

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS FRANCESCO PICCIONI COSTA

**REDES NEURAIAS ARTIFICIAIS PARA GERAÇÃO DE ACORDES EM
MELODIAS MUSICAIS**

PONTA GROSSA

2022

LUCAS FRANCESCO PICCIONI COSTA

**REDES NEURAIAS ARTIFICIAIS PARA GERAÇÃO DE ACORDES EM
MELODIAS MUSICAIS**

Artificial Neural Networks for Musical Melodies Chord Generation

Dissertação apresentada como requisito para obtenção do título de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Hugo Valadares Siqueira

Coorientador: Prof^ª. Dr^ª. Marcella Scoczynski
Ribeiro Martins

PONTA GROSSA

2022



[4.0 Internacional](https://creativecommons.org/licenses/by/4.0/)

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.



Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa



LUCAS FRANCESCO PICCIONI COSTA

REDES NEURAIS ARTIFICIAIS PARA GERAÇÃO DE ACORDES EM MELODIAS MUSICAIS

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Ciência Da Computação da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Sistemas E Métodos De Computação.

Data de aprovação: 25 de Novembro de 2022

Dr. Hugo Valadares Siqueira, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Allan Kardec Duailibe Barros Filho, Doutorado - Universidade Federal do Maranhão (Ufma)

Elder Elisandro Schemberger, - Universidade Tecnológica Federal do Paraná

Dr. Erikson Freitas De Moraes, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 25/11/2022.

Dedico este trabalho a Deus,
porque d'Ele, e por Ele, e
para Ele, são todas as coisas.
Glória somente a Deus!

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu forças para conclusão deste trabalho, me trouxe consolo e renovo continuamente, me manteve firme e por tantas bênçãos mais que não se podem contar.

À minha noiva Ana Clara Gibim, que me apoiou em cada passo, foi paciente, amorosa e compreensiva em todo o tempo, onde eu encontrei e continuo a encontrar sempre o repouso, a paz, o ânimo, a alegria de viver e ser.

Aos meus pais João Batista e Elaine da Costa, porque tornaram tudo isso possível e realidade, supriram todas as necessidades desde o princípio, de tudo. Estiveram sempre presentes, com abundância de amor, carinho, consolo, “colo”, encorajamento e boa vontade, além de prover um lar, um ambiente tão precioso e ímpar.

Aos meus orientadores prof. Dr. Hugo Siqueira e prof^a. Dr^a. Marcella Martins, pela crença de que esse trabalho seria possível e por me aceitarem como seu orientando, me tratando de forma tão especial e cordial, sempre dispostos e atenciosos para que chegássemos até aqui, sendo de fato, orientadores.

Aos meus amigos, colegas e professores do curso de graduação em Engenharia de Computação da UTFPR Campus Toledo, que me encorajaram a seguir essa ideia e realizar este trabalho, tanto no início quanto na continuidade, dos quais cito alguns nomes: Daniel Jeronymo, Elder Schemberger, Felipe Stark, Fernando Hartwig e Lucas Wolff.

Ao meu primeiro orientador universitário, prof. Dr. Andrés Coca Salazar, que me ensinou como escrever, como pesquisar, como fazer ciência e muito mais no âmbito acadêmico e também na vida, com sua sabedoria calma e assertiva.

À Universidade Tecnológica Federal do Paraná, por ser ambiente da minha formação acadêmica e por oportunizar, com sua excelente estrutura física e corpo de docentes tanto nos Campus de Toledo e Ponta Grossa, a execução e conclusão deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, processo 88887.613066/2021-00, pelo apoio financeiro provido.

“Bem-aventurado o homem que
acha sabedoria, e o homem que
adquire conhecimento.”
(BÍBLIA, Provérbios, 3, 13).

RESUMO

Harmonia pode ser definida como a arte de unir conjuntos de sons musicais com objetivo de emoldurar uma melodia. E como em toda arte, é necessário tempo, dedicação, estudo e experiência para que seja possível elaborar harmonias musicais coerentes, uma tarefa complexa para iniciantes. Para tanto, um sistema capaz de automatizar este processo é de grande valia no auxílio de estudantes de harmonia e composição, tanto novatos quanto experientes, uma vez que podem encontrar novas formas de construir estruturas musicais. A literatura apresenta diversas alternativas de solução como métodos computacionais baseados em regras usando algoritmos evolutivos e também aplicando Redes Neurais Artificiais. Trabalhos que empregaram Redes LSTM Bidirecionais a partir de dados simbólicos de melodias padronizadas, atingiram resultados consideráveis levando em conta a quantidade de classes possíveis. Não obstante, este trabalho propõe novas abordagens ao problema de harmonização musical automática, com alvo em resultados que sejam mais precisos que os encontrados na literatura e diversificados mediante a harmonias preexistentes, além de apresentar uma nova forma de representação musical usando imagens, e uma arquitetura inovadora denominada Sistema Harmonizador Inteligente. Sendo assim, uma melodia descrita em formato simbólico é processada para servir como entrada ao sistema, que consiste de uma Rede Neural Convolutiva, treinada com base em dados de melodias e harmonias originais. Deste modo, tem-se na saída uma harmonia que esteja dentro dos objetivos propostos. A análise dos resultados deu-se com uso de instrumentos estatísticos, que mostraram taxas razoáveis de acerto, acima do que encontrado na literatura, estabelecendo um novo marco. O sistema resultante apresenta vantagens tanto para o campo da música, com utilidade para músicos de todos níveis de conhecimento e também ao estudo teórico da música, quanto para a ciência da computação, contribuindo com o desenvolvimento de codificações de estruturas musicais e a construção de arquiteturas de Redes Neurais inovadoras.

Palavras-chave: teoria musical; sistemas inteligentes; avaliação de performance.

ABSTRACT

Harmony can be defined as the art of combining sets of musical sounds in order to frame a melody. Furthermore, as in any form of art, it takes time, dedication, study, and experience to make coherent musical harmonies possible, a complex task for beginners. A system capable of automating this process is of great value in helping harmony and composition students, both beginners and experienced since they can find new ways to build musical structures. The literature presents several solution alternatives, such as computational methods based on rules, evolutionary algorithms, and applying Artificial Neural Networks. Works that used Bidirectional LSTM Networks from symbolic data of standardized melodies achieved considerable results considering the number of possible classes. This work proposes new approaches to the problem of automatic musical harmonization, aiming at results that are more accurate than those found in the literature and diversified through preexisting harmonies, in addition to presenting a new form of musical representation using images and an innovative architecture system called Intelligent Harmonizer System. Thus, a melody described in symbolic format is processed to serve as input to the system, which consists of a Convolutional Neural Network trained based on data from original melodies and harmonies. In this way, there is harmony at the exit within the proposed objectives. The results were analyzed using statistical instruments, which showed reasonable success rates above what is found in the literature, establishing a new milestone. The resulting system presents advantages both for the field of music, useful for musicians of all levels of knowledge and theoretical study of music, and computer science, contributing to the development of musical structures encodings and the construction of innovative neural network architectures.

Keywords: musical theory; intelligent systems; performance evaluation.

LISTA DE FIGURAS

Figura 1 – Quadro comparativo da divisão proporcional dos valores das figuras rítmicas desde a semibreve até a semifusa.	24
Figura 2 – As sete notas musicais representadas na pauta usando a clave de sol e os seus respectivos nomes e intervalos de tons (T) e semitons (S).	24
Figura 3 – Os sete acordes musicais disponíveis a partir da escala de Dó maior e suas respectivas cifras e graus.	27
Figura 4 – Tríades dispostas no círculo de terças, rotuladas por funções harmônicas.	28
Figura 5 – Esquema simplificado de um neurônio biológico e sua organização em dendritos, corpo celular, núcleo, axônio e terminais axônicos.	30
Figura 6 – O neurônio artificial de McCulloch e Pitts. Duas entradas são somadas e seu resultado passa por uma função de ativação antes da saída.	31
Figura 7 – Esquema de construção de um neurônio artificial para x_k entradas que são multiplicadas por pesos sinápticos antes de passar pela junção somadora e função de ativação, até a saída.	32
Figura 8 – Diversas funções de ativação nomeadas individualmente.	33
Figura 9 – Uma MLP de quatro camadas: a Camada de Entrada, que não realiza nenhuma operação servindo apenas de entrada para os dados, a Primeira e Segunda Camada Intermediária ou camadas escondidas, e a Camada de Saída, contendo neste caso, três neurônios.	36
Figura 10 – Operação do <i>backpropagation</i> : primeiro acontece a propagação da entrada ilustrado como sendo da esquerda para a direita, e depois a propagação do erro no sentido oposto.	37
Figura 11 – Derivadas de três funções de ativação diferentes.	43
Figura 12 – Arquitetura da CNN idealizada por LeCun <i>et al.</i> (1998), onde uma imagem na entrada de 32×32 <i>pixels</i> sofre sucessivas convoluções e subamostragens, até no final passar por três camadas totalmente conectadas.	46

Figura 13 – Processo aplicado na camada convolucional: um conjunto de entradas x (em azul) é multiplicado pelos parâmetros w do filtro de acordo com sua posição e em seguida integrados em u que passa por uma função de ativação chegando na saída como y . O mesmo acontece para o outro conjunto de entradas (em alaranjado), porém passando pelos mesmos parâmetros do filtro, o que explica a condição de compartilhamento de parâmetros.	47
Figura 14 – Operação de subamostragem usando a função de máximo, onde o máximo de cada quadrado colorido da esquerda é selecionado para compor a saída.	48
Figura 15 – Ilustração do efeito de <i>dropout</i> em uma rede de duas camadas escondidas, onde alguns nós são excluídos aleatoriamente.	49
Figura 16 – Ilustração da otimização Bayesiana de uma função unidimensional, com previsões mostradas como uma linha preta sólida e o modelo substituto como linha tracejada. A área azul representa a incerteza, enquanto que a área alaranjada é a função de aquisição.	51
Figura 17 – Níveis subjetivos de concordância para diferentes intervalos de valores de κ	57
Figura 18 – Diagrama de blocos explicativo da metodologia proposta neste trabalho.	62
Figura 19 – Trecho da música <i>Amazing Grace</i> , de John Newton, em notação musical.	63
Figura 20 – Compasso musical de exemplo escrito em notação musical.	65
Figura 21 – Módulo “ <i>inception</i> ” existente na arquitetura da CNN <i>GoogLeNet</i>	69
Figura 22 – Porcentagem de notas (também representadas pelo sistema de letras), acordes maiores e menores existentes na totalidade da base de dados.	72
Figura 23 – Análise da harmonia original e gerada pelos modelos de RNA propostos para os primeiros oito compassos da música <i>America</i>	75
Figura 24 – Matriz de confusão normalizada para classificação de acordes usando o modelo MLP2.	76
Figura 25 – Componentes responsáveis por formar uma imagem RGBA. O valor de cada componente, quando unido em um conjunto ou vetor, forma um <i>pixel</i> de coloração específica.	77

Figura 26 – Exemplo de imagem que faz uso das componentes RGBA, com a componente alfa enfatizada: sua intensidade é máxima no topo da figura e decresce até ser nula na base.	78
Figura 27 – Composição da Imagem de Compasso (IC): na vertical, cada <i>pixel</i> representa uma nota musical, enquanto que na horizontal se tem a duração, sendo que <i>pixels</i> azuis são a reprodução continuada de uma determinada nota e <i>pixels</i> vermelhos mostram a sua finalização.	78
Figura 28 – Equivalência de duração entre três quiálteras de colcheia (tercina) e duas colcheias normais, ambas duram o mesmo que uma semínima. . .	79
Figura 29 – Compasso musical de exemplo para demonstrar construção da IC. . . .	81
Figura 30 – Passo a passo de exemplificação de construção da IC, dividido em quatro etapas: indicação da primeira nota, pausa, apojatura e a figura completa.	83
Figura 31 – Compasso de exemplo e IC, com detalhes da sua construção: (1) a aplicação de <i>pixels</i> de cor azul e vermelha, (2) a resolução da figura, (3) a diferença de intensidade de cor, (4) a possibilidade de representar apojaturas e (5) o uso do canal alfa.	84
Figura 32 – Arquiteturas que podem ser formadas a partir da variação de hiperparâmetros. O número de filtros (2) aparece nas camadas convolutivas, podendo ter quantidade igual a duas ou quatro vezes seu valor, a depender da profundidade da rede (3), que por sua vez é o encadeamento de camadas adicionais. A camada densa, com borda pontilhada vermelha, pode ou não ser inserida (4) e tem total de neurônios definido pela configuração 5.1, enquanto que a camada densa da sequência tem neurônios definidos por 5.2. A configuração de função de ativação (1), embora não esteja ilustrada, é definida para todas as camadas convolucionais e densas, a não ser pela camada de saída, que será sempre de tamanho 24, e ativação <i>softmax</i> , devido ao número de classes considerado.	88
Figura 33 – Gráfico de coordenadas paralelas onde cada eixo representa as configurações de hiperparâmetros testadas, e as linhas são execuções completas. Linhas mais escuras estão relacionadas com menor valor de ECC de validação.	89

Figura 34 – Gráfico de coordenadas paralelas para a segunda etapa de testes do processo de otimização.	91
Figura 35 – Gráfico de coordenadas paralelas, referente ao segundo teste do processo de otimização, considerando somente a profundidade da rede (3), quantidade de neurônios da camada densa (5.2) e η (7).	92
Figura 36 – Arquitetura final resultante do processo de otimização e após adaptações manuais denominada de SIH.	94
Figura 37 – Análise da harmonia original e gerada pela MLP2 da Seção 8.2 e pela SIH para os primeiros oito compassos da música <i>America</i>.	95
Figura 38 – Matrizes de confusão resultante da MLP2 e SIH em caráter comparativo, geradas com resultados da melhor execução de cada.	95
Figura 39 – Pôster apresentado durante congresso internacional <i>Music Encoding Conference</i>.	134

LISTA DE TABELAS

Tabela 1 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para cada modelo testado. A quantidade de neurônios é mostrada entre parênteses na coluna de Experimento.	74
Tabela 2 – Classificação geral usando o método de contagem Borda considerando diferentes métricas e resultados de algoritmos.	74
Tabela 3 – Configurações das arquiteturas de CNN usadas durante a experimentação, como definidas pela literatura. A primeira linha (Qtde. Camadas) se refere ao total de camadas convolucionais e densamente conectadas, ou seja, somente camadas que tenham parâmetros ajustáveis.	86
Tabela 4 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para cada modelo de CNN testado. A última linha traz os resultados do melhor modelo usando entrada vetorial (MLP2) obtido na experimentação da Seção 8.2, a título de comparação.	86
Tabela 5 – Classificação geral dos modelos de CNN a partir do método de contagem Borda para as diferentes métricas aplicadas.	87
Tabela 6 – Hiperparâmetros considerados para processo de otimização, bem como as variações possíveis de configuração para a primeira etapa de testes.	89
Tabela 7 – Correlação de Kendall calculada para cada hiperparâmetro testado na primeira etapa do processo de otimização e configuração que trouxe a melhor execução.	90
Tabela 8 – Hiperparâmetros considerados na segunda etapa da otimização, correlação de Kendall resultante e configurações da melhor execução.	91
Tabela 9 – Hiperparâmetros considerados para a terceira etapa da otimização, resultados de correlação de Kendall e configurações da execução de melhor retorno.	92
Tabela 10 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para o teste da arquitetura otimizada, <i>AlexNet</i> sem alteração e MLP2 com entrada vetorial.	93

Tabela 11 – Média de ECC e porcentagem média de AC, F_{1M}, MCC e κ para cada adaptação manual, comparado com o teste da arquitetura otimizada, <i>AlexNet</i> e MLP2 com entrada vetorial.	94
---	-----------

LISTA DE QUADROS

Quadro 1 – Nomes dos intervalos, forma de representação, quantidade de semitons que contêm e exemplo de aplicação com relação à nota Dó.	25
Quadro 2 – Principais escalas musicais, estrutura de tons e semitons, e exemplo de construção.	26
Quadro 3 – Modos possíveis para as tríades, a regra para sua composição e exemplo montado a partir da nota Dó como tônica.	27
Quadro 4 – Construção de uma matriz de confusão binária com métricas relativas calculadas utilizando índices diretamente da matriz.	53
Quadro 5 – Matriz de confusão para o caso multiclasse para um total de \mathcal{C} classes.	54
Quadro 6 – Conteúdo do trecho da música <i>Amazing Grace</i> , da Figura 19, dentro do banco de dados <i>CSV Leadsheet</i>	63
Quadro 7 – Durações das figuras rítmicas de acordo com a padronização da base de dados selecionada, comparada com a duração convencional, que são frações da semibreve.	64
Quadro 8 – Acordes considerados durante a tarefa de classificação do sistema: tríades maiores e menores.	65

LISTA DE ABREVIATURAS E SIGLAS

Siglas

AC	Acerto ou Acurácia
AG	Algoritmo Genético
BLSTM	Bidirectional Long Short-Term Memory
CNN	<i>Convolutional Neural Network</i>
CSV	<i>Comma Separated Values</i>
EC	Entropia Cruzada
ECC	Entropia Cruzada Categórica
ELM	<i>Extreme Learning Machine</i>
ELU	<i>Exponential Linear Unit</i>
ESN	<i>Echo State Network</i>
FN	Falso negativo
FP	Falso positivo
IA	Inteligência Artificial
IC	Imagem de Compasso
MCC	Correlação de Coeficiente de Matthews
MIDI	<i>Musical Instrument Digital Interface</i>
MIR	<i>Music Information Retrieval</i>
MLP	<i>Multilayer Perceptron</i>
MPPI	Moore-Penrose Pseudo-Inversa
MSE	<i>Mean Squared Error</i>
RBF	<i>Radial Basis Function</i>
RGB	<i>Red, Green, Blue</i>
RGBA	<i>Red, Green, Blue, Alpha</i>

RNA	Rede Neural Artificial
SIH	Sistema Inteligente Harmonizador
SGD	<i>Stochastic Gradient Descent</i>
VN	Verdadeiro negativo
VP	Verdadeiro positivo
XML	<i>Extensible Markup Language</i>

LISTA DE SÍMBOLOS

Letras Latinas

FC	Fórmula de compasso
\dot{d}	Vetor de durações de um compasso
$\dot{d}^{(N)}$	Vetor de durações normalizadas de um compasso
x	Entrada de um neurônio
u	Produto integrador das entradas de um neurônio
w	Peso sináptico
y	Saída de um neurônio
d	Saída desejada
t	Instante de tempo
$f(\cdot)$	Função de ativação
$E(\cdot)$	Função de custo
F	Medida F

Letras Gregas

\mathcal{C}	Número total de classes de uma base de dados
α	Valor do canal alfa de uma imagem RGBA
η	Taxa de aprendizado

Notações

\sharp	Sustenido
\flat	Bemol
\boxplus	Bequadro
C	Clave de Sol
F	Clave de Fa
C	Clave de Dó
\circ	Semibreve
\downarrow	Mínima
\downarrow	Semínima
\curvearrowright	Colcheia

SUMÁRIO

1	INTRODUÇÃO	20
1.1	Objetivos	21
1.1.1	Geral	21
1.1.2	Específicos	22
1.2	Estrutura do Trabalho	22
2	TEORIA MUSICAL BÁSICA	23
2.1	Teoria de Harmonia	25
3	REDES NEURAIS ARTIFICIAIS	29
3.1	Neurônio Artificial	30
3.2	Funções de Ativação	32
3.3	Regra de Aprendizado	35
3.4	Perceptron de Múltiplas Camadas	35
3.5	Treinamento da Rede	37
3.5.1	<i>Backpropagation</i>	38
3.5.2	Métricas de Erro	40
3.5.3	Otimizadores	41
3.5.4	Problema da Superfície Plana	42
3.6	Aprendizado, Generalização e Validação Cruzada	43
4	APRENDIZADO PROFUNDO	45
4.1	Rede Neural Convolutacional	45
4.1.1	Camada Convolutacional	46
4.1.2	Camada de Subamostragem	47
4.2	Questões Práticas para Aplicação de Redes Neurais	48
4.2.1	<i>Dropout</i>	48
4.2.2	Normalização em Lotes	49
4.2.3	Otimização de Hiperparâmetros	50
5	MEDIDAS DE DESEMPENHO	53
5.1	Matriz de Confusão e Métricas Relacionadas	53
5.1.1	Acurácia	55
5.1.2	Medida F	55

5.1.3	Correlação de Coeficiente de Matthews	56
5.1.4	Coeficiente de Concordância de Kappa	57
6	ESTADO DA ARTE	59
7	METODOLOGIA	62
7.1	Descrição da Base de Dados	62
7.2	Padronização da Base de Dados	64
7.2.1	Padronização de Informações Melódico-Harmônicas	64
7.2.2	Padronização de Informações Rítmicas	65
7.3	Codificação das Entradas e Saídas	66
7.4	Configuração das Redes Neurais Artificiais	67
7.5	Análise de Resultados	70
7.5.1	Tesde de Friedman	70
7.5.2	Método de Contagem Borda	70
7.5.3	Correlação de Classificação de Kendall	71
8	RESULTADOS	72
8.1	Análise da Base de Dados	72
8.2	Comparação de Desempenho Entre Redes Neurais e Ensembles Para Entrada Vetorial	73
8.3	Representação de Compassos Melódicos por Meio de Imagens	77
8.4	Aplicação de Arquiteturas de Aprendizado Profundo Usando a Imagem de Compasso Como Entrada	84
8.4.1	Comparação de Arquiteturas da Literatura	85
8.4.2	Otimização de Hiperparâmetros	87
8.4.2.1	<u>Primeira Etapa</u>	88
8.4.2.2	<u>Segunda Etapa</u>	90
8.4.2.3	<u>Terceira Etapa</u>	91
8.4.3	Avaliação da Arquitetura Final	93
9	CONCLUSÃO	96
	REFERÊNCIAS	97
	APÊNDICE A ARTIGO SUBMETIDO A REVISTA <i>APPLIED AR- TIFICIAL INTELLIGENCE</i>	107

APÊNDICE B	ARTIGO SUBMETIDO AO CONGRESSO <i>MUSIC</i>	
	<i>ENCODING CONFERENCE</i>	127

1 INTRODUÇÃO

Quando se trata de teoria musical, um aspecto importante é o estudo da harmonia. Tem-se por som harmônico a união de duas ou mais notas musicais reproduzidas simultaneamente. O conjunto de vários destes sons organizados e ordenados define uma harmonia, cuja função é de emoldurar e acompanhar uma melodia (GUEST, 2010).

Entretanto, construir uma harmonia que atenda a essa função não é uma tarefa trivial. Requer por parte do músico compositor conhecimento aprofundado sobre teoria musical, em especial teoria de harmonia, além de tempo e experiência (CHEDIAK, 1986). Analisando a tarefa de harmonização como um processo, pode-se definir como a busca de uma harmonia para certa melodia dentre várias opções existentes cuja quantidade é geralmente limitada por conjuntos de regras e convenções estéticas (NOUGUÉ, 2011).

Dessa forma, apresenta-se como viável a construção de um sistema capaz de automatizar tal processo, apresentando contribuição no auxílio de estudantes de teoria musical, bem como músicos experientes, para que seja possível encontrar formas inovadoras de harmonizar músicas. Segundo Makris, Kayrdis e Sioutas (2013), a harmonização é extensão e aplicação natural da análise harmônica. A pesquisa dos passos que constituem este processo e uma forma de automatização apresenta contribuição valiosa para os campos de estudo de IA e Recuperação de Informação Musical (MIR).

Ao observar o problema de harmonização objetivando a construção de um acompanhamento (harmonia funcional ou ainda “popular”), a abordagem aplicada é a de gerar acordes preocupando-se com a função de cada um, com estruturação flexível principalmente no que diz respeito às regras de condução de vozes, sendo que não necessariamente deve acompanhar as notas da melodia (SANTOS, 2012), o que não acontece na harmonia tradicional. Esta, por sua vez, tem por tarefa a harmonização de quatro partes em que constrói-se uma sequência de acordes condizente com a melodia ou as vozes (MAKRIS; KAYRDIS; SIOUTAS, 2013). De forma geral, pode-se determinar que a tarefa de harmonizar uma melodia é caracterizada como sendo de classificação, isto é, encontrar qual o melhor acorde para uma determinada sequência de notas musicais.

Nesse sentido, diversas tentativas de solução computacional podem ser encontradas, frequentemente focados em métodos baseados em regras ou usando algoritmos evolutivos. Rybnik e Homenda (2012) elaboraram um modelo de harmonização baseado em conhecimento e regras de teoria musical, no qual um fragmento melódico passa por camadas de processamento que decidem qual acorde melhor servirá, com resultados analisados por especialistas, pois afirmam que o nível de acerto de tal sistema não pode ser determinado de forma algorítmica. Por outro lado, Wiggins *et al.* (1998) apresentaram uma discussão sobre o uso de Algoritmos Genéticos (AGs) para a geração de harmonias de quatro vozes a partir de uma melodia especificada, interessados na simulação de comportamento humano durante a atividade composicional, além da obtenção de resultados musicais. Além destes, uma comparação de desempenho realizada por

Phon-Amnuaisuk e Wiggins (1999) mostrou diferenças entre o uso de sistemas de regras e AGs, tendo por conclusão uma relação entre qualidade e quantidade, ou seja, quanto maior a quantidade de conhecimento que o sistema possui sobre o tema, melhor a qualidade dos resultados.

Além destes métodos, também é possível encontrar abordagens usando Redes Neurais Artificiais (RNAs) na tentativa de solução deste problema, como é o caso do trabalho de Lim e Lee (2017), que fizeram uso de LSTM Bidirecionais (BLSTM) a partir de dados simbólicos de melodias padronizadas, atingindo resultados consideráveis levando em conta a quantidade de classes de acordes possíveis. Os trabalhos de Huang e Wu (2016), Dong *et al.* (2018) e Liu e Yang (2018) são outros exemplos que usam essa metodologia, porém com o objetivo de gerar músicas inteiras, desde a melodia e a harmonia, até o arranjo para a execução.

A proposta deste trabalho é utilizar RNAs como um sistema harmonizador. Para tanto, uma melodia descrita em formato simbólico é padronizada e processada de forma a conter o máximo de informação musical possível. Esta serve como entrada para a RNA, que foi treinada com base em dados anteriores de melodias completas, acompanhadas de suas harmonias originais. A saída do sistema é um conjunto que harmoniza a melodia de entrada. Aplicam-se arquiteturas diferentes de RNAs e comparam-se os respectivos desempenhos, considerando principalmente Perceptron de Múltiplas Camadas (*Multilayer Perceptrons* - MLPs) e Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNNs). Para analisar se os objetivos propostos foram atingidos, os resultados são examinados de maneira quantitativa, por meio de indicadores estatísticos e medidas de desempenho.

1.1 Objetivos

Objetivos a serem alcançados foram definidos para a resolução do problema deste estudo, divididos em um objetivo geral e três objetivos específicos.

1.1.1 Geral

O objetivo geral deste estudo é desenvolver um sistema automático de harmonização melódica por meio de RNAs, considerando dados existentes de músicas populares.

1.1.2 Específicos

Os objetivos específicos são:

- Atingir resultados mais precisos que os encontrados na literatura;
- Encontrar soluções diversificadas mediante a harmonias já existentes;
- Elaborar uma nova arquitetura de RNA que seja capaz de harmonizar melodias musicais.

1.2 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 estabelece conceitos básicos de teoria musical e de harmonia, necessários para compreensão de demais aspectos do trabalho. No Capítulo 3 é feito o referencial teórico das RNAs, continuando no Capítulo 4 dedicado às CNNs e o tema de aprendizado profundo. Na continuidade, diversas medidas estatísticas úteis para determinar a qualidade dos resultados são descritas no Capítulo 5, e o estado da arte é retratado no Capítulo 6. O Capítulo 7 organiza os materiais e métodos propostos para execução deste trabalho, com resultados apresentados no Capítulo 8. Por fim, é feita a conclusão no Capítulo 9, seguido pelas referências e apêndices.




2 TEORIA MUSICAL BÁSICA

Pode-se usar três elementos principais para se definir música (MED, 1996):

- **Melodia:** sons, alturas ou notas, organizadas sucessivamente, responsáveis por caracterizar a música, sendo por vezes o elemento mais marcante em toda a obra;
- **Harmonia:** conjuntos de notas dispostas e tocadas simultaneamente, emoldurando e dando sentido à melodia;
- **Ritmo:** organização e ordenação temporal dos sons da música, tanto melódicos como harmônicos.

As notas pelas quais se dão qualquer música correspondem com sons organizados em sistemas de afinação, sendo o Sistema Temperado o mais comumente aplicado nas músicas ocidentais (NOUGUÉ, 2011). Os nomes pelos quais se conhecem estas notas são, respectivamente: Dó, Ré, Mi, Fá, Sol, Lá, Si. Além destas sete notas, existem ainda aquelas consideradas como acidentes, nomeadas como sustenidos ou bemóis.

Um sustenido (\sharp) eleva a nota por ele referenciada em um semitom¹, por exemplo: Dó \sharp fica entre Dó e Ré. O bemol (\flat) por sua vez possui lógica oposta, diminuindo um semitom. Ainda, outra figura existente é o bequadro (\natural), um sinal que indica a anulação de um acidente (sustenido ou bemol) em um local específico, e extrapolado para o mesmo compasso apenas.

Todas estas notas podem ser representadas utilizando notação musical em um espaço denominado pauta ou pentagrama, um conjunto de cinco linhas. Para referenciar o nome de cada nota na pauta, é preciso ainda outro elemento chamado clave, havendo para esta três representações: clave de Sol () , clave de Fá () e clave de Dó () .

Existem formas diferentes de representar uma figura musical, para que se atribua a correta duração de uma nota, dando característica rítmica à música e as localizando no tempo. Para organizar as durações na pauta, os tempos são agrupados em espaços chamados compassos, os quais seguem como regra uma fórmula que é indicada no início da pauta. A fórmula de compasso é escrita com formato de fração, em que o numerador indica a quantidade de tempos de um compasso e o denominador indica a figura que tem valor de um tempo (CHEDIAK, 1986). Dessa forma, as quatro figuras rítmicas de duração mais longa são descritas a seguir (MED, 1996):

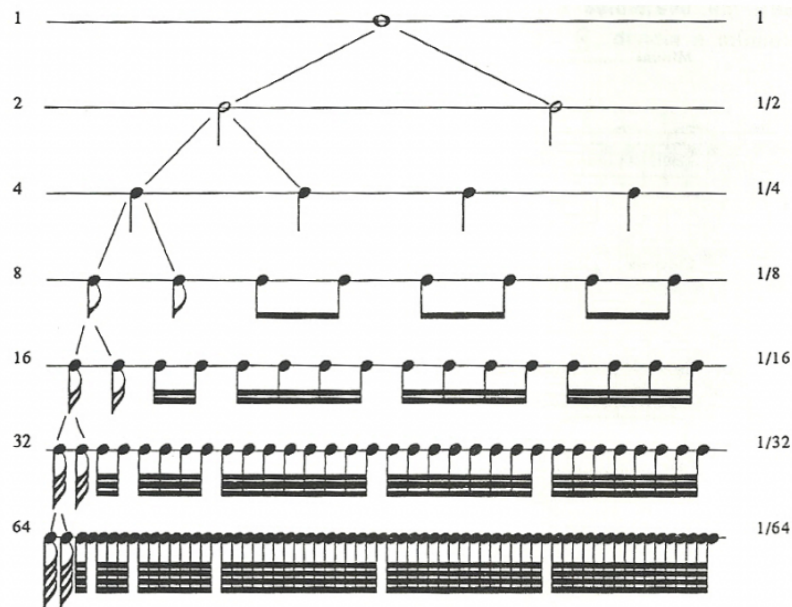
- **Semibreve** (\circ): figura de maior duração possível, sendo que as demais figuras são derivações desta;
- **Mínima** (\downarrow): vale a metade da semibreve, ou seja, quando duas mínimas são somadas, equivalem a uma semibreve;

¹ Distância entre notas musicais, explicada na Subseção 2.1.

- **Semínima** (♪): tem valor de metade da mínima; quatro semínimas equivalem a uma semibreve, e uma mínima tem duração igual a soma de duas semínimas;
- **Colcheia** (♩): é caracterizada por uma bandeirola (colchete), e vale a metade da semínima, seguindo a lógica constituída.

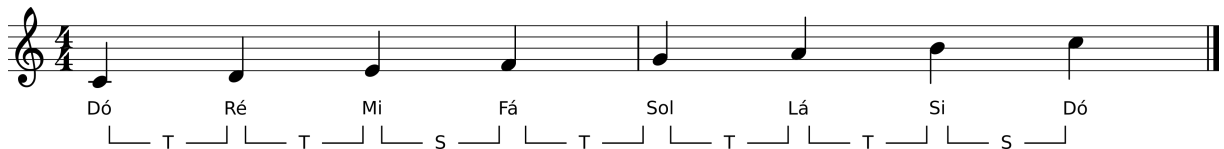
As demais formas e figuras podem ser resumidas pela Figura 1, que mostra uma comparação da quantidade de figuras que representam a anterior, a começar pela semibreve. Pode-se notar então que cada figura tem a metade da duração da superior.

Figura 1 – Quadro comparativo da divisão proporcional dos valores das figuras rítmicas desde a semibreve até a semifusa.



Fonte: Med (1996).

Figura 2 – As sete notas musicais representadas na pauta usando a clave de sol e os seus respectivos nomes e intervalos de tons (T) e semitons (S).



Fonte: Autorial Própria (2022). Adaptado de Med (1996).

Na Figura 2 tem-se um trecho completo usando os conceitos de notação musical até aqui apresentados. As notas estão referenciadas pela clave de Sol. É possível perceber que a clave dá indicativo de a nota Sol estar sobre a segunda linha de baixo para cima, e a partir dela é possível nomear as outras notas usando espaços e linhas (MED, 1996).

2.1 Teoria de Harmonia

Segundo Schoenberg (1978), entende-se por harmonia o estudo de sons simultâneos, chamados de acordes, e como realizar sua união relativos a seus valores originais, melódicos e rítmicos, além de compreender seu significado e seu peso relativo um ao outro. O estudo da harmonia estabelece regras precisas e conceitos importantes, dos quais alguns serão apresentados a seguir.

A distância entre notas musicais é medida em termos de tons (T) e semitons (S), sendo que um semitom, segundo Med (1996) é definido como o menor intervalo existente entre duas notas no Sistema Temperado de 12 notas. Além disso, a soma de um semitom com um tom equivalem a um tom e meio (Tm). Na Tabela 1 são apresentados os nomes usados para caracterizar os intervalos (distâncias) entre notas dentro do espaço de uma oitava justa (quando as notas começam a se repetir) e a quantidade de semitons que os representam, bem como a nota que está com essa distância relativa à nota Dó, como exemplo.

Quadro 1 – Nomes dos intervalos, forma de representação, quantidade de semitons que contêm e exemplo de aplicação com relação à nota Dó.

Nome do intervalo	Representação	Quantidade de semitons	Nota relativa de Dó
Segunda menor	2 ^a m	1	Dó#
Segunda maior	2 ^a M	2	Ré
Terça menor	3 ^a m	3	Mi \flat
Terça maior	3 ^a M	4	Mi
Quarta justa	4 ^a J	5	Fá
Quinta diminuta	5 ^a D	6	Sol \flat
Quinta justa	5 ^a J	7	Sol
Quinta aumentada	5 ^a A	8	Sol#
Sexta maior	6 ^a M	9	Lá
Sétima menor	7 ^a m	10	Si \flat
Sétima maior	7 ^a M	11	Si
Oitava justa	8 ^a J	12	Dó

Fonte: Adaptado de Kaizer (2017).

Um intervalo de terça pode ser maior ou menor, sendo que o primeiro representa uma distância de três semitons de uma nota de partida, e o segundo, quatro. Por exemplo, de Dó até Mi, há uma distância de quatro semitons e conseqüentemente um intervalo de terça maior. Já entre Dó e Ré#, um intervalo de três semitons, uma terça menor.

As escalas musicais, por sua vez, são seqüências ordenadas de notas, de acordo com regras de intervalos entre notas. São muitas as variações possíveis para organizar as notas em escalas (COCA; ZHAO, 2016), das quais três são principais: cromática, maior e menor, esta última dividida em natural, harmônica e melódica (MED, 1996). A Tabela 2 contém informações sobre a construção em intervalos de T, S e Tm destas escalas, bem como um exemplo.

Quadro 2 – Principais escalas musicais, estrutura de tons e semitons, e exemplo de construção.

Escala	Estrutura de intervalos	Exemplo de construção
Maior	T T S T T T S	Dó Ré Mi Fá Sol Lá Si Dó
Menor natural	T S T T S T T	Lá Si Dó Ré Mi Fá Sol Lá
Menor harmônica	T S T T S Tm S	Lá Si Dó Ré Mi Fá Sol#
Menor melódica ascendente	T S T T T T S	Lá Si Dó Ré Mi Fá# Sol#

Fonte: Adaptado de Chediak (1986).

A escala cromática, em especial, contém todas as 12 notas existentes no Sistema Temperado, sendo elas (usando notação sustentada): Dó, Dó#, Ré, Ré#, Mi, Fá, Fá#, Sol, Sol#, Lá, Lá# e Si. Por ser composta de todas as notas, não há definição de tonalidade.

A escala maior se baseia no uso de sete notas diatônicas² (KOELLREUTER, 1978). Na Figura 2 está ilustrada a escala maior de Dó, com intervalos entre notas indicados. Importante entender que, na mudança da nota inicial, as distâncias entre notas são mantidas para a construção da escala. Na verdade, são as distâncias entre as notas que de fato caracterizam a forma da escala.

A escala menor possui três formas de construção básica. A primeira, escala menor natural, é semelhante à escala maior se iniciada a partir da terça menor descendente da escala maior. Pode-se ver para o exemplo referente da Tabela 2, que as notas da escala menor natural são iguais às da maior, mudando apenas o ponto de início. Tendo por base a escala menor natural, se aumentada com um sustenido a sétima nota, de forma a criar uma sétima maior, tem-se a menor harmônica. Por fim, a menor melódica é como a menor harmônica, porém com a sexta nota também aumentada na forma ascendente mas sem essa alteração na forma descendente.

Uma forma comum de construir acordes é fazendo uso de tríades, consistindo de três notas tocadas ao mesmo tempo (acordes de três notas), feitas da sobreposição de notas em intervalos de terças. Dependendo da qualidade destes intervalos, poderá chamar o acorde de maior, menor, aumentado ou diminuto. A Tabela 3 mostra as possibilidades de modos de acordes disponíveis e a definição dos mesmos para a tônica³ (1ª) e suas respectivas terça (3ª) e quinta (5ª), sendo que a 3ª pode ser maior ou menor, a 5ª é dita justa, e ambas (3ª e 5ª) podem ser aumentadas ou diminutas.

Para se referir ao grupo de acordes que pode ser executado a partir de uma escala, é usada a expressão “campo harmônico”. Dentro deste, estão contidos sobretudo acordes maiores e menores que constituem uma relação funcional entre si. A construção de um campo harmônico se dá de forma a obter um acorde para cada nota da escala selecionada, usando a sobreposição de terças e mantendo as notas dos acordes dentro da mesma escala. Os acordes formados a partir da escala maior podem ser representados como na Figura 3. Em uma música elaborada desta

² Ordem de notas que procedem de acordo com a sucessão natural de tons e semitons, nos modos e nas escalas maior e menor.

³ A nota fundamental de um acorde é chamada de tônica, a qual também é responsável por dar nome ao acorde.

Quadro 3 – Modos possíveis para as tríades, a regra para sua composição e exemplo montado a partir da nota Dó como tônica.

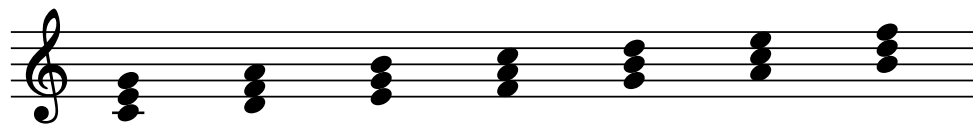
Modo	Composição	Exemplo a partir de Dó
Aumentado	1 ^a – 3 ^a maior – 5 ^a aumentada	Dó Mi Sol \sharp
Maior	1 ^a – 3 ^a maior – 5 ^a justa	Dó Mi Sol
Menor	1 ^a – 3 ^a menor – 5 ^a justa	Dó Mi \flat Sol
Diminuto	1 ^a – 3 ^a menor – 5 ^a diminuta	Dó Mi \flat Sol \flat

Fonte: Adaptado de Kaizer (2017).

maneira e sem modulação⁴, os acordes montados estarão dentro deste campo harmônico e cada acorde terá uma função diferente (KAIZER, 2017).

Outra forma comum para a representação de acordes é a utilização de letras (ou cifras) ao invés do nome das notas, como também ilustrado na Figura 3. Para este sistema, troca-se a nota Lá pela letra A (maiúscula), e segue-se sequencialmente até a letra G (Sol). Isso facilita a cifragem, e permite outras convenções, como a representação de acordes menores (letra “m” minúscula). Os números romanos, por sua vez, servem para indicação dos graus dos acordes, generalizando a representação.

Figura 3 – Os sete acordes musicais disponíveis a partir da escala de Dó maior e suas respectivas cifras e graus.



Dó	Ré	Mi	Fá	Sol	Lá	Si
maior	menor	menor	maior	maior	menor	diminuto
C	Dm	Em	F	G	Am	Bmdim
I	IIIm	IIIIm	IV	V	VIIm	VIIIdim

Fonte: Adaptado de Chediak (1986).

Entende-se por função, em termos de harmonia, a característica de um acorde e seu valor em relação aos demais, no contexto da obra, que é determinado pela somatória relacional de todos os acordes com um centro tonal, por onde se estabelece o conceito de tonalidade. Esta é definida então pelo conjunto de categorias ou funções tradicionalmente denominadas de tônica, dominante e subdominante, respectivamente. Cada uma pode ter suas funções definidas como (KOELLREUTER, 1978; CHEDIAK, 1986):

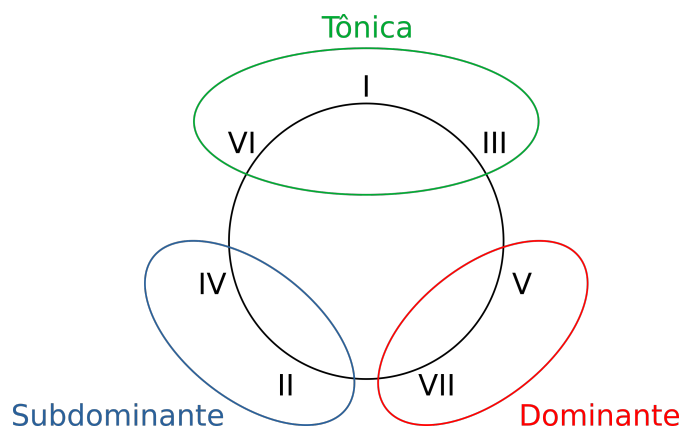
- **Tônica:** concede sensação de repouso, tendo funcionalidade de início ou finalização. Sua representante é o grau I do campo harmônico;

⁴ Mudança de tom de caráter provisório.

- **Dominante:** dá o sentido de movimento de aproximação, uma vez que gera tensão e necessidade de resolução, geralmente provida pela tônica. É o grau V do campo harmônico;
- **Subdominante:** também relacionado ao movimento, porém de afastamento, um meio termo entre tônica e dominante, sendo representada pelo grau IV.

Para os demais acordes (ou graus) presentes no campo harmônico, suas funções são herdadas das funções principais, chamadas funções secundárias, e estes acordes são denominados de relativos. Segundo Koellreuter (1978), acordes relativos compartilham o mesmo significado harmônico da tônica, dominante ou subdominante dos quais são vizinhos de terça, ou seja, são acordes diatônicos cujas fundamentais estão a uma distância de terça da função principal. Em linhas gerais, aplica-se o ilustrado na Figura 4, em que os graus III e VI são relativos da tônica, VII da dominante e II da subdominante. Isso quer dizer que estes graus compartilham da mesma função principal, em nível menos intenso, e podem funcionar como substitutos.

Figura 4 – Tríades dispostas no círculo de terças, rotuladas por funções harmônicas.



Fonte: Shaffer, Hughes e Moseley (2014).

Respeitando as funções de cada grau, estabelece-se o movimento padrão dos acordes dentro da música: a função dominante é comumente precedida pela tônica, dando a resolução da dissonância; a função subdominante tem maior liberdade, entretanto sua sensação de movimento normalmente leva à dominante; a função tônica por sua vez é geralmente aplicada no início ou final de uma música ou frase musical, provendo a ideia de conclusão (CHEDIAK, 1986).

A harmonia funcional, portanto, tem por objetivo estudar o funcionamento destas funções, bem como estabelecer regras e conceitos para a sua correta utilização dentro da música. A partir do entendimento das funções é possível harmonizar músicas de forma coerente, que façam sentido aos ouvintes. Sendo também regras que, quando respeitadas, indicam a qualidade da obra.

3 REDES NEURAIS ARTIFICIAIS

A Inteligência Artificial (IA) é um ramo da computação que engloba diversas disciplinas, como processamento de linguagem natural, representação de conhecimento, raciocínio automatizado, aprendizado de máquina, visão computacional e robótica (RUSSELL; NORVIG, 2009). Não obstante, o estudo de Redes Neurais Artificiais (RNAs) está incluído em algumas destas, como se verá a seguir.

As RNAs, também reconhecidas pelos nomes de redes conexionistas ou computação neural, são sistemas paralelos e distribuídos construídos com base em unidades de processamento chamadas “neurônios”, capazes de calcular funções matemáticas, uma inspiração pautada no funcionamento do sistema nervoso dos organismos superiores (BRAGA, 2000).

De forma geral, RNAs são sistemas que tentam modelar como o sistema nervoso performa determinada função ou tarefa, tendo a propensão natural de guardar conhecimento com base em experiências e permitir seu uso de alguma forma. Diz-se inspirado no sistema nervoso principalmente por dois aspectos: possui um processo de aprendizagem cujo conhecimento é adquirido através de informações do seu ambiente e contém pesos entre conexões de unidades de processamento (pesos sinápticos) que são capazes de armazenar conhecimento (HAYKIN, 2008).

O fato de que as RNAs tem potencial de aprendizado é importante, pois permite a generalização, ou seja, produção de resultados razoáveis para dados desconhecidos (HAYKIN, 2008), além de atuar como mapeadoras universais de funções multivariáveis (BRAGA, 2000). Outro fato importante é que o conhecimento é distribuído, ou seja, não depende de uma unidade mas sim de toda a estrutura do sistema. Sendo assim, são três aspectos que caracterizam uma RNA: a existência de neurônios, que são as unidades de processamento do sistema; o padrão de conexão entre estes, denominado de estrutura ou arquitetura da rede; e a necessidade de um método para determinar os pesos entre conexões, chamado de algoritmo de treinamento (CASTRO, 2006).

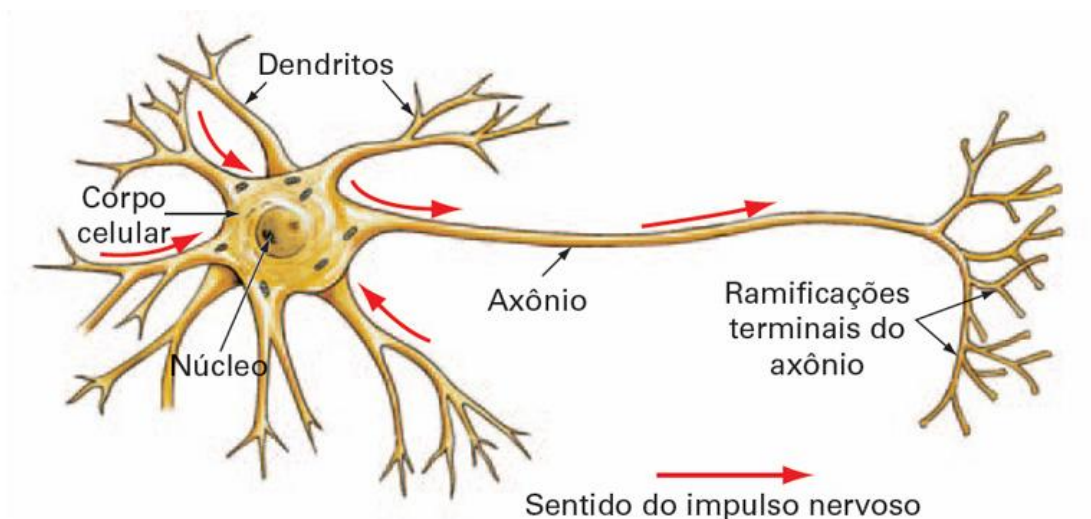
Indo além, a capacidade de aprendizado confere às RNAs o caráter de ferramentas gerais para solução dos mais diferentes tipos de problemas. Dessa maneira, são frequentemente aplicadas em tarefas como classificação de padrões, mineração de dados, regressão de funções e processamento da informação, sendo úteis em diversas áreas do conhecimento, dentre as quais a composição musical (HAYKIN, 2008).

A concepção das RNAs começa na compreensão do funcionamento do neurônio artificial, pois é de onde partiu o estudo para se alcançar estruturas complexas, fundando o campo de estudo da neurociência computacional ou neurocomputação, graças a sua popularidade e formas eficientes de aprendizado (RUSSELL; NORVIG, 2009).

3.1 Neurônio Artificial

O neurônio biológico, como ilustrado na Figura 5, é a fonte de inspiração do neurônio artificial. Nele, impulsos nervosos (pulsos elétricos) enviados por outros neurônios ou pelos sensores biológicos contidos em órgãos como a pele, são recebidos por meio dos dendritos e processados no corpo celular. A depender do resultado da integração dos sinais recebidos, o neurônio pode ou não produzir um novo impulso (potencial de ação) com uma determinada modulação (ponderação), que será por sua vez transmitido para outros neurônios que estiverem conectados aos seus terminais axônicos (CASTRO, 2006). Essa união do axônio com dendritos chama-se sinapse, que funciona como uma válvula controladora do fluxo de informação (impulsos) entre neurônios. A sinapse é variável (modulável), e é isso que capacita a adaptação e o aprendizado (BRAGA, 2000).

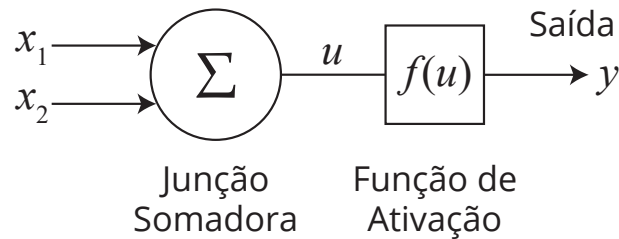
Figura 5 – Esquema simplificado de um neurônio biológico e sua organização em dendritos, corpo celular, núcleo, axônio e terminais axônicos.



Fonte: Academy (2021).

Em 1943, surgiu a primeira modelagem do que viria a ser um neurônio artificial. McCulloch e Pitts (1943) realizaram um trabalho pioneiro que se concentrava na descrição de um modelo artificial de um neurônio e apresentação das suas capacidades (BRAGA, 2000). Em seu trabalho, assumiram que o neurônio responde de forma binária e que tem uma estrutura estática (CASTRO, 2006). Uma representação do neurônio de McCulloch e Pitts pode ser observada na Figura 6.

Figura 6 – O neurônio artificial de McCulloch e Pitts. Duas entradas são somadas e seu resultado passa por uma função de ativação antes da saída.



Fonte: Castro (2006).

Para duas entradas x_1 e x_2 , o neurônio integra (soma) seus valores gerando a saída u . Tal saída é processada por uma função de ativação $f(\cdot)$, definida pela Equação 1:

$$y = f(u) = \begin{cases} 0, & u < 0 \\ 1, & u \geq 0 \end{cases}, \quad (1)$$

sendo y a saída do neurônio.

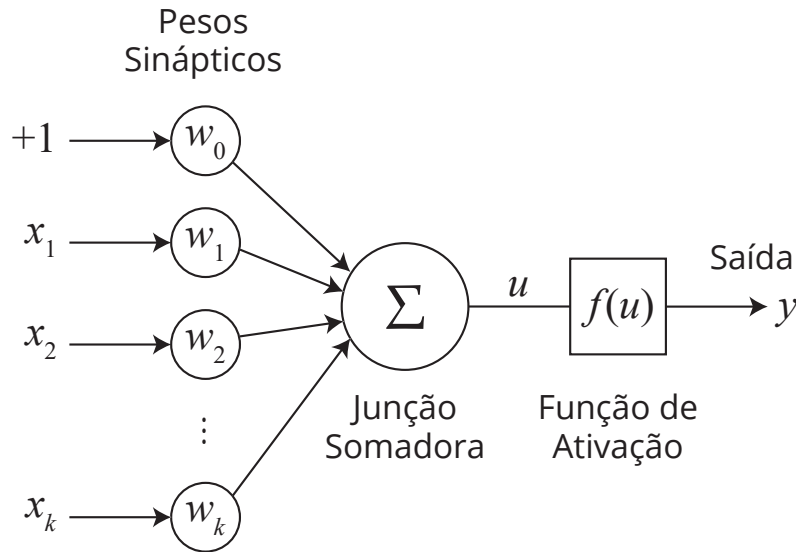
Dessa forma, se u for maior ou igual a 0, então o neurônio responde com $y = 1$, ou seja, passa para o estado ativo. Caso contrário, fica inativo, respondendo com $y = 0$.

Embora seja de construção simples, notam-se algumas características importantes em comum com modelos de RNA atuais, como a presença de uma função de ativação e a integração das entradas para determinar a saída. Entretanto, também é de várias limitações, como o fato de conseguir implementar apenas funções linearmente separáveis e ausência de pesos ajustáveis, consequentemente incapaz de aprendizado (BRAGA, 2000).

O estudo das RNAs continuou nos anos seguintes, com fatos marcantes como citado a seguir: Hebb (1949) apresenta uma nova proposta com capacidade de aprendizado e propõe uma forma de elaboração de algoritmos de aprendizagem; Widrow e Hoff (1960) sugerem uma nova regra de aprendizagem que se baseia no método do gradiente para minimização do erro na saída de um neurônio com resposta linear; Rosenblatt (1958) propõe o Perceptron, uma estrutura com quantidade maior de sinapses ajustáveis capaz de classificar determinados padrões.

Por fim, pode-se estabelecer o modelo do neurônio genérico dos dias atuais conforme o diagrama da Figura 7, o qual é constituído de três elementos básicos: 1) entradas com pesos, 2) unidade de integração ou junção somadora, e 3) uma função de ativação. Nota-se também a presença de uma entrada fixa $x_0 = +1$, chamada de polarização ou *bias* (HAYKIN, 2008).

Figura 7 – Esquema de construção de um neurônio artificial para x_k entradas que são multiplicadas por pesos sinápticos antes de passar pela junção somadora e função de ativação, até a saída.



Fonte: Haykin (2008).

Considerando esta estrutura, tem-se que uma entrada x_j na conexão j é multiplicada pelo peso w_j . Assim, a saída da unidade de integração u pode ser descrita como a Equação 2:

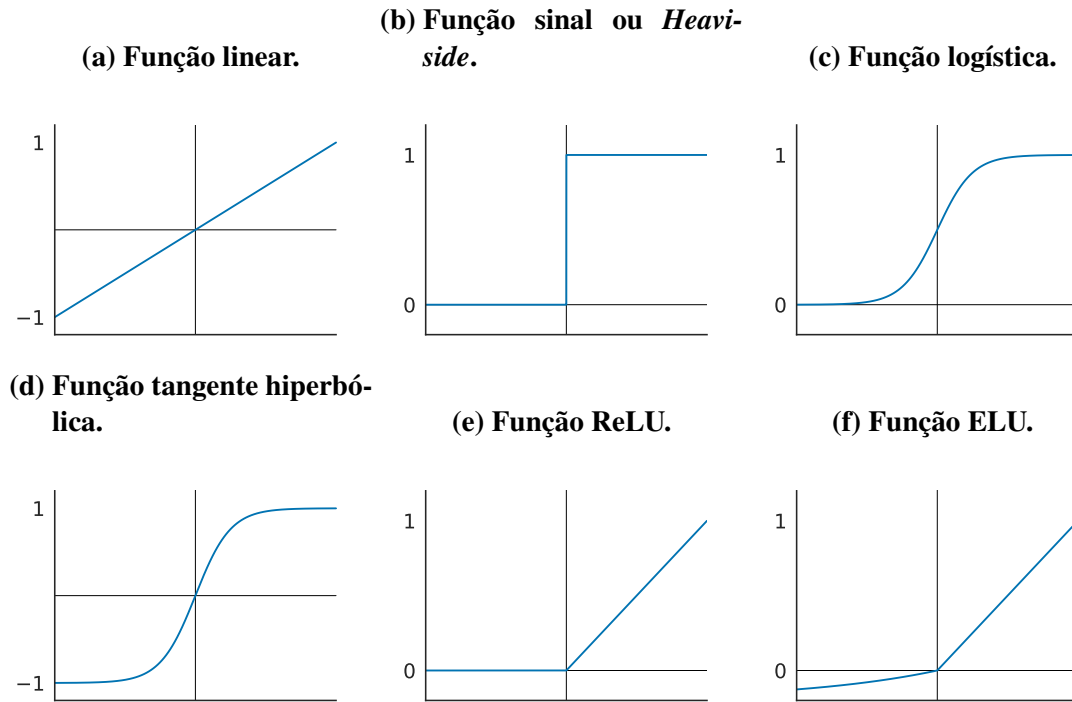
$$u = \sum_{j=0}^k w_j x_j, \quad (2)$$

para k sendo o total de entradas, passando pela função de ativação $f(\cdot)$, sendo $\mathbf{x} = [+1; x_1; x_2; \dots; x_k]^T$ os sinais de entrada, $\mathbf{w} = [w_0; w_1; w_2; \dots; w_k]^T$ os respectivos pesos e y a saída do neurônio. O *bias* é um parâmetro cujo efeito é de aumentar o grau de liberdade da função e a capacidade de aproximação da rede, além de tornar possível saídas não nulas mesmo que todas as entradas sejam (ACADEMY, 2021). Grosso modo, ajusta o nível ou *offset* da resposta.

3.2 Funções de Ativação

Uma função de ativação $f(\cdot)$ define qual será a saída de determinado neurônio respeitando o valor de integração u . Partindo do modelo definido por McCulloch e Pitts, diversas outras funções foram desenvolvidas que permitem a produção de saídas quaisquer, não somente zero e um (binárias) (HAYKIN, 2008; BRAGA, 2000). A seguir, pode-se encontrar a definição de algumas delas e sua visualização gráfica na Figura 8.

Figura 8 – Diversas funções de ativação nomeadas individualmente.



Fonte: Autoria Própria (2022).

A função de ativação linear é uma função simples que pode ser descrita pela Equação 3:

$$f(u) = au, \quad (3)$$

para a um número real que define por sua vez uma saída linear. A mesma pode ser vista na Figura 8a.

Outra função comum é a função sinal, também conhecida como *Heaviside* (HAYKIN, 2008), que pode responder de maneira binária $[0, 1]$ ou bipolar $[-1, 1]$, sendo a forma binária a utilizada no neurônio de McCulloch e Pitts, como mostrado na Equação 1 e Figura 8b.

Um grupo de funções que é mais comumente utilizada na literatura de RNAs são as sigmóides (CASTRO, 2006; HAYKIN, 2008). Seu gráfico tem a forma peculiar de um “S” e apresenta um balanço entre comportamento linear e não linear, podendo ser obtida a partir de várias funções, como a logística e a tangente hiperbólica. A função logística (Figura 8c) pode ser definida pela Equação 4:

$$f(u) = \frac{1}{1 + e^{-u}}, \quad (4)$$

e a tangente hiperbólica (Figura 8d), pela Equação 5 (JEFFREY; DAI, 2008):

$$f(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}, \quad (5)$$

A tangente hiperbólica possui a interessante característica de resposta bipolar, que pode trazer benefícios frente a função logística cuja resposta é binária, muito embora elas sejam essencialmente equivalentes (MENON *et al.*, 1996). Além disso, funções não lineares, por consequência, inserem não linearidade na resposta de saída do neurônio, característica principal para mapeadores universais.

Recentemente, com o avanço dos estudos de RNAs e mais especificamente de arquiteturas de aprendizado profundo, novas funções de ativação foram desenvolvidas com objetivo de resolver problemas como o de dissipação do gradiente (*vanishing gradients*) ou superfície plana (FAHLMAN, 1988), abordado em maior detalhe na Subseção 3.5.4.

Uma destas, desenvolvida por Maas, Hannun e Ng (2013), é denominada de Unidade Linear Retificada ou ReLU (*Rectified Linear Unit*) e pode ser expressa como na Equação 6:

$$f(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \end{cases}, \quad (6)$$

ou seja, caso u seja maior que zero, a resposta será igual a entrada.

É de certa forma similar a função de ativação linear (Equação 3) para valores maiores do que zero, com exceção da presença de um número real a . Está ilustrada na Figura 8e.

Outra função de desenvolvimento recente é a Unidade Linear Exponencial ou ELU (*Exponential Linear Unit*), concebida por Clevert, Unterthiner e Hochreiter (2016). Sua definição é similar a da Equação 6 e é da forma da Equação 7:

$$f(u) = \begin{cases} u, & u > 0 \\ a(e^u - 1), & u \leq 0 \end{cases}. \quad (7)$$

O que a diferencia da ReLU é apenas a resposta para u menor do que zero, permitindo saídas com valores negativos, como também pode-se observar na Figura 8f.

Ainda, a função exponencial normalizada, também chamada de *softmax* é de interesse pois é uma generalização da função logística, ou seja, é uma função sigmoide, porém adaptada para múltiplas dimensões (BISHOP, 2006). Sua definição matemática é vista na Equação 8:

$$f(u)_m = \frac{e^{u_m}}{\sum_{i=1}^k e^{u_i}}, \quad (8)$$

sendo $\mathbf{u} = [u_1; u_2; \dots; u_k]^T$ as saídas, obtendo o valor para a saída m .

Esta função retorna uma distribuição de probabilidade para casos em que há mais de uma saída para a rede, configuração comum quando o objetivo é de realizar tarefas de classificação com múltiplas classes.

3.3 Regra de Aprendizado

O conceito de aprendizado de um neurônio artificial foi introduzido somente quando surgiu o Perceptron de Rosenblatt, pois os modelos desenvolvidos até então continham pesos sinápticos fixos, não ajustáveis (BRAGA, 2000).

A regra de aprendizado do Perceptron, também chamada de algoritmo de aprendizado por correção de erro (HAYKIN, 2008) tem por base o modelo ilustrado na Figura 7. Para tanto, pode-se definir as entradas como um vetor $\mathbf{x}(t) = [+1; x_1(t); x_2(t); \dots; x_k(t)]^T$, para t representando o momento temporal de aplicação do algoritmo, uma vez que se trata de um algoritmo iterativo. Conseqüentemente, o vetor de pesos é definido como $\mathbf{w}(t) = [w_0(t); w_1(t); w_2(t); \dots; w_k(t)]^T$, e o combinador linear é da forma da Equação 9:

$$u(t) = \sum_{j=0}^k w_j(t)x_j(t) = \mathbf{w}^T(t)\mathbf{x}(t). \quad (9)$$

Considerando a função de ativação sinal (Equação 1) ajustada de maneira bipolar, a regra de adaptação de pesos pode ser definida como a Equação 10:

$$\mathbf{w}(t + 1) = \mathbf{w}(t) + \eta(t)(d(t) - y(t))\mathbf{x}(t), \quad (10)$$

onde $\eta(t)$ é o parâmetro de taxa de aprendizado, que controla o ajuste aplicado ao vetor de pesos na iteração t .

Importante notar que, caso $\eta(t) = \eta > 0$ seja constante independente do número da iteração t , tem-se uma regra de adaptação com incremento fixo.

Dessa forma, para η constante, tem-se que $y(t)$ é a saída na iteração t e $d(t)$ representa a resposta desejada quantizada, em que a diferença $d(t) - y(t)$ é considerada como sinal de erro. É relevante ressaltar que o parâmetro da taxa de aprendizado constante η é definido no intervalo $0 < \eta \leq 1$, devendo equilibrar entre alterações estáveis da variação dos pesos (valores menores de η) e velocidade de aprendizado (valores maiores de η) (LIPPMANN, 1987).

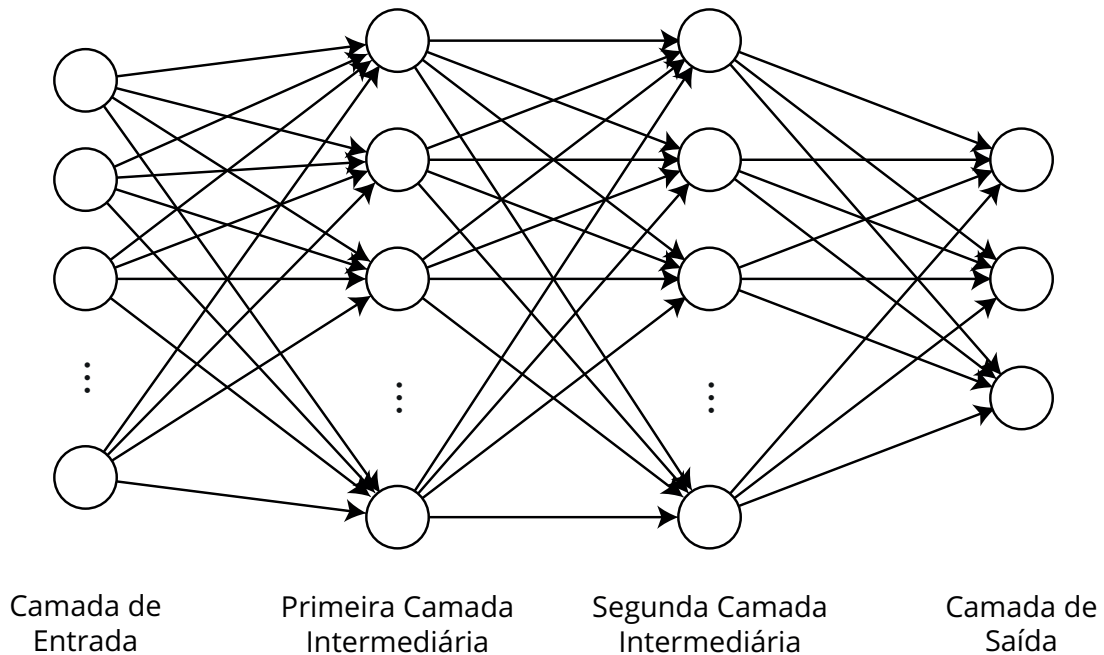
3.4 Perceptron de Múltiplas Camadas

O neurônio é uma unidade limitada a fazer classificações de dados linearmente separáveis, considerado como uma rede de camada única ou singular. Quando mais destes neurônios são interligados de diferentes formas, entende-se o conceito de camadas em redes neurais.

Uma camada é a disposição de neurônios paralelos, que recebem todos os sinais das entradas anteriores porém não se comunicam entre si, e enviam a resposta somente adiante, o que se conhece por estrutura *feedforward*. Várias camadas de neurônios podem ser encadeadas,

sendo que a camada inicial é a de entrada; a final, de saída; e as intermediárias são chamadas de camadas escondidas (*hidden*) (HAYKIN, 2008). Um exemplo está ilustrado na Figura 9, em que os círculos representam os neurônios artificiais.

Figura 9 – Uma MLP de quatro camadas: a Camada de Entrada, que não realiza nenhuma operação servindo apenas de entrada para os dados, a Primeira e Segunda Camada Intermediária ou camadas escondidas, e a Camada de Saída, contendo neste caso, três neurônios.



Fonte: Castro (2006).

Percebe-se que todos os neurônios de uma camada se conectam a todos os neurônios da próxima camada, o que caracteriza a rede como sendo totalmente conectada. Caso contrário, esta seria uma rede parcialmente conectada (HAYKIN, 2008).

Não obstante, define-se uma RNA que apresenta ao menos uma camada escondida como Perceptron de Múltiplas Camadas, ou MLP (*Multilayer Perceptron*) (BRAGA, 2000). As MLPs são RNAs capazes de solucionar problemas não linearmente separáveis. Na verdade, segundo Cybenko (1988) e Cybenko (1989), é possível resolver qualquer função contínua quando há presença de uma camada escondida, e com duas, pode-se aproximar qualquer função matemática, desde que corretamente definido o número de neurônios.

Para problemas de classificação binária, ou seja, quando deve-se distinguir entre duas classes, é natural utilizar apenas um neurônio na camada de saída, já que o mesmo pode responder de forma binária ou bipolar, indicando a qual classe se refere. No entanto, para problemas multiclasse, quando há pelo menos três classes diferentes, é possível empregar outras abordagens. Uma delas é definir a quantidade de neurônios de saída igual à quantidade de classes que se quer categorizar, sendo que a resposta de cada neurônio é a probabilidade de determinado

dado pertencer a uma das classes codificadas usando o método *one-hot*¹ (HARRIS; HARRIS, 2013), situação em que a função de ativação *softmax* se torna de interesse.

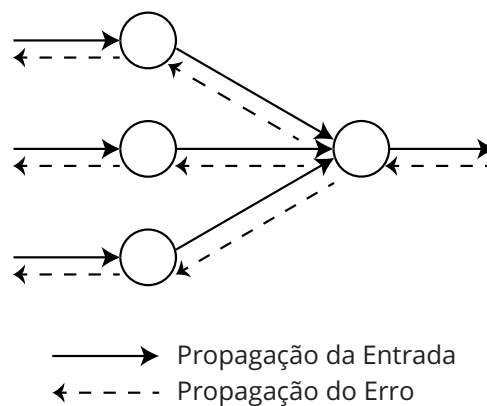
Para realizar o treinamento deste tipo de rede, o método mais usual da literatura é o denominado de retropropagação de erro ou *backpropagation*, que se baseia em gradiente descendente, popularizado por Rumelhart, Hinton e Williams (1986).

3.5 Treinamento da Rede

Estão disponíveis diversos algoritmos com objetivo de treinar MLPs, geralmente supervisionados estáticos (variação dos pesos entre conexões) ou dinâmicos (alteram a estrutura da rede) (BRAGA, 2000). No entanto, o mais popular é o de *backpropagation*, que é dividido em duas fases: *forward* (para frente) e *backward* (para trás) (HAYKIN, 2008).

Na primeira fase, os pesos não sofrem alteração, ou seja, são fixos, e os dados de entrada são propagados até a saída da rede. Após, um sinal de erro é produzido comparando a saída com a resposta desejada. Então o erro se propaga de volta na rede, porém desta vez realizando sucessivos ajustes nos pesos entre conexões (HAYKIN, 2008). A Figura 10 ilustra o processo descrito, que é detalhado a seguir na Subseção 3.5.1.

Figura 10 – Operação do *backpropagation*: primeiro acontece a propagação da entrada ilustrado como sendo da esquerda para a direita, e depois a propagação do erro no sentido oposto.



Fonte: Castro (2006).

Todo o procedimento do *backpropagation* pode ser visto como um problema de otimização não linear irrestrita com a minimização de uma função de custo $E(\cdot)$ baseada em alguma métrica de erro de aproximação, sendo o método de gradiente descendente o mais comum (ZUBEN, 1996).

¹ Um vetor formado de “0”s cujo tamanho é igual à quantidade de classes, em que apenas uma posição é igual a “1”, posição essa indicativa da classe.

3.5.1 Backpropagation

A primeira fase na aplicação do *backpropagation* é a de propagação do sinal de entrada passando pelas camadas ocultas até alcançar a camada de saída (*forward*). Para que isso aconteça, a rede precisa ter seus pesos inicializados de maneira aleatória e então segue-se com a apresentação de dados de treinamento. Se houver algum conhecimento prévio sobre o processo de mapeamento que está sendo trabalhado, é possível inicializar os pesos com valores pré-fixados. Abaixo, seguem-se as definições segundo Haykin (2008).

Considerando como um dado de treinamento $(\mathbf{x}(n), \mathbf{d}(n))$ para a iteração n , em que $\mathbf{x}(n)$ é o vetor de entrada aplicado à camada de entrada da RNA e $\mathbf{d}(n)$ é o vetor da resposta desejada, a propagação do sinal de entrada se dá pelo cálculo dos campos locais induzidos $u_j^{(l)}(n)$ para o neurônio j na camada l , feito como na Equação 11:

$$u_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) y_i^{(l-1)}(n), \quad (11)$$

para $y_i^{(l-1)}(n)$ igual ao sinal de saída do neurônio j da camada anterior $l - 1$, e $w_{ji}^{(l)}(n)$ é o peso do neurônio j na camada l que é alimentado pelo neurônio i da camada $l - 1$.

Quando $i = 0$, então $y_0^{(l-1)}(n) = +1$, sendo este o sinal de bias aplicado ao neurônio j da camada l . A saída do neurônio j da camada l pode ser definida pela Equação 12:

$$y_j^{(l)} = f_j(u_j(n)), \quad (12)$$

uma vez que, se o neurônio j estiver na primeira camada escondida, ou seja, $l = 1$, então (Equação 13):

$$y_j^{(0)} = x_j(n), \quad (13)$$

em que $x_j(n)$ é o j -ésimo elemento do vetor de entrada $\mathbf{x}(n)$.

Agora, se o neurônio j estiver na camada de saída, ou seja, $l = L$ para L igual ao total de camadas da rede, então tem-se a Equação 14:

$$y_j^{(L)} = o_j(n). \quad (14)$$

Na sequência, o erro para cada neurônio da camada de saída é calculado e inicia-se a segunda fase do *backpropagation* (*backward*): a propagação do erro e ajuste dos pesos. Este pode ser visto como um problema de otimização com a minimização de uma função de custo $E(\cdot)$ que se baseia em alguma métrica.

Para tanto, uma forma simples de representar o sinal de erro pode ser ilustrado na Equação 15:

$$e_j(n) = d_j(n) - o_j(n), \quad (15)$$

para $d_j(n)$ como sendo o j -ésimo elemento do vetor de resposta desejada $\mathbf{d}(n)$.

Em seguida, realiza-se o cálculo dos gradientes locais δ , em que o gradiente da camada de saída L é obtido como na Equação 16:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n) f_j'(u_j^{(L)}(n)). \quad (16)$$

Os das demais camadas são de acordo com a Equação 17:

$$\delta_j^{(l)}(n) = f_j'(u_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), \quad (17)$$

para os k neurônios diretamente posteriores ao neurônio j , sendo que $f_j'(\cdot)$ é a diferenciação da função objetivo com respeito ao argumento.

Por fim, acontece o ajuste dos pesos da camada l de acordo com a regra delta generalizada da Equação 18:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n). \quad (18)$$

Uma forma de melhorar o processo de ajuste de pesos e evitar instabilidades é modificar a Equação 18 adicionando um termo de *momentum* γ , alterando o cálculo para como na Equação 19:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \gamma [\Delta w_{ji}^{(l)}(n-1)] + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n), \quad (19)$$

sendo γ uma constante geralmente positiva e $\Delta w_{ji}^{(l)}(n) = w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)$.

Depois disso, o processo deve ser repetido apresentando novas épocas de dados de treinamento até que algum critério de parada seja atingido. Pode-se definir o algoritmo de *backpropagation* como descrito no Algoritmo 1.

O algoritmo recebe como parâmetros a taxa de aprendizado η , a constante de *momentum* γ , os dados de treinamento e uma variável *max_it* que é o critério de parada do algoritmo, ou seja, quando este atingir o número de iterações definido, o treinamento para e são retornados os pesos da rede. Outro critério de parada comum é calcular o erro ao final de cada iteração e verificar se este se encontra em um nível aceitável: se sim, o treinamento pode parar, mas continua caso contrário.

Algoritmo 1 – Algoritmo de *Backpropagation*

requer max_it, η , γ , u , d
inserir w
 1: Inicializar $w(1)$
 2: $n \leftarrow 1$
 3: **enquanto** $n < \text{max_it}$ **faça**
 4: **para** l de 1 até L **faça**
 5: **para** j de 1 até J **faça**
 6: **para** i de 1 até I **faça**
 7: $u_j^{(l)}(n) \leftarrow u_j^{(l)}(n) + w_{ji}^{(l)}(n)y_i^{(l-1)}(n)$ (Equação 11)
 8: **finaliza para**
 9: $e_j(n) \leftarrow d_j(n) - o_j(n)$ (Equação 15)
 10: **finaliza para**
 11: **finaliza para**
 12: **para** l de L decrescendo para 1 **faça**
 13: **para** j de 1 até J **faça**
 14: **se** $l = L$ **então**
 15: $\delta_j^{(L)}(n) \leftarrow e_j^{(L)}(n)f'_j(u_j^{(L)}(n))$ (Equação 16)
 16: **senão,**
 17: **para** k de 1 até K **faça**
 18: $\delta_j^{(l)}(n) \leftarrow \delta_j^{(l)}(n) + f'_j(u_j^{(l)}(n))\delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n)$ (Equação 17)
 19: **finaliza para**
 20: **finaliza se**
 21: **para** i de 1 até I **faça**
 22: $w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \gamma[\Delta w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n)$ (Equação 19)
 23: **finaliza para**
 24: **finaliza para**
 25: **finaliza para**
 26: $n \leftarrow n + 1$
 27: **finaliza enquanto**

Fonte: Autoria Própria (2022).

Por se tratar da computação do gradiente, um fator definitivo para o correto funcionamento deste algoritmo está relacionado a função de ativação empregada: ela precisa ser contínua, diferenciável e não decrescente (BRAGA, 2000), como é o caso das funções sigmóides, por exemplo.

3.5.2 Métricas de Erro

Como supracitado, o *backpropagation* tem por objetivo minimizar uma função de custo $E(\cdot)$ com base em uma métrica de erro. A métrica clássica é a do Erro Quadrático Médio, ou MSE (*Mean Square Error*), já que o algoritmo original se baseia na regra delta proposta por Widrow e Hoff (1960). O MSE pode ser definido pela Equação 20:

$$\text{MSE}(t) = \frac{1}{N} \sum_{i=1}^N (y(t)_i - d(t)_i)^2, \quad (20)$$

para N sendo a quantidade de amostras (DEGROOT; SCHERVISH, 2014).

Outra métrica de erro interessante quando enfrentado o problema de multiclassificação é a Entropia Cruzada (EC), também conhecida como erro logístico multinomial, pois quantifica a diferença entre distribuições de probabilidade. Sua definição é pela Equação 21:

$$\text{EC}(t) = - \sum_{i=1}^{\mathcal{C}} d(t)_i \log(y(t)_i), \quad (21)$$

para \mathcal{C} como o total de classes possíveis, e agora $\mathbf{d}(t) = [d_1(t); d_2(t); \dots; d_{\mathcal{C}}(t)]^T$ e $\mathbf{y}(t) = [y_1(t); y_2(t); \dots; y_{\mathcal{C}}(t)]^T$ vetores de resposta desejada e saída, respectivamente, pois estão sendo consideradas múltiplas saídas para o sistema.

Aplicando a função de ativação *softmax* (Equação 8), tem-se a Equação 22:

$$\text{ECC}(t) = - \sum_{m=1}^{\mathcal{C}} d(t)_m \log \left(\frac{e^{u_m}}{\sum_{i=1}^{\mathcal{C}} e^{u_i}} \right) \quad (22)$$

sendo ECC a Entropia Cruzada Categórica, também chamada de erro *softmax* (GÓMEZ, 2018).

3.5.3 Otimizadores

O *backpropagation* faz uso de um algoritmo otimizador iterativo para reduzir o erro de saída da rede. Essas iterações, também chamadas de ciclos ou épocas, são executadas até que algum critério de parada seja acionado. O algoritmo padrão utilizado no *backpropagation* é o de gradiente descendente cujo objetivo é dar passos repetidos na direção oposta do gradiente da função no ponto atual (CASTRO, 2006).

Existem métodos de aprendizado diferentes para tratar o ajuste de pesos da rede com base no gradiente descendente (HAYKIN, 2008; BENGIO, 2012):

- Aprendizado em lote (*batch*): quando os ajustes dos pesos acontecem depois que todas as amostras são apresentadas à rede, ou seja, ao final de uma época de treinamento;
- Aprendizado em linha (*on-line*): quando a cada amostra, os pesos são atualizados, fazendo com que a busca por uma solução no espaço de pesos multidimensional aconteça de forma estocástica, já que as amostras são apresentadas em ordem randômica;
- Aprendizado em mini lotes (*mini-batch*): um intermediário entre os aprendizados em lote e em linha, realizando atualizações após computar tamanhos fixos de lotes.

Outros otimizadores que podem ser empregados ao invés do gradiente descendente estocástico² (SGD, *Stochastic Gradient Descent*) são RMSprop (HINTON; SRIVASTAVA; SWERSKY, 2014) e Adam (nome derivado de “estimativa de momento adaptativo”, ou *adaptive moment estimation*) (KINGMA; BA, 2014). Ambos, assim como SGD, são algoritmos baseados no cálculo do gradiente para encontrar um mínimo local de uma função diferenciável.

A diferença entre eles está na forma de lidar com a taxa de aprendizado: o SGD utiliza apenas uma taxa de aprendizado constante η para a atualização de todos os pesos da rede; RMSprop divide a taxa de aprendizado por uma média exponencialmente decrescente de gradientes quadrados; Adam, por sua vez, mantém taxas de aprendizado diferentes para cada parâmetro (peso), os quais são separadamente ajustados conforme o aprendizado se desenvolve.

3.5.4 Problema da Superfície Plana

Durante a execução do *backpropagation*, é preciso calcular o gradiente local de cada neurônio da MLP, o que requer conhecimento sobre a derivada da função de ativação associada àquele neurônio, justificando a necessidade desta ser contínua, diferenciável e não decrescente (HAYKIN, 2008).

Quando a saída do neurônio se aproxima de zero ou um, a depender da função de ativação utilizada, pode ser que a resposta, ou seja, o valor da derivada seja zero ou muito próximo a zero. Como o ajuste de pesos é feito baseando-se neste valor, este pode acabar sendo insignificante ou até mesmo não acontecer.

Esse problema é conhecido como superfície plana ou dissipação do gradiente (*vanish gradient*) (FAHLMAN, 1988) e pode ser bem ilustrado quando se observa a derivada da função logística, ilustrada na Figura 11a e definida como na Equação 23.

$$f'(u) = \frac{e^{-u}}{(1 + e^{-u})^2}. \quad (23)$$

Percebe-se, como descrito anteriormente e pela análise do gráfico que, para valores muito próximos de zero e um, a resposta da função de ativação será quase nula.

A função ReLU oferece uma solução a este problema. A sua derivada é como a ilustrada na Figura 11b e na forma da Equação 24:

$$f'(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases}, \quad (24)$$

Ou seja, sua resposta será sempre zero ou um, eliminando o problema da função logística. No entanto, caso muitas respostas sejam de ordem negativa, incorre-se noutro problema, conhecido como *dying ReLU* (LU *et al.*, 2020), pois não haverá atualização de pesos da rede. Por sua vez,

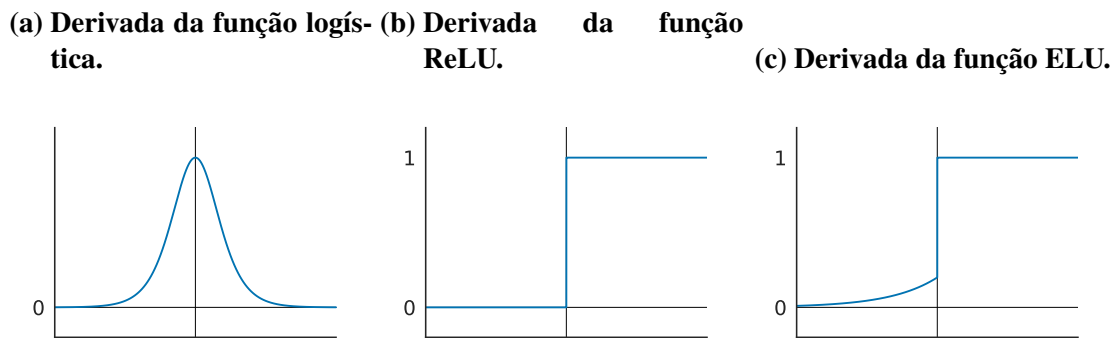
² Diz-se “estocástico” quando aplicado aos métodos de aprendizado em linha ou em mini lotes.

esse problema é resolvido quando utilizadas funções como ELU ilustrada na Figura 11c cuja derivada é (Equação 25):

$$f'(u) = \begin{cases} 1, & u > 0 \\ ae^u, & u \leq 0 \end{cases}, \quad (25)$$

Dessa forma, é fácil notar que o problema com a função ReLU é evitado para valores negativos, produzindo atualizações ajudando a rede a direcionar pesos e polarizações nas direções corretas (HANSEN, 2019).

Figura 11 – Derivadas de três funções de ativação diferentes.



Fonte: Autoria Própria (2022).

3.6 Aprendizado, Generalização e Validação Cruzada

O objetivo do treino das RNAs é alcançar um nível de generalização tal que seja possível fazer inferências corretas sobre dados que o sistema não conhece, já que todo o processo se resume a encontrar um mapeamento entre entradas e saídas para um conjunto de pesos. Entretanto, pode acontecer de que a RNA aparentemente encontre um conjunto generalizador para os dados com que foi treinada, mas não tenha bom desempenho com dados novos, o que caracteriza o fenômeno de *overfitting*, também chamado de sobre-treino (HAYKIN, 2008).

Para evitar esse problema, é preciso utilizar métodos que permitam avaliar a qualidade do processo de aprendizado. Um procedimento comum é separar os dados disponíveis em três conjuntos de dados distintos: treino, validação e teste.

O conjunto de treino é usado para o ajuste de pesos da RNA. Porém, ao final de cada época de treinamento, a rede usa o conjunto de validação para realizar previsões, um conjunto de dados imparcial que não foi utilizado durante a computação dos parâmetros, provendo uma estimativa melhor de taxa de erro ou acerto (JAMES *et al.*, 2013).

No entanto, usar apenas um valor final de estimativa para o sistema é geralmente insignificante sem haver um intervalo de confiança (KOHAVI, 1995). Por isso, um conjunto de teste, cujos dados ficam separados de todo o processo de treinamento, são usados para avaliar a qualidade final do modelo. Ao comparar os resultados obtidos com o conjunto de treino e o

conjunto de teste, se estes tiverem pouca diferença, pode-se concluir que não houve *overfitting*, algo desejável. Um conjunto de teste é, portanto, usado apenas para avaliar a generalização final de um modelo (RIPLEY, 2005).

Ainda, objetivando obter resultados mais estáveis, uma mesma base de dados pode ser dividida repetidamente em vários conjuntos de dados de treinamento e de validação, processo chamado de validação cruzada. Ou seja, para validar o desempenho do modelo, o processo de treinamento é repetido com conjuntos de dados de treino e validação diferentes ou não e avaliados ao final de cada repetição por um conjunto de testes. A média dos resultados de cada repetição é então considerada como sendo o verdadeiro resultado final (JAMES *et al.*, 2013).

É importante ressaltar que a escolha dos conjuntos de treino, validação e teste devem ser suficientemente representativos do mapeamento que se deseja alcançar. Em problemas de classificação, casos em que a base de dados apresente desbalanceamento de classes, isto é, quando há um número diferente de exemplos para cada classe, é recomendável fazer a separação estratificada, mantendo as divisões em grupos não sobrepostos. Isso pode ser feito com uma alocação proporcional, que usa uma fração de amostragem em cada um dos grupos que é proporcional ao total de amostras (HUNT; TYRRELL, 2004).

4 APRENDIZADO PROFUNDO

O Aprendizado Profundo (*Deep Learning*) é considerado uma subárea da IA, relacionado a aprendizado de máquina e RNAs, sendo considerado como um dos melhores métodos para realizar classificação de imagens (BENGIO, 2009).

A inspiração para o desenvolvimento desta área de estudo está no resultado de pesquisas neurocientíficas sobre a forma como o cérebro representa informação, em especial no fato de que não há necessariamente um pré-processamento dos sinais sensoriais, mas sim sua propagação em uma hierarquia complexa de módulos que aprendem com base na frequência que aparecem (AREL; ROSE; KARNOWSKI, 2010).

Métodos convencionais de aprendizado de máquina são limitados na habilidade de processar dados sem pré-processamento, sendo a tarefa de extração de características, que é a transformação de dados puros em representação interna adequada, uma necessidade inevitável (LECUN; BENGIO; HINTON, 2015). Uma forma mais interessante seria depender exclusivamente do algoritmo de aprendizado para esta tarefa, objetivo chave do Aprendizado Profundo (LECUN *et al.*, 1998).

O Aprendizado Profundo tem possibilitado o avanço e a solução de problemas que resistiam há anos na comunidade de IA, batendo recordes em tarefas de reconhecimento de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), reconhecimento de fala (HINTON *et al.*, 2012a), análise de dados de aceleração de partículas (CIODARO *et al.*, 2012) e reconstrução de circuitos cerebrais (HELMSTAEDTER *et al.*, 2013), por exemplo.

4.1 Rede Neural Convolutacional

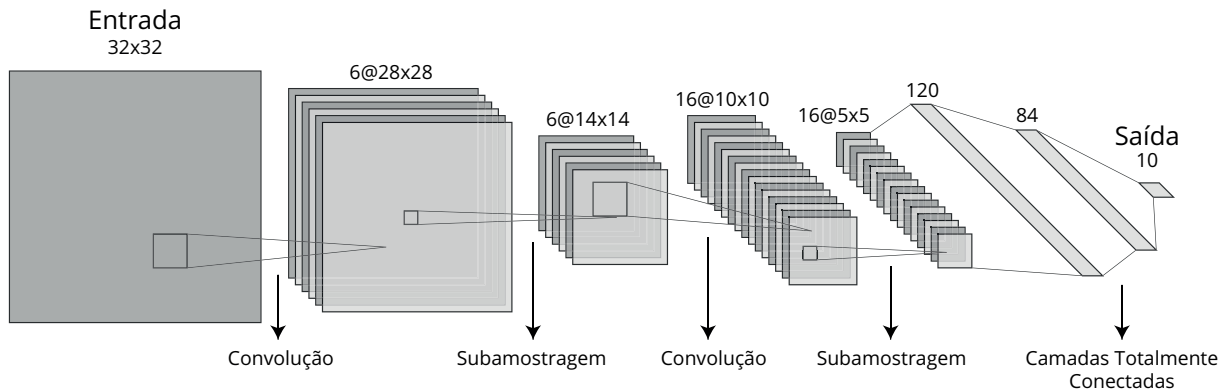
Com suas origens no trabalho de LeCun *et al.* (1998), a rede neural convolutacional, também chamada de ConvNet ou CNN (*Convolutional Neural Network*) é uma MLP desenhada para reconhecimento de padrões em duas dimensões com um alto grau de invariância à formas de distorção (HAYKIN, 2008), mas não limitada a isso, podendo trabalhar com vetores de mais ou menos dimensões, ideal para processamento de imagens e vídeos, por exemplo (LECUN; BENGIO; HINTON, 2015).

Segundo LeCun, Bengio e Hinton (2015), são quatro propriedades principais que ajudam as CNNs a ter melhor proveito de sinais sem pré-processamento: 1) conexões locais, 2) pesos compartilhados em uma mesma camada, 3) subamostragem (*pooling*) e 4) o uso de diversas camadas.

Diferentes arquiteturas de CNNs servem para diferentes propósitos. Por exemplo, a desenvolvida por Szegedy *et al.* (2015a) chamada de *GoogLeNet* tem por objetivo classificação de imagens, diferente da S-CNN (*Smoothed-CNN*) de Borovykh, Bohte e Oosterlee (2018), que serve para predição de séries temporais.

Uma arquitetura simples pode ser visualizada na Figura 12, sendo a idealizada por LeCun *et al.* (1998) e chamada de *LeNet*, com objetivo de classificação binária de imagens.

Figura 12 – Arquitetura da CNN idealizada por LeCun *et al.* (1998), onde uma imagem na entrada de 32×32 pixels sofre sucessivas convoluções e subamostragens, até no final passar por três camadas totalmente conectadas.



Fonte: LeCun *et al.* (1998).

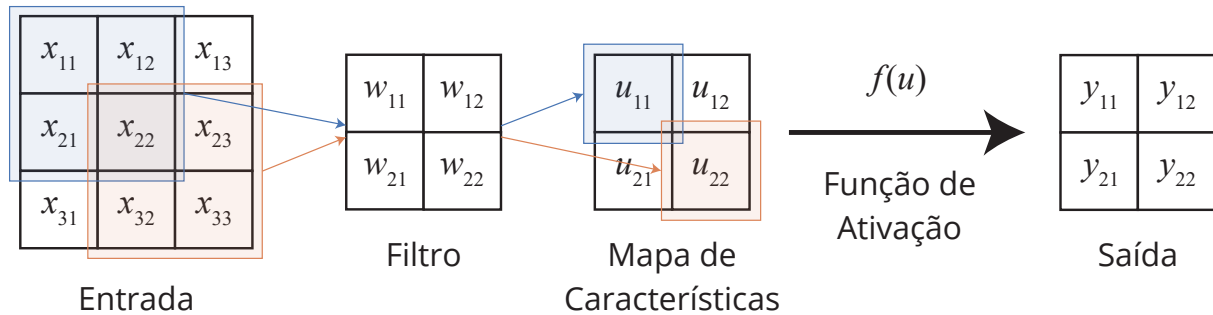
São três os tipos principais de camadas utilizadas em CNNs: camadas convolucionais, geralmente seguidas por camadas de subamostragem, e no final, camadas densamente conectadas que nada mais são do que MLPs, conforme discutido na Subseção 3.4.

4.1.1 Camada Convolutiva

A camada convolutiva pode ser considerada a mais importante em uma CNN, tendo como principal função a extração de características. Ela é composta por filtros bidimensionais (*kernels*) treináveis que são pequenos espacialmente, mas se estendem por toda a profundidade do volume de entrada. Esses filtros são então “deslizados” (operação de convolução) computando o produto escalar entre as entradas do filtro e a entrada em qualquer posição. A aplicação da convolução acontece por um número de distância em passos (*stride*). Cada filtro é responsável por detectar um tipo de característica, produzindo mapas de características (*feature maps*) individuais e diferentes (LI; KRISHNA; XU, 2021).

É natural que, após a operação de convolução, o resultado passe por uma função de ativação, por vezes denotado como uma camada da CNN. Na verdade, poder-se-ia considerar cada filtro e função de ativação que segue como um neurônio artificial, que compartilha seus parâmetros (pesos) com os demais do mesmo mapa de características. Ou seja, o filtro funciona como o conjunto de pesos do neurônio e como são compartilhados no mesmo mapa, tem-se uma operação de convolução. A Figura 13 ilustra essa afirmação (LI; KRISHNA; XU, 2021).

Figura 13 – Processo aplicado na camada convolucional: um conjunto de entradas x (em azul) é multiplicado pelos parâmetros w do filtro de acordo com sua posição e em seguida integrados em u que passa por uma função de ativação chegando na saída como y . O mesmo acontece para o outro conjunto de entradas (em alaranjado), porém passando pelos mesmos parâmetros do filtro, o que explica a condição de compartilhamento de parâmetros.



Fonte: Autoria Própria (2022).

Na Figura 13, uma entrada de tamanho 3×3 é processada por um filtro de 2×2 . O espaço analisado pelo filtro é inicialmente representado pelo quadrado azulado, sendo cada entrada multiplicada por um dos parâmetros do filtro, integrados e assim gerando uma única saída. O filtro desliza até que todas as entradas sejam processadas, sendo a posição ilustrada pelo quadrado alaranjado a última, gerando um mapa de características também de tamanho 2×2 . Por fim, esses valores ainda passam por uma função de ativação, constituindo a saída.

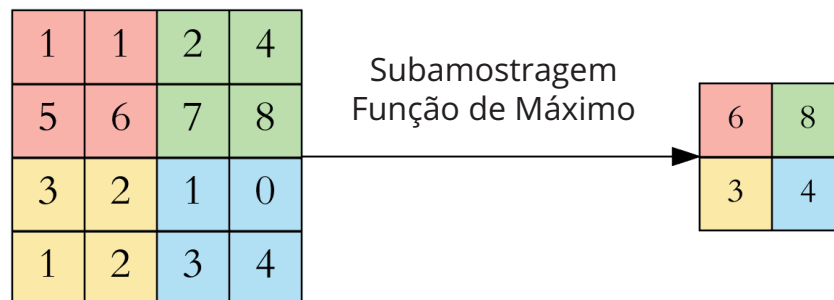
4.1.2 Camada de Subamostragem

A camada de subamostragem (ou *pooling*) é responsável por reduzir o tamanho espacial da representação dos dados, consequentemente diminuindo a quantidade de parâmetros da rede, operando de maneira independente. Em outras palavras, segundo Goodfellow, Bengio e Courville (2016), a função de subamostragem substitui a saída da rede em um determinado local por uma estatística resumida das saídas próximas. Isso é importante, porque com a aplicação de convoluções usando vários filtros, o volume de dados cresce muito, comprometendo o processamento do sistema. A subamostragem auxilia, reduzindo esse volume e possibilitando o funcionamento adequado do processo.

Diversas funções podem ser usadas para a subamostragem, como a média, norma L^2 , média ponderada ou o máximo de uma área retangular da entrada, sendo esta última a mais comum. Na Figura 14 é possível observar esta operação em detalhe.

Percebe-se pela Figura 14 que a subamostragem diminuiu a dimensão da entrada pela metade, como no exemplo: a entrada de 8×8 passou a ser de 4×4 . Isso porque ocorreu a análise de espaços de tamanho 2×2 , escolhendo sempre o maior valor dentre eles, para compor a saída.

Figura 14 – Operação de subamostragem usando a função de máximo, onde o máximo de cada quadrado colorido da esquerda é selecionado para compor a saída.



Fonte: Dertat (2017).

A sequência mais comum de aplicação destas camadas no desenvolvimento das CNNs é utilizar camadas de convolução seguidas de subamostragem diversas vezes, aumentando cada vez mais a profundidade da rede, tendo ao final as camadas densamente conectadas (MLPs) (HAYKIN, 2008). A ideia de convolução seguida de subamostragem é inspirada na noção de células “simples” seguidas de células “complexas”, descrita por Hubel e Wiesel (1962). Embora comum, não é uma regra, como pode-se observar nas estruturas de redes como a VGG19 (SIMONYAN; ZISSERMAN, 2014), que tem sequências de duas e quatro camadas convolucionas ligada diretamente, ou a *ResNet* (HE *et al.*, 2016), que tem diversas camadas convolucionais concatenadas. Assim sendo, há liberdade para a confecção das mais variadas arquiteturas de CNNs.

4.2 Questões Práticas para Aplicação de Redes Neurais

Existem diversos problemas e desafios relativos ao treinamento de RNAs para a tarefa de classificação, que afetam seu desempenho. A seguir, são discutidas três formas para melhorar o desempenho e solucionar alguns destes problemas.

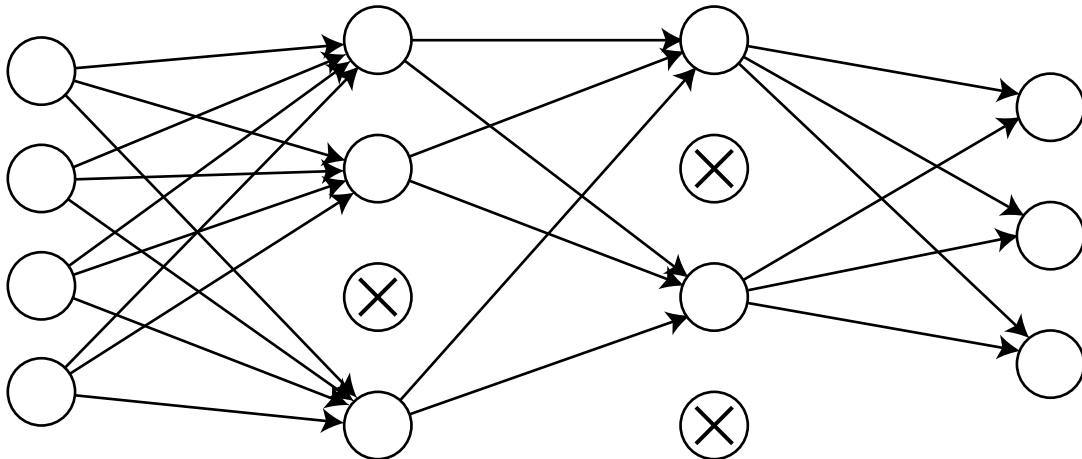
4.2.1 Dropout

Apesar das técnicas observadas na Seção 3.6, o problema de *overfitting* continua a representar um sério desafio de treinamento não só de CNNs, mas de RNAs em geral, principalmente quando estas são profundas e não há uma base de dados grande o suficiente, o que leva a rede geralmente se sair bem nos dados de treino, porém não nos dados de teste.

Uma forma de reduzir o *overfitting* é utilizando a técnica de *dropout* (significa abandonar, deixar fora), que consiste em ignorar a saída de neurônios das camadas escondidas com algum nível de probabilidade (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Ou seja, para

cada dado apresentado à rede durante o treinamento, um neurônio aleatório é omitido, fazendo com que este não possa contar com a presença de outros neurônios. A Figura 15 ilustra esse efeito.

Figura 15 – Ilustração do efeito de *dropout* em uma rede de duas camadas escondidas, onde alguns nós são excluídos aleatoriamente.



Fonte: Autoria Própria (2022).

Para uma rodada de treino, de acordo com o método de ajuste de pesos empregado (em linha, por lotes ou mini lotes), uma quantidade determinada de neurônios não irá operar. Na próxima rodada ou atualização, outros neurônios serão escolhidos. Isso evita coadaptações complexas nas quais uma camada só é útil no contexto de várias outras camadas específicas. Dessa forma, cada neurônio aprende a detectar uma característica que geralmente é adequada para produzir a resposta correta, dada a grande variedade combinatória de contextos internos nos quais deve operar (HINTON *et al.*, 2012b).

4.2.2 Normalização em Lotes

A normalização em lotes (*batch normalization*) é um método de reparametrização adaptativa motivada pela dificuldade em treinar RNAs muito profundas (GOODFELLOW; BENGIO; COURVILLE, 2016). Essa dificuldade está relacionada ao fato de que a distribuição das entradas de cada camada muda durante o treinamento como consequência dos ajustes dos parâmetros das camadas anteriores, o que causa diminuição da velocidade de treinamento pois requer valores mais baixos de taxa de aprendizagem, fenômeno nomeado de “mudança de covariável interna” pelos autores do método (IOFFE; SZEGEDY, 2015).

Embora não seja considerado um algoritmo de otimização (GOODFELLOW; BENGIO; COURVILLE, 2016), mas sim uma camada complementar das RNAs, trata-se tão somente da normalização das entradas de uma determinada camada para cada lote de treinamento, ou seja, é uma etapa de normalização que fixa as médias e variações das entradas de cada camada. Isso permite a utilização de valores maiores de taxa de aprendizado e, em alguns casos, eli-

mina a necessidade de uso de *dropout* por atuar como um regularizador (IOFFE; SZEGEDY, 2015). Outro impacto relacionado ao uso de *batch normalization* é tornar a função objetivo significativamente mais suave, induzindo um comportamento mais preditivo e estável dos gradientes (SANTURKAR *et al.*, 2018).

4.2.3 Otimização de Hiperparâmetros

Em um modelo de RNA, são considerados como parâmetros da rede os seus pesos, que são ajustados automaticamente durante a fase de treinamento. Hiperparâmetros, por sua vez, são as configurações relacionadas a arquitetura do modelo, por exemplo: quantidade de neurônios em determinada camada da rede, quantidade de camadas, tipos de camadas, valor da taxa de aprendizado, critério de parada, dentre outros. A escolha de hiperparâmetros pode acarretar em grande impacto no resultado final do classificador, e são muitas vezes escolhidos de maneira manual (YANG; SHAMI, 2020; WU *et al.*, 2019; SNOEK; LAROCHELLE; ADAMS, 2012).

A otimização de hiperparâmetros diz respeito a um processo automatizado com objetivo de encontrar o melhor conjunto p^* para um modelo, sendo este capaz de retornar o menor valor da função de custo $E(p)$, ou seja (Equação 26):

$$p^* = \arg \min_{p \in \rho} E(p) \quad (26)$$

para ρ como o domínio de valores de p (YANG; SHAMI, 2020).

Em outras palavras, quer-se encontrar os hiperparâmetros do modelo que geram o melhor resultado na métrica do conjunto de validação.

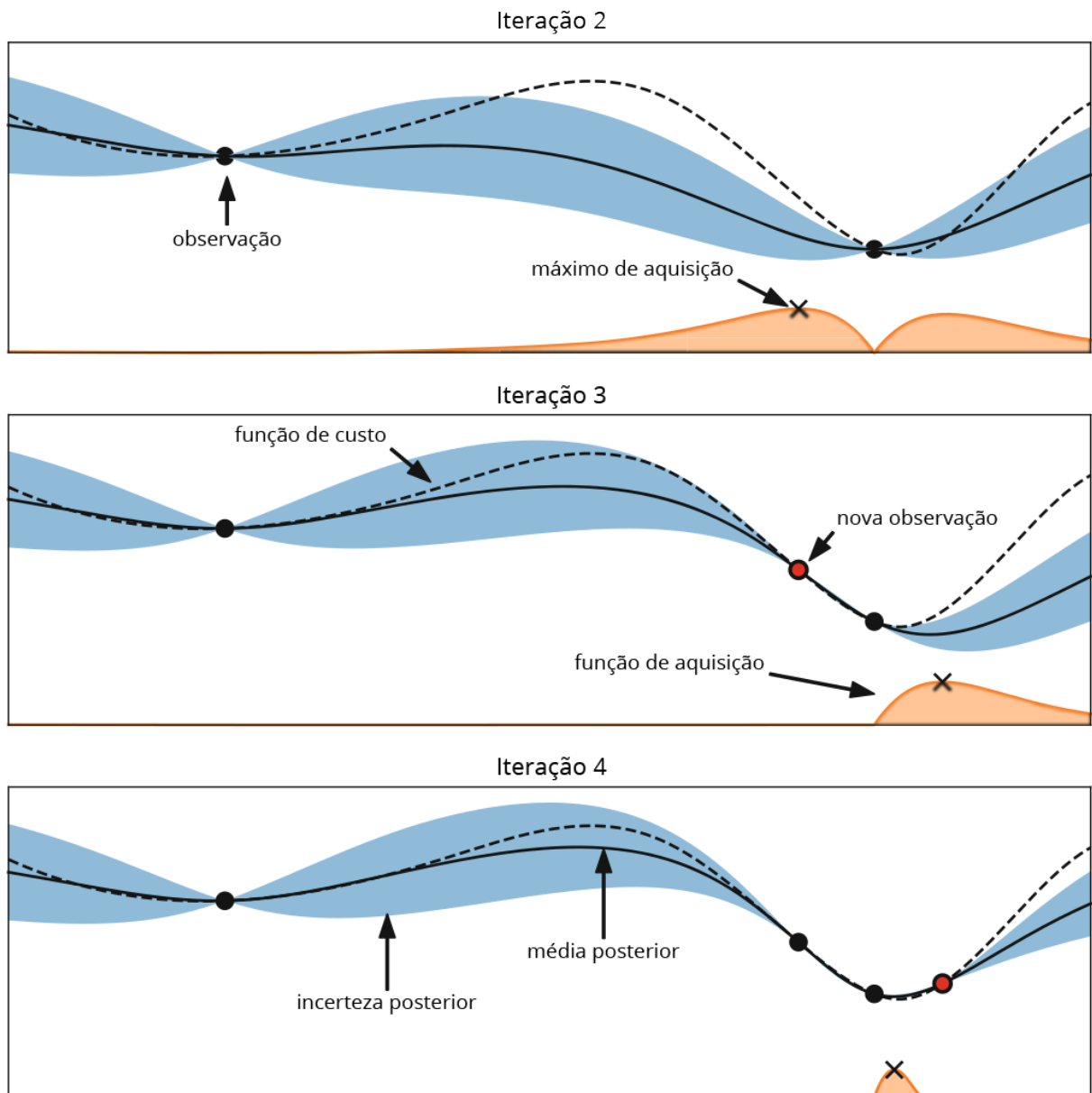
São diversos os benefícios do uso de métodos de otimização de hiperparâmetros, como redução de esforço humano, melhoria de performance dos modelos e melhoria da reprodutibilidade de estudos científicos (FEURER; HUTTER, 2019).

Existem diversas formas de aplicar a otimização de hiperparâmetros com algoritmos que independem do modelo em questão, como buscas em grade ou randômicas, otimização baseada no gradiente, Bayesiana, algoritmos de otimização multi-fidelidade e meta-heurísticos, cada qual com suas vantagens e desvantagens (YANG; SHAMI, 2020; FEURER; HUTTER, 2019).

A otimização Bayesiana é um método que leva em consideração o histórico de execuções do modelo. É uma forma similar a lógica humana de escolha, optando por configurações próximas que retornam melhores resultados. É composto por um modelo “substituto” e uma função de aquisição. Para cada iteração, o modelo substituto tem por objetivo ser treinado em todos os pontos observados até então na função de custo. Em seguida, a função de aquisição, que usa a distribuição preditiva do modelo probabilístico, determina a utilidade de diferentes pontos candidatos, alternando entre exploração e exploração, que são os atos de amostrar as instâncias nas áreas que não foram amostradas, e amostrar nas regiões atualmente promissoras onde o

ótimo global é mais provável de ocorrer, respectivamente. A Figura 16 ilustra a otimização de uma função unidimensional entre a segunda e quarta iteração do algoritmo.

Figura 16 – Ilustração da otimização Bayesiana de uma função unidimensional, com previsões mostradas como uma linha preta sólida e o modelo substituto como linha tracejada. A área azul representa a incerteza, enquanto que a área alaranjada é a função de aquisição.



Fonte: Feurer e Hutter (2019).

Na Figura 16, o objetivo é minimizar a linha tracejada usando um modelo substituto com um processo gaussiano maximizando a função de aquisição representada pela curva laranja inferior. No primeiro quadro, o valor de aquisição é baixo em torno das observações, e o valor de aquisição mais alto está em um ponto onde o valor da função predita é baixo e a incerteza preditiva é relativamente alta. Na terceira iteração, embora ainda haja muita variação à esquerda

da nova observação, a média prevista à direita é muito menor, portanto a próxima observação é realizada naquela área. Na quarta iteração, mesmo quase não havendo incerteza em torno da localização do máximo verdadeiro, a próxima avaliação é feita naquele espaço devido à sua melhoria esperada em relação ao melhor ponto até o momento.

Com o embasamento sobre RNAs e aprendizado profundo concluído, é importante entender os métodos para avaliar o desempenho destes sistemas. Para tanto, diversas medidas estatísticas podem ser aplicadas, como se verá a seguir no Capítulo 5.

5 MEDIDAS DE DESEMPENHO

A avaliação da produção de um sistema harmonizador de melodias automático pode ser feita usando uma abordagem quantitativa, com o uso de medidas estatísticas ou indicadores calculados a partir dos resultados de execução de um determinado modelo. Estes são úteis quando o objetivo é avaliar e comparar diferentes modelos de classificação ou técnicas de aprendizado (GRANDINI; BAGLI; VISANI, 2020).

A métrica mais conhecida e utilizada na literatura em um problema de classificação, é a taxa de acerto ou acurácia. Entretanto, a depender da distribuição de classes da base de dados e outros fatores, esta não é a ideal para comparação entre sistemas (PROVOST; FAWCETT; KOHAVI, 1998).

Não obstante, é possível fazer uso de diversas outras métricas que oferecem maior confiabilidade e comparabilidade provenientes da matriz de confusão, uma tabela que identifica a quantidade de ocorrências entre valores reais ou esperados e as saídas previstas pelo modelo.

5.1 Matriz de Confusão e Métricas Relacionadas

Também chamada de matriz de erro ou tabela de confusão, a matriz de confusão é uma estrutura que contém basicamente quatro categorias: Verdadeiros Positivos (VP) são exemplos corretamente classificados como positivos e Verdadeiros Negativos (VN) são classificações negativas corretas; Falsos Positivos (FP) são exemplos negativos incorretamente classificados como positivos, enquanto os Falsos Negativos (FN) são o oposto, exemplos positivos classificados como negativos de maneira incorreta (DAVIS; GOADRICH, 2006). O Quadro 4 ilustra a matriz de confusão, bem como algumas das diversas medidas que se podem dela extrair.

Quadro 4 – Construção de uma matriz de confusão binária com métricas relativas calculadas utilizando índices diretamente da matriz.

		Verdadeiro		Acerto (AC) $= \frac{VP+VN}{VP+FP+VN+FN}$
		Condição Positiva	Condição Negativa	
Predito	Condição Positiva Prevista	Verdadeiro Positivo (VP)	Falso Positivo (FP)	Precisão (P) $= \frac{VP}{VP+FP}$
	Condição Negativa Prevista	Falso Negativo (FN)	Verdadeiro Negativo (VN)	
		Sensibilidade (S) $= \frac{VP}{VP+FN}$		Medida F $= 2 \frac{P \cdot S}{P+S}$

Fonte: Autoria Própria (2022).

Todas estas medidas são de interesse quando se deseja avaliar quantitativamente um ou vários modelos de classificação. Entretanto, é importante notar que estão todas definidas para o tipo de classificação binária, ou seja, apenas duas classes: positivo ou negativo. Mesmo assim, é possível elaborar a matriz de confusão para um problema de multiclassificação, tal como ilustrado no Quadro 5.

Quadro 5 – Matriz de confusão para o caso multiclasse para um total de \mathcal{C} classes.

		Verdadeiro			
Predito		C_{11}	C_{12}	\dots	$C_{1\mathcal{C}}$
		C_{21}	C_{ij}		
		\vdots		\ddots	
		$C_{\mathcal{C}1}$			$C_{\mathcal{C}\mathcal{C}}$

Fonte: Autoria Própria (2022).

O Quadro 5 ilustra como seria a construção da matriz de confusão para \mathcal{C} classes. Em suma, os elementos diagonais são classificações corretas, e todos os outros, incorretas (JAMES *et al.*, 2013). Portanto, para poder se utilizar das métricas definidas no Quadro 4, é preciso realizar algumas adaptações, as quais são vistas a seguir.

Antes, porém, faz-se útil definir as equações para cálculo dos índices centrais do Quadro 4 para o caso multiclasse. Portanto, para cada classe m , calcula-se (KAUTZ; ESKOFIER; PASLUOSTA, 2017):

- Verdadeiros Positivos da classe m (Equação 27):

$$VP_m = C_{mm}. \quad (27)$$

- Verdadeiros Negativos da classe m (Equação 28):

$$VN_m = \sum_{i=1, i \neq m}^{\mathcal{C}} \sum_{j=1, j \neq m}^{\mathcal{C}} C_{ij}. \quad (28)$$

- Falsos Positivos da classe m (Equação 29):

$$FP_m = \sum_{i=1, i \neq m}^{\mathcal{C}} C_{mi}. \quad (29)$$

- Falsos Negativos da classe m (Equação 30):

$$FN_m = \sum_{i=1, i \neq m}^{\mathcal{C}} C_{im}. \quad (30)$$

5.1.1 Acurácia

A acurácia ou taxa de acerto (AC) é uma medida que diz de forma geral o quão bem um modelo está classificando corretamente os dados que lhe são apresentados. O cálculo da AC para o caso multiclasse pode ser definido pela Equação 31:

$$AC = \frac{\sum_{m=1}^{\mathcal{C}} VP_m}{\sum_{m=1}^{\mathcal{C}} VP_m + FP_m}, \quad (31)$$

ou seja, a soma de todas as classificações corretas (valores da diagonal) dividido pela quantidade total de elementos. Em outras palavras, a AC é a probabilidade de que a classificação de um modelo esteja correta (GRANDINI; BAGLI; VISANI, 2020).

A AC é uma medida problemática quando se trata de uma base de dados desbalanceada porque o impacto dos exemplos menos representados, mas mais importantes, é reduzido quando comparado ao da classe majoritária (BRANCO; TORGO; RIBEIRO, 2015).

5.1.2 Medida F

Mais comumente conhecida como F -score ou F -measure, a medida F traça uma relação entre precisão e sensibilidade. No caso binário, a precisão (P) é a proporção de predições positivas entre as realmente positivas, enquanto que a sensibilidade (S) se refere a proporção de predições positivas que são realmente positivas (POWERS, 2007).

A forma geral de definição da F -score é utilizando um valor β que decide quantas vezes a sensibilidade é mais importante que a precisão, chamada de F_β e calculada como na Equação 32:

$$F_\beta = (1 + \beta^2) \frac{P \cdot S}{(\beta^2 \cdot P) + S}. \quad (32)$$

No entanto, quando não há diferença de importância entre classes, ou seja, quando a precisão é tão importante quanto a sensibilidade, é comum aplicar $\beta = 1$, transformando a F -score em uma média harmônica entre precisão e sensibilidade (GRANDINI; BAGLI; VISANI, 2020).

Quando analisado o problema multiclasse, a F -score precisa envolver todas as classes, o que leva ao desenvolvimento de duas métricas diferentes: Micro F -score ($F_{1\mu}$) e Macro F -score (F_{1M}). Os termos Micro e Macro estão relacionados à forma que a matriz de confusão multiclasse é analisada. Sob a perspectiva Macro, os valores de performance são computados para cada classe separadamente e depois obtém-se sua média, enquanto que para a forma Micro esses valores são computados diretamente da matriz de confusão (JURAFSKY; MARTIN, 2008).

A $F_{1\mu}$ depende da definição de Microprecisão (P_μ) e Microsensibilidade (S_μ), cujo objetivo é considerar todo o conjunto sem levar em consideração diferenças entre classes.

É interessante notar que a soma de todos os FP_m é igual a soma de todos FN_m , ou seja, $\sum_{m=1}^{\mathcal{C}} FP_m = \sum_{m=1}^{\mathcal{C}} FN_m$. Isso permite concluir que P_μ é igual à S_μ e, conseqüentemente, igual à $F_{1\mu}$. Indo além, pode-se dizer que a $F_{1\mu}$ é o mesmo cálculo que da AC (Equação 31) (GRANDINI; BAGLI; VISANI, 2020).

Por outro lado, a F_{1M} considera primeiramente o valor da *F-score* para cada classe, calculando precisão e sensibilidade individualmente e depois, obtendo a média aritmética. Sendo assim, a precisão de uma classe m é definida como na Equação 33:

$$P_m = \frac{VP_m}{VP_m + FP_m}, \quad (33)$$

e a sensibilidade é na forma da Equação 34:

$$S_m = \frac{VP_m}{VP_m + FN_m}. \quad (34)$$

Usando as Equações 33 e 34 e adaptando a Equação 32 com $\beta = 1$, pode-se definir a F_{1M} na Equação 35:

$$F_{1M} = \frac{1}{\mathcal{C}} \sum_{m=1}^{\mathcal{C}} 2 \frac{P_m \cdot S_m}{P_m + S_m}. \quad (35)$$

O cálculo da F_{1M} pode ser interpretado de outra forma: em vez de obter a precisão e sensibilidade de cada classe e depois uma média da *F-score*, pode-se calcular antes a média de precisão e sensibilidade e usar seus valores diretamente na Equação 32 (SOKOLOVA; LAPALME, 2009). Entretanto esta não é recomendável por ser menos robusta quanto a distribuição do tipo de erro (OPITZ; BURST, 2019).

Outras duas métricas interessantes que podem ser obtidas da matriz de confusão são o Coeficiente de Correlação de Matthews (MCC, *Matthews Correlation Coefficient*) e o Coeficiente de Concordância de Kappa (κ).

5.1.3 Correlação de Coeficiente de Matthews

A Correlação de Coeficiente de Matthews (MCC) é uma medida desenvolvida por Matthews (1975) definida de maneira similar ao Coeficiente Phi de Pearson (CRAMER, 1962). É uma métrica de interesse pois leva em conta todos os índices da matriz de confusão, verdadeiros e falsos positivos e negativos, com um resultado no intervalo $[-1, 1]$. Valores próximos de 1 indicam boas predições, em outras palavras, uma correlação positiva forte entre valores preditos e reais; valores próximos de -1 indicam correlação inversa ao passo que 0 é o resultado de quando não há correlação.

A definição do MCC para cálculo multiclasse é como na Equação 36 (GORODKIN, 2004; SCIKIT-LEARN, 2021):

$$\text{MCC} = \frac{cs - \sum_{m=1}^{\mathcal{C}} p_m r_m}{\sqrt{(s^2 - \sum_{m=1}^{\mathcal{C}} p_m^2)(s^2 - \sum_{m=1}^{\mathcal{C}} r_m^2)}}, \quad (36)$$

para:

- $c = \sum_{m=1}^{\mathcal{C}} \text{VP}_m$, total de predições corretas;
- $s = \sum_{m=1}^{\mathcal{C}} (\text{VP}_m + \text{FP}_m)$, soma de todas as predições;
- $r_m = \text{VP}_m + \text{FN}_m$, número de vezes que a classe m realmente aconteceu;
- $p_m = \text{VP}_m + \text{FP}_m$, número de vezes que a classe m foi predita.

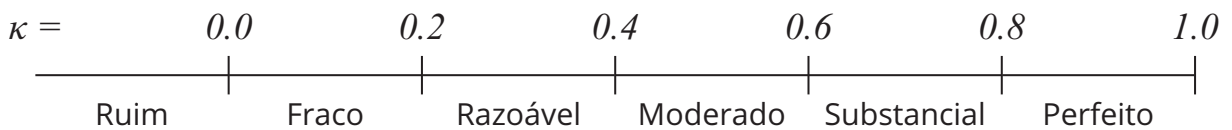
O MCC apresenta um bom compromisso entre discriminação e consistência, além de comportamento coerente com número variável de classes, conjuntos de dados desbalanceados e randomização (JURMAN; RICCADONNA; FURLANELLO, 2012).

5.1.4 Coeficiente de Concordância de Kappa

Também conhecido por *Cohen's kappa* (κ), é um coeficiente para medir o grau de concordância em escalas nominais e para fornecer meios de testar hipóteses e definir limites de confiança (COHEN, 1960). Em outras palavras, é um cálculo de concordância entre observadores ou classes levando em consideração a concordância esperada ao acaso, de forma randômica (RANGANATHAN; PRAMESH; AGGARWAL, 2017).

Assim como o MCC, o valor de κ varia no limite $[-1, 1]$, podendo ser interpretado de forma arbitrária, sendo que $\kappa = 0$ significa concordância randômica e conforme o valor for aumentando, maior o nível de concordância, sendo $\kappa = 1$ indicativo de total concordância entre variáveis. Valores negativos indicam que o nível de concordância é ainda pior do que randômico (RANGANATHAN; PRAMESH; AGGARWAL, 2017). A Figura 17 ilustra os níveis de concordância para cada intervalo de valores do coeficiente.

Figura 17 – Níveis subjetivos de concordância para diferentes intervalos de valores de κ .



Fonte: Artstein e Poesio (2008).

Para calcular κ da matriz de confusão multiclasse, considerando as mesmas variáveis c , s , r_m e p_m usadas na Equação 36, tem-se a Equação 37 (TALLÓN-BALLESTEROS; RIQUELME, 2014):

$$\kappa = \frac{cs - \sum_{m=1}^{\mathcal{C}} p_m r_m}{s^2 - \sum_{m=1}^{\mathcal{C}} p_m^2 r_m^2}, \quad (37)$$

similar à definição de MCC, diferindo no denominador da equação, fazendo com que o resultado prático final geralmente seja maior.

Agora que se sabe sobre conceitos básicos da teoria musical (Capítulo 2), RNAs (Capítulo 3) e aprendizado profundo (Capítulo 4), bem como formas de determinar o desempenho destes sistemas, visto neste Capítulo 5, é importante estabelecer o estado da arte do tema, ou seja, os meios encontrados na literatura para a solução do problema de harmonização musical automática, exposto no Capítulo 6.

6 ESTADO DA ARTE

Há décadas computadores vêm sendo utilizados para a tarefa de composição musical. Com foco na composição automática, pode-se entender a harmonização automática de melodias como uma extensão da composição algorítmica, a qual consiste na aplicação de algoritmos de inteligência artificial, especialmente de aprendizado de máquina, durante o processo de concepção de uma obra musical (MAKRIS; KAYRDIS; SIOUTAS, 2013). A harmonização automática possibilita, então, um processo de composição assistida. As técnicas mais empregadas para alcançar tal feito são algoritmos baseados em regras (SHEN; LEE, 2011; WU; CHEN, 2016b; CHATHURANGA; RATNAYAKE; PREMARATNE, 2017), modelos matemáticos (em especial, usando HMM ou *Hidden Markov Model*) (SIMON; MORRIS; BASU, 2008; SOYSA; LOKUGE, 2010; RADICIONI; ESPOSITO, 2010), métodos evolutivos, frequentemente sendo usados Algoritmos Genéticos (AGs) (PRISCO; ZACCAGNINO, 2009; YOU; LIU, 2016), com aprendizado de máquina, como com emprego de RNAs (LIM; LEE, 2017; DUA *et al.*, 2020; DE BOOM *et al.*, 2020) ou com abordagens híbridas (RACZYŃSKI; FUKAYAMA; VINCENT, 2013; MAJIDI; TOROGHI, 2021), nas suas mais diversas formas, com o uso de informações estatísticas e descritivas extraídas de conjuntos de treinamento, sendo estes normalmente feitos de obras musicais existentes, representadas de maneira simbólica, por exemplo, através do protocolo MIDI ou em MusicXML (MAKRIS; KAYRDIS; SIOUTAS, 2016).

Além disso, percebem-se diferentes objetivos quanto às entradas e saídas dos sistemas. Embora, em suma, seja colocada uma melodia na entrada e para esta gerados acordes, outros trabalhos têm por objetivo a geração de progressão de acordes com base em regras de teoria musical (BERNARDES *et al.*, 2016; CHUAN; CHEW, 2011) ou emoções (WU; CHEN, 2016a; MAGALHÃES; KOOPS, 2014), ou buscam fazer o processo inverso: a partir de acordes, gerar melodias ou toda uma música (DONG *et al.*, 2018; HAO, 2019). Como produção destes sistemas, os objetivos são normalmente a geração de notação de acordes (símbolos) (RYBNIK; HOMENDA, 2012), acompanhamentos já arranjados para execução (YOU; LIU, 2016) ou harmonização de vozes (FUKUMOTO, 2014).

A análise de resultados de cada trabalho varia de acordo com os métodos empregados. Para sistemas baseados em regras, os resultados são em maioria expressos de forma descritiva, analisando as respostas geradas e seguindo regras de harmonia (EBCIOĞLU, 1988; RYBNIK; HOMENDA, 2012). Modelos matemáticos fazem análises quantitativas, usando métricas de performance relacionadas ao próprio algoritmo e nem sempre à qualidade da música em si (TSUSHIMA; NAKAMURA; YOSHII, 2020; WASSERMANN; GLICKMAN, 2020). Os que usam AGs tendem a analisar resultados provenientes do processo de busca do próprio algoritmo, como a análise de funções de avaliação (*fitness*) (WIGGINS *et al.*, 1998; FREITAS; GUIMARÃES; RUELA, 2011). Por fim, trabalhos que envolvem o uso de RNAs fazem uso de avaliações qualitativas de forma similar ao que acontece com modelos matemáticos, com métricas próprias elaboradas para avaliação específica (DUA *et al.*, 2020; DE BOOM *et al.*,

2020). Poucos usam métricas estatísticas que permitem comparabilidade de maneira genérica, como taxa de acertos e erros frente a harmonias de músicas reais (LIM; LEE, 2017), bem como avaliações qualitativas com a participação de ouvintes ou análises de profissionais (DE BOOM *et al.*, 2020).

Sistemas de composição baseados em regras abordam o campo da representação de conhecimento. O conjunto de regras definido é considerado como todo o conhecimento que o algoritmo tem para a produção, semelhante a um estudante que aprende profundamente a teoria, mas não possui experiência. Dessa forma, Rybnik e Homenda (2012) apresentam um sistema capaz de gerar acordes a partir de cálculos de pesos sobre as notas de uma melodia, definidos de acordo com regras da teoria musical, mais especificamente com a determinação de importância das notas levando em conta suas durações, posições no compasso, altura e acentuação. Ao final, o sistema foi capaz de produzir harmonias complexas ou simples, de acordo com configurações propostas.

Koops, Magalhães e Haas (2013) propuseram um sistema automático para harmonização chamado FHARM, baseado em um modelo de harmonia funcional. O seu diferencial está na garantia de que ao menos regras básicas de harmonia sejam seguidas no resultado final. A partir de dois experimentos, o primeiro com resultados analisados por especialistas em harmonia e o segundo com comparação da harmonia gerada para determinada melodia com a harmonia elaborada por um estudante de música, foi possível confirmar que o sistema gera harmonias realísticas.

Métodos matemáticos apresentam análises interessantes e complexas, como é o caso do trabalho de Hu, Guan e Zhou (2010) em que os autores introduzem uma abordagem híbrida para modelar o processo de harmonização por meio da expansão de uma árvore binária e também do emprego de um algoritmo de busca bidirecional, de forma que foi possível capturar as dependências globais da progressão de acordes e captar toda a métrica da melodia em vez de apenas considerar dependências locais. De maneira similar, Tsushima, Nakamura e Yoshii (2020) trabalham com árvores descritas por um algoritmo semi-supervisionado de gramática probabilística livre de contexto, encontrando resultados melhores que os utilizando HMM convencionais.

AGs baseiam-se na ideia do processo evolutivo por seleção natural. Por causa disso, esses algoritmos têm sido amplamente utilizados para a tarefa de harmonização automática. Freitas, Guimarães e Ruela (2011), por exemplo, em seu trabalho, definem duas funções de *fitness*, uma de simplicidade e outra de dissonância. Em conjunto, estas caracterizam um processo multiobjetivo que revelou flexibilidade para aplicação ou não de todas as regras definidas, dando uma característica de criatividade ao sistema, ao apresentar soluções factíveis, algo inalcançável por um sistema somente baseado em regras. Uma importante conclusão a que chegaram foi de que sistemas capazes de analisar compassos de música como um todo seriam interessantes, já que a maior parte o faz considerando no máximo apenas o acorde seguinte.

De forma híbrida, o trabalho de Majidi e Toroghi (2021) apresenta o uso de um AG multiobjetivo (*Multi-Objective Genetic Algorithm* ou MO-GA) e BLSTMs, sendo o primeiro para geração de partes de músicas polifônicas, e o segundo para dar total autonomia ao sistema, removendo a necessidade de ouvintes humanos, com duas RNAs simulando tanto ouvintes regulares quanto experientes. Os resultados mostraram que o uso de RNAs e outras funções objetivo de métodos generativos em conjunto acelera a convergência para a solução.

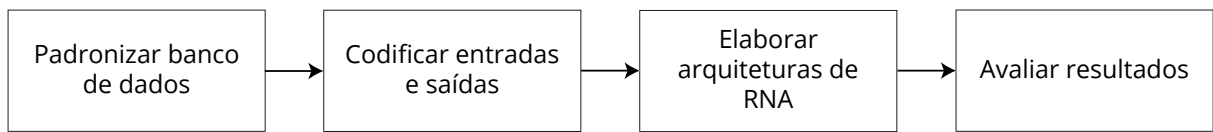
Nesse sentido, Lim e Lee (2017) apresentam uma possibilidade de composição de harmonias usando Redes *Bidirectional Long Short-Term Memory* (BLSTM) treinadas com base em dados simbólicos de harmonias de músicas reais. Tais redes são capazes de refletir sobre o contexto musical nos dois sentidos temporais, tendo em mente que acordes são formados com base na ordem de seleção de notas, tanto à frente quanto passadas. Sendo assim, cada compasso de música era analisado e apresentava, então, um acorde gerado correspondente. Comparado com resultados de outros métodos convencionais, ouvintes preferiram as harmonias geradas por este modelo.

Outra abordagem similar foi feita por Dua *et al.* (2020), estendendo a problemática e fazendo a análise de melodias a partir de trechos de áudio cantados em língua inglesa, usado arquiteturas de Redes LSTM. Uma conclusão interessante dos autores foi de que, para o problema de geração de acordes, ainda não foram alcançadas taxas de acerto consideráveis e ainda há muito espaço para estudos que visam melhoria.

7 METODOLOGIA

O método proposto neste trabalho consiste nos passos ilustrados pelo fluxograma da Figura 18. Primeiramente, deve-se realizar a padronização do banco de dados em níveis melódicos, rítmicos e harmônicos. Posteriormente, é necessário elaborar esquemas de codificação para que estes dados possam ser usados e entendidos pelos sistemas harmonizadores. Com base nas formas de codificação possíveis e avaliando o problema, é preciso modelar RNAs que sejam capazes de atingir os objetivos propostos, sendo que o sucesso deverá ser avaliado com base em resultados qualitativos e quantitativos.

Figura 18 – Diagrama de blocos explicativo da metodologia proposta neste trabalho.



Fonte: Autoria Própria (2022).

7.1 Descrição da Base de Dados

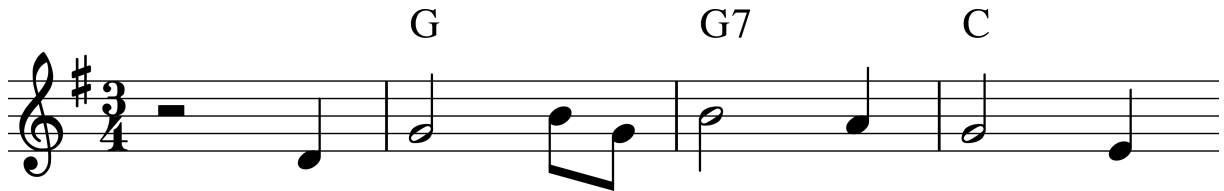
A forma convencional de se escrever acordes usando notação musical é colocando todas as notas diretamente na pauta. Entretanto, introduzida no século XX, a notação passou a ser mais flexível para um formato de partitura denominado “folha de condução” (*lead sheet*), que consiste em um sistema para denotar acordes usando símbolos da música popular. Ou seja, em vez de especificar exatamente quais notas devem ser tocadas para a harmonia, o compositor deixa o músico na liberdade de escolher, anotando somente com um símbolo o acorde que deve ser tocado em união à melodia (BENWARDS; SAKER, 2008).

As formas de se representar ou armazenar música são das mais variadas possíveis. Desde gravações em áudio, simples letras com cifras, usando notação musical ou com formatos digitais dedicados, como MIDI (ASSOCIATION, 2014) e MusicXML (GOOD, 2013). O formato MusicXML é um padrão de acesso livre desenvolvido em 2000 que usa sintaxe similar à de arquivos XML para escrever estruturas musicais, ou seja, com notação musical, como as próprias *lead sheets*. Além disso, músicas escritas neste formato também podem ser lidas por diversos softwares sintetizadores, e a música pode ser tocada ou reproduzida, ou seja, é possível sintetizar de forma audível o que está escrito. Também, é um tipo de arquivo de fácil processamento, graças a sua estrutura hierárquica de blocos, permitindo a extração de informações que podem ser usadas computacionalmente.

Diante disso, Lim, Rhyu e Lee (2017) processaram 2252 *lead sheets* em formato MusicXML provenientes de um banco de dados de público acesso à época chamado Wikifonia.org, desativado em 2013. Aproveitando-se da formatação dos arquivos, extraíram diversas caracte-

terísticas musicais e transformaram em uma base de dados CSV. A transformação é melhor entendida se visualizada a Figura 19 e a Tabela 6.

Figura 19 – Trecho da música *Amazing Grace*, de John Newton, em notação musical.



Fonte: Autoria Própria (2022).

Quadro 6 – Conteúdo do trecho da música *Amazing Grace*, da Figura 19, dentro do banco de dados *CSV Leadsheet*.

Tempo	Compasso	Tônica do acorde	Tipo do acorde	Nota da melodia	Duração da nota
3/4	1	Sem acorde	Sem acorde	Pausa	8
3/4	1	Sem acorde	Sem acorde	Ré	4
3/4	2	Sol	Maior	Sol	8
3/4	2	Sol	Maior	Si	2
3/4	2	Sol	Maior	Sol	2
3/4	3	Sol	Maior com 7 ^a	Si	8
3/4	3	Sol	Maior com 7 ^a	Lá	4
3/4	4	Dó	Maior	Sol	8
3/4	4	Dó	Maior	Mi	4
...

Fonte: Autoria Própria (2022).

O mesmo compasso musical escrito em formato MusicXML pode ser descrito pela tabela, que tem informações sobre tempo, compasso, tonalidade, nota e tipo de acorde, nota da melodia e sua duração, para cada ocorrência da música. Por exemplo, comparando a primeira nota do segundo compasso com a terceira linha da tabela, percebe-se que as informações são condizentes, uma vez que esse momento da música tem fórmula de compasso igual a $\frac{3}{4}$, a localização é o segundo compasso, a nota do acorde tocado é Sol (G) e seu tipo é maior, a nota da melodia também é Sol e sua duração é de 8 o que caracteriza uma mínima (\downarrow) de acordo com a padronização adotada na base de dados. Na base de dados *CSV Leadsheet*, a duração de cada nota está padronizada em valores de acordo com a Tabela 7¹.

¹ Como exposto no Capítulo 2, a duração da semibreve é comumente definida pelo valor inteiro 1, e as demais figuras rítmicas são frações desta. A diferença é que, nesta base de dados, definiu-se a semibreve como sendo igual a 16.

Quadro 7 – Durações das figuras rítmicas de acordo com a padronização da base de dados selecionada, comparada com a duração convencional, que são frações da semibreve.

Figura rítmica	Duração	Duração convencional
Semibreve (o)	16	1
Mínima (♩)	8	$\frac{1}{2}$
Semínima (♪)	4	$\frac{1}{4}$
Colcheia (♫)	2	$\frac{1}{8}$
...

Fonte: Autoria Própria (2022).

Tal base de dados se mostra ideal para a tarefa proposta neste trabalho, facilitando as fases que seguirão necessárias no desenvolvimento do projeto, sendo então a selecionada. Portanto, entende-se que a natureza do problema de harmonização automática é de classificação (HAYKIN, 2008), ou seja, dadas as notas musicais melódicas de uma entrada, deve-se classificar qual acorde melhor a harmoniza. Por isso tal estrutura de base de dados é tão importante, pois contém todas as informações relacionando melodia e harmonia.

7.2 Padronização da Base de Dados

Como os dados da base *CSV Leadsheet* tratam-se de músicas dos mais diferentes estilos e tipos, é necessário realizar uma padronização em termos musicais, para que haja conformidade entre obras. Essa padronização deve acontecer em três níveis: melódico, harmônico e rítmico.

7.2.1 Padronização de Informações Melódico-Harmônicas

No sentido melódico, é preciso padronizar a tonalidade para que todas as músicas compartilhem o mesmo tom. Isso acontece realizando a transposição tanto de notas como acordes para uma tonalidade comum, para que exista consistência e capacidade comparativa entre músicas, pois haverá então um conjunto de notas esperadas (escala) e um conjunto de acordes comum (campo harmônico). Para tanto, poder-se-á valer da escolha de qualquer tom, sendo que o mais natural é usar o de Dó maior.

A padronização harmônica, por sua vez, se dá na simplificação da quantidade de acordes. A possibilidade de construção de acordes é imensa, o que pode tornar a tarefa de classificação em algo demasiadamente difícil, bem como a análise de resultados. Portanto, uma forma de conduzir o processo é limitar-se ao uso de tríades (acordes de três notas) maiores e menores, pois estas sozinhas conseguem representar as funções harmônicas de uma música. A Tabela 8 mostra todos os acordes que serão considerados. Dessa forma, a classificação fica restrita a escolher dentre 24 possibilidades de acordes.

Quadro 8 – Acordes considerados durante a tarefa de classificação do sistema: tríades maiores e menores.

Maiores	Menores
C	Cm
C#	C#m
D	Dm
D#	D#m
E	Em
F	Fm
F#	F#m
G	Gm
G#	G#m
A	Am
A#	A#m
B	Bm

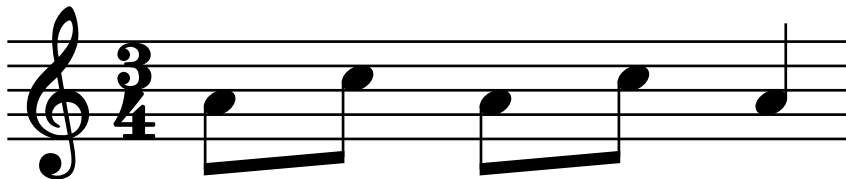
Fonte: Autoria Própria (2022).

7.2.2 Padronização de Informações Rítmicas

Como observado no Capítulo 2, a duração de um compasso musical é regida por uma fórmula de compasso, que indica a quantidade e o tipo de figura rítmica que o preenche. Existem diversas fórmulas de compasso possíveis, o que causa um problema, pois compassos de músicas com fórmulas diferentes terão duração final diferente. Isso evidencia a necessidade de normalização.

Fazendo memória da Figura 1, que contém uma comparação da divisão de durações das figuras rítmicas, constata-se que por padrão, a duração s da semibreve (figura com maior duração musical) é igual a 1, e as demais durações são frações dela, sendo que a fração está diretamente relacionada ao número do denominador da fórmula de compasso FC . O numerador por sua vez, indica o total desse tipo de figura que vai completar um compasso. Por exemplo, quando $FC = \frac{3}{4}$, significa que um compasso estará completo quando a soma de durações das figuras rítmicas for equivalente a três semínimas (\downarrow , $\frac{1}{4}$ da duração de s). Mas se $FC = \frac{7}{8}$, então são sete colcheias (\downarrow , $\frac{1}{8}$ de s) que completarão o compasso. Ilustrando, a Figura 20 mostra um compasso completo com $FC = \frac{3}{4}$.

Figura 20 – Compasso musical de exemplo escrito em notação musical.



Fonte: Autoria Própria (2022).

Para representar um compasso musical completo, poderia se construir os vetores de cada nota do compasso, os sobrepor e normalizar os valores dentro do intervalo $[0, 1]$. Exemplificando, para o compasso da Figura 20, tem-se os vetores de cada nota e sua sobreposição:

$$\begin{aligned}
 \mathbf{n}^{(Lá)} &= [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0]^T \\
 \mathbf{n}^{(Dó)} &= [1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]^T \\
 \mathbf{n}^{(Lá)} &= [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0]^T \\
 \mathbf{n}^{(Dó)} &= [1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]^T \\
 + \mathbf{n}^{(Lá)} &= [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0]^T \\
 \mathbf{n}^{(Lá,Dó,Lá,Dó,Lá)} &= [2; 0; 0; 0; 0; 0; 0; 0; 0; 0; 3; 0; 0]^T,
 \end{aligned}$$

que normalizado, é da forma da Equação 42:

$$\mathbf{n}^{(Lá,Dó,Lá,Dó,Lá)} = [0,67; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0]^T. \quad (42)$$

Porém, esta forma de codificação apresenta problemas: não se sabe nem a duração de cada nota nem a ordem em que aparecem.

7.4 Configuração das Redes Neurais Artificiais

O sucesso do sistema harmonizador automático também depende da escolha adequada e correta configuração de arquiteturas de RNAs. Tendo em vista as diferentes formas de codificação de entradas e saídas, pode-se sugerir a construção de algumas.

Os formatos descritos pelas Equações 41 para a entrada e 42 para a saída se mostram adequadas para uso em arquiteturas MLP genéricas. Além desta, as seguintes arquiteturas foram avaliadas:

- **Rede Neural com Função de Base Radial (RBF, *Radial Basis Function*):** Redes locais de aprendizagem que consistem em três camadas, sendo a camada de entrada composta apenas de nós de fonte para conectar os dados à rede. A camada oculta não calcula pesos com a camada de entrada e usa funções de ativação radiais não lineares, como a função Gaussiana. O treinamento dessa camada é realizado por meio de algoritmos de agrupamento. Finalmente, a camada de saída calcula pesos nas suas conexões com a camada oculta, sendo esses pesos ajustados usando *backpropagation* ou a operação Moore-Penrose Pseudo-Inversa (MPPI) (HAYKIN, 2008). É possível usar funções lineares ou não lineares como função de ativação dos neurônios de saída (SIQUEIRA; LUNA, 2019). O uso sequencial de transformações lineares e não lineares

tira vantagem do fato de que, para um problema de classificação, aumenta a dimensão do espaço de informações e leva a maior probabilidade de encontrar uma separação linear (HAYKIN, 2008).

- **Máquina de Aprendizado Extremo (ELM, *Extreme Learning Machine*):** A grande vantagem da ELM quando comparada a MLP é seu rápido processo de aprendizado e bom desempenho de generalização com a capacidade de aproximação universal (TADANO; SIQUEIRA; ALVES, 2016; SIQUEIRA *et al.*, 2012b). É uma arquitetura *feedforward* de apenas uma camada oculta, na qual os pesos não são ajustados, ou seja, possui parâmetros fixos. Assim, durante a fase de treinamento, não há manipulação da função de custo, que se resume a encontrar o melhor conjunto de pesos da camada de saída. Esta tarefa pode ser realizada com um combinador linear, o que pode ser feito usando a operação MPPI (HUANG; ZHU; SIEW, 2006).
- **Rede Neurais com Estados de Eco (ESN, *Echo State Network*):** Ao contrário das anteriores, esta é uma Rede Neural Recorrente, ou seja, apresenta repetições de *feedback* de informação, apresentando uma capacidade de memória intrínseca (RIBEIRO; REYNOSO-MEZA; SIQUEIRA, 2020). Uma estrutura de três camadas também é considerada, com uma camada de entrada de fonte e um combinador linear baseado na operação MPPI como saída. A camada oculta é referida como *reservatório dinâmico* e contém neurônios interconectados esparsos com pesos fixos (JAEGER, 2010). Possui semelhanças com a ELM, pois o processo de aprendizagem apenas modifica os pesos para a camada de saída, sendo eficiente devido à rápida convergência (SIQUEIRA *et al.*, 2012a).

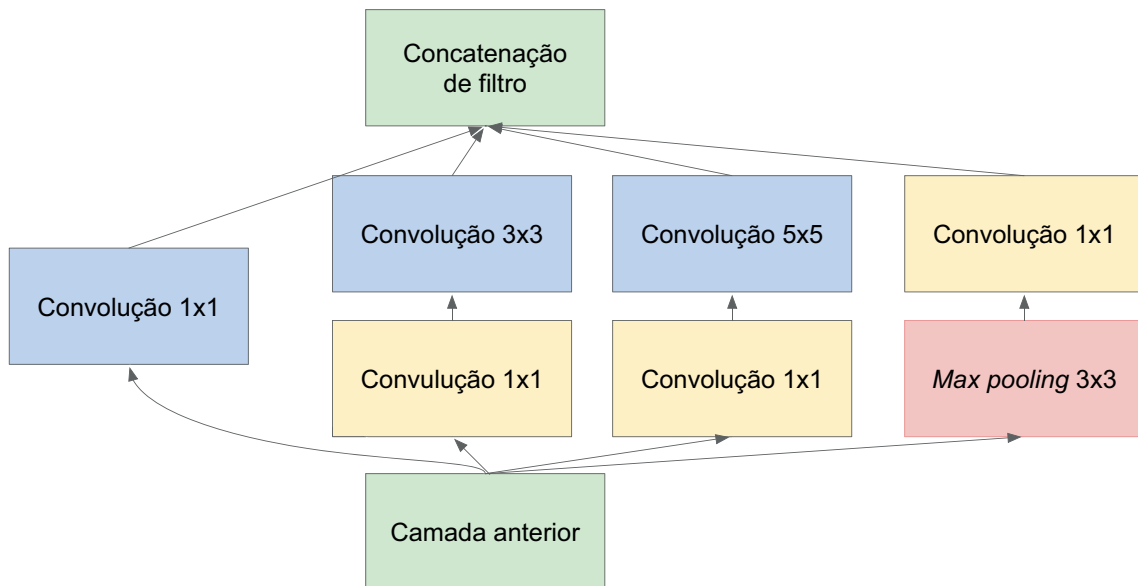
Além destas arquiteturas, após a fase de treinamento de cada uma, pode-se ainda criar agrupamentos de RNAs, também conhecidos como *ensembles*. O *ensemble* é uma metodologia de combinação de vários modelos já treinados para melhorar a resposta final de um sistema (WICHARD; OGORZALEK, 2004). Essa combinação ocorre porque métodos diferentes produzem comportamentos diferentes a partir das mesmas entradas. Um modelo pode apresentar melhores respostas para alguma entrada, enquanto outro funciona melhor para um tipo diferente. Uma abordagem de combinação é então aplicada para gerar a saída final do conjunto, por exemplo, média, votação ou outra rede neural (BACZYŃSKI; KOPYT; GULCZYŃSKI, 2022). Para que um *ensemble* funcione adequadamente, cada modelo precisa apresentar simultaneamente diversidade e ter previsões. O propósito de um *ensemble* é melhorar bons resultados já existentes. *Ensembles* têm sido usados para resolver muitos problemas (BELOTTI *et al.*, 2020).

Como será abordado na Seção 8.3, é possível construir uma imagem a partir de um compasso musical, que por sua vez pode servir de entrada para uma RNA, mais especificamente uma CNN dada sua capacidade de processamento de imagens. A definição de sua arquitetura, em especial no que diz respeito à quantidade e ordenação de camadas, pode ser tanto como as existentes na literatura, ou inspirada por elas.

Por exemplo, poderia se aplicar a lógica de construção da *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), que alterna camadas de convolução e subamostragem, diminuindo a área dos filtros e aumentando sua quantidade conforme a rede fica mais profunda. Ou então, fazer como a VGG16 (SIMONYAN; ZISSERMAN, 2014) que começa intercalando duas camadas de convolução com uma de subamostragem e depois, aumenta para três camadas de convolução seguidas e uma de subamostragem, mantendo fixa a área dos filtros e dobrando sua quantidade a cada conjunto de camadas.

Também poderia se valer de estratégias específicas, como o uso de módulos “*inception*” como acontece na *GoogLeNet* (SZEGEDY *et al.*, 2015a), um conjunto de camadas que operam de forma paralela, como ilustrado na Figura 21. Ou então usar os “blocos residuais” da *ResNet* (HE *et al.*, 2016), uma forma de “pular” camadas.

Figura 21 – Módulo “*inception*” existente na arquitetura da CNN *GoogLeNet*.



Fonte: Adaptado de Szegedy *et al.* (2015a).

Não obstante, neste trabalho foi avaliado o desempenho das redes *AlexNet*, VGG16, *Inception* (SZEGEDY *et al.*, 2015b) (que é a versão melhorada da *GoogLeNet*), *ResNet* e *DenseNet* (HUANG *et al.*, 2016) para determinar qual performa melhor para o formato de entrada desenvolvido. Após, a que obteve melhores resultados foi selecionada como modelo para ser otimizado e testado com diferentes configurações, com objetivo de encontrar uma arquitetura própria para a solução do problema proposto.

Uma questão interessante é o fato de que essas redes foram desenhadas para trabalhar com classificação de imagens reais e não sintéticas, sem variabilidade, ruídos e afins. Portanto, o desafio está em encontrar uma arquitetura específica que funcione para este caso, podendo contribuir para problemas similares cujas imagens são gerações abstratas.

7.5 Análise de Resultados

Conforme elaborado no Capítulo 5, existem formas quantitativas para avaliar as produções do sistema. Por se tratar de um problema de classificação com número definido de 24 classes, é possível a construção da matriz de confusão resultante, de onde pode-se extrair medidas de desempenho: AC (Equação 31), F_{1M} (Equação 35), MCC (Equação 36) e κ (Equação 37), além da própria medida de erro, no caso, ECC (Equação 22).

O uso de todas estas medidas possibilita a comparação de desempenho entre os modelos testados. Ainda assim, outras ferramentas estatísticas são necessárias para chegar a conclusões concretas, como o teste de Friedman, o método de contagem Borda e a correlação de classificação de Kendall.

7.5.1 Teste de Friedman

O objetivo do teste de Friedman (FRIEDMAN, 1937) é testar a hipótese nula de que amostras repetidas dos mesmos indivíduos têm a mesma distribuição, sendo um teste estatístico não paramétrico. É usado para avaliar a consistência entre amostras obtidas de diferentes maneiras (TECHNOLOGY, 2015). O mesmo é aplicado para determinar a individualidade de cada classificador. Exemplificando, para o caso analisado neste trabalho, a hipótese nula pode ser definida como: os modelos testados operam de maneira idêntica, ou seja, não há diferença estatística entre modelos; e a hipótese alternativa como: ao menos um modelo opera de maneira diferente. Se a hipótese nula for verdadeira, quer dizer que não há diferenças significativas entre modelos testados, ou seja, o experimento falhou.

Como se trata de um teste de hipóteses, pode-se calcular o valor-p resultante, que é a probabilidade de obter resultados de teste tão extremos quanto o resultado realmente observado, sob a suposição de que a hipótese nula está correta. Portanto, um valor-p pequeno é o que se quer atingir (WASSERSTEIN; LAZAR, 2016; FERREIRA; PATINO, 2015).

7.5.2 Método de Contagem Borda

O método de contagem Borda (EMERSON, 2013) foi originalmente desenvolvido com propósito de auxiliar na tomada de decisão em sistemas de votos, em processos eleitorais, por exemplo. É um método simples, no qual para cada candidato o eleitor deve colocar um número entre 1 e o total de candidatos n , sendo que o melhor candidato levaria o número n , e o pior, 1. Então, os pontos de cada candidato são somados e aquele que tiver maior quantidade de pontos é eleito.

Este método se prova útil quando se quer comparar o desempenho entre diferentes sistemas (GUERREIRO *et al.*, 2021), uma vez que várias medidas estatísticas são calculadas para

cada modelo e teste executado, a saber: erro ECC, AC, F_{1M} , MCC e κ . No contexto da contagem Borda, pode-se entender cada medida como um eleitor, e cada modelo ou teste como um candidato.

7.5.3 Correlação de Classificação de Kendall

Correlação diz respeito a um tipo de análise que mede a força da relação entre duas variáveis, bem como a direção desta relação. O valor do coeficiente de correlação geralmente varia no intervalo $[-1, 1]$, sendo que 0 significa que não há relação entre as variáveis, e ± 1 indica um grau perfeito de associação. A direção da relação, por sua vez, é indicada pelo sinal do coeficiente: um sinal “+” indica uma relação positiva, ou seja, quanto maior a primeira variável, maior será a segunda; e um sinal “-” indica justamente o oposto, uma relação negativa, de modo que quanto maior a primeira variável, menor será a segunda (CHEN; POPOVICH, 2002; BOBKO, 2001).

São quatro os tipos convencionais de cálculo de correlação: correlação de Pearson, correlação de classificação de Kendall, correlação de Spearman e correlação de ponto-biserial. A correlação de classificação de Kendall (KENDALL, 1938) desperta interesse por ser um teste não paramétrico e por poder ser usado com dados ordinais, que são informações categóricas em escala. É de interesse para auxílio na compreensão da relação entre configurações testadas no processo de otimização de hiperparâmetros, como exposto na Subseção 8.4.2 de resultados.

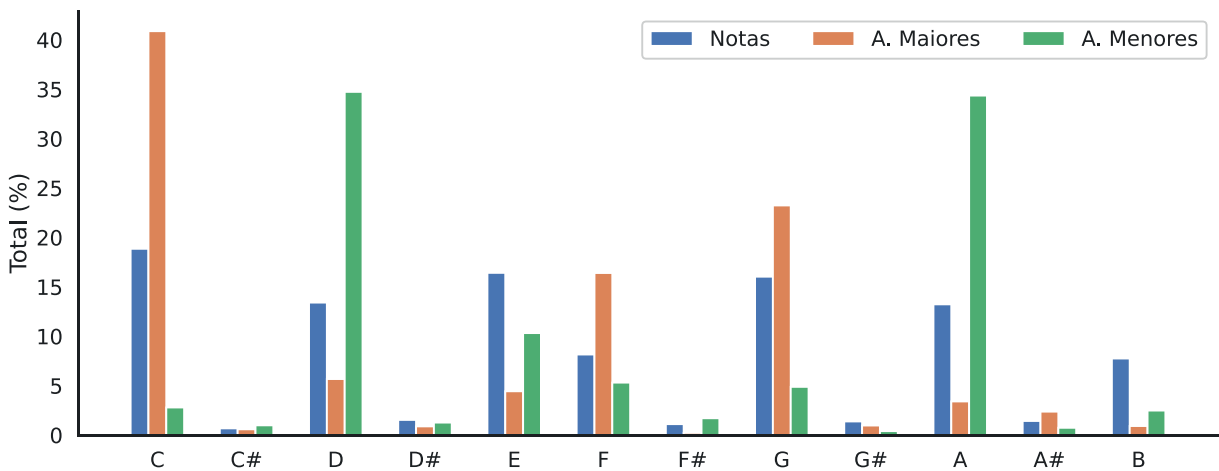
8 RESULTADOS

A seguir, serão apresentados resultados, seguindo o processo da metodologia proposta. Primeiramente, a base de dados foi analisada, e modelos de RNAs foram testados para tipos de entrada vetoriais. Depois, usando a Imagem de Compasso (IC) como entrada, testes com diferentes arquiteturas de CNNs da literatura foram realizados. A partir dos resultados, uma arquitetura foi selecionada para um processo de otimização de hiperparâmetros, para determinar um modelo final de classificação, configurando este o sistema de harmonização automática.

8.1 Análise da Base de Dados

Na base de dados utilizada para este trabalho, nomeadamente *CSV Leadsheet Database*, estão presentes ao todo, 2252 músicas. Estas, após a fase de processamento, formam um total de 304215 notas musicais e 89868 acordes e compassos¹, desconsiderando vazios, ou seja, compassos sem notas melódicas. A Figura 22 mostra a porcentagem da quantidade total de notas e acordes maiores e menores após a fase de padronização de dados.

Figura 22 – Porcentagem de notas (também representadas pelo sistema de letras), acordes maiores e menores existentes na totalidade da base de dados.



Fonte: Autoria Própria (2022).

Pode-se notar uma maior presença das notas Dó, Ré, Mi, Sol e Lá, que juntas correspondem a 77,95% do total. As notas Fá e Si aparecem em segundo lugar (15,91%), completando a escala de Dó maior, conforme esperado após a fase de transposição tonal. No entanto, notas com acidentes (#) também estão presentes (6,14%).

Sob uma perspectiva musical, é interessante observar que as notas de maior presença fazem parte de uma escala musical denominada “pentatônica”, muito utilizada em estilos como o blues e o rock, evidentemente presentes em vários outros estilos devido à sua funcionalidade.

¹ O número de acordes é igual ao de compassos já que está se considerando um acorde por compasso.

As notas Fá e Si, por sua vez, são chamadas de “notas a evitar”, geralmente usadas como notas de passagem. Da mesma forma, acordes de função básica (C, F e G) e relativos principais (Dm e Am), são a presença mais forte nas músicas, muito embora todos os tipos de acordes existam em alguma proporção, para os 24 tipos considerados. São acordes de ampla utilização dentro harmonia considerada tradicional ocidental, característica tangente a base de dados

É interessante notar que a base de dados está desbalanceada, uma característica comum e esperada, dada a natureza da música de utilização mais frequente de algumas notas do que outras, o mesmo para acordes. Ao aplicar esta base nos processos seguintes de treinamento dos sistemas inteligentes, é necessário que as amostras selecionadas sejam estratificadas. Além disso, a avaliação a partir de métricas de desempenho deve ser criteriosa, como foi observado na Subseção 5.1.1.

8.2 Comparação de Desempenho Entre Redes Neurais e Ensembles Para Entrada Vetorial

Com objetivo de testar os formatos de entrada e saída colocados nas Equações 42 e 41, respectivamente, ambas utilizando a notação one-hot, cinco modelos diferentes de RNA tiveram seu desempenho avaliado e comparado para a seleção do melhor dentre eles: MLP de uma camada escondida (MLP1), MLP de duas camadas (MLP2), RBF, ESN e ELM. Destes, três modelos de *ensemble* também foram avaliados usando votação como abordagem combinada, considerando todas as cinco redes (ENS5), apenas as três primeiras (ENS3) e duas (ENS2) melhores quanto aos resultados de perda.

Por questões de limitação computacional em termos de memória e tempo de processamento, uma amostra de 20% da base de dados foi usada, de maneira estratificada, totalizando 18651 compassos. Para realizar as etapas de treinamento e teste da rede, a amostra foi separada em 60% para treinamento, 20% para validação e 20% para teste.

Os testes foram realizados considerando uma variação do número de neurônios na camada oculta, usando os valores 64, 128 e 256. Para os modelos que necessitam de épocas de treinamento, o critério de parada selecionado foi um número total de épocas igual a 200, utilizando SGD com $\eta = 0,001$ como otimizador, salvando os melhores pesos quando o valor mínimo de erro de validação é alcançado para ECC.

Para analisar os resultados aplicando validação cruzada, cada modelo foi executado 30 vezes. Na Tabela 1 é possível observar os resultados médios de ECC, AC, F_{1M} , MCC e κ de cada modelo para o número de neurônios que trouxe o melhor desempenho. Portanto, ENS3 é o conjunto de MLP1, MLP2 e RBF, e ESN2 é composto de MLP1 e MLP2. Em média, os modelos tiveram um F_{1M} de 13,42%, um MCC de 32,54% e um κ de 31,15%. Como o banco de dados está desequilibrado, a AC como medida de avaliação pode trazer conclusões equivocadas sobre os resultados, mas por motivos de comparação e padronização, eles também são incluídos.

Tabela 1 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para cada modelo testado. A quantidade de neurônios é mostrada entre parênteses na coluna de Experimento.

Experimento	ECC	AC (%)	F1 (%)	MCC (%)	κ (%)
MLP1 (64)	1,7130	47,86	14,87	33,22	31,84
MLP2 (128)	1,7352	47,21	14,89	33,57	32,50
RBF (64)	1,7630	46,00	9,53	31,42	30,04
ELM (256)	2,0513	47,19	16,67	33,40	32,08
ESN (256)	2,0787	44,73	9,24	29,43	27,42
ENS5	1,7770	47,79	13,74	33,04	31,63
ENS3	1,7103	47,77	13,96	33,10	31,80
ENS2	1,7064	47,79	14,43	33,18	31,89

Fonte: Autoria Própria (2022).

O teste de Friedman foi aplicado aos resultados das 30 execuções de cada modelo em relação a ECC no conjunto de teste. O valor-p alcançado foi igual a $2,38 \times 10^{-41}$. Portanto, é possível afirmar que há mudanças significativas nos resultados para diferentes arquiteturas ou modelos.

Considerando as métricas de ECC, F_{1M} , MCC e κ , para cada modelo foi verificado um desempenho geral (Tabela 2), utilizando o método de contagem de Borda. O primeiro colocado recebeu 8 pontos, o segundo 7, até que o último colocado recebeu 1 ponto. Para valores de ECC, o primeiro lugar é o menor valor e, para todas as outras métricas, o primeiro lugar é o maior valor.

Tabela 2 – Classificação geral usando o método de contagem Borda considerando diferentes métricas e resultados de algoritmos.

Experimento	ECC	F_{1M}	MCC	κ	Total
MLP2	5	7	8	8	28
ENS2	8	5	5	6	24
ELM	2	8	7	7	24
MLP1	6	6	6	5	23
ENS3	7	4	4	4	19
ENS5	3	3	3	3	12
RBF	4	2	2	2	10
ESN	1	1	1	1	4

Fonte: Autoria Própria (2022).

Em geral, a MLP2 obteve o melhor desempenho, com o melhor MCC e κ e o segundo melhor F_{1M} . ENS2 e ELM empataram em segundo lugar, apesar do erro do ELM ser o segundo pior dentre todos. Na sequência, MLP1, com diferença de apenas um ponto, sendo este e MLP2 os componentes do ENS2. O fato de um modelo único ter apresentado melhores resultados é relevante, ainda que os métodos *ensemble* sejam a união dos melhores estimadores.

Em seguida, é feita uma análise mais aprofundada da resposta de cada modelo, considerando o ponto de vista musical, de forma descritiva e subjetiva.

Um exemplo de melodia e sua harmonia gerada pelo modelo, bem como a harmonia original, é mostrado na Figura 23 para os primeiros oito compassos da música *America* de Stephen Sondheim e Leonard Bernstein, composta para o musical *West Side Story* de 1957. A música tem uma construção rítmica de *hemiola*², uma melodia repetitiva e harmonia simples (MILLER, 2006). Um ponto interessante é entre os compassos 5 e 7, há uma rápida modulação³(ROIG-FRANCOLÍ, 2010) passando pela tonalidade de Cm, tornando este um exemplo adequado para perceber a capacidade de generalização para os modelos escolhidos.

Figura 23 – Análise da harmonia original e gerada pelos modelos de RNA propostos para os primeiros oito compassos da música *America*.

(a) Melodia da música *America* escrita utilizando notação musical.



(b) Harmonia original e resultante de cada modelo de RNA, um acorde por compasso.

Original:	C	F	C	G	Cm7	A#	G#	C	
MLP1:	C	F	C	G	D#	A#	G#	C	
MLP2:	C	F	C	G	G#	A#	G#	C	
RBF:	C	F	C	G	C	G	E	C	
ESN:	C	F	C	G	A#	G	E	C	
ELM:	C	F	C	G	D#	A#	G#	C	
ESN5:	C	F	C	G	D#	A#	G#	C	
ESN3:	C	F	C	G	D#	A#	G#	C	
ESN2:	C	F	C	G	D#	A#	G#	C	

Fonte: Autoria Própria (2022).

Para os primeiros quatro compassos, todos os modelos responderam com os mesmos acordes da harmonia original, provavelmente porque as notas da melodia são muito indicativas do acorde apropriado. Por exemplo, nos compassos 2 e 4, as notas juntas são as mesmas que compõem os acordes F (Fá, Lá e Dó) e G (Sol, Si e Ré), respectivamente. O mesmo vale para o último compasso analisado, em que as notas são as mesmas do acorde de C (Dó, Mi e Sol).

² ritmo alternado entre métricas binárias e terciárias.

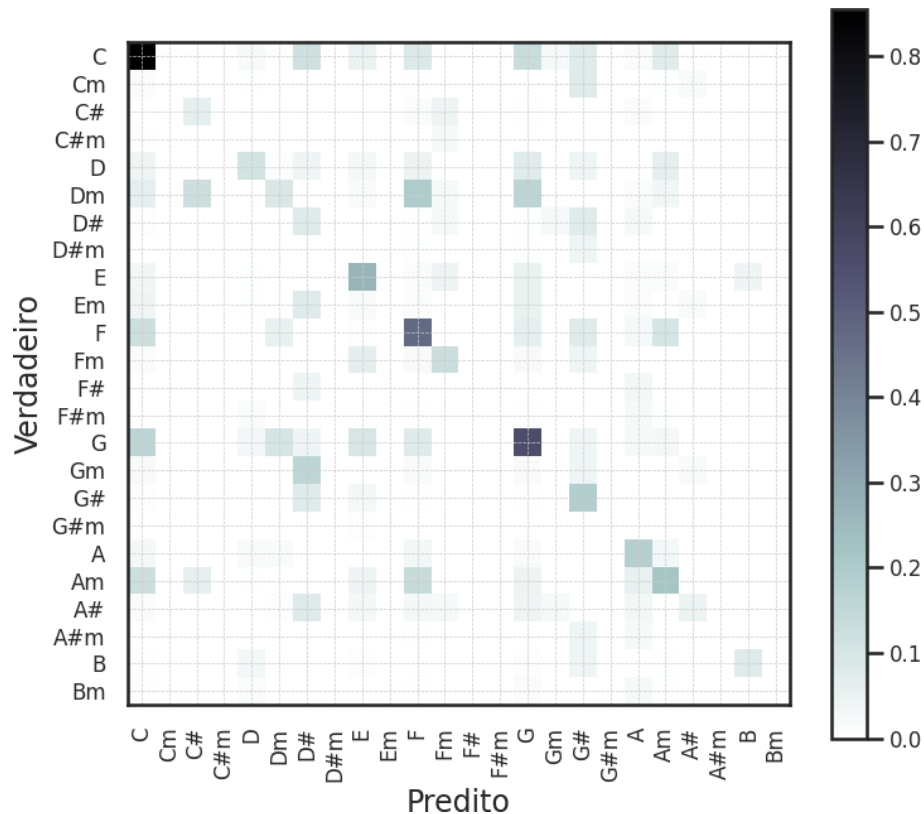
³ mudança provisória de tom.

É diferente quando se observa o intervalo de 5 a 7: a RBF fez as piores escolhas, mostrando-se incapaz de identificar a modulação; ESN da mesma forma, exceto por uma escolha possivelmente mais apropriada para o compasso 5, que tem a nota A \sharp presente; MLPs, ELM e todos os *ensembles* fizeram as mesmas escolhas para os compassos 6 e 7, iguais ao original, mas fica claro que houve dificuldade em encontrar um acorde para o compasso 5. Uma razão para isso pode ser que a harmonia original usa um acorde que não é considerado, Cm7 e apenas duas notas estão sendo tocadas neste compasso, Lá \sharp e Ré \sharp . A escolha mais adequada seria o acorde de D \sharp , pois as notas do compasso são sua fundamental e quinta, respectivamente, a qual foi escolhida por MLP1, ELM e todos os *ensembles*. A escolha feita pela MLP2, embora não seja a melhor em relação às anteriores, pode ser considerada funcional, com capacidade de trazer uma novidade harmônica e uma possível alternativa.

A análise anterior permite inferir diferenças entre o desempenho e a interpretação de cada modelo. Além disso, com base nos dados apresentados, a MLP de duas camadas é escolhida como o modelo mais adequado, com sua melhor execução atingindo 13,89% F_{1M} , 33,08% MCC e 32,01% κ . Este valor κ nos permite afirmar que este é um resultado razoável.

Uma forma eficiente de analisar a resposta do modelo selecionado é olhar para sua matriz de confusão, que pode ser vista na Figura 24.

Figura 24 – Matriz de confusão normalizada para classificação de acordes usando o modelo MLP2.



Fonte: Autoria Própria (2022).

A formação de uma linha diagonal na matriz é levemente perceptível, uma característica desejável, pois significa atribuições de classe corretas. Isto é notório em pontos específicos, como nos acordes de C, F e G, que são as principais funções tônicas. Além disso, destacam-se algumas colunas e linhas de classes: C, F e G. Estas colunas e linhas levam a entender que o modelo foi capaz de compreender o papel que esses acordes desempenham na harmonia musical e pode generalizar os resultados, utilizando-os de forma simplificada.

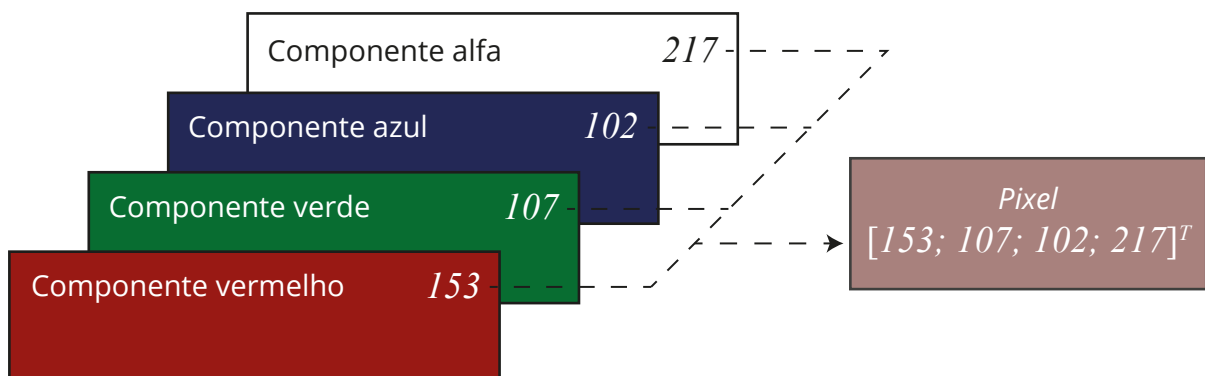
Os resultados expostos nesta Seção fazem parte do artigo “Neural Networks And Ensemble Based Architectures To Automatic Musical Harmonization: A Performance Comparison”, submetido para a revista *Applied Artificial Intelligence*. O mesmo pode ser conferido integralmente no Apêndice A.

8.3 Representação de Compassos Melódicos por Meio de Imagens

Um compasso de música contém muitas informações, principalmente as durações e alturas de cada nota, quando observado o contexto melódico. Para que nenhuma dessas informações se perca, propõe-se a construção de uma imagem que seja capaz de as conter.

Considera-se por imagem uma matriz tridimensional $x \times y \times z$, em que x e y são coordenadas de valores discretos e finitos denominados de *pixels*, e a dimensão z é uma composição de dimensões RGBA, sendo R = *red* (vermelho), G = *green* (verde), B = *blue* (azul) e A = *alpha* (alfa), sendo que as três primeiras componentes especificam cores e a quarta, a transparência (GONZALEZ; WOODS, 2010; ADLER *et al.*, 2003). A intensidade de cada *pixel* é um valor de 8 *bits*, ou seja, um inteiro entre 0 e 255. Portanto, em uma imagem, um *pixel* azul pode ser representado pelo vetor $[0; 0; 255; 255]^T$, enquanto que outro *pixel* amarelo 50% transparente é da forma $[255; 255; 0; 127]^T$. A Figura 25 ilustra a relação entre essas dimensões, e a Figura 26 mostra uma imagem que faz uso das quatro dimensões RGBA.

Figura 25 – Componentes responsáveis por formar uma imagem RGBA. O valor de cada componente, quando unido em um conjunto ou vetor, forma um *pixel* de coloração específica.



Fonte: Adaptado de Adler *et al.* (2003).

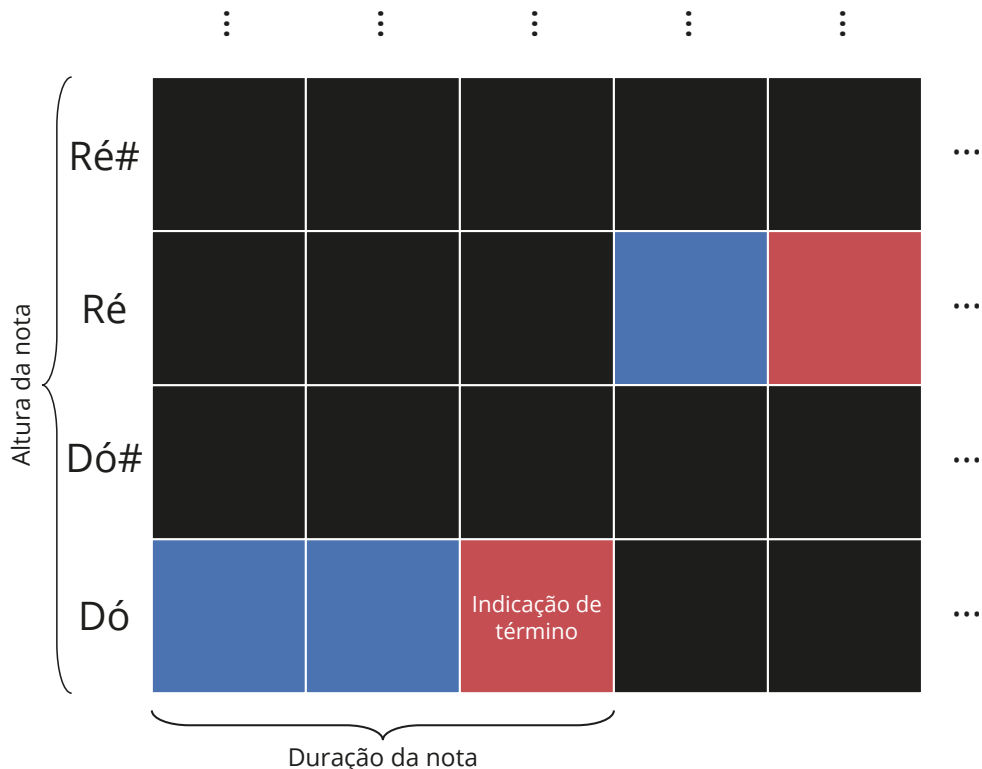
Figura 26 – Exemplo de imagem que faz uso das componentes RGBA, com a componente alfa enfatizada: sua intensidade é máxima no topo da figura e decresce até ser nula na base.



Fonte: Autoria Própria (2022).

Para compor uma imagem que tenha informações melódicas, aplica-se a abordagem gráfica de *piano roll*, uma forma de representação que usa linhas horizontais para definir a duração de notas e cada linha representa uma altura na vertical (FL STUDIO, 2021). Dessa forma, a Figura 27 demonstra o significado de cada *pixel* da Imagem de Compasso (IC), proposta inovadora deste trabalho.

Figura 27 – Composição da Imagem de Compasso (IC): na vertical, cada *pixel* representa uma nota musical, enquanto que na horizontal se tem a duração, sendo que *pixels* azuis são a reprodução continuada de uma determinada nota e *pixels* vermelhos mostram a sua finalização.

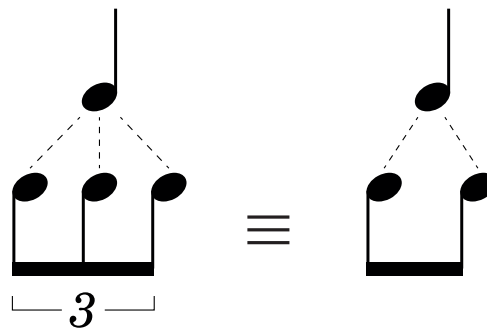


Fonte: Autoria Própria (2022).

Cada *pixel* significa uma unidade de tempo ou duração. Quando este estiver azul, quer dizer que uma nota naquela posição está sendo executada, mas se for vermelho, então a execução da nota acabou. Para que seja possível criar uma IC abrangente, é preciso aumentar a sua resolução, ou seja, fazer com que a maior quantidade de figuras rítmicas possível seja representável.

Uma vez que *pixels* são unidades discretas, é interessante conseguir representar ao menos fusas (♩, sexta linha da figura) como valor inteiro. Significa que a IC deve ter ao menos 32 *pixels* de largura, ou seja, $s = 32$. Outro fator impactante é a presença de quiálteras, que são grupos de notas cujo valor temporal é diferente do que o estabelecido pela subdivisão de valores (MED, 1996). Por exemplo, a Figura 28 mostra três quiálteras (ou tercina) que são um conjunto de três valores iguais que valem por dois da mesma categoria.

Figura 28 – Equivalência de duração entre três quiálteras de colcheia (tercina) e duas colcheias normais, ambas duram o mesmo que uma semínima.



Fonte: Adaptado de Med (1996).

Logo, faz-se relevante aumentar a resolução da IC, multiplicando a largura anterior por um fator de três (a tercina é a quiáltera mais comum), resultando em uma largura de 96 *pixels* ($s = 96$), que é o valor usado no protocolo MIDI digital padrão chamado *ticks* (algo como “riscos”) (MESSICK, 1997). Com objetivo de manter a proporção, pode-se determinar a altura da IC também de 96 *pixels*, onde cada intervalo de 8 *pixels* ($96 \div 12$) representa uma nota musical.

Também é necessário indicar a qual oitava cada nota está localizada. Considerando que um *pixel* é igual a um valor inteiro no intervalo $[0, 255]$ para cada componente dimensional RGBA, pode-se variar a intensidade de cor relativa à oitava. O Sistema Padrão Internacional Atual De Afinação considera ao todo a existência de nove oitavas musicais (YOUNG, 1939). Portanto, a definição do valor de intensidade da cor na IC depende da divisão do máximo de intensidade possível pela oitava em que a nota se encontra, como na Equação 43:

$$\text{Intensidade de cor} = \left\lfloor \frac{255}{\text{Oitava da nota}} \right\rfloor. \quad (43)$$

Outro fator de impacto é a ocorrência de apojeturas, que são ornamentos que precedem a nota real, ou seja, notas executadas antes da definitiva de maneira muito rápida, cujo valor temporal é quase que insignificante (MED, 1996). Essa questão pode ser contornada adicionando um *pixel* de coloração diferente (verde, por exemplo) junto da nota definitiva.

A composição da IC depende de que as durações das figuras rítmicas sejam números inteiros, já que não é possível preencher frações de *pixels*. E mesmo usando a resolução definida (96×96), ainda ocorrerão casos em que após a normalização (Subseção 7.2.2), a duração das figuras será fracionária. Poderia se pensar no arredondamento destes valores, porém isso acarretaria um resultado final $\sum_{i=1}^k d_i^{(N)}$ diferente de s , fazendo com que as imagens não fiquem com tamanho padronizado. Cortar ou preencher espaços vazios também não é uma boa opção, porque informações serão perdidas. É nesse caso que a componente A (alfa) da imagem se torna útil.

Para tanto, propõe-se o Algoritmo de Resto (Algoritmo 2) que, para o vetor de durações $\mathbf{d}^{(N)}$, retorna a matriz $\mathbf{D}^{(\alpha)}$ formada por um conjunto de pares $\{d_i^{(\alpha)}, \alpha_i\}$ em que $d_i^{(\alpha)}$ é um valor de duração inteiro e α_i é a intensidade alfa que deve estar presente naquele grupo de *pixels* referente a uma nota musical. Esta proposta é definida como na Equação 44:

$$\mathbf{D}^{(\alpha)} = \begin{bmatrix} d_1^{(\alpha)} & d_2^{(\alpha)} & \dots & d_k^{(\alpha)} \\ \alpha_1 & \alpha_2 & \dots & \alpha_k \end{bmatrix} \quad (44)$$

Cada valor de $\mathbf{d}^{(N)}$ é separado em sua parte inteira I e fracionária R pelo Algoritmo de Resto, apresentado no Algoritmo 2. Uma variável de controle de resto R_C é usada para verificar, a cada iteração, se um inteiro menos a soma das partes fracionárias das durações é maior que um erro aceitável (10^{-6} , por exemplo), já que essa soma pode resultar em valores muito próximos de 1. Se for o caso, então I é incrementado. Isso garante que ao final a soma de todos $d_i^{(\alpha)}$ seja igual a s . Ainda, para que a informação fracionária de cada duração não se perca, multiplica-se por α , pois R sempre será um valor menor do que 0. Caso $R = 0$, α terá intensidade máxima, em outras palavras, o *pixel* a ele relativo não terá transparência.

A aplicação do valor de α_i em cada nota é arbitrário, podendo compor a cor de toda a barra ou então, somente a parte vermelha, por exemplo, que corresponde ao final da nota, sendo a opção adotada neste trabalho.

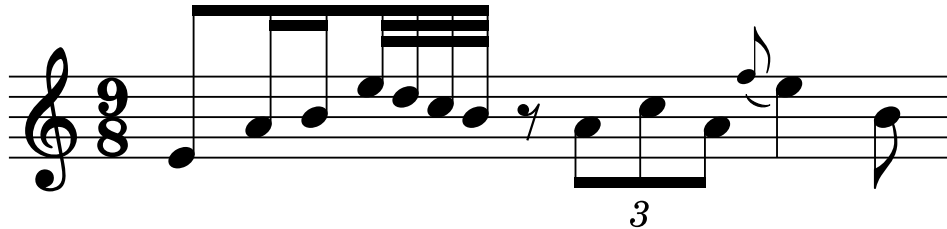
Para exemplificar o processo de construção da IC, será utilizado o compasso ilustrado na Figura 29, que contempla os desafios listados anteriormente: figuras de curta duração, notas iguais em oitavas diferentes, apojeturas e fórmula de compasso que resulta em notas com duração fracional.

Algoritmo 2 – Algoritmo de Resto

requer $\mathbf{d}^{(N)}$
inserir $\mathbf{D}^{(\alpha)}$
 1: $R_C \leftarrow 0$
 2: **para** i de 1 a k **faça**
 3: $I \leftarrow \lfloor d_i^{(N)} \rfloor$
 4: $R \leftarrow d_i^{(N)} - I$
 5: $R_C \leftarrow R_C + R$
 6: **se** $1 - R_C < \text{erro aceitavel}$ **então**
 7: $I \leftarrow I + 1$
 8: $R_C \leftarrow R_C - 1$
 9: **finaliza se**
 10: **se** $R_C < \text{erro aceitavel}$ **então**
 11: $R_C \leftarrow 0$
 12: **finaliza se**
 13: $a \leftarrow \text{valor máximo de intensidade}$
 14: **se** $R_C \neq 0$ **então**
 15: $a \leftarrow \lfloor (a \cdot R_C) + 0,5 \rfloor$
 16: **finaliza se**
 17: $d_i^{(\alpha)} \leftarrow I$
 18: $\alpha_i \leftarrow a$
 19: **finaliza para**

Fonte: Autoria Própria (2022).

Figura 29 – Compasso musical de exemplo para demonstrar construção da IC.



Fonte: Autoria Própria (2022).

O vetor de durações normalizado $\mathbf{d}^{(N)}$ resultante da padronização rítmica, para $s = 96$, é da forma:

$$\mathbf{d}^{(N)} = [10,67; 5,33; 5,33; 2,67; 2,67; 2,67; 2,67; 10,67; 7,11; 7,11; 7,11; 21,33; 10,67]^T.$$

Observa-se que, mesmo usando um valor elevado de s , todas as durações têm parte fracional. Aplicando o Algoritmo de Resto, tem-se portanto:

$$\mathbf{D}^{(\alpha)} = \begin{bmatrix} 10 & 6 & 5 & 3 & 2 & 3 & 3 & 10 & 7 & 7 & 8 & 21 & 11 \\ 170 & 255 & 85 & 255 & 170 & 85 & 255 & 170 & 198 & 227 & 255 & 85 & 255 \end{bmatrix}.$$

Agora, as durações têm valor inteiro, e a segunda linha da matriz são os valores que serão usados no canal alfa da IC. Assim, o passo a passo para construção da IC é mostrado na Figura 30.

Na Figura 30a, a representação da primeira nota é feita na IC. Pode-se observar que se trata da nota Mi, que tem duração equivalente a 10 *pixels* de largura. Oito *pixels* de altura indicam a posição ou altura da nota, e a última coluna de *pixels* vermelhos é que contém aplicação do valor na camada alfa, visualmente mais clara ou branqueada, ao se considerar um fundo branco.

Adiante, no segundo passo (Figura 30b), mais notas são representadas e a IC começa a tomar forma. Um ponto curioso é a indicação de uma pausa, que não tem altura de nota, portanto apenas *pixels* de cor preta são usados.

O terceiro passo (Figura 30c) demonstra a apojatura: uma coluna de *pixels* de cor verde são usados, aparecendo com o início da próxima nota, uma vez que a duração de uma apojatura considera-se nula. Por fim, o último passo da Figura 30d ilustra a IC em sua completude, e a Figura 31 sumariza cada aspecto da IC.

Observando a construção de arquiteturas de CNN da literatura, nota-se comumente a entrada como uma imagem de somente três canais, como seria o caso RGB, sem canal alfa. Mesmo assim, é possível adaptar a IC para que tenha somente três canais, trocando transparência pela cor branca. Essa transformação consiste em balancear as três camadas de cor distribuindo o valor existente de transparência. Para tanto, usa-se a Equação 45 para cada *pixel* da IC:

$$g(c) = \lfloor c \cdot \alpha + (1 - \alpha) \cdot 255 \rfloor, \quad (45)$$

sendo c a componente RGB que se quer transformar, considerando $\alpha = A/255$.

Exemplificando, para o *pixel* RGBA $[255; 255; 0; 127]^T$ (amarelo 50%), sua versão RGB passaria pelos cálculos:

$$g(R) = \lfloor 255 \cdot 0,5 + (1 - 0,5) \cdot 255 \rfloor = 255$$

$$g(G) = \lfloor 255 \cdot 0,5 + (1 - 0,5) \cdot 255 \rfloor = 255$$

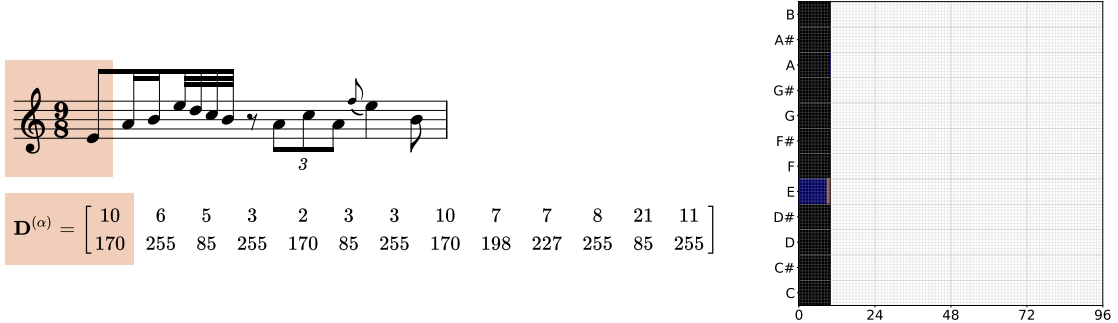
$$g(B) = \lfloor 0 \cdot 0,5 + (1 - 0,5) \cdot 255 \rfloor = 127,$$

formando o *pixel* RGB $[255; 255; 127]^T$.

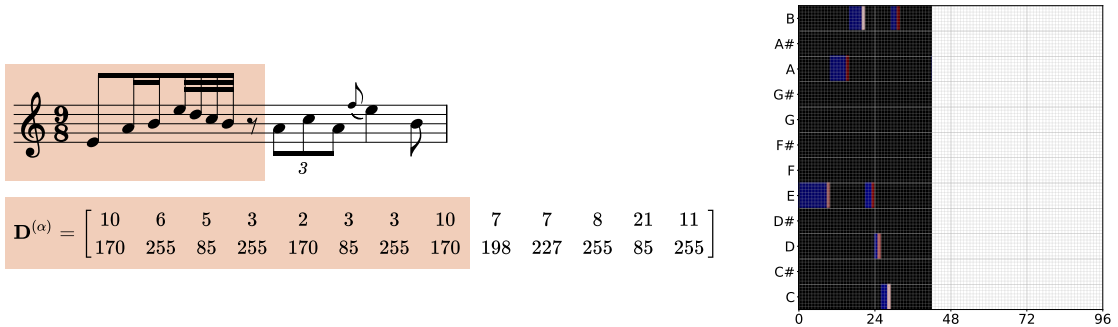
Assim sendo, a base de dados formada por ICs em formato RGBA pode ser transformada para o formato RGB e aplicadas a CNNs de acordo com suas configurações padrão de maneira aceitável.

Figura 30 – Passo a passo de exemplificação de construção da IC, dividido em quatro etapas: indicação da primeira nota, pausa, apojatura e a figura completa.

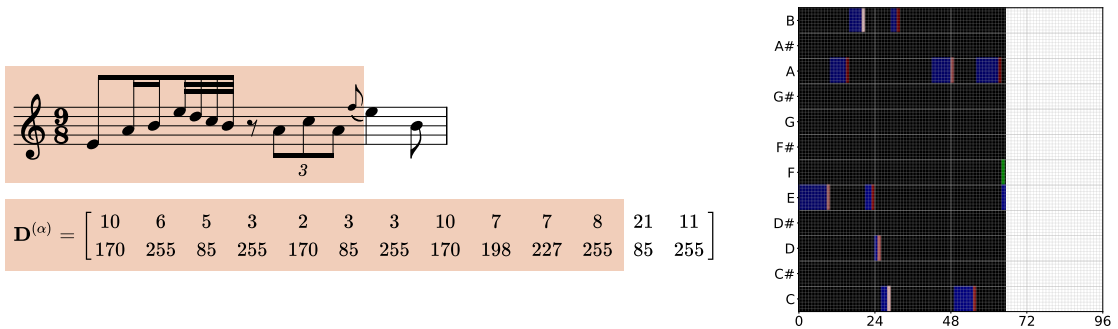
(a) Primeiro passo da construção da IC, indicando o posicionamento da primeira nota do compasso de exemplo.



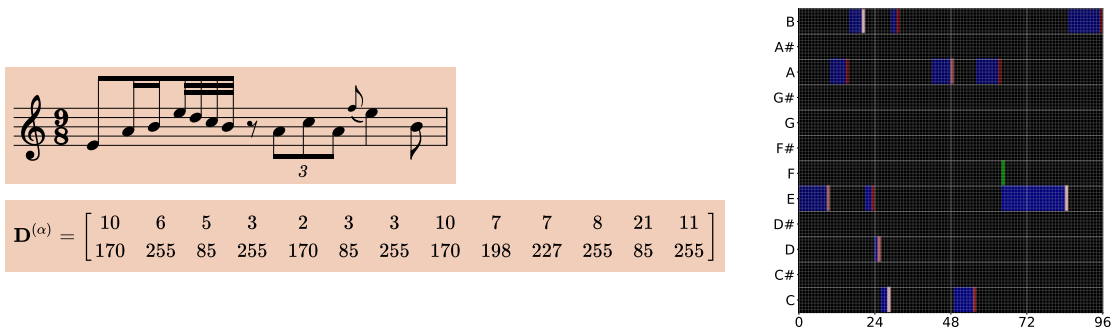
(b) Segundo passo da construção da IC, mostrando como fica a presença de pausas.



(c) Terceiro passo da construção da IC, onde há a ocorrência de um apojatura.

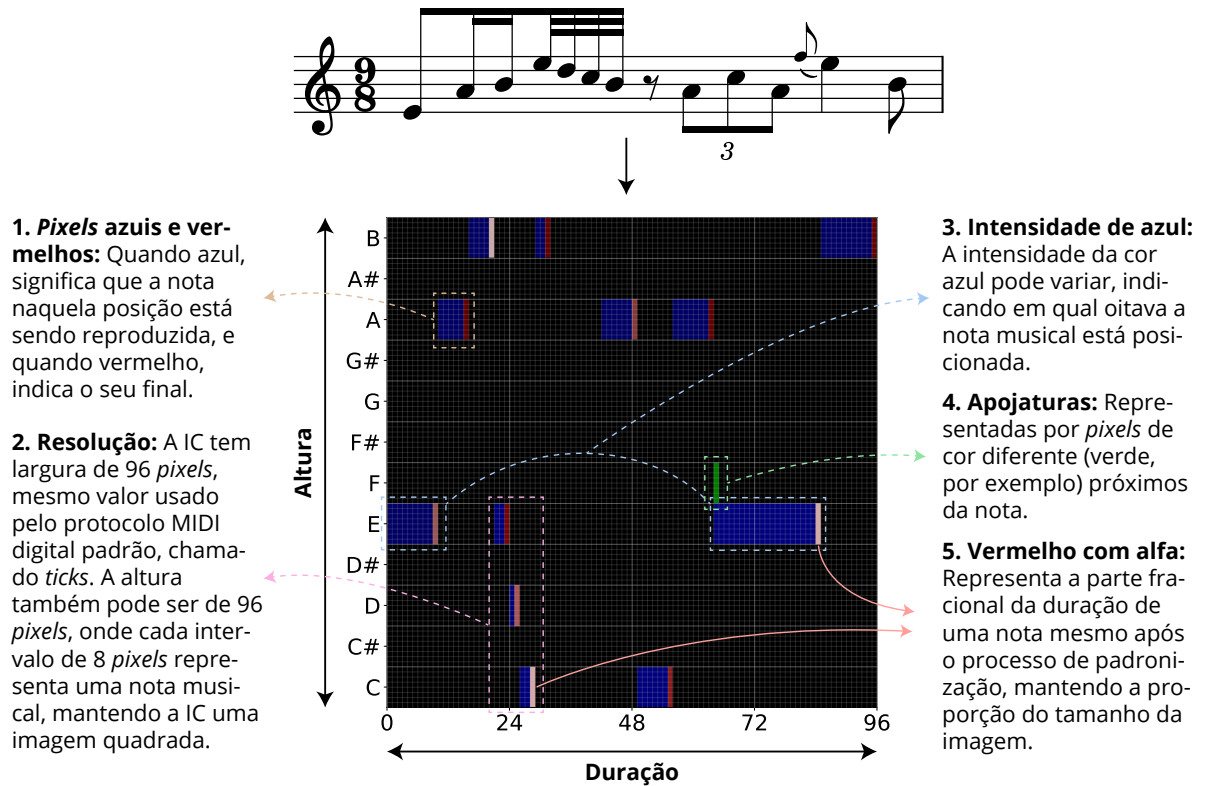


(d) Último passo da construção da IC, a definição da imagem completa.



Fonte: Autoria Própria (2022).

Figura 31 – Compasso de exemplo e IC, com detalhes da sua construção: (1) a aplicação de *pixels* de cor azul e vermelha, (2) a resolução da figura, (3) a diferença de intensidade de cor, (4) a possibilidade de representar apojeturas e (5) o uso do canal alfa.



Fonte: Autoria Própria (2022).

Os resultados e um exemplo de aplicação da IC compõe o artigo intitulado “Developing a Measure Image and Applying to Deep Learning” (COSTA *et al.*, 2022) apresentado em modalidade de pôster durante o congresso internacional *Music Encoding Conference* por autores envolvidos neste trabalho. O artigo e o pôster podem ser conferidos na íntegra no Apêndice B.

8.4 Aplicação de Arquiteturas de Aprendizado Profundo Usando a Imagem de Compasso Como Entrada

Tendo testado arquiteturas mais tradicionais, agora serão apresentados resultados fazendo uso de estratégias de aprendizado profundo, com a IC servindo de entrada aos sistemas, proposta inovadora deste trabalho.

Primeiramente, o desempenho de arquiteturas da literatura foi comparado, sendo elas: *AlexNet*, *VGG16*, *ResNet*, *Inception* e *DenseNet*. Depois, a melhor arquitetura passou por um processo de otimização de hiperparâmetros usando o método Bayesiano, para encontrar o conjunto de hiperparâmetros que retorna melhores resultados. Ao final, após alguns ajustes manuais, a arquitetura definitiva é apresentada e testada.

Para realização dos testes, por motivos de limitação de memória e processamento, foi usado um tamanho de amostra com grau de 99% de confiança, que corresponde a 15,57% do total de amostras da base de dados, ou seja, 14004 compassos de música. A separação dos conjuntos de treinamento, validação e teste são como no experimento anterior: 60%, 20% e 20%, respectivamente.

8.4.1 Comparação de Arquiteturas da Literatura

O teste comparativo entre arquiteturas da literatura tem por objetivo ser um ponto de partida no entendimento do comportamento de CNNs frente a IC. Isto porque o propósito original destas e outras propostas envolve o processamento de imagens reais e não sintéticas, como é o caso da IC. Cada uma dessas redes possui arquiteturas diferentes, como na forma de encadear camadas e tamanhos ou profundidades. Vale observar qual tem melhor desempenho, para então otimizar e adaptar, gerando uma nova arquitetura.

A configuração de cada rede foi definida de maneira similar ao encontrado na literatura original como segue:

- *AlexNet*: modelo construído como redigido por Krizhevsky, Sutskever e Hinton (2012), com camadas de convolução e subamostragem alternadas, com valores crescentes de filtros a duas camadas densamente conectadas antes da camada de saída, usando função de ativação ReLU entre camadas, *dropout* de 50% e normalização em lotes. O treinamento usa SGD com $\eta = 0,1$ e $\gamma = 0,9$, com mini lotes de tamanho igual a 128 e 90 épocas;
- VGG16: diferente da *AlexNet*, a VGG16 é mais profunda (16 camadas com pesos ajustáveis) e usa um padrão diferente, com até três camadas convolucionais seguidas, sem subamostragem entre elas (SIMONYAN; ZISSERMAN, 2014). A saída também é composta de três camadas densamente conectadas, antes da camada de saída. Usa o mesmo otimizador (SGD) com $\eta = 0,01$ e $\gamma = 0,9$, mini lotes de 256 e 100 épocas de treino;
- *ResNet*: a versão empregada neste trabalho foi a de 101 camadas descrita por He *et al.* (2016), uma das redes mais profundas testadas. É treinada com mini lotes de 256 e 400 épocas, usando SGD com $\eta = 0,0001$ e $\gamma = 0,9$;
- *Inception*: foi utilizada a arquitetura “Inception-v3”, que aplica os módulos chamados de “inception”, e que obteve melhores resultados no trabalho de Szegedy *et al.* (2015b). Diferente das anteriores, o otimizador usado é o RMSprop, com $\eta = 0,045$, mini lotes de tamanho 32 e 100 épocas de execução;

- *DenseNet*: foi usada a versão de 201 camadas como no trabalho de Huang *et al.* (2016), ainda mais profunda que a *ResNet*. Similar às demais, foi usado SGD com $\eta = 0,1$, $\gamma = 0,9$, mini lotes de 256 e 90 épocas.

A Tabela 3 sumariza a configuração de cada arquitetura para facilitar a comparação. Para execução dos testes, a base de dados foi codificada como IC para entrada, usando formato RGB, padrão aplicado na literatura.

Tabela 3 – Configurações das arquiteturas de CNN usadas durante a experimentação, como definidas pela literatura. A primeira linha (Qtde. Camadas) se refere ao total de camadas convolucionais e densamente conectadas, ou seja, somente camadas que tenham parâmetros ajustáveis.

Configuração	<i>AlexNet</i>	VGG16	<i>ResNet</i>	<i>Inception</i>	<i>DenseNet</i>
Qtde. camadas	8	16	101	42	201
Otimizador	SGD	SGD	SGD	RMSprop	SGD
η	0,1	0,01	0,0001	0,045	0,1
γ	0,9	0,9	0,9	-	0,9
Tam. lotes	128	256	256	32	256
Épocas	90	100	400	100	90

Fonte: Autoria Própria (2022).

Os modelos foram executados independentemente 10 vezes, e os resultados estão expostos na Tabela 4 para as mesmas medidas usadas no experimento anterior da Seção 8.2. A média geral de F_{1M} foi de 12,72%, 30,11% de MCC e 29,33% de κ , resultados levemente inferiores aos encontrados com uso de arquiteturas mais simples e entrada vetorial.

Tabela 4 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para cada modelo de CNN testado. A última linha traz os resultados do melhor modelo usando entrada vetorial (MLP2) obtido na experimentação da Seção 8.2, a título de comparação.

Experimento	ECC	AC (%)	F_{1M} (%)	MCC (%)	κ (%)
<i>AlexNet</i>	1,7986	46,58	14,59	33,08	32,18
VGG16	1,8082	46,41	13,51	32,82	31,75
<i>ResNet</i>	2,1870	40,20	12,18	25,42	25,05
<i>Inception</i>	1,9588	42,17	11,65	27,55	26,92
<i>DenseNet</i>	1,8663	45,63	11,69	31,69	30,76
MLP2	1,7352	47,21	14,89	33,57	32,50

Fonte: Autoria Própria (2022).

Aplicando o teste de Friedman, obteve-se um valor-p igual a $1,77 \times 10^{-7}$, possibilitando afirmar que os modelos tem diferenças significativas entre seus resultados. Fazendo uso novamente do método de Borda, para métricas de ECC, F_{1M} , MCC, e κ , o desempenho geral de cada resultado é apresentado na Tabela 5.

Tabela 5 – Classificação geral dos modelos de CNN a partir do método de contagem Borda para as diferentes métricas aplicadas.

Experimento	ECC	F_{1M}	MCC	κ	Total
<i>AlexNet</i>	5	5	5	5	20
VGG16	4	4	4	4	16
<i>DenseNet</i>	3	2	3	3	11
<i>Inception</i>	2	1	2	2	7
<i>ResNet</i>	1	3	1	1	6

Fonte: Autoria Própria (2022).

A ordenação permite concluir que a arquitetura que obteve melhor resultado foi a *AlexNet*, sendo esta consideravelmente mais simples de todas avaliadas, dado o menor número de camadas e, conseqüentemente, parâmetros. Sua melhor execução obteve 14,26% de F_{1M} , 33,52% de MCC e 32,48% de κ .

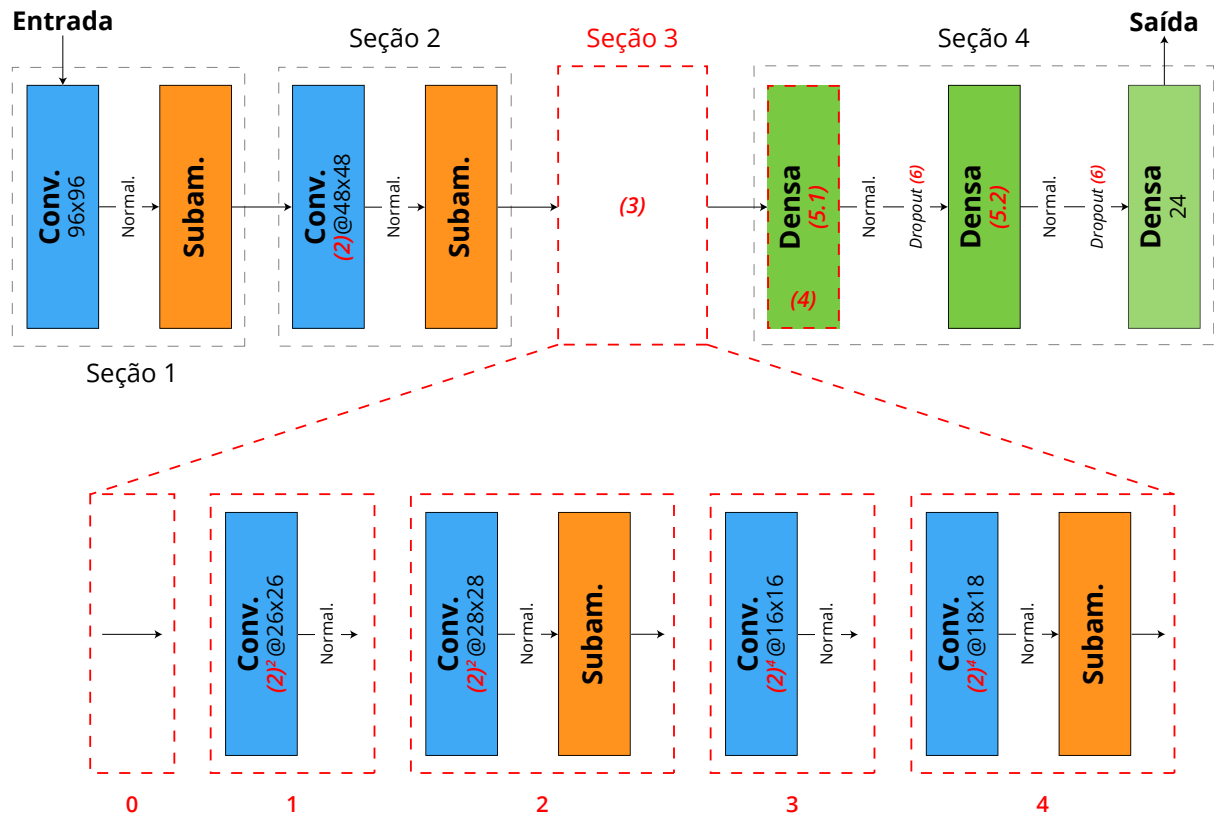
8.4.2 Otimização de Hiperparâmetros

A fase de otimização de hiperparâmetros teve por objetivo encontrar o conjunto de configurações arquiteturais que retornasse melhores resultados. Para o processo de otimização, foi escolhido o método Bayesiano com limitação de 10 iterações máximas (testes de configuração), com objetivo de redução da métrica de erro, sendo neste caso considerada a ECC. A Figura 32 exemplifica as arquiteturas possíveis testadas com a variação de hiperparâmetros. As configurações avaliadas foram:

- 1 Funções de ativação;
- 2 Número de filtros das camadas de convolução;
- 3 Profundidade da rede;
- 4 Inclusão de camada extra densamente conectada antes do final da rede;
- 5 Número de neurônios da (1) primeira e (2) segunda camada densamente conectada;
- 6 Valores de *dropout*;
- 7 Taxa de aprendizado η ;
- 8 Algoritmo otimizador.

O teste foi realizado em três etapas, reduzindo o espaço de busca de configurações a cada etapa, até definir um modelo ótimo. A análise de resultados de cada etapa aconteceu por meio de análise gráfica e cálculos de correlação de Kendall em comparação com o erro de validação.

Figura 32 – Arquiteturas que podem ser formadas a partir da variação de hiperparâmetros. O número de filtros (2) aparece nas camadas convolutivas, podendo ter quantidade igual a duas ou quatro vezes seu valor, a depender da profundidade da rede (3), que por sua vez é o encadeamento de camadas adicionais. A camada densa, com borda pontilhada vermelha, pode ou não ser inserida (4) e tem total de neurônios definido pela configuração 5.1, enquanto que a camada densa da sequência tem neurônios definidos por 5.2. A configuração de função de ativação (1), embora não esteja ilustrada, é definida para todas as camadas convolucionais e densas, a não ser pela camada de saída, que será sempre de tamanho 24, e ativação *softmax*, devido ao número de classes considerado.



Fonte: Autoria Própria (2022).

8.4.2.1 Primeira Etapa

A primeira etapa de otimização buscou variar todos os hiperparâmetros listados de acordo com o definido na Tabela 6, de acordo com a numeração da listagem anterior. No caso da configuração de profundidade, o valor 0 é o mesmo que dizer que não foram adicionadas camadas de convolução intermediárias.

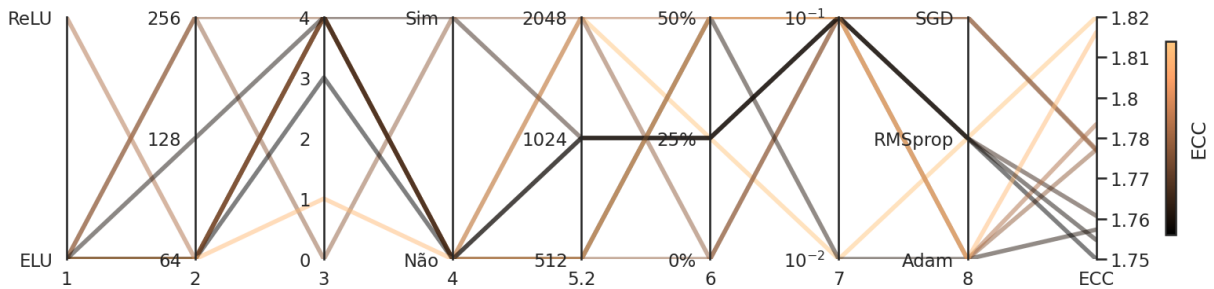
O gráfico de coordenadas paralelas das 10 execuções do processo de otimização pode ser visualizado na Figura 33. Este tipo de gráfico permite analisar quais configurações retornaram melhores resultados, seguindo as linhas formadas e esquema de cores, sendo que o objetivo é encontrar as configurações de menor erro de validação.

Tabela 6 – Hiperparâmetros considerados para processo de otimização, bem como as variações possíveis de configuração para a primeira etapa de testes.

Hiperparâmetro	Configurações
1	ELU, ReLU
2	64, 128 ou 256
3	De 0 a 4
4	Sim ou não
5.1	1024, 2048 ou 3072
5.2	512, 1024 ou 2048
6	0, 25% ou 50%
7	1×10^{-1} , 1×10^{-2} ou 1×10^{-3}
8	SGD, RMSprop ou Adam

Fonte: Autoria Própria (2022).

Figura 33 – Gráfico de coordenadas paralelas onde cada eixo representa as configurações de hiperparâmetros testadas, e as linhas são execuções completas. Linhas mais escuras estão relacionadas com menor valor de ECC de validação.



Fonte: Autoria Própria (2022).

Pela Figura 33, percebe-se que a melhor função de ativação é ELU, embora não de forma definitiva. A menor quantidade de filtros trouxe melhores resultados, de modo que três ou quatro camadas de profundidade parecem ser ideais. A inclusão de uma camada densamente conectada aparentemente não impactou no resultado, o mesmo se aplicando a sua quantidade de neurônios, diferentemente da seguinte, com indicativos positivos para valores menores, como 1024 e 512. A inserção do *dropout* teve melhores respostas para 25%, e η e otimizadores divididos entre 1×10^{-1} e 1×10^{-2} , e RMSprop e Adam, respectivamente.

Para auxiliar na análise, a correlação de Kendall entre os hiperparâmetros e o erro de validação de cada execução é mostrado na Tabela 7, bem como os hiperparâmetros da melhor execução. Hiperparâmetros categóricos (como função de ativação e otimizador) foram traduzidos para escala ordinal.

Tabela 7 – Correlação de Kendall calculada para cada hiperparâmetro testado na primeira etapa do processo de otimização e configuração que trouxe a melhor execução.

Hiperparâmetro	Correlação	Melhor execução
1	0,2485	ELU
2	-0,0311	64
3	-0,0609	3
4	0,0000	Não
5.1	0,0000	
5.2	0,1816	1024
6	0,1054	25%
7	-0,1491	1×10^{-1}
8	0,3162	RMSprop

Fonte: Autoria Própria (2022).

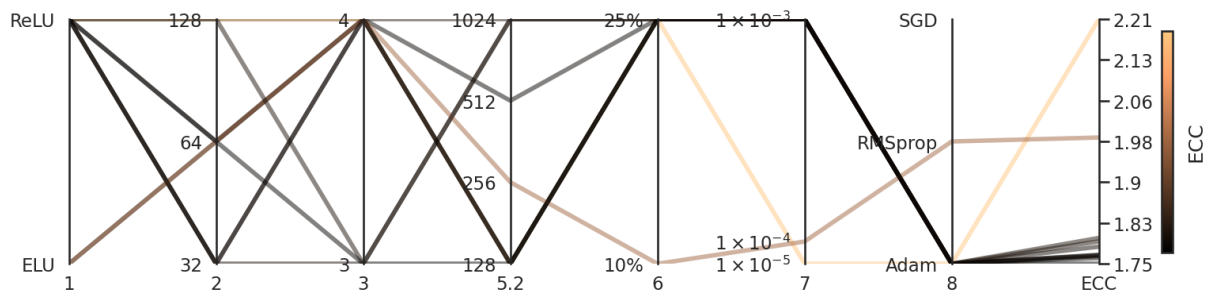
O que chama atenção é que o número de neurônios da camada densa (**5.2**) e *dropout* (**6**) tem correlação positiva, quer dizer que quanto menores seus valores, menor será o erro. O inverso ocorre para número de filtros (**2**), profundidade da rede (**3**) e η (**7**), indicativo de que, quão maiores, menor é o erro. A correlação de inclusão da camada densa extra (**4**) bem como seu número de neurônios (**5.1**) é igual a zero, o que confirma o baixo ou nulo impacto da adição desta camada. Um ponto importante é o fato de que a maioria das execuções convergiu em apenas três épocas, indicativo de que os valores de η estão demasiado altos, o que sugere o oposto do indicado pela correlação, motivando um próximo teste que considere valores menores.

8.4.2.2 Segunda Etapa

Para a realização do segundo teste, as opções de função de ativação e otimizadores foram mantidas iguais e a inclusão de uma camada extra foi descontinuada por não demonstrar impacto no resultado. Apenas duas opções foram mantidas para profundidade da rede e *dropout*. Os demais hiperparâmetros foram testados com quantitativos menores, dados pela última análise. O gráfico de coordenadas paralelas desta etapa está na Figura 34, e as opções testadas, correlações e resultado da melhor execução estão na Tabela 8.

Graficamente, percebe-se que a função de ativação ReLU foi fortemente preferida. A quantidade de filtros ficou entre 64 e 32, o mesmo para a profundidade, entre três e quatro, e para a quantidade de neurônios, entre 512 e 128. A η mais adequada foi de 1×10^{-3} , maior valor disponível, e o otimizador foi Adam. As correlações foram positivas para profundidade e ainda mais para quantidade de neurônios, de modo que sua redução também reduz o erro. Quantidade de filtros, *dropout* e η tem correlações negativas, portanto devem ser preferidos valores maiores.

Figura 34 – Gráfico de coordenadas paralelas para a segunda etapa de testes do processo de otimização.



Fonte: Autoria Própria (2022).

Tabela 8 – Hiperparâmetros considerados na segunda etapa da otimização, correlação de Kendall resultante e configurações da melhor execução.

Hiperparâmetro	Configurações	Correlação	Melhor execução
1	ELU, ReLU	-0,0745	ReLU
2	32, 64 ou 128	-0,0259	128
3	3 ou 4	0,0976	4
5.2	128, 256, 512 ou 1024	0,3689	128
6	10% ou 25%	-0,3478	25%
7	1×10^{-3} , 1×10^{-4} ou 1×10^{-5}	-0,6146	1×10^{-3}
8	SGD, RMSprop ou Adam	-0,3478	Adam

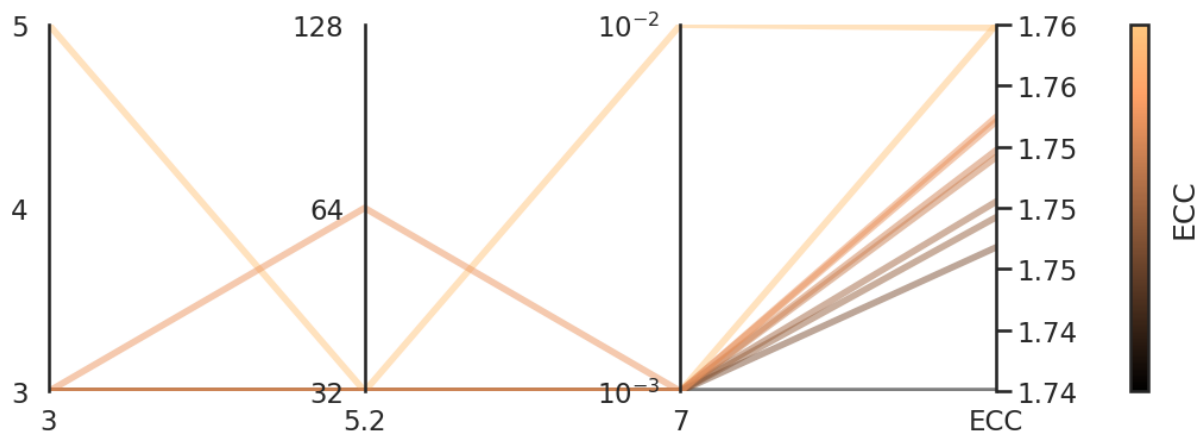
Fonte: Autoria Própria (2022).

Em resumo, considerando as duas últimas etapas, para a próxima, a função de ativação pode ser definida como ReLU pela ocorrência mais frequente, a quantidade de filtros igual a 64 pela baixa correlação e menor complexidade, *dropout* de 25% com base no resultado da primeira etapa, confirmados nesta, e otimizador como Adam novamente em virtude de ocorrências.

8.4.2.3 Terceira Etapa

Nesta terceira etapa de otimização, os hiperparâmetros que continuam a compor o espaço de busca são: profundidade da rede, quantidade de neurônios e η . Embora na segunda etapa houvesse indicativo de a melhor profundidade ser igual a três, vale testar para alguma quantidade maior. O número de neurônios em contrapartida, deve ser considerado para menores quantidades. A η apresentou maior valor absoluto de correlação, levando ao teste com valores maiores. Como anteriormente, a Figura 35 tem o gráfico de coordenadas da terceira etapa, e a Tabela 9 tem opções, correlações e configuração da melhor execução.

Figura 35 – Gráfico de coordenadas paralelas, referente ao segundo teste do processo de otimização, considerando somente a profundidade da rede (3), quantidade de neurônios da camada densa (5.2) e η (7).



Fonte: Autoria Própria (2022).

Tabela 9 – Hiperparâmetros considerados para a terceira etapa da otimização, resultados de correlação de Kendall e configurações da execução de melhor retorno.

Hiperparâmetro	Configurações	Correlação	Melhor execução
3	3, 4 ou 5	0,3478	3
5.2	32, 64 ou 128	0,2484	32
7	1×10^{-2} ou 1×10^{-3}	0,3478	1×10^{-3}

Fonte: Autoria Própria (2022).

Pela Figura 35, a melhor profundidade deve ser igual a três, e η igual a 1×10^{-3} , mas a quantidade de neurônios ainda fica dividida entre 32 e 128. Dessa vez, as correlações são todas positivas, indicando valores menores. Dessa forma, pode-se também decidir pela quantidade de neurônios igual a 32, sendo esta inclusive a configuração da melhor execução. Define-se então, a arquitetura otimizada resultante com as seguintes características:

- Função de ativação ReLU;
- 64 filtros;
- Três níveis de profundidade;
- Sem camada densamente conectada extra;
- 32 neurônios na camada densamente conectada antes da saída;
- *Dropout* de 25%;
- $\eta = 1 \times 10^{-3}$;

- Adam como algoritmo otimizador.

8.4.3 Avaliação da Arquitetura Final

Em posse da arquitetura otimizada, seguiram-se testes com adaptações manuais, para realizar um ajuste fino da rede na busca de resultados ainda melhores. Primeiramente, a arquitetura foi posta a prova com 50 épocas de treinamento e executada 30 vezes com validação cruzada. A Tabela 10 mostra as métricas obtidas com a arquitetura otimizada, e o comparativo com o resultado da *AlexNet* sem alteração e MLP2 com entrada vetorial. Vale lembrar que o mesmo conjunto de dados aplicado nos testes de redes da literatura foi usado para o processo de otimização e continua até o presente momento.

Tabela 10 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para o teste da arquitetura otimizada, *AlexNet* sem alteração e MLP2 com entrada vetorial.

Experimento	ECC	AC (%)	F_{1M} (%)	MCC (%)	κ (%)
Arquitetura otimizada	1,8265	46,23	13,80	32,71	31,76
<i>AlexNet</i>	1,7986	46,58	14,59	33,08	32,18
<i>MLP2</i>	1,7352	47,21	14,89	33,57	32,50

Fonte: Autoria Própria (2022).

São resultados muito próximos aos obtidos com a *AlexNet* original, e inferiores comparados ao uso de arquiteturas simples e entrada vetorial. Outrossim, a versão otimizada é mais simples em termos de estrutura e quantidade de parâmetros, com menos filtros nas camadas convolucionais, menor número de neurônios na camada densamente conectada, e total de sete camadas com parâmetros ajustáveis. Portanto, o custo computacional é mais baixo para se obter resultados similares ao do sistema sem otimização.

Não obstante, a convergência ainda acontece muito rápido, sugerindo um teste com η menor. Outro aspecto que se pode implementar é adicionar *dropout* entre conjuntos de camadas convolutivas e de subamostragem, tal qual acontece em outras redes da literatura.

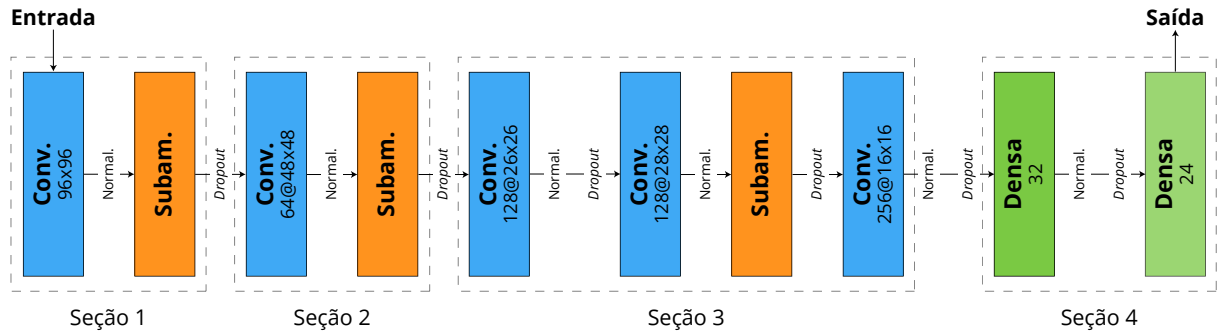
Alterando η para 1×10^{-4} e adicionando camadas de *dropout* (Adaptação 1), o sistema foi executado novamente com as mesmas quantidades (50 épocas e 30 vezes). Dessa forma, a ECC média passou a ser de 1,7939, com melhor execução atingindo 1,7778, resultado melhor que da *AlexNet* sem alteração.

Um aspecto interessante é que, até então, estava-se usando a entrada em formato RGB, para se adequar aos requisitos das redes da literatura. No entanto, como esta se trata de uma arquitetura própria, é válido utilizar a entrada como projetada, no formato RGBA. Aplicando essa alteração (Adaptação 2), a ECC média chegou a 1,7487, com 1,7366 para a melhor execução.

Como teste final, pode-se aumentar a quantidade de dados usada para treinamento, já que somente 15.57% da base estava sendo aplicada. Ao utilizar 100% da base, com 60 épocas e

30 execuções, a ECC média decaiu para 1,6069 com melhor execução chegando a 1,5999, superando todos os resultados anteriores (Adaptação 3). A Figura 36 mostra a versão da arquitetura final após processos de otimização e adaptação manual, que foi batizada de SIH, abreviação de Sistema Inteligente Harmonizador.

Figura 36 – Arquitetura final resultante do processo de otimização e após adaptações manuais denominada de SIH.



Fonte: Autoria Própria (2022).

Os resultados médios para as demais métricas de comparação são mostrados na Tabela 11 para todas as adaptações realizadas, com comparativo do modelo otimizado, *AlexNet* e MLP2 com entrada vetorial.

Tabela 11 – Média de ECC e porcentagem média de AC, F_{1M} , MCC e κ para cada adaptação manual, comparado com o teste da arquitetura otimizada, *AlexNet* e MLP2 com entrada vetorial.

Experimento	ECC	AC (%)	F_{1M} (%)	MCC (%)	κ (%)
Adaptação 1	1,7939	46,86	14,59	33,13	31,96
Adaptação 2	1,7487	47,67	15,52	34,24	33,05
Adaptação 3	1,6069	52,47	26,04	40,80	39,85
<i>AlexNet</i>	1,7986	46,58	14,59	33,08	32,18
Arquitetura otimizada	1,8265	46,23	13,80	32,71	31,76
MLP2	1,7352	47,21	14,89	33,57	32,50

Fonte: Autoria Própria (2022).

É interessante notar que todas as alterações tiveram impacto positivo nos resultados, a partir da arquitetura otimizada, com diferença significativa ao alterar o total de dados usado para treinamento.

Buscando-se a harmonização realizada na Seção 8.2 para os primeiros oito compassos da música America, mantendo-se somente a harmonia gerada pela MLP2 e adicionando-se a harmonização feita pelo SIH, foi colocada a Figura 37.

Pode-se notar que o SIH acertou todos os acordes originais, a não ser pela adição da 7ª no quinto compasso por não ser considerada uma resposta válida dentre as possibilidades de

Figura 37 – Análise da harmonia original e gerada pela MLP2 da Seção 8.2 e pela SIH para os primeiros oito compassos da música *America*.

(a) Melodia da música *America* em notação musical.



(b) Harmonia original e resultante da MLP2 e SIH.

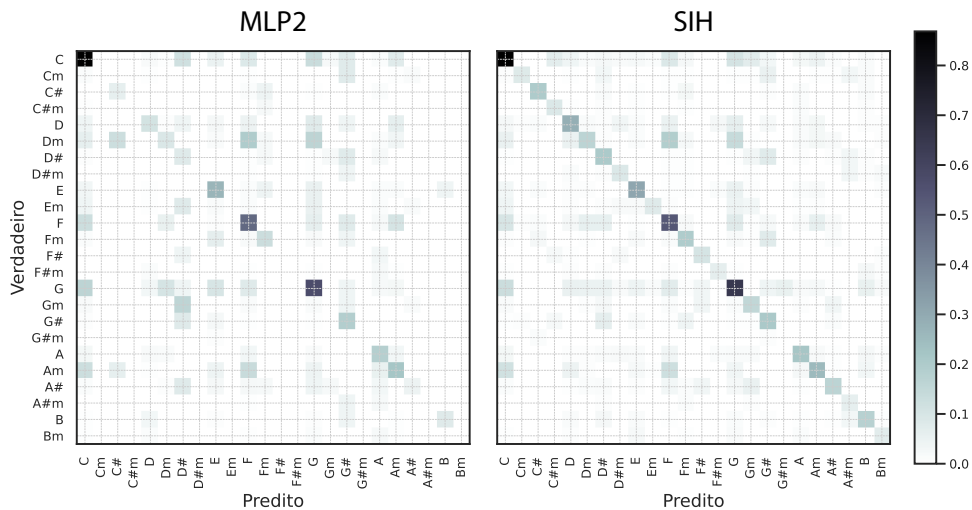
Original:	C	F	C	G	-----	Cm7	A#	G#	C	
MLP2:	C	F	C	G	-----	G#	A#	G#	C	
SIH:	C	F	C	G	-----	Cm	A#	G#	C	

Fonte: Autoria Própria (2022).

acordes. Mesmo assim, a percepção de classificação do acorde de Cm é um diferencial interessante se comparado ao resultado da MLP2, sendo preferido o SIH como sistema harmonizador.

Na Figura 38 foi colocada à esquerda a matriz de confusão obtida pela MLP2 no experimento da Seção 8.2 de forma comparativa, com a matriz de confusão do SIH.

Figura 38 – Matrizes de confusão resultante da MLP2 e SIH em caráter comparativo, geradas com resultados da melhor execução de cada.



Fonte: Autoria Própria (2022).

Pode-se ver claramente a diferença, com uma linha diagonal mais definida indicando mais classificações corretas. Há também semelhanças, como maior número de acertos para os acordes C, F e G, e leves linhas verticais e horizontais para os mesmos acordes, mesma característica simplificadora que observada anteriormente. Por fim, pode-se observar que o SIH é capaz de realizar classificações melhores que todos outros sistemas observados, mantendo a capacidade generalista de harmonização.

9 CONCLUSÃO

No presente trabalho, foi apresentado o desenvolvimento de um sistema inteligente capaz de harmonizar melodias musicais. Diversos conceitos de música, redes neurais, aprendizado profundo e métricas de desempenho foram revisados, bem como levantado o estado da arte sobre o tema. Uma metodologia foi proposta, envolvendo padronizações e codificações de informações musicais, além de configurações possíveis de RNAs e formas de analisar resultados. Por fim, diferentes arquiteturas de RNAs foram postas à prova, usando desde métodos convencionais da literatura, até a criação de uma nova forma de representar música, por meio da IC, usada em uma arquitetura própria gerada para os fins deste projeto.

Métodos que faziam uso de arquiteturas mais simples, como é o caso da MLP, retrataram resultados razoáveis e compatíveis com o encontrado na literatura, em estudos similares. Por sua vez, o uso de CNNs convencionais mostrou resultados interessantes, considerando uma entrada diferente do seu propósito original, uma imagem sintética. O processo de otimização de hiperparâmetros contribuiu para a observação de que, para a tarefa em questão, uma rede mais simples era ideal. Por fim, após suscetivas adaptações, encontrou-se um modelo final capaz de realizar adequadamente a tarefa proposta, com resultados mais precisos que os da literatura, dando respostas mais diversas com característica simplificadora, sendo por si só uma arquitetura única passível de harmonizar melodias musicais.

A continuidade do projeto, por outro lado, poderia contar com alguns ajustes, melhorias e incrementos, por exemplo:

- Além das medidas estatísticas definidas e usadas para avaliar a performance de cada modelo, poder-se-ia usar medidas musicais especificamente criadas para metrificar qualidades musicais, como é o caso de Sun, Wu e Yuan (2022);
- Nesse mesmo sentido, medidas também podem ser extraídas das melodias musicais, como observado no trabalho de Coca *et al.* (2010), podendo ser inseridas juntamente no conjunto de dados de entrada do sistema, provendo ainda mais informação e possivelmente auxiliando no processo de treinamento e previsão;
- Outra possibilidade seria conduzir um levantamento qualitativo das produções do sistema, dirigindo questionários com ouvintes e verificando sua preferência frente a harmonias originais e aquelas geradas pelo sistema, ou então a saída do sistema poderia ser analisada por especialistas, como se o sistema fosse um aluno de teoria musical e sua produção, uma prova, para a qual o especialista poderia atribuir uma nota e avaliar de forma subjetiva.

Estas sugestões e outras mais são exemplos de como este é um tema que pode ser amplamente explorado e aprofundado, com contribuições relevantes para a integração entre arte e máquina, compositor e computador.

REFERÊNCIAS

- ACADEMY, D. **O Neurônio, Biológico e Matemático**. 2021. Disponível em: <https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/>. Acesso em: 26 de out. de 2021.
- ADLER, M. *et al.* **Portable Network Graphics (PNG) Specification**. 2003. Disponível em: <https://www.w3.org/TR/2003/REC-PNG-20031110/>. Acesso em: 17 de nov. de 2021.
- AREL, I; ROSE, D; KARNOWSKI, T. Deep Machine Learning – A New Frontier. **IEEE Computational Intelligence Magazine**, v. 5, n. 4, p. 13–18, 2010.
- ARTSTEIN, R; POESIO, M. Inter-Coder Agreement for Computational Linguistics. **Computational Linguistics**, v. 34, n. 4, p. 555–596, 2008.
- ASSOCIATION, M. **The Complete MIDI 1.0 Detailed Specification**. 3. ed. La Habra, CA: MMA, 2014.
- BACZYŃSKI, D; KOPYT, M; GULCZYŃSKI, T. Advanced Ensemble Methods Using Machine Learning and Deep Learning for One-Day-Ahead Forecasts of Electric Energy Production in Wind Farms. **Energies**, Multidisciplinary Digital Publishing Institute, v. 15, n. 4, p. 1252, 2022.
- BELOTTI, J. *et al.* Neural-based ensembles and unorganized machines to predict streamflow series from hydroelectric plants. **Energies**, Multidisciplinary Digital Publishing Institute, v. 13, n. 18, p. 4769, 2020.
- BENGIO, Y. Learning deep architectures for AI. **Foundations and Trends in Machine Learning**, v. 2, n. 1, p. 1–27, 2009.
- BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 7700 LECTU, p. 437–478, 2012.
- BENWARDS, B; SAKER, M. **Music in Theory and Practice**. 8. ed. New York, USA: McGraw-Hill, 2008.
- BERNARDES, G. *et al.* Conchord: An application for generating musical harmony by navigating in the tonal interval space. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9617 LNCS, p. 243–260, 2016.
- BÍBLIA, A. T. Provérbios. *In*: BÍBLIA. Português. **Bíblia Sagrada**: contendo o Antigo e o Novo Testamento. Edição Revista e Corrigida. Santo André, SP: Geográfica editora, 2012. p. 776.
- BISHOP, C. **Pattern Recognition and Machine Learning**. 1. ed. Cambridge: Springer, 2006.
- BOBKO, P. **Correlation and Regression: Applications for Industrial Organizational Psychology and Management**. 2. ed. London, UK: SAGE Publications, 2001.
- BOROVYKH, A; BOHTE, S; OOSTERLEE, C. Dilated convolutional neural networks for time series forecasting. **Journal of Computational Finance**, v. 22, n. 4, p. 73–101, 2018.

- BRAGA, A; CARVALHO, A; LUDERMIR, T. **Redes Neurais Artificiais: Teoria e aplicações**. 4. ed. Rio de Janeiro: LTC, 2000.
- BRANCO, P; TORGO, L; RIBEIRO, R. A Survey of Predictive Modelling under Imbalanced Distributions. **ACM Computing Surveys**, v. 49, n. 31, p. 1–50, 2017.
- CASTRO, L. **Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications**. 1. ed. Boca Raton: Chapman and Hall/CRC, 2007.
- CHATHURANGA, E; RATNAYAKE, H; PREMARATNE, I. An expert system to generate chords for melodies composed in eastern music format. **International Conference on Computer, Communications and Electronics**, p. 501–504, 2017.
- CHEDIAK, A. **Harmonia e Improvisação**. 7. ed. Rio de Janeiro: Lumiar Editora, 1986.
- CHEN, P; POPOVICH, P. **Correlation: parametric and nonparametric measures**. 07-139. ed. London, UK: Sage University Paper Series on Quantitative Applications in the Social Sciences, 2002.
- CHUAN, C; CHEW, E. Generating and evaluating musical harmonizations that emulate style. **Computer Music Journal**, v. 35, n. 4, p. 64–82, 2011.
- CIODARO, T. *et al.* Online particle detection with Neural Networks based on topological calorimetry information. **Journal of Physics: Conference Series**, v. 368, p. 012030, 2012.
- CLEVERT, D; UNTERTHINER, T; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (ELUs). **International Conference on Learning Representations**, p. 1–14, 2016.
- COCA, A. *et al.* Characterizing chaotic melodies in automatic music composition. **Chaos – An Interdisciplinary Journal**, v. 20, n. 3, p. 33125, 2010.
- COCA, A; ZHAO, L. Musical Scales Recognition via Deterministic Walk in a Graph. **Brazilian Conference on Intelligent Systems**, p. 151–156, 2016.
- COHEN, J. A Coefficient of Agreement for Nominal Scales. **Educational and Psychological Measurement**, v. 20, n. 1, p. 37–46, 1960.
- COSTA, L. *et al.* Developing a Measure Image and Applying to Deep Learning. **Music Encoding Conference**. Halifax, CA: Music Encoding Initiative, 2022.
- CRAMER, H. **Mathematical Methods Of Statistics**. 1. ed. Bombay: Asia Publishing House, 1962.
- CYBENKO, G. **Continuous valued neural networks with two hidden layers are sufficient**. Illinois: University of Illinois at Urbana-Champaign, 1988.
- CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals and Systems**, v. 2, n. 4, p. 303–314, 1989.
- DAVIS, J; GOADRICH, M. The relationship between Precision-Recall and ROC curves. **International Conference on Machine learning**, p. 233–240, 2006.
- DE BOOM, C. *et al.* Rhythm, Chord and Melody Generation for Lead Sheets Using Recurrent Neural Networks. **Communications in Computer and Information Science**, v. 1168 CCIS, p. 454–461, 2020

- TADANO, Y; SIQUEIRA, H; ALVES, T. Unorganized machines to predict hospital admissions for respiratory diseases. **Latin American Conference on Computational Intelligence**, p. 1–6, 2016.
- DEGROOT, M; SCHERVISH, M. **Probability and Statistics**. 4. ed. Harlow: Pearson, 2014.
- DERTAT, A. **Applied Deep Learning – Part 4: Convolutional Neural Networks**. 2017. Disponível em: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>. Acesso em: 13 de nov. de 2021.
- DONG, H.-W. *et al.* Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. **AAAI Conference on Artificial Intelligence**, p. 34–41, 2018.
- DUA, M. *et al.* An Improved RNN-LSTM based Novel Approach for Sheet Music Generation. **Procedia Computer Science**, Elsevier B.V., v. 171, p. 465–474, 2020.
- EBCIOĞLU, K. An Expert Four-part Harmonizing Chorales. **Computer Music Journal**, v. 12, n. 3, p. 43–51, 1988.
- EMERSON, P. The original Borda count and partial voting. **Social Choice and Welfare**, v. 40, n. 2, p. 353–358, 2013.
- FAHLMAN, S. An empirical study of learning speed in back-propagation networks. **Neural Networks**, v. 6, n. 3, p. 1–19, 1988.
- FERREIRA, J; PATINO, C. What does the p-value really mean? **Jornal Brasileiro de Pneumologia**, v. 41, n. 5, p. 485–485, 2015.
- FEURER, M; HUTTER, F. Hyperparameter Optimization. *In*: HUTTER, F; KOTTHOFF, L; VANSCHOREN, J. **Automated Machine Learning: Methods, Systems, Challenges**. 1. ed. Manhattan: Springer, 2019. p. 3–33.
- FL STUDIO. **Piano roll**. 2021. Disponível em: <https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/pianoroll.htm>. Acesso em: 17 de nov. de 2021.
- FREITAS, A; GUIMARÃES, F; RUELA, A. Computação evolutiva multiobjetivo para harmonização de melodias. **Simpósio Brasileiro de Pesquisa Operacional**, p. 2000–2011, 2011.
- FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. **Journal of the American Statistical Association**, v. 32, n. 200, p. 675–701, 1937.
- FUKUMOTO, M. Creation of music chord progression suited for user’s feelings based on interactive genetic algorithm. **International Conference on Advanced Applied Informatics**, p. 757–762, 2014.
- GÓMEZ, R. **Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names**. 2018. Disponível em: https://gombu.github.io/2018/05/23/cross_entropy_loss/. Acesso em: 2 de nov. de 2021.
- GONZALEZ, R; WOODS, R. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Prentice Hall, 2010.
- GOOD, M. **Beyond PDF – Exchange and Publish Scores with MusicXML**. [S.l.: s.n.], 2013.

- GOODFELLOW, I; BENGIO, Y; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.
- GORODKIN, J. Comparing two K-category assignments by a K-category correlation coefficient. **Computational Biology and Chemistry**, v. 28, n. 5-6, p. 367–374, 2004.
- GRANDINI, M; BAGLI, E; VISANI, G. **Metrics for Multi-Class Classification: an Overview**. arXiv, 2020.
- GUERREIRO, M. *et al.* Anomaly Detection in Automotive Industry Using Clustering Methods – A Case Study. **Applied Sciences**, v. 11, n. 21, p. 9868, 2021.
- GUEST, I. **Harmonia – Método Prático**. 1. ed. [S.l.]: Lumiar, 2010.
- HANSEN, C. **Activation Functions Explained – GELU, SELU, ELU, ReLU and more**. 2019. Disponível em: <https://mlfromscratch.com/activation-functions-explained/>. Acesso em: 3 de nov. de 2021.
- HAO, T. ChordAL: A chord-based approach for music generation using Bi-LSTMs. **International Conference on Computational Creativity**, p. 364–365, 2019.
- HARRIS, D; HARRIS, S. **Digital Design and Computer Architecture**. 2. ed. Waltham, MA: Elsevier, 2013.
- HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Hamilton: Prentice Hall, 2008.
- HE, K. *et al.* Deep residual learning for image recognition. **Computer Society Conference on Computer Vision and Pattern Recognition**, p. 770–778, 2016.
- HEBB, D. The Organization of Behavior; A Neuropsychological Theory. **The American Journal of Psychology**, v. 63, n. 4, p. 633, 1949.
- HELMSTAEDTER, M. *et al.* Connectomic reconstruction of the inner plexiform layer in the mouse retina. **Nature**, v. 500, n. 7461, p. 168–174, 2013.
- HINTON, G. *et al.* Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. **IEEE Signal Processing Magazine**, v. 29, n. 6, p. 82–97, 2012.
- HINTON, G; SRIVASTAVA, N; SWERSKY, K. **Overview of mini-batch gradient descent**. 2014. Disponível em: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Acesso em: 30 de set. de 2022.
- HINTON, G. *et al.* **Improving neural networks by preventing co-adaptation of feature detectors**. arXiv, 2012.
- HU, J; GUAN, Y; ZHOU, C. A hierarchical approach to simulation of the melodic harmonization process. **International Conference on Intelligent Computing and Intelligent Systems**, p. 780–784, 2010.
- HUANG, A; WU, R. **Deep Learning for Music**. arXiv, 2016.
- HUANG, G. *et al.* **Densely Connected Convolutional Networks**. arXiv, 2016.
- HUANG, G.-B; ZHU, Q.-Y; SIEW, C.-K. Extreme learning machine: Theory and applications. **Neurocomputing**, v. 70, n. 1-3, p. 489–501, 2006.
- HUBEL, D; WIESEL, T. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of Physiology**, v. 160, n. 1, p. 106–154, 1962.

- HUNT, N; TYRRELL, S. **Stratified Sampling**. 2004. Disponível em: <https://archive.ph/20131013132818/http://nestor.coventry.ac.uk/~nhunt/meths/strati.html>. Acesso em: 8 de nov. de 2021.
- IOFFE, S; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. **International Conference on Machine Learning**, p. 448–456, 2015.
- JAEGER, H. **The “echo state” approach to analysing and training recurrent neural networks – with an Erratum note**. [S.l.: s.n.], 2010.
- JAMES, G. *et al.* **An Introduction to Statistical Learning**. 2. ed. New York, NY: Springer New York, 2013.
- JEFFREY, A; DAI, H.-H. **Handbook of Mathematical Formulas and Integrals**. 4. ed. Burlington: Elsevier, 2008.
- JURAFSKY, D; MARTIN, J. **Speech and Language Processing**. 2. ed. Stanford: Pearson, 2008.
- JURMAN, G; RICCADONNA, S; FURLANELLO, C. A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. **PLoS ONE**, v. 7, n. 8, p. e41882, 2012.
- KAIZER, W. **Curso de Harmonia**. 1. ed. Vila Velha: [s.n.], 2017.
- KAUTZ, T; ESKOFIER, B; PASLUOSTA, C. Generic performance measure for multiclass-classifiers. **Pattern Recognition**, v. 68, p. 111–125, 2017.
- KENDALL, M. A NEW MEASURE OF RANK CORRELATION. **Biometrika**, v. 30, n. 1-2, p. 81–93, 1938.
- KINGMA, D; BA, J. Adam: A Method for Stochastic Optimization. **International Conference on Learning Representations**, 2015.
- KOELLREUTER, H.-J. **Harmonia Funcional: Introdução À Teoria Das Funções Harmônicas**. 3. ed. Campos Elíseos, SP: Ricordi Brasileira S. A., 1978.
- KOHAVI, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. **International Joint Conference of Artificial Intelligence**, p. 1137–1143, 1995.
- KOOPS, H; MAGALHÃES, J; HAAS, W. A functional approach to automatic melody harmonisation. **ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design** p. 47–58, 2013.
- KRIZHEVSKY, A; SUTSKEVER, I; HINTON, G. ImageNet Classification with Deep Convolutional Neural Networks. **Advances in Neural Information Processing Systems**. v. 25, p. 1097–1105, 2012.
- LECUN, Y; BENGIO, Y; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.
- LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.
- LI, F.-F; KRISHNA, R; XU, D. **Convolutional Neural Networks for Visual Recognition**. 2021. Disponível em: <https://cs231n.github.io/convolutional-networks/>. Acesso em: 4 de nov. de 2021.

- LIM, H; LEE, K. Chord Generation from Symbolic Melody Using BLSTM Networks. **International Society for Music Information Retrieval Conference**, p. 621–627, 2017.
- LIM, H; RHYU, S; LEE, K. **CSV Leadsheet Database**. 2017. Disponível em: http://marg.snu.ac.kr/chord_generation/. Acesso em: 6 de jun. de 2019.
- LIPPMANN, R. An introduction to computing with neural nets. **IEEE ASSP Magazine**, v. 4, n. 2, p. 4–22, 1987.
- LIU, H.-M; YANG, Y.-H. Lead Sheet Generation and Arrangement by Conditional Generative Adversarial Network. **International Conference on Machine Learning and Applications**, p. 722–727, 2018.
- LU, L. *et al.* Dying ReLU and initialization: Theory and numerical examples. **Communications in Computational Physics**, v. 28, n. 5, p. 1671–1706, 2020.
- MAAS, A; HANNUN, A; NG, A. Rectifier nonlinearities improve neural network acoustic models. **International Conference on Machine Learning**, v. 28, 2013.
- MAGALHÃES, J; KOOPS, H. Functional generation of harmony and melody. **ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design**, p. 11–21, 2014.
- MAJIDI, M; TOROGHI, R. **A Combination of Multi-Objective Genetic Algorithm and Deep Learning for Music Harmony Generation**. arXiv, 2021.
- MAKRIS, D; KAYRDIS, I; SIOUTAS, S. Automatic Melodic Harmonization. **International Joint Conference on Neural Networks**, p. 146–165, 2013.
- MAKRIS, D; KAYRDIS, I; SIOUTAS, S. Automatic Melodic Harmonization. *In*: KOSTAGIOLAS, P; MARTZOUKOU, K; LAVRANOS, C. **Trends in Music Information Seeking, Behavior, and Retrieval for Creativity**. 1. ed. Hershey, PA: IGI Global, 2016. p. 146–165.
- MATTHEWS, B. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. **Biochimica et Biophysica Acta (BBA) – Protein Structure**, v. 405, n. 2, p. 442–451, 1975.
- MCCULLOCH, W; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943.
- MED, B. **Teoria da Música**. 4. ed. Brasília, DF: MusiMed, 1996.
- MENON, A. *et al.* Characterization of a class of sigmoid functions with applications to neural networks. **Neural Networks**, v. 9, n. 5, p. 819–835, 1996.
- MESSICK, P. **Maximum MIDI: Music Applications in C++**. 1. ed. [S.l.]: Manning, 1997.
- MILLER, N. **Heritage and Innovation of Harmony: A Study of West Side Story**. 54f. Tese (Doutorado) – University of North Texas, 2006.
- NOUGUÉ, C. **O que é o Sistema Temperado**. 2011. Disponível em: <http://www.aboamusica.com.br/2011/11/o-que-e-o-sistema-temperado.html>. Acesso em: 26 de jun. de 2019
- OPITZ, J; BURST, S. **Macro F1 and Macro F1**. arXiv, 2019.
- PHON-AMNUAISUK, S; WIGGINS, G. The Four-Part Harmonisation Problem. **Symposium on Musical Creativity**, p. 28–34, 1999.

- POWERS, D. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. **International Journal of Machine Learning Technology**, v. 2, n. 1, p. 37–63, 2011.
- PRISCO, R; ZACCAGNINO, R. An evolutionary music composer algorithm for bass harmonization. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 5484 LNCS, p. 567–572, 2009.
- PROVOST, F; FAWCETT, T; KOHAVI, R. The Case against Accuracy Estimation for Comparing Induction Algorithms. **International Conference on Machine Learning**, p. 445–453, 1998.
- RACZYŃSKI, S; FUKAYAMA, S; VINCENT, E. Melody Harmonization With Interpolated Probabilistic Models. **Journal of New Music Research**, v. 42, n. 3, p. 223–235, 2013.
- RADICIONI, D; ESPOSITO, R. BREVE: An HMPerceptron-Based Chord Recognition System. *In*: RAŚ, Z; WIECZORKOWSKA, A. **Advances in Music Information Retrieval**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 143–164.
- RANGANATHAN, P; PRAMESH, C; AGGARWAL, R. Common pitfalls in statistical analysis: Measures of agreement. **Perspectives in Clinical Research**, v. 8, n. 4, p. 187, 2017.
- RIBEIRO, V; REYNOSO-MEZA, G; SIQUEIRA, H. Multi-objective ensembles of echo state networks and extreme learning machines for streamflow series forecasting. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 95, p. 103910, 2020.
- RIPLEY, B. **Pattern Recognition and Neural Networks**. 8. ed. Cambridge: Cambridge University Press, 2005.
- ROIG-FRANCOLÍ, M. **Harmony in Context**. 2. ed. Cincinnati, OH: McGraw-Hill, 2010.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958.
- RUMELHART, D; HINTON, G; WILLIAMS, R. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.
- RUSSELL, S; NORVIG, P. **Artificial Intelligence: A Modern Approach**. New Jersey: Prentice Hall, 2009.
- RYBNIK, M; HOMENDA, W. Extension of knowledge-driven harmonization model for tonal music. **International Joint Conference on Neural Networks**, p. 10–15, 2012.
- SANTOS, A. **Harmonia 1**. [S.l.]: Câne Musical, 2012.
- SANTURKAR, S. *et al.* How Does Batch Normalization Help Optimization? **Advances in Neural Information Processing Systems**, v. 31, p. 2483–2493, 2018.
- SCHOENBERG, A. **Theory of Harmony**. [S.l.]: University of California Press, 1978.
- SCIKIT-LEARN. **3.3.2.13. Matthews correlation coefficient**. 2021. Disponível em: https://scikit-learn.org/stable/modules/model_evaluation.html#matthews-correlation-coefficient. Acesso em: 11 de nov. de 2021.
- SHAFFER, K; HUGHES, B; MOSELEY, B. **Open Music Theory: Harmonic Functions**. 2014. Disponível em: <http://openmusictheory.com/harmonicFunctions.html>. Acesso em: 26 de jun. de 2019

- SHEN, H; LEE, C. An interactive Whistle-to-Music composing system based on transcription, variation and chords generation. **Multimedia Tools and Applications**, v. 53, n. 1, p. 253–269, 2011.
- SIMON, I; MORRIS, D; BASU, S. MySong: Automatic accompaniment generation for vocal melodies. **Conference on Human Factors in Computing Systems**, p. 725–734, 2008.
- SIMONYAN, K; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **International Conference on Learning Representations**, p. 1–14, 2014.
- SIQUEIRA, H. *et al.* Echo state networks for seasonal streamflow series forecasting. **International Conference on Intelligent Data Engineering and Automated Learning**, p. 226–236, 2012.
- SIQUEIRA, H. *et al.* Echo State Networks in Seasonal Streamflow Series Prediction. **Learning and Nonlinear Models**, v. 10, p. 181–191, 2012.
- SIQUEIRA, H; LUNA, I. Performance comparison of feedforward neural networks applied to streamflow series forecasting. **Mathematics In Engineering Science And Aerospace**, v. 10, n. 1, p. 41–53, 2019.
- SNOEK, J; LAROCHELLE, H; ADAMS, R. **Practical Bayesian Optimization of Machine Learning Algorithms**. arXiv, 2012.
- SOKOLOVA, M; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information Processing and Management**, v. 45, n. 4, p. 427–437, 2009.
- SOYSA, A; LOKUGE, K. Interactive machine learning for incorporating user emotions in automatic music harmonization. **International Conference on Information and Automation for Sustainability**, p. 114–118, 2010.
- SUN, W; WU, J; YUAN, S. Melodic Skeleton: A Musical Feature for Automatic Melody Harmonization. **International Conference on Multimedia and Expo Workshops**, p. 1–6, 2022.
- SZEGEDY, C. *et al.* Going deeper with convolutions. **Computer Society Conference on Computer Vision and Pattern Recognition**, p. 1–9, 2015.
- SZEGEDY, C. *et al.* **Rethinking the Inception Architecture for Computer Vision**. arXiv, 2015.
- TALLÓN-BALLESTEROS, A; RIQUELME, J. Data Mining Methods Applied to a Digital Forensics Task for Supervised Machine Learning. *In*: MUDA, A. **Computational Intelligence in Digital Forensics: Forensic Investigation and Applications**. 1. ed. [S.l.]: Springer Cham, 2014. p. 413–428.
- TECHNOLOGY, N. **FRIEDMAN TEST**. 2015. Disponível em: <https://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/friedman.htm>. Acesso em: 5 de out. de 2022.
- TSUSHIMA, H; NAKAMURA, E; YOSHII, K. Bayesian Melody Harmonization Based on a Tree-Structured Generative Model of Chord Sequences and Melodies. **IEEE/ACM Transactions on Audio Speech and Language Processing**, v. 28, p. 1644–1655, 2020.
- WASSERMANN, G; GLICKMAN, M. Automated Harmonization of Bass Lines from Bach Chorales: A Hybrid Approach. **Computer Music Journal**, v. 43, n. 2-3, p. 142–157, 2020.
- WASSERSTEIN, R; LAZAR, N. The ASA Statement on p -Values: Context, Process, and Purpose. **The American Statistician**, v. 70, n. 2, p. 129–133, 2016.

- WICHARD, J; OGORZALEK, M. Time series prediction with ensemble models. **International Joint Conference on Neural Networks**, p. 1625–1630, 2004.
- WIDROW, B; HOFF, M. Adaptive switching circuits. **IRE WESCON Convention Record**, p. 96–104, 1960.
- WIGGINS, G. *et al.* Evolutionary Methods for Musical Composition. **International Journal of Computing Anticipatory Systems**, 1998.
- WU, J. *et al.* Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. **Journal Of Electronic Science And Technology**, v. 17, n. 1, p. 26–40, 2019.
- WU, Y.-C; CHEN, H. Emotion-flow guided music accompaniment generation. **International Conference on Acoustics, Speech and Signal Processing**, p. 574–578, 2016.
- WU, Y.-C; CHEN, H. Generation of Affective Accompaniment in Accordance with Emotion Flow. **IEEE/ACM Transactions on Audio Speech and Language Processing**, v. 24, n. 12, p. 2277–2287, 2016.
- YANG, L; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. **Neurocomputing**, v. 415, p. 295–316, 2020.
- YOU, S; LIU, P. Automatic chord generation system using basic music theory and genetic algorithm. **International Conference on Consumer Electronics-Taiwan**, p. 1–2, 2016.
- YOUNG, R. Terminology for Logarithmic Frequency Units. **The Journal of the Acoustical Society of America**, v. 11, n. 1, p. 134–139, 1939.
- ZUBEN, F. **Modelos paramétricos e não-paramétricos de redes neurais artificiais e aplicações**. 244f. Tese (Doutorado) – Universidade Estadual de Campinas, 1996.

APÊNDICE A – Artigo Submetido a Revista *Applied Artificial Intelligence*

Um artigo com os resultados mostrados na Seção 8.2 foi submetido para a revista internacional *Applied Artificial Intelligence*. Na sequência, o artigo na íntegra como foi submetido, e o e-mail de confirmação de submissão podem ser observados.

Neural Networks And Ensemble Based Architectures To Automatic Musical Harmonization: A Performance Comparison

Lucas F. P. Costa^a, Tathiana M. Barchi^a, Erikson F. de Morais^a,
Andrés E. Coca^c, Elder E. Schemberger^c Marcella S. R. Martins^b and Hugo V.
Siqueira^{a,b}

^aGraduate Program in Computer Sciences (PPGCC) - Federal University of Technology - Paraná (UTFPR)

^bDepartment of Electronics Engineering (DAELE) - Federal University of Technology - Paraná (UTFPR)

^cGraduate Program in Computer Engineering (COENC) - Federal University of Technology - Paraná (UTFPR)

ABSTRACT

Harmony can be defined in a musical way as art that combines several musical notes reproduced simultaneously to create sounds that are coherent to human ears and serve as accompaniment and filling. However, working out harmony is not a simple task. It requires knowledge, experience, and an intense study of music theory, which takes time to reach good skills. Thus, systems capable of automatically harmonizing melodies are beneficial for experienced and novice musicians. In this paper, a comparative study between distinct architectures and ensembles of Artificial Neural Networks was proposed to solve the problem of musical harmonization, seeking consistent results with rules of music theory: Multilayer Perceptron (MLP), Radial Basis Function network (RBF), Echo State Network (ESN), and Extreme Learning Machines (ELM). For this, a processed and defined melody with symbolic musical data serves as input to the system, having been trained from a musical database that contains melody and harmony. The output is the chord sequence to be applied to the melody. The results were analyzed with quantitative measures and the ability to melody adaptation. The performances were favorable to the MLP, which could generate harmonies according to the objectives.

KEYWORDS

Music theory; neural network; ensemble; performance evaluation

1. Introduction

When three or more musical sounds are played simultaneously, a sound set called a chord is formed. When several chords are put in order, there is harmony. Naturally, such harmonies are designed to accompany musical melodies, which bring meaning, proportion, and symmetry to a song Roig-Francolí (2010). Therefore, harmony is the art of choosing chords that correctly complement a musical line.

The harmonic choices for a given melody are limited, although several options can fit the exact musical moment. However, like all art, the harmonization of a given melody requires time, experience, and musical study, especially concerning harmony theory Koops, Pedro, and de Haas (2013).

Understanding the harmonization task as an analytical process, research on its constitution, and a way to automate it represents a valuable contribution to the study of music, artificial intelligence, and the Musical Information Retrieval (MIR) Koops et al. (2013).

Several attempts to solve the problem of automatic music harmonization exist in the literature, often using evolutionary methods, such as with Genetic Algorithms (GAs) Nakashima, Imamura, Ogawa, and Fukumoto (2010); Wiggins, Papadopoulos, and Phon-Amnuaisuk (1998), statistics Chuan (2011) or with rule-based systems Ebcioğlu (1988); Koops et al. (2013).

Neural Networks have also been used. In Lim and Lee (2017), the authors applied Bidirectional Long Short-Term Memory (BLSTM) networks, trained in symbolic data of actual song harmonies, reaching 50% accuracy. Other works, such as Dong, Hsiao, Yang, and Yang (2017); A. Huang and Wu (2016); Liu and Yang (2018), seek to use similar methodologies, however, with a different objective, which is the whole music generation, including melody and even the arrangement.

In this paper, different architectures and ensembles of Artificial Neural Networks (ANNs) were applied to solve the problem of harmonizing music, and their performance was compared for the following algorithms: Multilayer Perceptron (MLP), Radial Basis Function network (RBF), Echo State Network (ESN) and Extreme Learning Machines (ELM). Using a database of symbolically represented songs containing melodies and chords, the ANNs were trained and later became able to generate new harmonies. Statistical and performance measures were calculated on the results of each model to describe the production quality, making it possible to compare them quantitatively and choose the best one. The results met the objectives, reaching consistent and reasonable performance levels.

This paper is divided into sections of which: Sections 2 and 3 briefly describe musical harmony theory and establish the applied methodology, respectively. Results are exposed and discussed in Section 4, of which conclusions are drawn in Section 5.

2. Musical Harmony Theory

According to Schoenberg (1978), harmony is the study of sounds played simultaneously and their relationship between architectural, melodic, and rhythmic values, as well as their meaning and relative strength to each other Schoenberg (1978). The study of harmony establishes essential rules and concepts, some of which are presented below.

The construction of chords can be commonly done using triads, which are three musical notes played simultaneously, usually consisting of notes overlapping in intervals of thirds. Depending on the type of these intervals, the chords can be named as major, minor, augmented, or diminished Terefenko (2014).

The term *harmonic field* is used to refer to the group of chords that can be played on a given note scale. In it are inserted the chords that constitute a functional relationship with each other. For example, the chords used in the key of C major are C major (or maj), D minor (or min), E min, F maj, G maj, A min, and B min flatted fifth (or diminished) Roig-Francolí (2010).

In harmony terms, *function* is understood as the characteristic of a chord, having its value concerning the others. The tone is defined by essential functions traditionally called tonic, dominant, and subdominant, respectively, each represented by a chord. For the other chords present in the harmonic field, their functions are inherited from the main functions, called secondary functions, and these chords can be named as

relatives Terefenko (2014).

Respecting these functions, the normal movement of the chords within the song is established as Roig-Francolí (2010):

- The dominant function is commonly preceded by the tonic, giving the resolution of the dissonance (in the tonic);
- The subdominant function has greater freedom, but its sense of movement usually leads to the dominant;
- The tonic function is usually applied at the beginning or at the end of a song or musical phrase, giving the idea of completion.

Exemplifying using the C major harmonic field chords: G maj chord has a dominant function and B diminished is its relative; F maj is subdominant, and its relative is D min; and for the tonic function, there is C maj, with A min and E min being their relative.

Functional harmony aims to study the functioning of these functions and establish rules and concepts for their correct use within the music. By understanding the functions, it is possible to harmonize songs coherently. Also, rules that, when respected, indicate the quality of the work Roig-Francolí (2010); Terefenko (2014).

3. Proposed Method For Automatic Musical Harmonization

The proposed method for this work is illustrated in Fig 1 and can be described in some steps: a database containing both melody and harmony of several songs in symbolic format is standardized and encoded into numeric vectors, later used for training ANNs models, which will be able to generate conditioned harmonies by the melodies at the entrance; a chord by measure¹Roig-Francolí (2010). These ANNs models then will be used to create ensembles. The resulting harmonies will be evaluated quantitatively compared to the original harmonies.

[Figure 1 about here.]

The following will discuss how musical data processing is done, the description of the ANNs and ensembles used, and the measures used for quantitative evaluation.

3.1. Data Processing

To carry out the training of the ANNs, data are needed. We use the *CSV Leadsheet Database* Lim, Rhyu, and Lee (2017). It contains songs of rock, pop, country, jazz, folk, R&B, children's songs, etc., in CSV format, all in the major key, processed and organized, presenting information about time signature, measure, melody note, chord played, duration and other features for each song.

More processing phases are necessary to use this data for the training of ANNs. First, the songs need to be standardized, and then, information needs to be encoded to be understood by the models.

Standardization takes place at the following levels:

- **Tonality:** to keep information consistent, the songs are all transposed into a common tone. In this case, all the songs were transposed to C major tone,

¹Division of a piece of music into a time series.

which gives higher confidence about which notes to expect and defines a common harmonic field;

- **Note duration:** in that database, the duration of each note follows a numerical pattern independent of the metric or time signature²Roig-Francolí (2010). However, the sum of notes durations of a measure will also be different for different time signatures, which requires normalization. Therefore, the length of each note can be multiplied by the inverse of the song’s time signature Costa, Ogoshi, Martins, and Siqueira (2022). In this way, all measures will have the same total duration;
- **Chord types:** one must consider the vast amount of chords that the chaining of musical notes can form as a problem. To simplify the possible system responses, only major and minor triads were considered, resulting in 24 chord classes.

Following these steps, there is a standardized tone, rhythm, and harmony database. The encoding of both notes and chords is done using the *one-hot* method. In this sense, a vector \mathbf{n}_{note} is used to represent a musical note. This vector contains 12 elements representing each note of the chromatic scale plus one position for pauses. Thus, the note D \sharp could be written as (1):

$$\mathbf{n}_{\text{D}\sharp} = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]. \quad (1)$$

Likewise, the representation of a chord takes a vector $\mathbf{c}_{\text{chord}}$ of 24 positions, 12 for major chords and 12 for minor chords, interspersed with each other, for example, C maj, C minor, C \sharp maj, C \sharp minor, D maj, and so on. In that way, D maj chord is represented as (2):

$$\mathbf{c}_{\text{D maj}} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]. \quad (2)$$

Considering that the analysis takes place by musical measure, a chord and a variable number of notes and rests are used for one measure. For a given measure, consider that three musical notes are played in sequence: D, F, and C. The encoding of each note would result in the following matrix $\mathbf{N}_{\text{NPM} \times 12}$, where NPM stands for Notes Per Measure and 12 is the size of \mathbf{n}_{note} (3):

$$\mathbf{N}_{3 \times 12} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

To simplify this matrix into a single vector, the lines are added, which leads to the sum vector \mathbf{s} (4):

$$\mathbf{s} = [1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]. \quad (4)$$

Thus, for each measure, there is $\mathbf{c}_{\text{chord}}$ for the chord and \mathbf{s} representing the notes.

3.2. Artificial Neural Networks

For this work, the performance of four different architectures of ANNs is compared. They are described below.

²Grouping of values with musical meaning, for rhythmic determination.

3.2.1. Multilayer Perceptron (MLP)

Its main characteristic is the existence of at least one intermediate (*hidden*) layer of neurons between the input and output layers of the network, being conventional the use of only one or even two hidden layers Haykin (2008). It is a versatile model with wide application in areas such as universal functions approximation, time series forecasting, systems optimization, and pattern recognition Kachba et al. (2020); Siqueira and Luna (2019). In an MLP, neurons from different layers are densely connected, but there is no connection between those of the same layer. The training takes place in a supervised manner using the *backpropagation* algorithm Haykin (2008). This paper employed two MLPs: one and two hidden layers.

3.2.2. Radial Basis Function Networks (RBF)

Local learning networks like the RBF usually consist of three layers, the input layer consisting only of source nodes to connect the data to the network. The hidden layer does not calculate weights with the input layer and uses nonlinear radial-based activation functions like the Gaussian. The training of this layer is performed using clustering algorithms, such as the K-Means. Finally, the output layer presents weights between the hidden layer; being these weights adjusted using backpropagation or the Moore-Penrose Pseudo-Inverse (MPPI) operation Haykin (2008). It is possible to use linear or nonlinear functions as activation functions of the output neurons Siqueira and Luna (2019). The sequential use of nonlinear and linear transformations takes advantage of the fact that, for a classification problem, increasing the information space dimension gives a greater probability of finding a linear separation Haykin (2008). This finding is a desirable attribute since, for the problem of this paper, the number of classes is relatively large.

3.2.3. Extreme Learning Machine (ELM)

The great advantage of the ELMs when compared to the traditional MLP is its fast learning process, and good generalization performance with the universal approximation capability de Souza Tadano, Siqueira, and Alves (2016); Siqueira, Boccato, Attux, and Lyra Filho (2012b). It is a *feedforward* architecture and only one hidden layer, in which the weights are not adjusted; that is, it has fixed parameters. Thus, during the training phase, there is no manipulation of the cost function, which is summarized to find the best output layer weights. This task can be accomplished with a linear combiner, which can be done by using MPPI operation G.-B. Huang, Zhu, and Siew (2006). Furthermore, the multitude of musical note arrangements for a given chord requires a model with good generalization capabilities, making it enjoyable to use this network model.

3.2.4. Echo State Network (ESN)

Unlike the previous ones, this is a Recurrent Neural Network. That is, present feedback loops of information, presenting an intrinsic memory capability Ribeiro, Reynoso-Meza, and Siqueira (2020). A three-layer structure is also considered, with a source input layer and a linear combiner based on the MPPI operation as output. The hidden layer is referred to as *dynamic reservoir* and contains sparsely interconnected neurons with fixed weights Jaeger (2010). It has similarities with ELM since the learning process only modifies the weights for the output layer, being efficient due to the fast

convergence Siqueira, Boccato, Attux, and Lyra Filho (2012a). The presence of memory can be attractive because the choice of chord sequences depends not only on the notes played in some measures but also on the context in which they are inserted, that is, on the notes and chords previously played in the song.

3.3. Ensembles

After the training phase, we can create ensembles of ANNs and compare their results. The ensemble is a combination methodology of multiple already trained models to improve the final system response Wichard and Ogorzalek (2004). This combination is because different methods produce different behaviors with the same inputs. A model can present better responses for some input, while another works better for a different kind. A combination approach is then applied to generate the final ensemble output, for example, average, voting, or another neural network Piotrowski Paweł and Baczyński, Kopyt, and Gulczyński (2022).

There is still the necessity of simultaneously presenting diversity and having accurate predictions from each model. The purpose of an ensemble is to improve already existing good results. Ensembles have been used to solve many problems Belotti et al. (2020). This work tested three different ensembles: considering all ANNs, the best three, and the best two considering the error return.

3.4. Quantitative Evaluation

As this is a multiclassification problem, considering inputs and outputs as one-hot encoded vectors, the most indicated and commonly used training metric loss is the Categorical Cross-Entropy Loss, also called Softmax Loss Gómez (2018).

To assess the efficiency of each model, classically, accuracy would be applied. However, as the database used in the training process is composed of actual songs, we can expect it to be unbalanced due to the musical nature of using some notes and chords more than others. Therefore, care must be taken when using accuracy, as it is a problematic measure when dealing with an unbalanced database because the impact of less represented but more critical examples is reduced compared to the majority class Branco, Torgo, and Ribeiro (2015).

Other measures can be used to assess the quality of results, which take into account the issue of unbalance, such as Macro F1-score (F_{1M}) Powers (2007), Matthews Correlation Coefficient (MCC) Cramer (1962) and Cohen’s Kappa Coefficient (κ) Cohen (1960).

4. Experimental Results

The results of the applied proposed methodology for automatic musical harmonization are described. In all, 2249 songs were standardized and filtered. For computational performance and availability reasons, only 20% of the total data was used, with 68182 notes and 18651 chords and measures, as only one chord per measure is being considered. Fig. 2 shows the percentage of the total amount of notes and major and minor chords after the database processing phase.

[Figure 2 about here.]

It is interesting to note a more significant presence of notes C, D, E, G, and A, corresponding to 77.03% of the total notes. These notes are part of a musical scale called “pentatonic,” widely used in styles like blues and rock, evidently present in several other styles due to its functionality.

The notes F and B appear in second place, completing the C major scale, as expected after the tonal transposition phase. Nevertheless, notes with accidentals (\sharp) are also present, demonstrating the plurality of the database used. Similarly, basic function chords (C maj, F maj, and G maj) and primary relatives (D min and A min) are the most substantial presence in the pieces of music, leading to the belief that the harmony employed must be relatively simple for most songs, although all types of chords exist in some proportion, for the 24 types considered. Furthermore, the fact that the database is unbalanced is still notorious.

The following models were evaluated: one-layer MLP (MLP1), two-layers MLP (MLP2), RBF, ESN, and ELM. From those, three ensemble models were also evaluated using voting as combination approach, considering all five networks (ENS5), only the top three (ENS3) and two (ENS2) best regarding loss results. To carry out the network training and testing stages, 60% of the data were separated for training, 20% for validation, and 20% for testing.

Tests were performed considering the variation in the number of neurons in the hidden layer, passing the values 64, 128, and 256. For the models that need training epochs, the selected stopping criterion was a total number of epochs equal to 200, using Stochastic Gradient Descent with $\eta = 0.001$ as the optimizer, saving the best weights when the minimum validation error value is reached for Categorical Cross-Entropy as a function of losses.

To analyze the results, each model was run 30 times. In Table 1 it is possible to observe the average results of loss, accuracy, F_{1M} , MCC, and κ of each model for the number of neurons that brought the best performance, considering the test set. Therefore, ENS3 is the ensemble of MLP1, MLP2, and RBF, and ESN2 is made of MLP1 and MLP2. On average, the models had a F_{1M} of 13.42%, an MCC of 32.54%, and a κ of 31.15%. As we are dealing with an unbalanced database, accuracy as an evaluation measure can bring wrong conclusions about the results, but for reasons of comparison and standard, they are also included.

[Table 1 about here.]

Friedman Friedman (1937) test was applied to the results for the 30 runs of each model regarding the loss in the test set. The p -values achieved were equal to 2.38×10^{-41} . Therefore, it is possible to assume that there are significant changes in the results for different architectures.

Considering the metrics, loss, F_{1M} , MCC, and κ and each model was checked an overall performance (Table 2) for each result, using the Borda count method. The first place was awarded 8 points, the second 7, until the last place received 1 point. For loss, the first place is the smallest value, and for all other metrics, the first place is the biggest value.

[Table 2 about here.]

In general, MLP2 obtained the best performance result, with the best MCC and κ and second-best F_{1M} . ENS2 and ELM tied for second, despite ELM’s loss being the second-worst of all models. In the sequence, MLP1, with a difference of only one point, being this and MLP2 the components of ENS2. The fact that a singular model was

able to return better results is relevant, even though the ensemble methods are the union of the best estimators.

Next, a more in-depth analysis of the response of each model is made. An example of melody and its model-generated harmony, as well as the original harmony, is shown in Fig. 3 for the first eight measures of the song *America* by Stephen Sondheim and Leonard Bernstein, composed for the musical *West Side Story* from 1957. The song has a rhythmic construction of *hemiola*³, a repetitive melody, and simple harmony Miller (2006). An interesting point is between measures 5 and 7, where there is a rapid modulation⁴Roig-Francolí (2010) passing through the key of C minor, making this an appropriate example to perceive the ability of generalization for the chosen models. For the first four measures, all models responded with the same chords as the original harmony, probably because the melody notes are very indicative of the appropriate chord. For example, in measures 2 and 4, the notes together are the same that make up the F maj (F, A, and C) and G maj (G, B, and D) chords, respectively. The same goes for the last measure analyzed, where the notes are the same as the C maj chord (C, E, and G).

[Figure 3 about here.]

It is different when observing the interval from 5 to 7: RBF made the worst choices, proving to be unable to identify the modulation; ESN similarly, except for a possibly more appropriate choice for measure 5, that has the A♯ note present; MLPs, ELM and all ensembles made the same choices for measures 6 and 7, equal to the original, but it is clear that there was difficulty in finding a chord for measure 5. One reason for this could be that the original harmony uses a chord that is not considered, C min7 and only two notes are being played in this measure, A♯ and D♯. The most suitable choice would be D♯ maj, since the bar notes are its root and fifth, respectively, and it was chosen by MLP1, ELM, and all ensembles. The choice made by MLP2, although not the best compared to the previous ones, can be considered functional, with the ability to bring a harmonic novelty and a possible alternative.

The previous analysis allows us to infer differences between the performance and interpretation of each model. In addition, based on the data presented, the two-layer MLP model is chosen as the most appropriate, with its best execution reaching 13.89% F_{1M} , 33.08% MCC, and 32.01% κ . This κ value allows us to state that this is a reasonable result Artstein and Poesio (2008). An efficient way to analyze the response of the selected model is to look at its confusion matrix, which can be seen in Fig. 4.

[Figure 4 about here.]

The formation of a diagonal line in the matrix is slightly noticeable, a desirable characteristic since it means correct class assignments. This characteristic is most noticeable at specific points, as in the chords of C maj, F maj, and G maj, which are the main tonic functions. In addition, some columns of predicted classes stand out: C maj, F maj, and G maj. These columns lead us to understand that the model was able to comprehend the role that these chords play in musical harmony and can generalize the results, using them in a simplified way.

³Alternating rhythm between binary and tertiary metrics.

⁴Provisional character change of tone.

5. Conclusion

Different neural and ensemble models were proposed, tested, and evaluated for the task of automatic musical harmonization based on melodies.

The final harmonizing system was the MLP model with two hidden layers and 128 neurons, a model widely used in literature. It proved to be a reasonable nonlinear mapper in this field as well. The accuracy of the best model among 30 rounds reached 48.65%, a considerable result considering the results found in the literature for more complex models, and compared to a random guess. Keep in mind that 24 chord classes with uniform distribution would have a 4.17% chance of selection. The results generally show that the system has a simplifying capacity since harmony rules were not disregarded for the results generated.

For future work, we intend to explore models that can achieve higher classification accuracy values and use other means of quantifying the results to consider more rules of music theory, such as considering harmonic and relative functions. Also, carry out a study on other ways of representing the input data since simplification in a sum vector sacrifices the order in which the notes are played and their duration, potentially compromising the results.

Acknowledgment

The authors thank the Brazilian agencies Coordination for the Improvement of Higher Education Personnel (CAPES) - Financing Code 001, Brazilian National Council for Scientific and Technological Development (CNPq), processes number 40558/2018-5, 315298/2020-0, and Araucaria Foundation, process number 51497, and Federal University of Technology - Parana (UTFPR) for their financial support.

References

- Artstein, R., & Poesio, M. (2008, dec). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4), 555–596. Retrieved from <https://direct.mit.edu/coli/article/34/4/555-596/1999>
- Belotti, J., Siqueira, H., Araujo, L., Stevan, S. L., de Mattos Neto, P. S. G., Marinho, M. H. N., ... Sarubbo, L. A. (2020). Neural-based ensembles and unorganized machines to predict streamflow series from hydroelectric plants. *Energies*, 13(18), 4769.
- Branco, P., Torgo, L., & Ribeiro, R. (2015, may). A Survey of Predictive Modelling under Imbalanced Distributions. Retrieved from <http://arxiv.org/abs/1505.01658>
- Chuan, C. (2011). A comparison of statistical and rule-based models for style-specific harmonization. In *International society for music information retrieval conference* (pp. 221–226).
- Cohen, J. (1960, apr). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1), 37–46. Retrieved from <http://journals.sagepub.com/doi/10.1177/001316446002000104>
- Costa, L. F. P., Ogoshi, A. Y. C., Martins, M. S. R., & Siqueira, H. V. (2022). Developing a Measure Image and Applying to Deep Learning. In *Music encoding conference*. Halifax, CA: Music Encoding Initiative.
- Cramer, H. (1962). *Mathematical Methods Of Statistics* (1st ed.). Bombay: Asia Publishing House.
- de Souza Tadano, Y., Siqueira, H., & Alves, T. (2016). Unorganized machines to predict hospital admissions for respiratory diseases. In *Latin american conference on computational intelligence* (pp. 1–6).

- Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2017). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. In *Aaai conference on artificial intelligence* (pp. 34–41).
- Ebcioğlu, K. (1988). An Expert Four-part Harmonizing Chorales. *Computer Music Journal*, 12(3), 43–51.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200), 675–701.
- Gómez, R. (2018). *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. Retrieved 2021-11-02, from https://gombru.github.io/2018/05/23/cross_entropy_loss/
- Haykin, S. (2008). *Neural Networks and Learning Machines* (3rd ed.). Hamilton, ON: Prentice Hall.
- Huang, A., & Wu, R. (2016). Deep learning for music. *Deep Learning for Natural Language Processing*, abs/1606.04930.
- Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3), 489–501.
- Jaeger, H. (2010). *The “echo state” approach to analysing and training recurrent neural networks – with an Erratum note* (Tech. Rep. No. 148). GMD - German National Research Institute for Computer Science.
- Kachba, Y., Chiroli, D., Belotti, J., Alves, T., de Souza Tadano, Y., & Siqueira, H. (2020). Artificial neural networks to estimate the influence of vehicular emission variables on morbidity and mortality in the largest metropolis in south america. *Sustainability*, 12(7), 2621.
- Koops, H., Pedro, M., & de Haas, W. (2013). A functional approach to automatic melody harmonisation. In *Acm sigplan international conference on functional programming* (p. 47–58).
- Lim, H., & Lee, K. (2017). Chord Generation from Symbolic Melody Using BLSTM Networks. In *International society for music information retrieval conference* (pp. 621–627).
- Lim, H., Rhyu, S., & Lee, K. (2017). *CSV Leadsheet Database*. Retrieved 2019-06-08, from http://marg.snu.ac.kr/chord_generation/
- Liu, H.-M., & Yang, Y.-H. (2018). Lead Sheet Generation and Arrangement by Conditional Generative Adversarial Network. In *International conference on machine learning and applications* (pp. 722–727). IEEE.
- Miller, N. (2006). *Heritage and Innovation of Harmony: A Study of West Side Story* (Unpublished doctoral dissertation). University of North Texas.
- Nakashima, S., Imamura, Y., Ogawa, S., & Fukumoto, M. (2010). Generation of appropriate user chord development based on interactive genetic algorithm. In *International conference on p2p, parallel, grid, cloud and internet computing* (p. 450–453).
- Piotrowski Paweł and Baczyński, D., Kopyt, M., & Gulczyński, T. (2022). Advanced Ensemble Methods Using Machine Learning and Deep Learning for One-Day-Ahead Forecasts of Electric Energy Production in Wind Farms. *Energies*, 15(4), 1252.
- Powers, D. M. W. (2007). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation* (Tech. Rep.). Adelaide: Flinders University. Retrieved from <http://arxiv.org/abs/2010.16061>
- Ribeiro, V., Reynoso-Meza, G., & Siqueira, H. (2020). Multi-objective ensembles of echo state networks and extreme learning machines for streamflow series forecasting. *Engineering Applications of Artificial Intelligence*, 95, 103910.
- Roig-Francolí, M. (2010). *Harmony in Context* (2nd ed.). Cincinnati, OH: McGraw-Hill.
- Schoenberg, A. (1978). *Theory of Harmony* (1st ed.). Berkeley, CA: University of California Press.
- Siqueira, H., Boccato, L., Attux, R., & Lyra Filho, C. (2012a). Echo state networks for seasonal streamflow series forecasting. In *International conference on intelligent data engineering and automated learning* (pp. 226–236).
- Siqueira, H., Boccato, L., Attux, R., & Lyra Filho, C. (2012b). Echo state networks in seasonal streamflow series prediction. *Learning and Nonlinear Models*, 10, 181–191.

- Siqueira, H., & Luna, I. (2019). Performance comparison of feedforward neural networks applied to streamflow series forecasting. *Mathematics In Engineering Science And Aerospace*, 10(1), 41–53.
- Terefenko, D. (2014). *Jazz Theory: From Basic to Advanced Study* (1st ed.). Rochester, NY: Routledge.
- Wichard, J. D., & Ogorzalek, M. (2004). Time series prediction with ensemble models. In *International joint conference on neural networks* (Vol. 2, pp. 1625–1630). IEEE.
- Wiggins, G., Papadopoulos, G., & Phon-Amnuaisuk, S. (1998). Evolutionary methods for musical composition. *International Journal of Computing Anticipatory Systems*.

Table 1. Average loss and accuracy, F_{1M} , MCC and κ percentage for each model tested. Neurons are shown in parentheses.

Experiment	Loss	ACC (%)	F1 (%)	MCC (%)	κ (%)
MLP1 (64)	1.7130	47.86	14.87	33.22	31.84
MLP2 (128)	1.7352	47.21	14.89	33.57	32.50
RBF (64)	1.7630	46.00	9.53	31.42	30.04
ELM (256)	2.0513	47.19	16.67	33.40	32.08
ESN (256)	2.0787	44.73	9.24	29.43	27.42
ENS5	1.7770	47.79	13.74	33.04	31.63
ENS3	1.7103	47.77	13.96	33.10	31.80
ENS2	1.7064	47.79	14.43	33.18	31.89

Table 2. Overall ranking using Borda count method considering different metrics and algorithm results.

Experiment	Loss	F1	MCC	κ	Total
MLP2	5	7	8	8	28
ENS2	8	5	5	6	24
ELM	2	8	7	7	24
MLP1	6	6	6	5	23
ENS3	7	4	4	4	19
ENS5	3	3	3	3	12
RBF	4	2	2	2	10
ESN	1	1	1	1	4

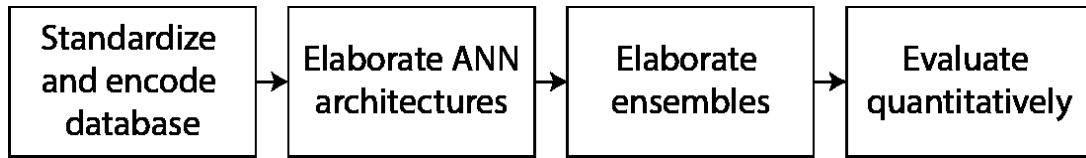


Figure 1. Block diagram showing the proposed methodology.

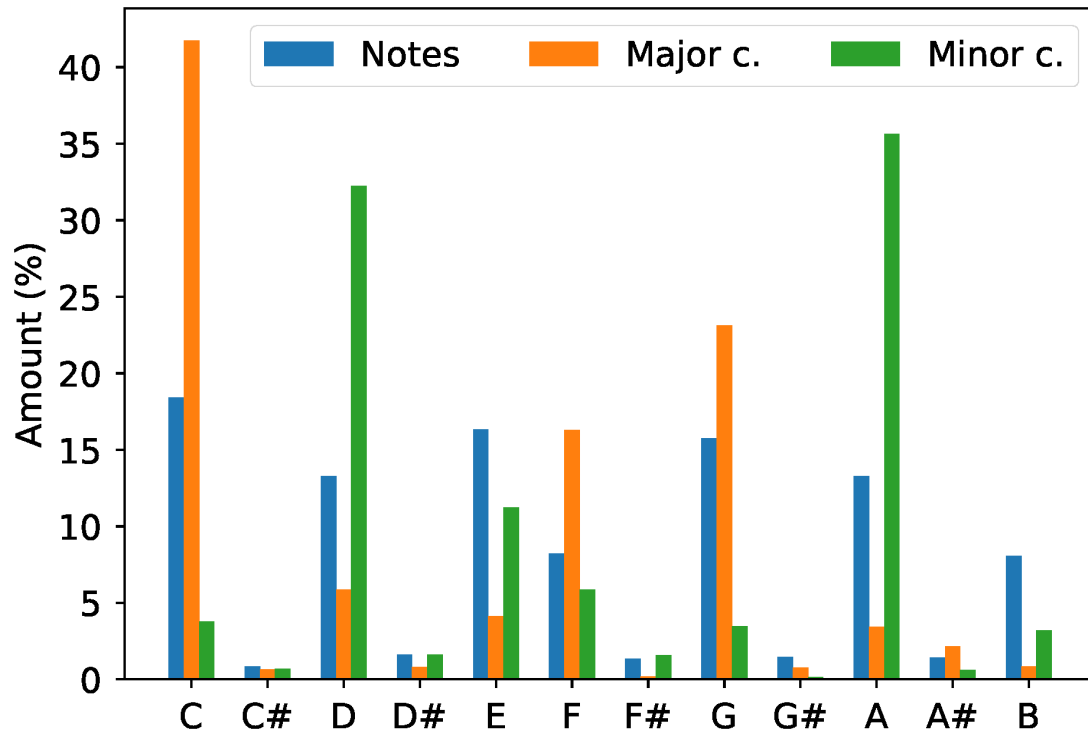


Figure 2. Percentage of notes, major and minor chords present in 20% of the database.



(a)

Original: C maj | F maj | C maj | G maj | C min7 | A# maj | G# maj | C maj ||
MLP1: C maj | F maj | C maj | G maj | D# maj | A# maj | G# maj | C maj ||
MLP2: C maj | F maj | C maj | G maj | G# maj | A# maj | G# maj | C maj ||
RBF: C maj | F maj | C maj | G maj | C maj | G maj | E maj | C maj ||
ESN: C maj | F maj | C maj | G maj | A# maj | G maj | E maj | C maj ||
ELM: C maj | F maj | C maj | G maj | D# maj | A# maj | G# maj | C maj ||
ENS5: C maj | F maj | C maj | G maj | D# maj | A# maj | G# maj | C maj ||
ENS3: C maj | F maj | C maj | G maj | D# maj | A# maj | G# maj | C maj ||
ENS2: C maj | F maj | C maj | G maj | D# maj | A# maj | G# maj | C maj ||

(b)

Figure 3. Melody of the song *America* represented with (a) score notation and (b) original and generated harmony by each evaluated model.

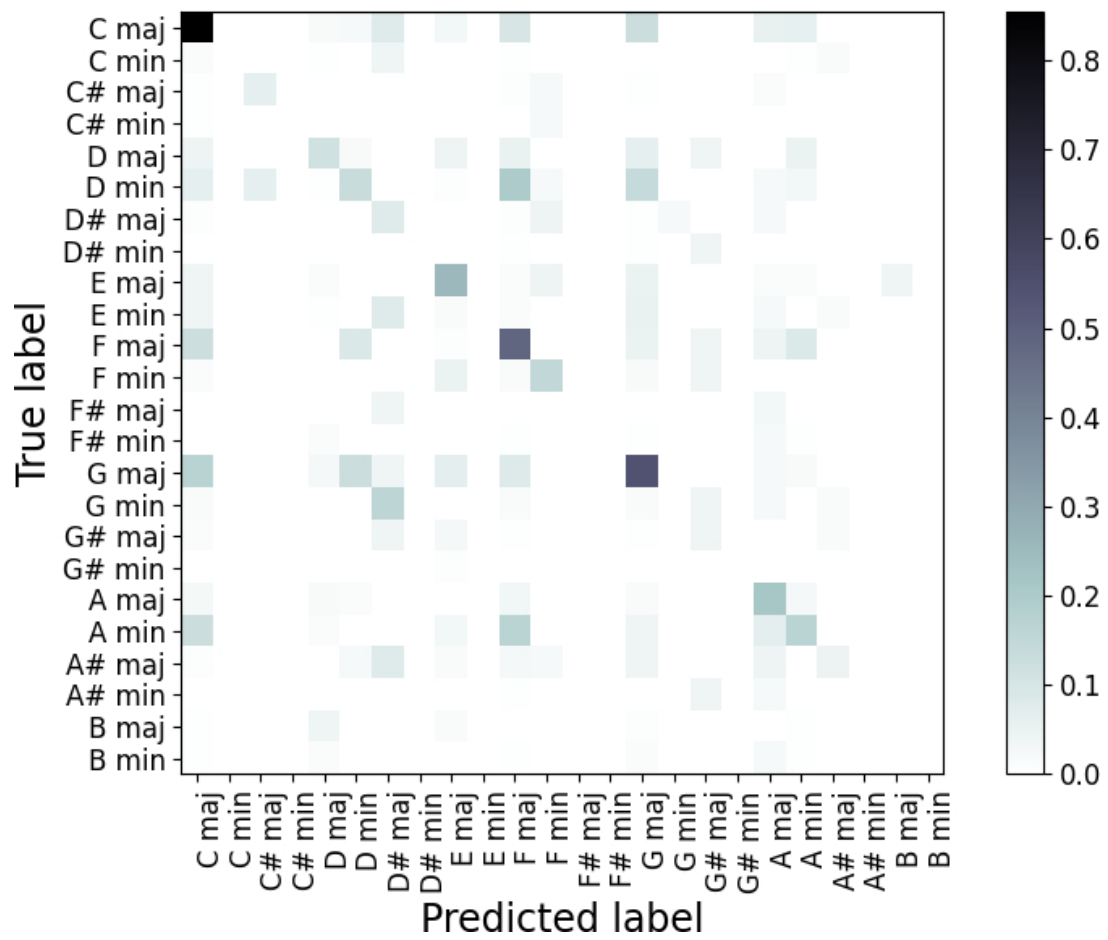


Figure 4. Normalized confusion matrix for classifying chords using the MLP2 model.

Submission received for Applied Artificial Intelligence (Submission ID: 229086030)

1 mensagem

UAAI-peerreview@journals.tandf.co.uk <UAAI-peerreview@journals.tandf.co.uk>
Para: luccos@alunos.utfpr.edu.br

26 de outubro de 2022 21:54

**Taylor & Francis**
Taylor & Francis Group

Dear Lucas Francesco Piccioni Costa,

Thank you for your submission.

Submission ID	229086030
Manuscript Title	Neural Networks And Ensemble Based Architectures To Automatic Musical Harmonization: A Performance Comparison
Journal	Applied Artificial Intelligence
Article Publishing Charge (APC)	USD \$1860.00 (plus VAT or other local taxes where applicable in your country)

**APC only payable if your article is accepted*

You can check the progress of your submission, and make any requested revisions, on the Author Portal.

Thank you for submitting your work to our journal.

If you have any queries, please get in touch with UAAI-peerreview@journals.tandf.co.uk.

For any queries relating to your APC, please get in touch with APC@tandf.co.uk

Kind Regards,
Applied Artificial Intelligence Editorial Office

Taylor & Francis is a trading name of Informa UK Limited, registered in England under no. 1072954.
Registered office: [5 Howick Place, London, SW1P 1W](#).

**APÊNDICE B – Artigo Submetido ao Congresso *Music Encoding*
*Conference***

Um artigo com os resultados expostos na Seção 8.3 foi submetido para publicação, aceito e devidamente apresentado no congresso internacional *Music Encoding Conference*. Nas próximas páginas, o artigo na íntegra e o pôster (Figura 39) podem ser conferidos.

Developing a Measure Image and Applying to Deep Learning

Lucas Francesco Piccioni Costa
Universidade Tecnológica
Federal do Paraná
Brazil
luccos@alunos.utfpr.edu.br

André Yoshio Caram Ogoshi
Universidade Estadual do
Centro-Oeste
Brazil
a.yoshio@hotmail.com

Marcella Scoczynski Ribeiro
Martins
Universidade Tecnológica
Federal do Paraná
Brazil
marcella@utfpr.edu.br

Hugo Valadares Siqueira
Universidade Tecnológica
Federal do Paraná
Brazil
hugosiqueira@utfpr.edu.br

Abstract

The use of intelligent systems linked to musical tasks such as automatic composition, classification, and Music Information Retrieval has increasingly shown itself to be a promising field of study, not only from a computational point of view but also musical. This paper aimed to develop an innovative method capable of producing a coded image that contains all the information of a musical measure, generating a structure that can use in several computational applications involving machine learning, especially deep learning and convolutional neural networks (CNNs). To illustrate the usefulness of this method, the measured image is applied to a CNN to solve the problem of automatic musical harmonization. The results of this brief application are better than those known in the literature, proving the method's efficiency.

Introduction

The art of music undergoes constant changes, sometimes appropriating science to achieve its transformations. Music and computing can be considered indivisible through the generation of new sounds and even acting in composition ([Webster, 2002](#)).

But this does not mean that computers can understand music: the human element is fundamental in this process. Specifically, when the subject is automatic musical computation, it is necessary to encode musical information, often wholly present in a musical score. It is natural that in the encoding process, some info is lost, which can negatively affect the performance of automated systems.

The most common ways to perform this encoding involve numeric vectors. That is clear in the papers of [Franklin \(2006\)](#), [Laden & Keefe \(1989\)](#), [Mozer \(1994\)](#), and [Todd \(1989\)](#). Other authors use coding approaches with image representations, such as [Velarde et al. \(2016\)](#) and [Modrzejewski et al. \(2019\)](#).

This paper aims to present a standardized way to elaborate musical visual representations focusing on intelligent systems applications. The analysis space is limited to one musical measure at a time. First, the measure has its rhythmic information standardized. Then, an image is built capable of containing all melodic and rhythmic information of that measure, regardless of the time signature or tempo. To illustrate the method's usefulness, we applied the Measure Image (MI) in the automatic harmonization task. For a melodic input, defining the best chord that harmonizes it is necessary. We can notice better results than those found in the literature, using a CNN with simple architecture.

1 Standardization of Rhythmic Information

The length of a musical measure is governed by a time signature, which indicates the amount and type of rhythmic figure that fills it. There are several possible time signatures, which causes a problem, as this way, songs with different time signatures will have different lengths, highlighting the need for standardization.

By default, the duration s of the semibreve is equal to 1, and the other durations are fractions of it. The fraction is directly related to the denominator number of the time signature TS . The numerator, in turn, indicates the total of that figure type that will complete a measure.

Knowing this, one can define the durations vector of the k rhythmic figures of a measure as in Equation 1:

$$\dot{\mathbf{d}} = \left[\dot{d}_1, \dot{d}_2, \dots, \dot{d}_k \right]^T \mid \dot{d}_i = \frac{s}{d_i}, d_i \geq 1, \sum_{i=1}^k \dot{d}_i = s \cdot TS. \quad (1)$$

To standardize measure durations independent of TS , we want to define the normalized durations vector. To do so, just perform the scalar multiplication of TS^{-1} by $\dot{\mathbf{d}}$ forming the vector $\mathbf{d}^{(N)}$, that is, $TS^{-1} \cdot \dot{\mathbf{d}} = \mathbf{d}^{(N)}$, which has properties according to Equation 2:

$$\mathbf{d}^{(N)} = \left[d_1^{(N)}, d_2^{(N)}, \dots, d_k^{(N)} \right]^T \mid d_i^{(N)} = \frac{s}{d_i^{(N)}}, d_i^{(N)} = d_i \cdot TS, \sum_{i=1}^k d_i^{(N)} = s. \quad (2)$$

In possession of this information, it is possible to perform rhythmic standardization. The next step is to encode this information into a Measure Image (MI).

2 Measure Image Construction

A musical measure contains a lot of information, especially the duration and pitch of each note when observing the melodic context. So that none of this information is lost, we proposed building an image capable of containing it.

A three-dimensional matrix $x \times y \times z$ is considered per image, where x and y are coordinates of discrete and finite values called pixels, and dimension z is a composition of RGBA dimensions ([Adler et al., 2003](#); [Gonzalez & Woods, 2010](#)).

The visual piano roll approach is applied to compose an image with melodic information, a form of representation that uses horizontal lines to define the duration of notes, and each line represents a pitch ([FL Studio, 2021](#)). Figure 1 demonstrates the construction and meaning of each component in the Measure Image (MI).

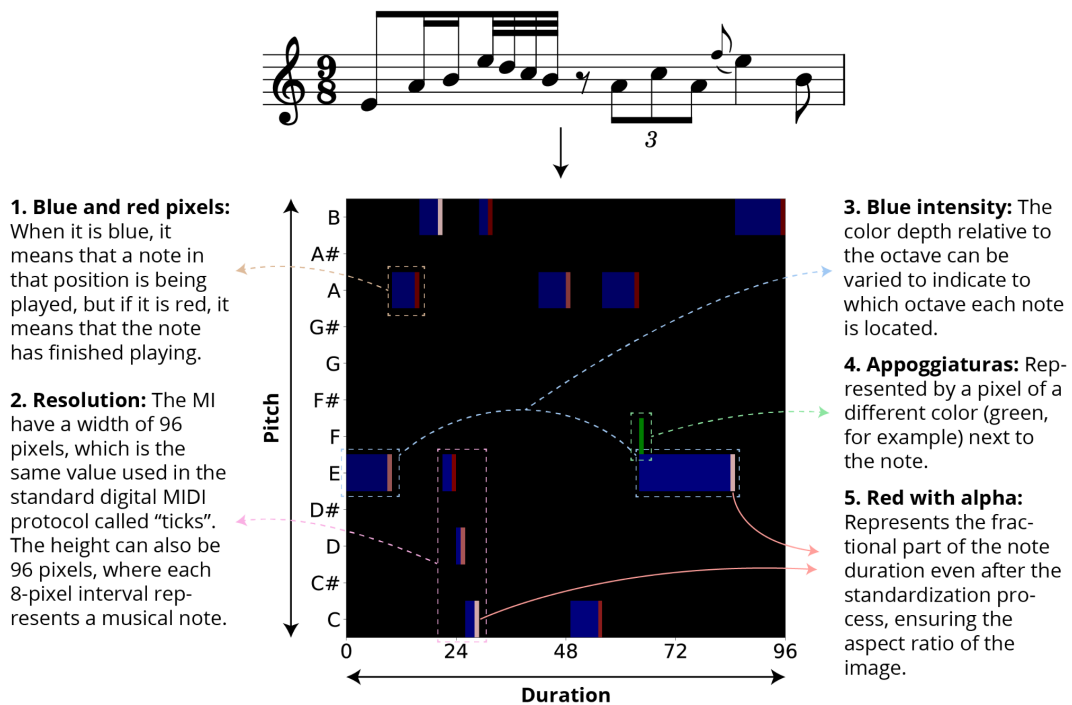


Figure 1: MI originated from a measure with all significant parts explained: (1) how to represent a note duration and pitch, (2) the resolution and size of the image, (3) how to represent different octaves, and (4) appoggiaturas, and (5) the use of the alpha channel to store fractional duration information.

3 Applying the Measure Image to Deep Learning

Aiming at applying MI as an input to an automatic harmonization system, we wanted to test a CNN architecture to obtain results using the *CSV Leadsheet Database* (Lim et al., 2017). The database was first standardized in terms of melody (all songs were transposed to the key of C major), harmony (only major and minor triads were considered), and rhythm (applying Equation 2 to the duration of each note). A simple model based on AlexNet (Krizhevsky et al., 2012) was used, as illustrated in Figure 2. applied.

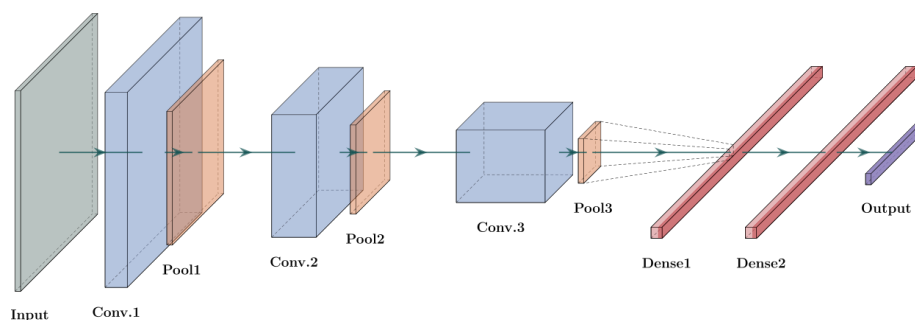


Figure 2: Illustrative structure of the CNN architecture applied during the experiment.

In this one, three layers of convolution (Conv.) and subsampling (Pool) applying the max function, i. e., max-pooling, are interspersed with an increasing number of filters (32, 64, and 128, respectively). In the sequence, three fully connected layers (Dense) at the end, where Dense1 and Dense2 have 128 neurons each and 24 neurons at the Output. Each

convolution and fully connected layer uses the ELU activation function, except the output layer that performs softmax activation. Dropout and batch normalization techniques were applied.

Only 30% of the database was used, being divided 60% for training, 20% validation, and 20% testing. The model was run 30 times to be cross-validated, resulting in average and best accuracy (ACC) of 50.88% and 52.34, respectively, and average and best Cohen’s Kappa (κ) of 38.31% and 40.37%, respectively.

Based on the subjective levels of κ , the average result can be defined as reasonable, with the best value being just above the lower limit to be considered a moderate or still adequate result ([Artstein & Poesio, 2008](#)).

The ACC values allow us to conclude an improvement compared to the results in the literature, especially when compared to the work of [Lim & Lee \(2017\)](#), and against a random guess, being equal to 4.17% for the 24 chord classes considered. Analyzing the resulting normalized confusion matrix illustrated in Figure 3 can better explain these statements.

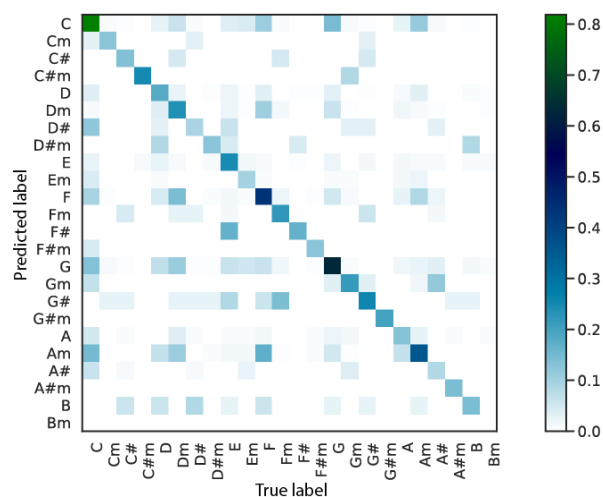


Figure 3: Confusion matrix of the CNN model generated with the results of its best execution.

It is easy to notice the present diagonal line, indicative of the good classification performance of the system. As a similarity, there is a greater density of correct classification for the chords of C major, F major, and G major.

The success of this simple model on classical techniques and the literature demonstrates how promising the possible results are to be obtained with the application of MI using CNNs.

Conclusion

A new way of representing and encoding musical measures was developed with its application in intelligent systems in mind. It was put to the test by serving as the input to an automatic harmonizing system that obtained reasonable results considering its simple construction and comparing its results with existing and random ones.

In the future, we intend to explore more possible applications for MI, such as classification and automatic composition tasks, in addition to the development of a generic application that can be used in development environments for quick and easy use.

Acknowledgments

The authors thank the Brazilian agencies Coordination for the Improvement of Higher Education Personnel (CAPES) - Financing Code 001, Brazilian National Council for Scientific and Technological Development (CNPq), processes number 40558/2018-5, 315298/2020-0, and Araucaria Foundation, process number 51497, and Federal University of Technology - Parana (UTFPR) for their financial support.

References

- Adler, M., Boutell, T., Bowler, J., Brunschen, C., Costello, A. M., Crocker, L. D., Dilger, A., Fromme, O., Gailly, J., Herborth, C., Jakulin, A., Kettler, N., Lane, T., Lehmann, A., Lilley, C., Martindale, D., Mortensen, O., Pickens, K. S., Poole, R. P., ... Wohl, J. (2003). *Portable Network Graphics (PNG) Specification*. <https://www.w3.org/TR/2003/REC-PNG-20031110/>
- Artstein, R., & Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4), 555–596. <https://doi.org/10.1162/coli.07-034-R2>
- FL Studio. (2021). *Piano roll*. <https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/pianoroll.htm>
- Franklin, J. A. (2006). Recurrent Neural Networks for Music Computation. *INFORMS Journal on Computing*, 18(3), 321–338. <https://doi.org/10.1287/ijoc.1050.0131>
- Gonzalez, R. C., & Woods, R. C. (2010). *Processamento digital de imagens* (3rd ed.). Pearson Prentice Hall.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- Laden, B., & Keefe, D. H. (1989). The Representation of Pitch in a Neural Net Model of Chord Classification. *Computer Music Journal*, 13(4), 12. <https://doi.org/10.2307/3679550>
- Lim, H., & Lee, K. (2017). Chord Generation from Symbolic Melody Using BLSTM Networks. *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 621–627. <https://doi.org/10.5281/zenodo.1417327>
- Lim, H., Rhyu, S., & Lee, K. (2017). *CSV Leadsheet Database*. Music and Audio Research Group. http://marg.snu.ac.kr/chord_generation/
- Modrzejewski, M., Dorobek, M., & Rokita, P. (2019). Application of Deep Neural Networks to Music Composition Based on MIDI Datasets and Graphical Representation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11508 LNAI, 143–152. https://doi.org/10.1007/978-3-030-20912-4_14
- Mozer, M. C. (1994). Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing. *Connection Science*, 6(2–3), 247–280. <https://doi.org/10.1080/09540099408915726>
- Todd, P. M. (1989). A Connectionist Approach to Algorithmic Composition. *Computer Music Journal*, 13(4), 27. <https://doi.org/10.2307/3679551>

- Velarde, G., Weyde, T., Chacón, C. C., Meredith, D., & Grachten, M. (2016). Composer recognition based on 2D-filtered piano-rolls. *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, 3, 115–121.
- Webster, P. (2002). Historical Perspectives on Technology and Music. *Music Educators Journal*, 89(1), 38–43. <https://doi.org/10.2307/3399883>

Figura 39 – Pôster apresentado durante congresso internacional *Music Encoding Conference*.



DEVELOPING A MEASURE IMAGE AND APPLYING TO DEEP LEARNING

Lucas Costa
 Universidade Tecnológica
 Federal do Paraná
 luccos@alunos.utfpr.edu.br

André Ogoshi
 Universidade Estadual do
 Centro-Oeste
 a.yoshio@hotmail.com

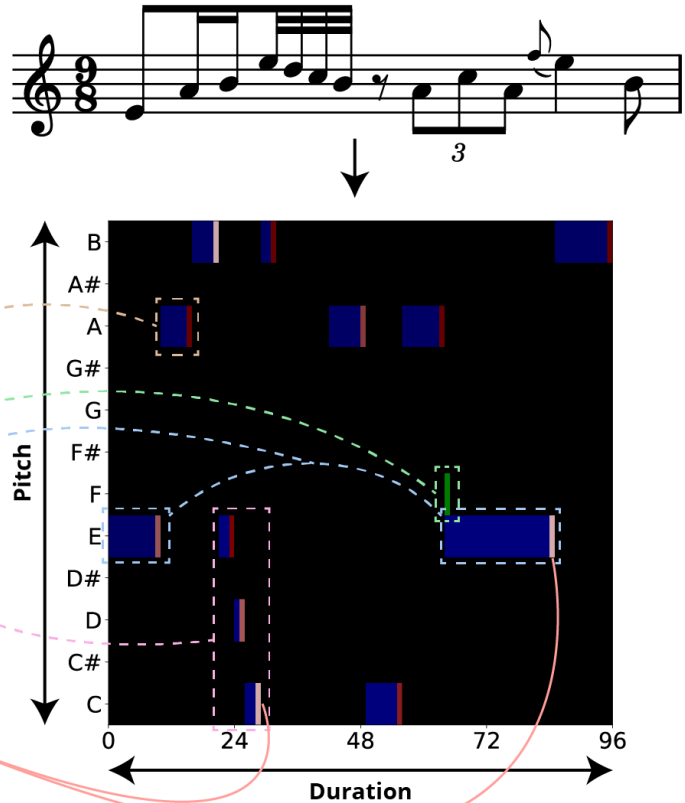
Marcella Martins
 Universidade Tecnológica
 Federal do Paraná
 marcella@utfpr.edu.br

Hugo Siqueira
 Universidade Tecnológica
 Federal do Paraná
 hugosiqueira@utfpr.edu.br

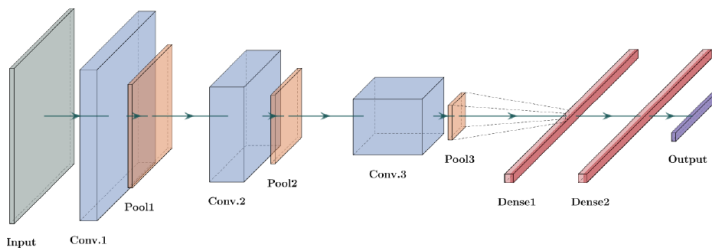
Abstract

This paper aimed to develop an innovative method capable of producing a coded image that contains all the information of a musical measure, generating a structure that can use in several computational applications involving machine learning, especially deep learning and convolutional neural networks (CNNs). To illustrate the usefulness of this method, the measured image is applied to a CNN to solve the problem of automatic musical harmonization. The results of this brief application are better than those known in the literature, proving the method's efficiency.

- 1. Blue and red pixels:** When it is blue, it means that a note in that position is being played, but if it is red, it means that the note has finished playing.
- 2. Appoggiaturas:** Represented by a pixel of a different color (green, for example) next to the note.
- 3. Blue intensity:** The color depth relative to the octave can be varied to indicate to which octave each note is located.
- 4. Resolution:** The MI have a width of 96 pixels, which is the same value used in the standard digital MIDI protocol called "ticks". The height can also be 96 pixels, where each 8-pixel interval represents a musical note.
- 5. Red with alpha:** Represents the fractional part of the note duration even after the standardization process, ensuring the aspect ratio of the image.



Deep Learning Application



- Input dimensions: 96 × 96 × 4;
- 3 × 3 convolutions and 2 × 2 maxpooling;
- ELU activation function and softmax on output;
- Dropout set to 30% and batch normalization;
- 30% of the database divided into 60% for training, 20% for validation and 20% for testing;
- Maximum number of training epochs: 300;
- Optimized with Adam.

Metric	Result
ACC (%) – average	50.88%
ACC (%) – best	52.34%
Kappa (κ) – average	38.31%
Kappa (κ) – best	40.37%

Conclusion

A new way of representing and encoding musical measures was developed with its application in intelligent systems in mind. It was put to the test by serving as the input to an automatic harmonizing system that obtained reasonable results considering its simple construction and comparing its results with existing and random ones.

