompanhamento foram mantidos na transferência, denotando a primeira parte do sucesso do procedimento.

Em um paralelo com a Figura 6.23, é notável que a sequência de acordes e estrutura harmônica, principalmente da linha de piano, não são idênticas, mas de forma geral apresentam um grau de semelhança considerável, denotando a preservação do conteúdo da entrada C.

Por outro lado, é notável a diferença presente na cadência dos instrumentos e na estrutura rítmica presente na Figura 6.25 em relação a Figura 6.23. Essa mudança ocorre devido a aplicação da entrada de estilo E. Em uma análise minuciosa, é possível perceber que tal variação rítmica é semelhante ao padrão existente na Figura 6.24, expondo um notável grau de sucesso na aplicação da entrada de estilo.

Verificando o áudio gerado pelo MIDI da transferência de C para E, disponibilizado por Maruffa e Tha (2023a), pode-se constatar que os fatos acima são contundentes. O resultado sonoro é uma combinação da sequência melódica e harmônica de C com o ritmo e acompanhamento de E. Dessa forma, é possível considerar que a transferência de estilo ocorreu com significativo sucesso.

Após a transferência, é necessário olhar para o ciclo resultante do G2G.

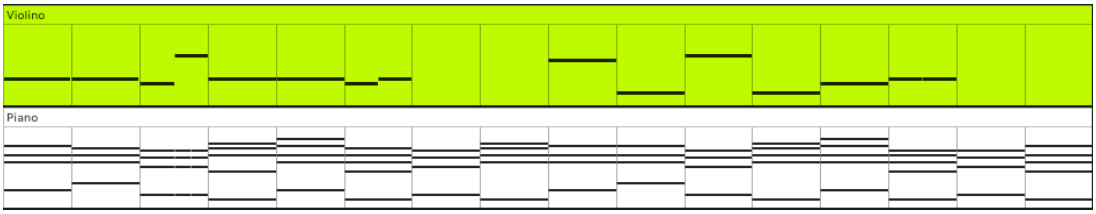


Figura 6.26: 16 primeiros compassos da representação MIDI do ciclo $C \rightarrow E \rightarrow C$ gerado pelo G2G.

A Figura 6.26 ilustra o resultado do ciclo $C \rightarrow E \rightarrow C$ gerado pelo Groove2Groove, isto é, o fruto da conversão da música *Adele - Hello* para o estilo de *Cheryl Lynn - Got To Be Real* (Figura 6.25) e então convertida novamente para o estilo de *Adele - Hello*. Idealmente, este ciclo deve ser o mais próximo possível do MIDI original (Figura 6.23).

Comparando as Figuras 6.26 e 6.23, observa-se que a quantidade de linhas MIDI é a mesma, sendo um ponto positivo da aplicação do estilo esperado. A linha de violino apresenta uma sequência de notas completamente diferente da original, mas por outro lado, a linha de piano apresenta a estrutura, sequência e cadência muito próximas da original, mesmo que com variações de notas e um resultado sonoro diferente. Avaliando o áudio do ciclo, disponibilizado por Maruffa e Tha (2023b), nota-se que a harmonia presente remete à original, mas no contexto geral é impossível dizer que as músicas são parecidas.

De maneira geral, analisando os artefatos produzidos, conclui-se que o G2G apresenta um resultado efetivo, porém qualitativamente razoável no processo de transferência de estilo.

6.6.2 Visualização do Symbolic Music Style Transfer

Diferentemente do G2G, o Symbolic Music Style Transfer necessita de apenas um arquivo de entrada, sendo que o estilo está definido internamente na estrutura do modelo a partir do processo de treinamento, vide a seção 4.1.

Dessa forma, nesta visualização será utilizado um modelo do Symbolic Music Style Transfer treinado para a transferência de *Pop* para *Jazz* e *Jazz* para *Pop*. A música escolhida como entrada foi *Adele - Hello* (Figura 6.23).

Ressalta-se que o Symbolic Music Style Transfer realiza o pré-processamento do arquivo MIDI antes de utilizá-lo como entrada do modelo. Neste processo, assim como explicado na seção 3.2.2, as múltiplas linhas de MIDI são mescladas em uma só, além das modificações ocorridas na métrica musical do arquivo. Com isso, o resultado da transferência e do ciclo serão, consequentemente, arquivos com uma única linha de MIDI.

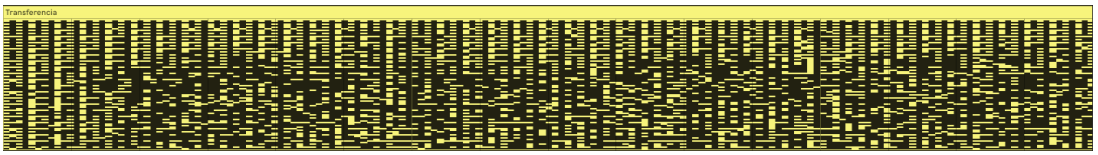


Figura 6.27: 16 primeiros compassos da representação MIDI da transferência de *Pop* para *Jazz* pelo SMST.

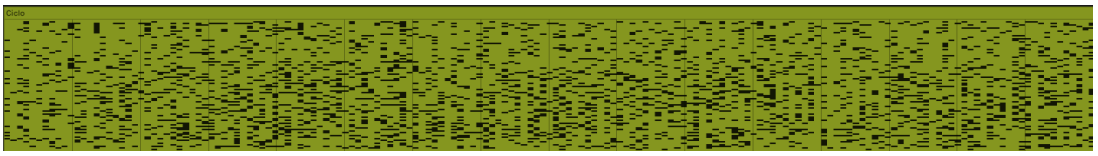


Figura 6.28: 16 primeiros compassos da representação MIDI do ciclo *Pop* → *Jazz* → *Pop* pelo SMST.

Na Figura 6.27 está representado o resultado da transferência de estilo da música *Adele - Hello* para *Jazz*. Do mesmo modo, na Figura 6.28 está a saída do ciclo *Pop* → *Jazz* → *Pop*. A partir daí é possível notar que os resultados gerados são arquivos MIDI com uma grande densidade de eventos consideravelmente aleatorizados. Portanto, ambos os arquivos geram áudios musicalmente ruins, que se assemelham a uma sequência aleatória de sons e não a uma música.

Acredita-se que esse fenômeno acontece pela consequência da aplicação do pré-processamento dos arquivos. Dado que as linhas de MIDI são unidas em uma única, julga-se que o resultado dessa conversão confunde todo o processamento do modelo, ocasionando uma saída sonoramente aleatória.

Porém, é possível notar que o resultado do ciclo se assemelha mais a cadência e métrica musical do arquivo original do que a saída da transferência. Isto mostra que a aplicação de estilo está ocorrendo, e que o modelo busca se aproximar da entrada original no processo de ciclo, mesmo que com resultados ruins.

6.7 ANÁLISE SUBJETIVA

Anteriormente são mostrados os pareceres sobre a qualidade dos resultados dos projetos a partir de um ponto de vista numérico e matemático. A análise subjetiva tem como objetivo adicionar a observação e avaliação humana nos resultados. Para isso, foram avaliados em torno de 25 resultados de ambos os projetos.

Sonoramente, o projeto Groove2Groove apresentou um resultado razoável para a transferência de estilo. É possível notar que a sequência de notas do *chord chart* da entrada de conteúdo é mantida na maioria dos casos, assim como a saída respeita os instrumentos, a interpretação e a métrica da entrada de estilo, assim como proposto. Porém, é perceptível que o resultado possui um certo grau de aleatoriedade e que não se assemelha completamente a uma música criada por um humano. Para o ciclo, o G2G também tem um resultado razoável, visto que a saída faz sentido musicalmente e também contém todos os elementos da música

original, mesmo apresentando uma significativa discrepância melódica, harmônica e estrutural em comparação à música original.

Por outro lado, as saídas de transferência e ciclo para o Symbolic Music Style Transfer não fazem sentido musical. Dessa forma, todos os resultados avaliados foram considerados ruins.

Por fim, conclui-se que o projeto Groove2Groove é musicalmente melhor do que o Symbolic Music Style Transfer, a partir de uma análise sonora subjetiva.

6.8 AVALIAÇÃO DAS HIPÓTESES

Na seção 4.3.3.3 foram levantadas 5 hipóteses de avaliação para auxiliar as análises dos resultados por meio de uma visão de alto nível. Aqui serão discutidas as hipóteses com base nos resultados apresentados pelos projetos:

1. O projeto Symbolic Music Style Transfer terá melhor desempenho que o Groove2Groove nas métricas de identidade

    - Para analisar as métricas de identidade é necessário considerar o *Score* de todas essas métricas para a transferência e para o ciclo.

Projeto	**Métrica**	**Valor**	**Média**
Groove2Groove	*Score* Espectrogramas	0.908	0,673
	Score RQA	0.983	
	Score FastDTW	0.128	
Symbolic Music Style Transfer	*Score* Espectrogramas	0.776	0,593
	Score RQA	0.910	
	Score FastDTW	0.094	

Tabela 6.19: Médias das métricas de identidade para a **transferência** agrupadas por projeto

Projeto	**Métrica**	**Valor**	**Média**
Groove2Groove	*Score* Espectrogramas	0.089	0,062
	Score RQA	0.015	
	Score FastDTW	0.084	
Symbolic Music Style Transfer	*Score* Espectrogramas	0.227	0,448
	Score RQA	0.206	
	Score FastDTW	0.913	

Tabela 6.20: Médias das métricas de identidade para o **ciclo** agrupadas por projeto

Nas Tabelas 6.19 e 6.20 estão representados os valores de *score* de identidade para a transferência e o ciclo, respectivamente. Com estes dados, obtêm-se a média geral de *score* da identidade por projeto, sendo **0,367** para o G2G e **0,520** para o SMST.

Com isso, nota-se que por uma margem bastante considerável o projeto SMST apresenta um valor mais alto. Entende-se que isso ocorre pela estrutura de uso da CycleGAN que funciona, fundamentalmente, gerando um objeto de ciclo. Com isso, a média dos valores de identidade do SMST para o ciclo é muito superior à do G2G, aumentando, consequentemente, a média geral. Portanto, esta hipótese é verdadeira.

2. O projeto Groove2Groove terá melhor desempenho que o Symbolic Music Style Transfer nas métricas de classificação de estilo

   - Para analisar as métricas de classificação é necessário considerar o *Score* de todas essas métricas para a transferência e para o ciclo.

Projeto	Métrica	Valor
Groove2Groove	*Score* CE	0.56
Symbolic Music Style Transfer	*Score* CE	0.368

Tabela 6.21: Médias das métricas de classificação para a **transferência** agrupadas por projeto

Projeto	Métrica	Valor
Groove2Groove	*Score* CE	0.713
Symbolic Music Style Transfer	*Score* CE	0.401

Tabela 6.22: Médias das métricas de classificação para o **ciclo** agrupadas por projeto

Nas Tabelas 6.21 e 6.22 estão representados os valores de *score* de classificação para a transferência e o ciclo, respectivamente. Com estes dados, obtêm-se a média geral de *score* da identidade por projeto, sendo **0,636** para o G2G e **0,384** para o SMST.

Com isso, nota-se que o Groove2Groove apresenta um *score* de classificação significativamente mais alto que o Symbolic Music Style Transfer. Acredita-se que isso ocorre por não existir pré-processamento na utilização do G2G. Dessa forma, os resultados contém características mais contundentes, facilitando o funcionamento do classificador e resultando em melhores classificações. Portanto, esta hipótese é verdadeira.

3. A performance das métricas oscilará dependendo dos estilos na direção da transferência

   - A hipótese provou-se verdadeira para certos *scores*, mas não para todos. Algumas instâncias onde tal diferença ocorre são:
     (a) Nas Figuras 6.4, 6.6, 6.12 e 6.14 que o Groove2Groove teve performance melhor em S^T_{CE} e S^C_{CE} no estilo *Classic*, enquanto o Symbolic Music Style Transfer teve performance melhor para o Jazz.
     (b) Na Tabela 6.6 o projeto Groove2Groove apresentou desempenho pior no estilo *Classic* na avaliação de S^T_{cv2}.
     (c) Na Tabela 6.7 o projeto Symbolic Music Style Transfer apresentou desempenho melhor no estilo *Classic* na avaliação de S^T_{rqa}.
     (d) Na Tabela 6.13 o projeto Symbolic Music Style Transfer apresentou desempenho melhor no estilo Jazz na avaliação de S^C_{rqa}.

4. O projeto Groove2Groove terá resultados musicalmente melhores do que o Symbolic Music Style Transfer

   - Levando em consideração a seção 6.7, pode-se concluir que, com base nas análises feitas, o projeto Groove2Groove apresenta resultados musicalmente superiores aos do Symbolic Music Style Transfer. Portanto, esta hipótese é verdadeira.

5. O projeto Groove2Groove será beneficiado pelo classificador treinado com D_O, enquanto Symbolic Music Style Transfer será favorecido pelo treinado com D_P

    - Para analisar quantidade de acertos do classificador por projeto é necessário considerar para a transferência e para o ciclo.

Projeto	Tipo	Dados do Classificador	Acertos
Groove2Groove	Transferência	D_O	**1439**
		D_P	1439
		D_M	904
	Ciclo	D_O	**1813**
		D_P	1419
		D_M	1793
Symbolic Music Style Transfer	Transferência	D_O	800
		D_P	**1166**
		D_M	800
	Ciclo	D_O	800
		D_P	**1008**
		D_M	400

Tabela 6.23: Acertos dos classificadores divididos por projeto, tipo e base de dados.

Analisando a Tabela 6.23, nota-se que o projeto Groove2Groove apresenta as maiores quantidades de acerto com os classificadores treinados pela base de dados D_O, tanto para os resultados de transferência quanto para os de ciclo. O mesmo fenômeno acontece com o Symbolic Music Style Transfer, mas com os dados de D_P. Acredita-se que isso acontece pela semelhança existente entre a saída dos projetos e os seus respectivos conjuntos de dados, fato que facilita o reconhecimento pelo classificador e consequente aumento da taxa de acerto. Portanto, está hipótese é verdadeira.

6.9 CONSIDERAÇÕES FINAIS DOS RESULTADOS

Na seção 6, foram apresentados e discutidos os resultados obtidos. Observou-se os resultados de diferentes classificadores, além da visualização de arquivos exemplos do processo da transferência de estilo. Os diversos tipos de *score* e as hipóteses traçadas anteriormente foram respondidas, além da avaliação subjetiva dos projetos. Optou-se pelo *score* Desafiante como o mais robusto, uma vez que os problemas que este se propõe a resolver realmente ocorreram.

A partir de todos os dados coletados e análises realizadas, o projeto Groove2Groove foi escolhido como o que apresenta melhores e mais robustos resultados. Entretanto, o projeto Symbolic Music Style Transfer apresenta bastante potencial caso melhorias sejam feitas, como a inclusão de vários instrumentos e uma redução nos efeitos do pré-processamento.

Para futuros trabalhos, considera-se importante o estabelecimento da constante ideal para as funções de identidade “ambíguas”. Caso o valor diferente na análise do projeto atual, os resultados poderiam ter tomado forma diferente, revelando outros aspectos dos projetos. Além disso, análises mais profundas do classificador poderiam ser exploradas, buscando maior robustez nos resultados.

7 CONCLUSÃO

7.1 DISCUSSÃO

O desenvolvimento do presente trabalho buscou entender o atual cenário da transferência de estilo musical com áudio simbólico, possibilitando a criação de métricas para a avaliação de projetos. Para permitir a comparação entre projetos, dois tipos de técnicas foram desenvolvidas, focando em diferentes aspectos da música: a Identidade e o Classificador de Estilo. Além disso, ressalta-se que os múltiplos classificadores de estilos para arquivos MIDI desenvolvidos possuem performance bastante robusta. Com os artefatos desenvolvidos para avaliação, foram comparados dois projetos: o Symbolic Music Style Transfer e o Groove2Groove, ambos projetos de transferência de estilo com técnicas consideravelmente diferentes. Para a análise final, 2400 casos de teste foram gerados, todos sendo avaliados pelas métricas desenvolvidas e gerando um *score* final. Diversas hipóteses a respeito da performance dos projetos quando avaliados pelas métricas foram estabelecidas, sendo todas investigadas e respondidas.

A análise dos projetos Symbolic Music Style Transfer e Groove2Groove mostrou-se bastante reveladora, permitindo uma investigação mais profunda dos resultados e maior compreensão de como os projetos operam. Destaca-se também a importância da avaliação subjetiva, uma vez que diversas características que determinam se uma música é boa estão além do que as técnicas atuais são capazes de identificar. Detalhes relevantes dos artefatos produzidos pelos projetos podem não ter sido identificados durante o desenvolvimento sem a constante atenção às produções da transferência de estilo.

As métricas desenvolvidas foram consideradas satisfatórias e robustas para a análise, uma vez que o proposto *Score Desafiante* foi capaz de se adaptar as peculiaridades das transformações e obter um resultado considerado suficiente para a avaliação dos projetos, definindo o G2G campeão.

Considerando a atual popularidade de técnicas de geração de arte usando inteligência artificial, o presente trabalho objetiva contribuir com a criação de métricas que podem ser aplicadas em outros projetos, auxiliando na padronização do ambiente de experimentação e comparação de resultados. Certamente, outras métricas podem ser desenvolvidas para a análise de outros aspectos da transferência.

7.2 TRABALHOS FUTUROS

Para pesquisas futuras, destaca-se a possibilidade da identificação do valor ideal, alvo para as métricas de identidade entre os projetos, especialmente nos casos onde não se deseja maximizar a similaridade. O limiar ótimo não é trivial e pode afetar consideravelmente os resultados de análises futuras, uma vez que pode determinar com mais clareza o sucesso de uma transferência de estilo.

Além disso, os classificadores de estilo desenvolvidos fazem uso de técnicas estatísticas para a aproximação da identificação do estilo original da música, não sendo perfeitos. Estudos futuros podem investigar a influência da não-exatidão do classificador, além de identificar possíveis enviesamentos com base nos estilos utilizados.

Trabalhos futuros podem abordar a comparação de outros projetos, além da otimização e melhora dos apresentados no presente trabalho. Estende-se também a ideia da inclusão de outros projetos que utilizam outras técnicas e formatos de representação musical, como arquivos

de áudio ou espectrogramas. Para tal, mudanças nas métricas para o uso de novos formatos seriam necessárias, abrindo a possibilidade para desenvolvimento de novas técnicas de avaliação.

REFERÊNCIAS

Ambrosius, W. (2007). *Topics in Biostatistics*. Humana.

Bahdanau, D., Cho, K. e Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

Baia, S. F. (2011). Partitura, fonograma e outros suportes: Fontes para a historiografia da mÚsica popular. Em *Anais do XXVI Simpósio Nacional de História*.

Best, H. e Wolf, C. (2014). *The SAGE Handbook of Regression Analysis and Causal Inference*. SAGE Publications.

Bhavsar, P., Safro, I., Bouaynaya, N., Polikar, R. e Dera, D. (2017). Chapter 12 - machine learning in transportation data analytics. Em Chowdhury, M., Apon, A. e Dey, K., editores, *Data Analytics for Intelligent Transportation Systems*, páginas 283–307. Elsevier.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Brownlee, J. (2020). How to develop a cyclegan for image-to-image translation with keras. `https://machinelearningmastery.com/cyclegan-tutorial-with-keras/`. Acessado em 15/02/2023.

Broze, Y. e Shanahan, D. (2013). Diachronic changes in jazz harmony: A cognitive perspective. *Music Perception: An Interdisciplinary Journal*, 31:32–45.

Brunner, G., Wang, Y., Wattenhofer, R. e Zhao, S. (2018). Symbolic music genre transfer with cyclegan. *CoRR*, abs/1809.07575.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T. et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. e Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65.

Cífka, O. (2020). Groove2groove. `https://github.com/cifkao/groove2groove/tree/master/experiments`.

Cífka, O., Şimşekli, U. e Richard, G. (2020). Groove2groove: One-shot music style transfer with supervision from synthetic data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2638–2650.

Dabakoglu, C. (2018). What is support vector machine (svm)? - with python. `https://medium.com/@cdabakoglu/what-is-support-vector-machine-svm-fd0e9e39514fhttps://medium.com/@cdabakoglu/what-is-support-vector-machine-svm-fd0e9e39514f`. Acessado em 15/02/2023.

Donoso, J. P. (2015). A física nos instrumentos musicais. https://www2.ifsc.usp.br/portal-ifsc/a-fisica-nos-instrumentos-musicais/. Acessado em 15/02/2023.

fastdtw (2015). Fastdtw is approximate and generally slower than the algorithm it approximates.

Fleck, L., Tavares, M. H. F., Eyng, E., Helmann, A. C. e Andrade, M. d. M. (2016). Redes neurais artificiais: Princípios básicos. *Revista Eletrônica Científica Inovação e Tecnologia*, 1(13):47–57.

Gatys, L. A., Ecker, A. S. e Bethge, M. (2015). A neural algorithm of artistic style. *CoRR*, abs/1508.06576.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. e Bengio, Y. (2014). Generative adversarial nets. Em Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. e Weinberger, K., editores, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Goródscy, F., Moura, S. e Queiroz, M. G. d. (2018). Audiotomidi similarity for music retrieval. Em *International Conference of Students of Systematic Musicology - SysMus*. UFMG.

Group, C. S. (2018). O que é uma floresta aleatória? https://www.tibco.com/pt-br/reference-center/what-is-a-random-forest. Acessado em 15/02/2023.

Haynie, D. (2018). Audio files explanation. https://www.quora.com/How-can-i-make-my-TV-play-an-audio-if-it-cant-recognize-the-file-type-of-the-audio. Acessado em 15/02/2023.

Kalita, D. (2022). Introdução ao perceptron passo a passo. https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/. Acessado em 15/02/2023.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. e Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Em Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. e Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kingma, D. P. e Ba, J. (2014). Adam: A method for stochastic optimization.

Korzeniowski, F., Sears, D. R. W. e Widmer, G. (2018). A large-scale study of language models for chord prediction.

Laurie, A. (2015). Adele - hello (official music video). https://youtu.be/YQHsXMglC9A. Acessado em 19/02/2023.

Le, J. (2019). Recurrent neural networks: The powerhouse of language modeling. https://builtin.com/data-science/recurrent-neural-networks-powerhouse-language-modeling. Acessado em 15/02/2023.

Li, Z., Liu, F., Yang, W., Peng, S. e Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019.

Lopes, P. (2019). Como ler partitura? https://www.musicdot.com.br/artigos/como-ler-partitura. Acessado em 15/02/2023.

Lorena, A. C. e de Carvalho, A. C. P. L. F. (2007). Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14(2):43–67.

Lynn, C. (2016). Cheryl lynn - got to be real (audio). https://youtu.be/fI569nw0YUQ. Acessado em 19/02/2023.

Maruffa, E. e Tha, G. (2022a). Adele - hello (Áudio do midi). https://on.soundcloud.com/3xeS5. Acessado em 19/02/2023.

Maruffa, E. e Tha, G. (2022b). Cheryl lynn - got to be real (Áudio do midi). https://on.soundcloud.com/18WB7. Acessado em 19/02/2023.

Maruffa, E. e Tha, G. (2023a). Adele - hello transferido para cheryl lynn - got to be real (Áudio do midi). https://on.soundcloud.com/jnAN9. Acessado em 19/02/2023.

Maruffa, E. e Tha, G. (2023b). Ciclo de tranferência para adele - hello e cheryl lynn - got to be real (Áudio do midi). https://on.soundcloud.com/9V9cx. Acessado em 19/02/2023.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E. e Nieto, O. (2015). librosa: Audio and music signal analysis in python. Em *Proceedings of the 14th python in science conference*, volume 8.

McKay, C. (2004). Automatic genre classification of midi recordings. Dissertação de Mestrado, McGill University.

McKay, C. (2018). jsymbolic 2.2 manual. https://jmir.sourceforge.net/manuals/jSymbolic_manual/home.html. Acessado em 19/02/2023.

McKay, C., Cumming, J. e Fujinaga, I. (2018). Jsymbolic 2.2: Extracting features from symbolic music for use in musicological and mir research. Em *International Society for Music Information Retrieval Conference*.

McKay, C. e Fujinaga, I. (2005). The bodhidharma system and the results of the mirex 2005 symbolic genre classification contest. *International Society for Music Information Retrieval*.

Mckay, C. e Fujinaga, I. (2010). jmir: Tools for automatic music classification.

Medsker, L. R. e Jain, L. (2001). Recurrent neural networks. *Design and Applications*, 5:64–67.

MIDI Manufacturers Association (1996). The complete MIDI 1.0 detailed specification: Incorporating all recommended practices.

Mitzenmacher, M. e Owen, S. (2001). Estimating resemblance of midi documents. Em *Workshop on Algorithm Engineering and Experimentation*, páginas 78–90.

Monard, M. C. e Baranauskas, J. A. (2003). Indução de regras e árvores de decisão. *Sistemas Inteligentes-fundamentos e aplicações*, 1:115–139.

Müller, M. (2007). *Information retrieval for music and motion.* Springer.

Pandala, S. R. (2019). Welcome to lazy predict's documentation!¶.

Pasini, M. (2019). Melgan-vc: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms.

Raffel, C. e Ellis, D. P. W. (2015). Large-scale content-based matching of midi and audio files. Em *International Society for Music Information Retrieval Conference*.

Rezende, S. O., Monard, M. C. e Carvalho, A. C. P. L. (1999). Sistemas inteligentes para engenharia: Pesquisa e desenvolvimento. Em *Anais III Workshop de Sistemas Inteligentes para Engenharia*, Belo Horizonte. Editora UFMG.

Richardson, I. E. G. ([2003]). *H.264 and MPEG-4 video compression : video coding for next generation multimedia*. Chichester ; Hoboken, NJ : Wiley, [2003] ©2003. Includes bibliographical references (page [277]) and index.

Rocha, J. C. (2017). Introdução ao perceptron passo a passo. `https://juliocprocha.wordpress.com/2017/07/27/perceptron-para-classificacao-passo-a-passo/`. Acessado em 15/02/2023.

Rossini, M. C. (2018). Novas técnicas geram síntese e efeitos de áudio. `https://aun.webhostusp.sti.usp.br/index.php/2018/06/12/novas-tecnicas-geram-sintese-e-efeitos-de-audio/https://aun.webhostusp.sti.usp.br/index.php/2018/06/12/novas-tecnicas-geram-sintese-e-efeitos-de-audio/`. Acessado em 15/02/2023.

Rui, L. R. e Steffani, M. (2007). Física: som e audição humana. Em *Simpósio Nacional de Ensino de Física*.

Salvador, S. e Chan, P. (2004). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11:70–80.

Serrà, J., Serra, X. e Andrzejak, R. G. (2009). Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11:093017.

Simoyama, C. (2019). Redes neurais. `https://pesquisa.ufabc.edu.br/lirte/coletivo_mina/redes-neurais/`. Acessado em 15/02/2023.

Soares, A. (2011). Composição Áudio e sonoplastia. `https://anasoares1.files.wordpress.com/2011/01/som_freq1.png`. Acessado em 15/02/2023.

Team, P. C. (2016). *Python: A dynamic, open source programming language*. Python Software Foundation. Python version 3.6.

Teng, Y., Zhao, A. e Goudeseune, C. (2017). Generating nontrivial melodies for music as a service. *CoRR*, abs/1710.02280.

Van der Maaten, L. e Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Wang, W., Chakraborty, G. e Chakraborty, B. (2021). Predicting the risk of chronic kidney disease (ckd) using machine learning algorithm. `https://www.mdpi.com/2076-3417/11/1/202`. Acessado em 15/02/2023.

Webber, C. e Marwan, N. (2015). *Recurrence Quantification Analysis – Theory and Best Practices*. Springer.

Wisnik, J. (1989). *O som e o sentido: uma outra história das músicas*. Companhia das Letras.

Wu, R. e Keogh, E. (2020). Fastdtw is approximate and generally slower than the algorithm it approximates. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1.

Xia, G., Huang, T., Ma, Y., Dannenberg, R. e Faloutsos, C. (2013). Midifind: Similarity search and popularity mining in large midi databases. Em *International Symposium on Computer Music Multidisciplinary Research*, páginas 259–276.

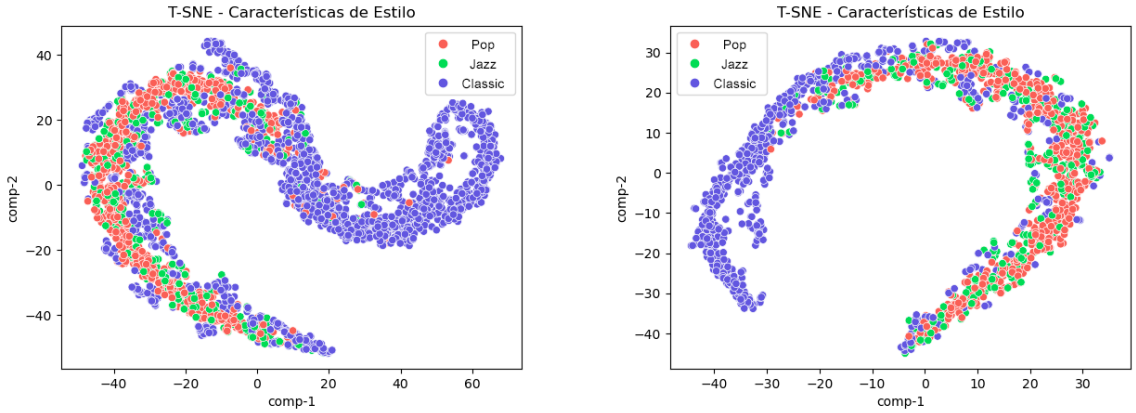
Zhu, J., Park, T., Isola, P. e Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593.

APÊNDICE A – T-SNE COM AS CARACTERÍSTICAS DO JSYMBOLIC

Utilizado amplamente para análise de dados, o T-SNE, abreviação de *T-Distributed Stochastic Neighbor Embedding*, é um algoritmo que permite "visualizar dados de grandes dimensões, dando a cada ponto dos dados uma localização em um mapa bi ou tridimensional" Van der Maaten e Hinton (2008). Em suma, seu funcionamento consiste na comparação da probabilidade de distância entre os pontos de alta dimensão, como forma de representá-los em dimensões menores, mas mantendo as relações mútuas e as propriedades estruturais do conjunto de dados original. Majoritariamente, o resultado produzido é uma representação visual dos dados em um gráfico de duas ou três dimensões, que pode ser usado para a identificação de padrões e *clusters* de dados.

Conforme apresentado, o conjunto de dados escolhido foi o do projeto Symbolic Music Style Transfer. Então, para o treinamento do classificador de estilo é necessário extrair as características de tais dados. Este processo foi realizado pela utilização do *software* jSymbolic, assim como descrito em 4.3.1.2, resultando em vetores de 232 posições. Portanto, a combinação dos vetores de todos os arquivos MIDI do conjunto resulta em um espaço 232-dimensional.

Como extensão deste trabalho, resolveu-se analisar brevemente como as características dos dados e as suas respectivas classes se relacionam. Para isso, torna-se propícia a utilização do T-SNE.



(a) Características de D_O

(b) Características de D_P

Figura A.1: Visualização das características com T-SNE

Na Figura A.1 estão representadas as visualizações dos conjuntos de dados D_O e D_P pela aplicação do algoritmo do T-SNE, sendo cada ponto a coordenada de uma entrada do conjunto correspondente após a redução da dimensionalidade e cada cor a identificação da classe daquela entrada.

Analisando a Figura A.1(a), pode-se notar que, no conjunto D_O, a quantidade de dados da classe *Classic* é maior do que as outras, assim como o esperado. Além disso, essa mesma classe possui dados distribuídos por todo o espectro de características, enquanto as outras classes não. As classes *Jazz* e *Pop* estão em uma região muito semelhante do mapa, fato que representa, de maneira geral, que os objetos dessas classes possuem muita semelhança nas características extraídas pelo jSymbolic. Essa análise corrobora, por exemplo, que o classificador 4.3.1.2 apresente menos confiança na predição destas duas classes quando comparado a classe *Classic*.

Na Figura A.1(b), temos o conjunto D_P, ou seja, dados pré-processados pelo Symbolic Music Style Transfer. De início, é perceptível que a quantidade de entradas diminuiu significativamente, mostrando que este pré-processamento descarta muitos dados. Além disso, o SMST transforma os dados em uma única linha de MIDI, assim como explicado em 3.2.2, fato que remove substancialmente diversas características dos dados, tornando-os mais homogêneos e menos distinguíveis. Isso é claramente ilustrado na Figura A.1(b), visto que os dados apresentam-se muito mais próximos e misturados, em comparação com A.1(a).

Essa menor diferenciação entre os dados apresenta consequências verdadeiramente impactantes nas análises dos projetos. Um exemplo claro é a taxa de acerto inferior do classificador de estilos para o conjunto D_P, mostrada em 6.1.

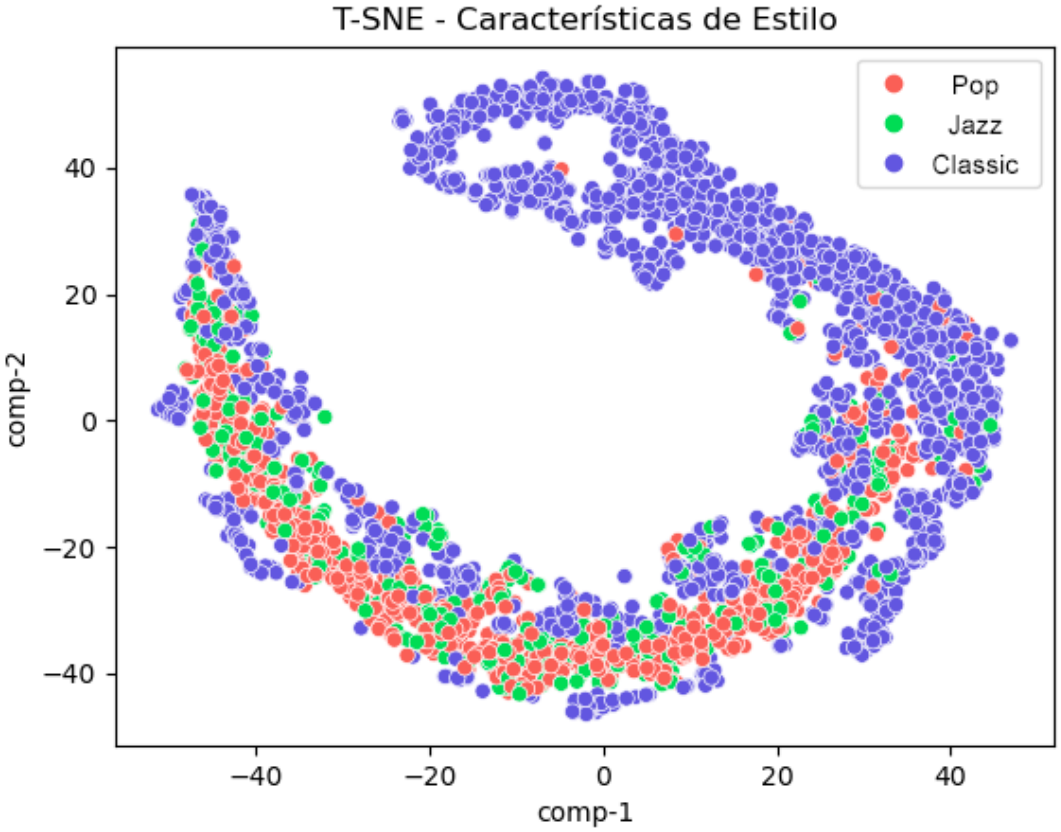


Figura A.2: Visualização das características do conjunto D_M com T-SNE

Na Figura A.2, temos a representação do conjunto D_M. Verifica-se que os dados estão mais densos, dada a maior quantidade de entradas. Porém, em comparação com A.1(a), é perceptível que os dados estão mais misturados e menos distintos, resultado da adição do conjunto D_P ao D_O.

Por fim, destaca-se que essas visualizações depende dos parâmetros de utilização do T-SNE, além de poderem não representar com total fidelidade as relações entre dos dados que existem no espaço multidimensional.