Universidade Federal
do Rio de Janeiro
Escola Politécnica

RECUPERAÇÃO DE INFORMAÇÕES MUSICAIS: UMA ABORDAGEM UTILIZANDO DEEP LEARNING

Heitor Rodrigues Guimarães

Projeto de Graduação apresentado ao Curso de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Orientador: Ricardo Guerra Marroquim

Rio de Janeiro
Setembro de 2018

RECUPERAÇÃO DE INFORMAÇÕES MUSICAIS: UMA ABORDAGEM UTILIZANDO DEEP LEARNING

Heitor Rodrigues Guimarães

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

Prof. Ricardo Guerra Marroquim, Dr.

Prof. Fernando Gil Vianna Resende Junior, Dr.

Prof. André de Almeida Maximo, Dr.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Guimarães, Heitor Rodrigues

Recuperação de informações musicais: Uma abordagem utilizando Deep Learning/Heitor Rodrigues Guimarães. – Rio de Janeiro: UFRJ/POLI – COPPE, 2018.

XII, 53 p.: il.; 29, 7cm.

Orientador: Ricardo Guerra Marroquim

Projeto (graduação) – UFRJ/ Escola Politécnica/ Curso de Computação e Informação, 2018.

Referências Bibliográficas: p. 51 – 53.

1. Aprendizado Profundo. 2. CNN. 3. MIR. 4. Classificação de Gêneros Musicais. I. Marroquim, Ricardo Guerra. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Computação e Informação. III. Título.

"The only truth is music - the only meaning is without meaning -
Music blends with the heartbeat universe and we forget the brain beat."
— Jack Kerouac, *Desolation Angels*

Agradecimentos

À minha família, que sempre esteve ao meu lado me apoiando e fazendo duros sacrifícios para que essa jornada acadêmica fosse possível.

Agradeço também ao Caíque Gomes e Thais Maciel, amigos de longa data, pelos momentos de risadas e todo o incentivo para seguir em frente com este trabalho.

À Jéssica Guimarães que, apesar da distância física, sempre esteve ao meu lado e me apoiando durante todo este tempo.

Agradeço à todas as pessoas com quem tive o prazer de trabalhar no CBPF e na GE, em especial ao Professor Márcio Portes de Albuquerque e meu supervisor Bruno Astuto, por me orientarem acadêmica e pessoalmente e me mostrarem a importância da ciência aplicada.

Ao Professor Marroquim e o Pedro Savarese, meus orientadores, por todo o apoio e motivação neste trabalho. Muito obrigado por terem confiado em mim durante o desenvolvimento deste trabalho, por todo o interesse e gosto pelo tema.

Agradeço à todos os professores da UFRJ que lecionam suas disciplinas com uma paixão indescritível, sem vocês essa trajetória não seria possível. Obrigado em especial aos professores Guilherme Travassos, Fernando Gil Vianna, Daniel Ratton e Sérgio Villas-Boas (*in-memoriam*).

Por fim, mas não menos importante, agradeço ao povo brasileiro por me possibilitar a formação em uma excelente instituição de ensino de forma gratuita. Espero poder ajudar e retribuir por esta oportunidade com os conhecimentos que aqui adquiri.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

RECUPERAÇÃO DE INFORMAÇÕES MUSICAIS: UMA ABORDAGEM UTILIZANDO DEEP LEARNING

Heitor Rodrigues Guimarães

Setembro/2018

Orientador: Ricardo Guerra Marroquim

Curso: Engenharia de Computação e Informação

A distribuição em larga escala de músicas em formato digital representou um marco para a indústria do entretenimento e o relacionamento com seus consumidores no século XXI. Para atender tais demandas houve um amplo investimento em Pesquisa e Desenvolvimento (P&D) na área de processamento de sinais e de *Recuperação de Informações Musicais*. Técnicas de recomendação baseadas em conteúdo são cada vez mais importantes para a sugestão de novos conteúdos e uma melhor experiência de seus clientes. Saber categorizar o áudio por suas propriedades é de fundamental importância para sugerir e agrupar músicas. Mesmo sendo possível a caracterização de uma música por propriedades estatísticas como instrumentação e estrutra rítmica, a *classificação de gêneros musicais* é uma tarefa difícil e de caráter subjetivo. O objetivo deste trabalho é propor e implementar uma metodologia para a classificação de gêneros musicais utilizando *Redes Neurais Convolucionais* em diferentes conjuntos de áudios, com o mínimo de pré-processamento possível, e comparar aspectos dessa abordagem com as técnicas clássicas de Machine Learning, onde precisamos projetar cuidadosamente cada feature do áudio.

Palavras-Chave: Aprendizado Profundo, CNN, MIR, Classificação de Gêneros Musicais.

Abstract of the Undergraduate Project presented to Poli/COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

MUSIC INFORMATION RETRIEVAL: A DEEP LEARNING APPROACH

Heitor Rodrigues Guimarães

September/2018

Advisor: Ricardo Guerra Marroquim

Course: Computer and Information Engineering

Large scale music distribution in digital format is a milestone for the entertainment industry and the relation with the customers in the 21st century. The straight consequences of this high demand are the large investments in Research and Development (R&D), specially for Signal Processing and *Music Information Retrieval*. Content-based recommendation are becoming more important to suggest and group personalized content to users for a better experience. Even being possible to categorize music using statistical properties such as rhythmic structure, pitch and instrumentation, *music genre classification* is hard and subjective problem. The goal of this work is to propose and implement a methodology to classify music genres using *Convolutional Neural Networks* in datasets of audio files with less pre-processing as possible, and compare with classical Machine Learning algorithms applied on hand crafted features.

Keywords: Deep Learning, CNN, MIR, Music Genre Classification.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contexto e Motivação

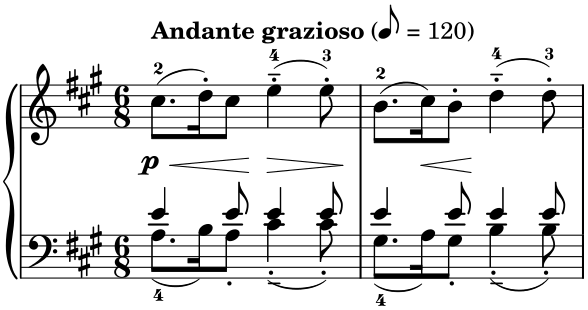
A indústria musical no início do século XXI presenciou uma drástica mudança na forma de consumo e distribuição de seu principal produto, a música. O modelo de venda de mídias físicas como CD, Vinil e Fita Cassete foi rapidamente substituído por mídias digitais com o aumento da popularidade dos computadores pessoais. Empresas como o Napster que utilizavam arquiteturas *peer-to-peer* (P2P) para a distribuição de arquivos de forma descentralizada se tornaram muito populares e por muitos anos acreditou-se que a indústria musical estava fadada à sua falência devido à pirataria.

De fato essa mudança no paradigma de distribuição levou a indústria a perder bilhões de dólares nessa transição, porém hoje o setor está reencontrando o seu espaço através de serviços de Streaming, por meio de plataformas como o Spotify, Apple Music e Tidal [5]. Essa distribuição de conteúdo digital em larga escala nos permite realizar análises sobre o comportamento de usuários e suas conexões, melhorando assim processos como sugestão de novos conteúdos. E uma área de pesquisa que tem ganho destaque nos últimos anos é a de *Recuperação de Informações Musicais* (MIR).

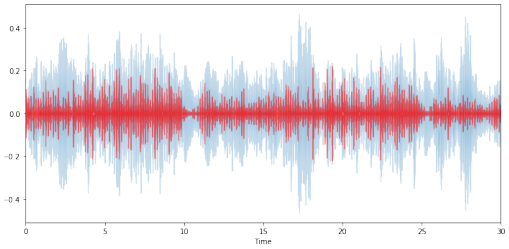
A MIR é uma área altamente interdisciplinar com o foco na extração de informações de músicas e suas aplicações, como por exemplo: Sistemas de recomendação, transcrição automática, reconhecimento de instrumentos e categorização de músicas [6]. Informações essas que são adquiridas ao utilizarmos algoritmos específicos em uma representação digital da música.

É importante ressaltar que existem muitas formas de se representar uma música. Usualmente correlacionamos o termo **música** com a representação através do sinal de áudio, mas não estamos restritos somente a esta: Temos as partituras no mundo físico, o protocolo MIDI (Musical Instrument Digital Interface) no mundo digital e

muitas outras. Neste trabalho nosso objetivo é analisar a música a partir de sua representação como sinal de áudio, um tipo de dado naturalmente desestruturado.



(a) Representação em Partitura



(b) Representação em Forma de onda (Harmônico e Percussão)

Figura 1.1: Duas representações de Sonata para Piano No. 11 in A major, K. 331

Analisar uma música a partir do sinal de áudio é o objetivo de diversas subáreas da MIR, entre elas podemos destacar a "*Recomendação baseada em conteúdo*" (Content-based Recommendation), onde o objetivo é permitir a classificação e agrupamento de músicas através de uma característica comum (e.g. gênero) com o objetivo de melhorar a experiência e sugestão de novos conteúdos para um usuário dado o seu histórico de uso [7]. Nos últimos anos a acurácia de sistemas de recomendação deste tipo tem alcançado níveis próximos ao *estado da arte* com a utilização de técnicas de **Machine Learning**, onde o desafio é permitir que o computador possa aprender a partir de experiências (ou exemplos) sobre o contexto do problema em termos de conceitos hierárquicos, de forma que cada conceito é definido por relações ou conceitos mais simples[3].

O principal algoritmo que estudaremos neste trabalho são as *Redes Neurais Artificiais* (RNA). As RNAs foram concebidas com o objetivo de modelar o comportamento do sistema nervoso biológico, mas desde a sua criação essa ideia divergiu bastante e hoje o principal objetivo é obter bons resultados nos problemas de Machine Learning [2]. Os detalhes técnicos do funcionamento das RNAs serão explicados nos próximos capítulos, por enquanto é importante saber que as redes neurais são **aproximadores universais de funções**. Seja x a entrada de uma rede neural e y a sua saída (e.g. categorias). Uma RNA define um mapeamento $y = f(x; w, b)$ e aprende os valores de w e b que resultam na melhor função aproximada possível [3].

A performance dos diversos algoritmos de Machine Learning dependem fortemente da representação dos dados fornecidos. Em muitas aplicações é possível resolver um problema com mais facilidade se escolhermos os atributos (**features**) corretos para uma dada tarefa. Porém a escolha de tais features pode não ser uma tarefa simples e depende dos conhecimentos específicos de um especialista na área de domínio do problema. Em contraste com essa alternativa, uma nova área de pesquisa

tem ganho popularidade nos últimos anos chamada de Aprendizado Profundo (Deep Learning), que nos permite usar técnicas com as quais podemos explorar os dados em sua forma mais bruta (desestruturada) e aprender progressivamente padrões a partir dessas representações para então realizar a classificação.

1.2 Objetivo

O objetivo deste trabalho é propor uma abordagem para classificar um conjunto de músicas (representados por sinais de áudio) de acordo com seus gêneros musicais (e.g Jazz, Rock e outros), utilizando Redes Neurais Convolucionais (CNN) e comparar com as abordagens tradicionais nas quais precisamos projetar e extrair features específicas do áudio antes de utilizar alguma técnica de aprendizado estatístico clássica.

1.3 Estruturação do texto

O presente documento está dividido em 6 capítulos, além deste introdutório, e foi organizado da seguinte forma:

Nos capítulos 2 e 3 encontram-se os fundamentos teóricos deste trabalho. No capítulo 2 faremos uma introdução sobre a área de Recuperação de Informações musicais. No capítulo 3 vamos nos aprofundar no aprendizado de máquina, em especial na técnica central deste trabalho, as Redes Neurais Artificiais.

No capítulo 4 discutimos sobre as abordagens e resultados de trabalhos clássicos e modernos na área. No capítulo 5 será apresentada a metodologia proposta para resolver o problema de classificação de gêneros musicais, mostrando em detalhes o fluxo de trabalho, informações sobre os conjuntos de dados utilizados e o processo de treinamento das redes neurais. No capítulo 6 vamos apresentar e discutir os resultados obtidos, as métricas utilizadas e fazer comparações com trabalhos similares.

Por fim, o capítulo 7 é o fim do nosso trabalho, onde apresentamos as conclusões sobre os métodos propostos, sua eficiência e uma discussão sobre possíveis trabalhos futuros.

Capítulo 2

Recuperação de Informações Musicais

Gêneros musicais são rótulos categóricos criados e utilizados com o objetivo de estruturar o vasto universo musical [8]. Note que as definições dos gêneros musicais são subjetivas, e as fronteiras de separação confusas, visto que são construídas a partir de iterações sociais complexas entre pessoas, história, marketing, fatores culturais e outros [9]. Entretanto, mesmo com essas fronteiras imprecisas, membros de um gênero em particular ainda compartilham de características relacionadas a instrumentação e estrutura rítmica, por exemplo.

Neste capítulo faremos uma breve introdução aos conceitos de processamentos de sinais e suas aplicações na música.

2.1 Conceitos

Um conceito da física muito importante neste trabalho é o de **onda**. Uma onda é qualquer sinal que se transmite de um ponto a outro de um meio com velocidade definida [10].

Especificamente, o som é uma onda mecânica gerada pela vibração de objetos, causando uma oscilação na pressão do meio em que está sendo propagada e transportando energia. Definimos o som como uma onda longitudinal pois a direção da energia transportada é paralela ao movimento de oscilação das partículas do meio. Por fim, para o som ser considerado audível pelo ser humano, deve ter uma frequência entre 20 Hz e 20 KHz.

Para a música não é tão diferente: O que distingue a música de um ruído sonoro é a sua periodicidade. Nas próximas seções vamos ver as características responsáveis pelas sensações subjetivas que sentimos ao ouvir uma música, que serão divididas em três grupos: *Timbre*, *Volume* e *Altura*.

2.1.1 Timbre

Podemos pensar no timbre de uma música como sendo a sua *cor* ou sua textura. Através dessa propriedade é que podemos entender o conceito de qualidade (Escutar um som e reconhecer uma flauta, por exemplo) e sua identidade (O som de uma flauta ser diferente de uma outra) [11].

A definição formal para timbre, definida pela *Acoustical Society of America* em 1994, é: O timbre é um atributo de sensação auditiva em termos do qual um ouvinte pode julgar que dois sons apresentados de maneira semelhante, e que têm o mesmo Volume e Altura, são diferentes.

Apesar de ser um padrão aceito, essa definição já foi criticada diversas vezes. Segundo Bregman, uma de suas falhas é que não conceituamos o que é o timbre, apenas dizemos o que não é (volume e altura) [12].

2.1.2 Volume

Antes de definir volume (ou **loudness**) é importante ressaltar que este não é o mesmo conceito que **intensidade**. Intensidade está relacionada a uma entidade física (como a magnitude) de um som, já o volume se refere a uma entidade perceptual que só pode ser medida através da resposta de observadores humanos [11].

O volume está correlacionado com a intensidade e a potência do som, que é a taxa de transferência de energia por unidade de tempo: Um aumento na magnitude do sinal leva também a um aumento na percepção do volume, mas esta relação é *não-linear*, ou seja, um aumento linear na intensidade não gera um aumento linear no volume.

2.1.3 Altura

Altura (ou *Pitch*) é a sensação responsável por nos permitir caracterizar sons em agudos e graves, e a característica física associada a esta sensação é a **frequência**. O Pitch é o bloco mais fundamental para outros conceitos como de tom, melodia e séries harmônicas [11].

2.2 Fundamentos de processamento de áudio

Como descrevemos, a música é um sinal contínuo variante no tempo onde existe periodicidade. Seja $x(t)$ um sinal de áudio com período T_0. A primeira propriedade que podemos representar desse sinal é que:

$$x(t) = x(t + T_0) \tag{2.1}$$

Onde,

$$f_0 = \frac{1}{T_0} \tag{2.2}$$

é chamada de *frequência fundamental* do sinal.

Para armazenar esse sinal contínuo em um formato digital, utilizamos equipamentos (e.g microfone) capazes de converter o nível de variação da pressão do ar em **tensão elétrica**. Nos próximos tópicos descreveremos os processos de amostragem e quantização de um sinal, que são responsáveis, respectivamente, pela discretização no tempo e na amplitude. Os processos aqui descritos são componentes do método conhecido como *Pulse-code modulation* (**PCM**).

2.2.1 Amostragem

Uma propriedade importante que levaremos em conta para a amostragem é que podemos escrever um sinal periódico como a sobreposição de ondas senoidais ponderadas, chamada de **série de Fourier**:

$$x(t) = c_0 + \sum_{k=1}^{\infty} c_n e^{j\omega_n t} \tag{2.3}$$

Para o caso de sinais de áudio, somente a parte real da série será de interesse para nossa análise. O objetivo do processo de amostragem é discretizar o sinal no tempo, ou seja, vamos escolher alguns pontos a serem armazenados em um vetor. A escolha desses pontos deve ser tal que permita a reconstrução do sinal original com o menor erro possível. Uma forma conhecida de se amostrar um sinal é utilizando uma multiplicação do sinal por um **trem de impulsos** periódicos, como podemos observar na figura 2.1.

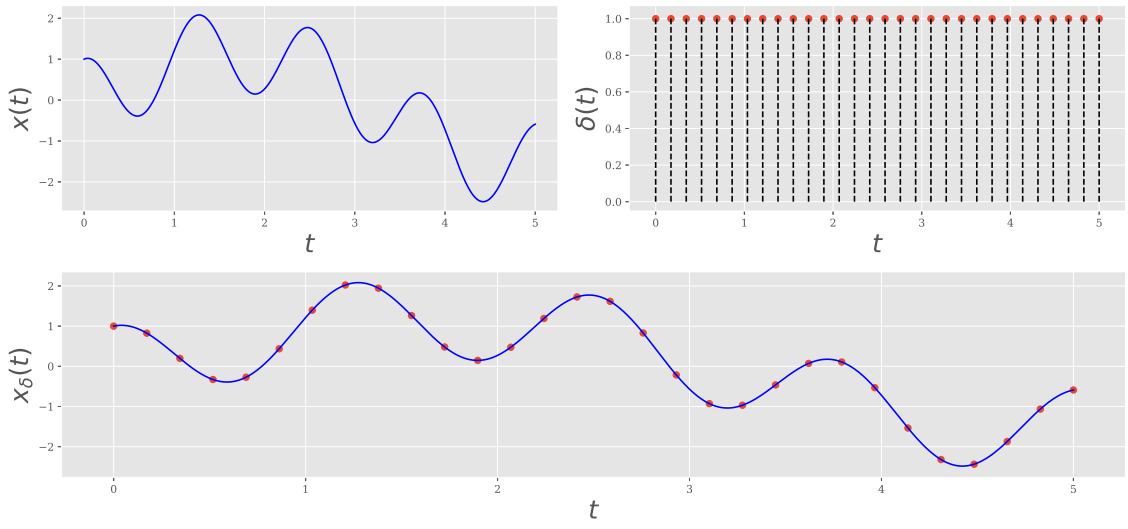


Figura 2.1: Processo de Amostragem

Matematicamente,

$$x_\delta(t) = x(t)p(t) \tag{2.4}$$

Onde,

$$p(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_0) \tag{2.5}$$

Um *sinal amostrado* só pode ser reconstruído sem perda de informação e sem ambiguidade se a frequência de amostragem é o dobro da maior frequência do sinal original. Esse principio é conhecido como o **Teorema da Amostragem**.

O principal objetivo desta seção foi mostrar ao leitor como um sinal de áudio pode ser representado discretamente, e que é possível do sinal discreto voltar à sua forma de onda original sem perda de informação.

2.2.2 Quantização

Depois do processo de amostragem temos uma sequência de números que podem assumir, em tese, qualquer valor em $\mathbb{R}$. Precisamos agora discretizar o sinal na amplitude, ou seja, precisamos representar esses valores contínuos como uma sequência finita de valores digitais.

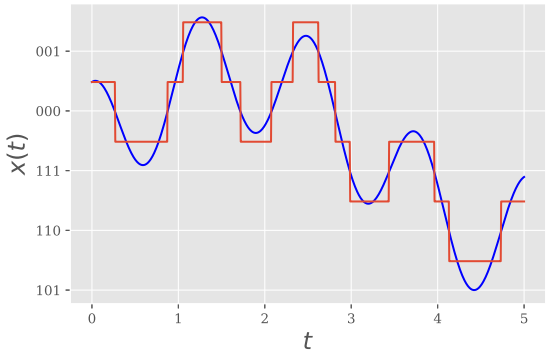


Figura 2.2: Processo de Quantização

Podemos tomar como exemplo o intervalo $[-1, 1] \subset \mathbb{R}$, cuja uma discretização linear poderia ser $(-1.0, -0.95, ..., 0, ..., 0.95, 1.0)$. O número de níveis diferente de amplitudes nesse intervalo é denotado por N_Δ. A diferença entre dois valores consecutivos nesse intervalo é chamada de *passo*, e é constante em toda a sequência (Esta modulação de passos constantes recebe o nome de **LPCM**). Denotamos o passo por Δ. Em nosso exemplo, $N_\Delta = 41$ e $\Delta = 0.05$

O mapeamento de um número qualquer x para seu nível de quantização mais próximo x_Δ, em geral ocasiona um pequeno erro, que chamamos de **erro de quantização**, matematicamente representado por:

$$e_\Delta = x_\Delta - x \tag{2.6}$$

Onde $e_\Delta \in [-\Delta/2, \Delta/2)$. Pensando no erro de quantização, agora para cada ponto do sinal discretizado e denotado por $e_\Delta[n]$, uma hipótese válida é considerar esse erro como um ruído branco descorrelacionado ao sinal, estocástico e com distribuição de probabilidade uniforme no intervalo descrito acima. A função de densidade de probabilidade é dada por:

$$p_{e_\Delta}(e) = \begin{cases} 1/\Delta & \text{se } e \in [-\Delta/2, \Delta/2) \\ 0 & \text{Caso contrário} \end{cases} \tag{2.7}$$

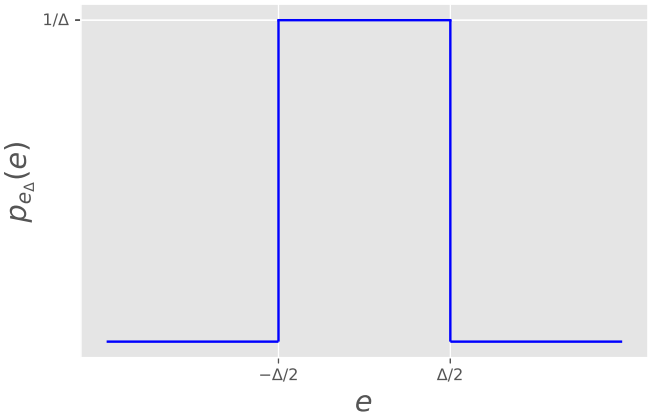


Figura 2.3: Função Densidade de Probabilidade para Distribuição Uniforme

Nosso principal objetivo é minimizar o erro de quantização. Para tal, vamos minimizar o erro médio quadrático, dado por:

$$E[e_\Delta^2] = \int_{-\infty}^{\infty} e^2 p_{e_\Delta}(e) de \tag{2.8}$$

$$E[e_\Delta^2] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2 \frac{1}{\Delta} de = \frac{\Delta^2}{12} \tag{2.9}$$

Podemos ainda relacionar essa grandeza Δ com o número de bits utilizado para a quantização (*bit-depth*) e a amplitude máxima do sinal. Para uma amplitude variando entre $[-A, A]$ e B o número de bits para a quantização, temos 2^B possíveis valores em um intervalo de altura $2A$. Assim,

$$\Delta = \frac{2A}{2^B} \tag{2.10}$$

$$E[e_{\Delta}^2] = \frac{A^2}{3 \times 2^{2B}} \tag{2.11}$$

O objetivo desta seção foi mostrar ao leitor como armazenamos os valores contínuos de um sinal de forma discreta no meio digital. Quanto maior o número de bits usado para a quantização, menor será o passo de quantização e menor será o nosso erro associado. Hoje grande parte das músicas que se encontram em formato digitais realizam a quantização utilizando 16 ou 24 bits.

2.2.3 Espectrogramas

Seja $x(t)$ um sinal de áudio de 30 segundos, transmitido em um único canal (**mono**) com taxa de amostragem de 22050 Hz e 16-bits de *bit-depth*. Digitalmente $x(t)$ pode ser representado no tempo por $x[n]$, que é um vetor com $22050 \times 30 = 661500$ elementos, onde cada entrada é um número real.

Uma forma usual e mais compacta de se representar o áudio é através de sua representação no domínio **tempo-frequência**, representação esta conhecida como **espectrograma**, que representa a distribuição da densidade espectral de energia do sinal. O espectrograma consiste em uma forma visual de representar o *volume* ao longo do tempo para as diferentes frequências presentes no sinal de áudio [13].

Na transformada de Fourier sobre todo o sinal, mudanças bruscas e variações locais no áudio, como início e final de eventos, não podem ser bem identificados pois esses fenômenos locais se tornarão globais. Na análise da densidade espectral de energia tomamos um pequeno trecho do áudio, processo chamado de **janelamento**, onde podemos dizer que o sinal é aproximadamente estacionário e calculamos a transformada de Fourier em cada segmento. Esse processo é conhecido como *Transformada de Fourier de tempo curto* (**STFT**).

Matematicamente, a STFT é dada por:

$$\mathbf{STFT}\Big\{x[n]\Big\}(m,\omega) = \chi(m,\omega) = \sum_{n=0}^{N-1} x[n]w[n-m]e^{-j\omega n} \tag{2.12}$$

Onde $w[n]$ é a função janela, $x[n]$ é o sinal de áudio discretizado, n é o tempo discreto (indíce do nosso vetor), ω a frequência em tempo contínuo e N é o comprimento da janela.

A função janela é uma função matemática que é não zero por um pequeno período de tempo, que é onde queremos janelar o sinal. A multiplicação dessa função por um sinal gera o que definimos como **sinal janelado**. Para obter a informação de frequência em diferentes tempos, fazemos um deslocamento da janela no tempo e calculamos a transformada de fourier de cada sinal janelado.

É comum também escrevermos a equação 2.12 em função de um parâmetro adicional chamado de **hop-size**, que indica em quantas unidades a janela deve ser deslocada no sinal.

$$\chi(m,\omega) = \sum_{n=0}^{N-1} x[n+mH]w[n]e^{-j\omega n} \tag{2.13}$$

Neste trabalho iremos utilizar a chamada **janela de Hann**, que é uma das mais populares em trabalhos de processamento de áudio e possuí um formato de sino similar à distribuição normal. A principal vantagem dessa função é suavidade nas bordas, evitando possíveis artefatos na transformada de Fourier do sinal janelado, o que é um problema comum com janelas retangulares, pois a função janelada terá pontos de descontinuidade. A janela de Hann é dada pela equação 2.14 e sua representação gráfica encontra-se na figura 2.4.

$$w(n) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right] \tag{2.14}$$

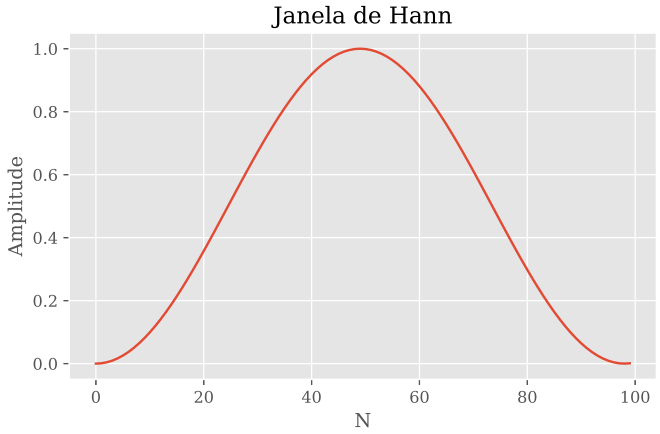


Figura 2.4: Gráfico da janela de Hann para um vetor de 100 amostras

Por fim, podemos definir o espectrograma γ como:

$$\gamma(m,\omega) = |\chi(m,\omega)|^2$$

De forma mais simples, podemos pensar nos espectrogramas como gráficos bidimensionais: O eixo horizontal representa o tempo, o eixo vertical representa as frequências presente no áudio. Os valores de $\gamma(m,\omega)$ são representados graficamente como cores na imagem. Através do espectrograma também podemos visualizar de forma mais fácil como a energia varia no tempo.

Existem outras formas usuais de se representar o eixo das frequências no espectrograma, como veremos abaixo. Da forma como definimos, o eixo das frequências está exposto em escala linear.

Escala-Log

Utilizamos uma transformação logarítmica no eixo das frequências com o objetivo de dar ênfase nas relações tonais ou musicais de um áudio, de forma mais próxima a percepção humana.

Escala-Mel

A escala *Mel* foi baseada em estudos sobre a percepção da audição humana. A hipótese testada nesses experimentos é de que o aparelho auditivo humano atua como um filtro que seleciona certas componentes de frequência que são espaçados de maneira não uniforme. Com esse mesmo estudo foi possível notar a concentração de filtros nas regiões de baixa-frequência. [14]

A conversão de uma escala linear para a escala de frequências Mel é dada pela seguinte equação:

$$\text{Mel}(f) = 1127ln\left(1 + \frac{f}{700}\right)$$

Na figura 2.5 podemos ver a diferença da utilização da escala linear, Log e a Mel em um espectrograma de uma música do gênero Jazz.

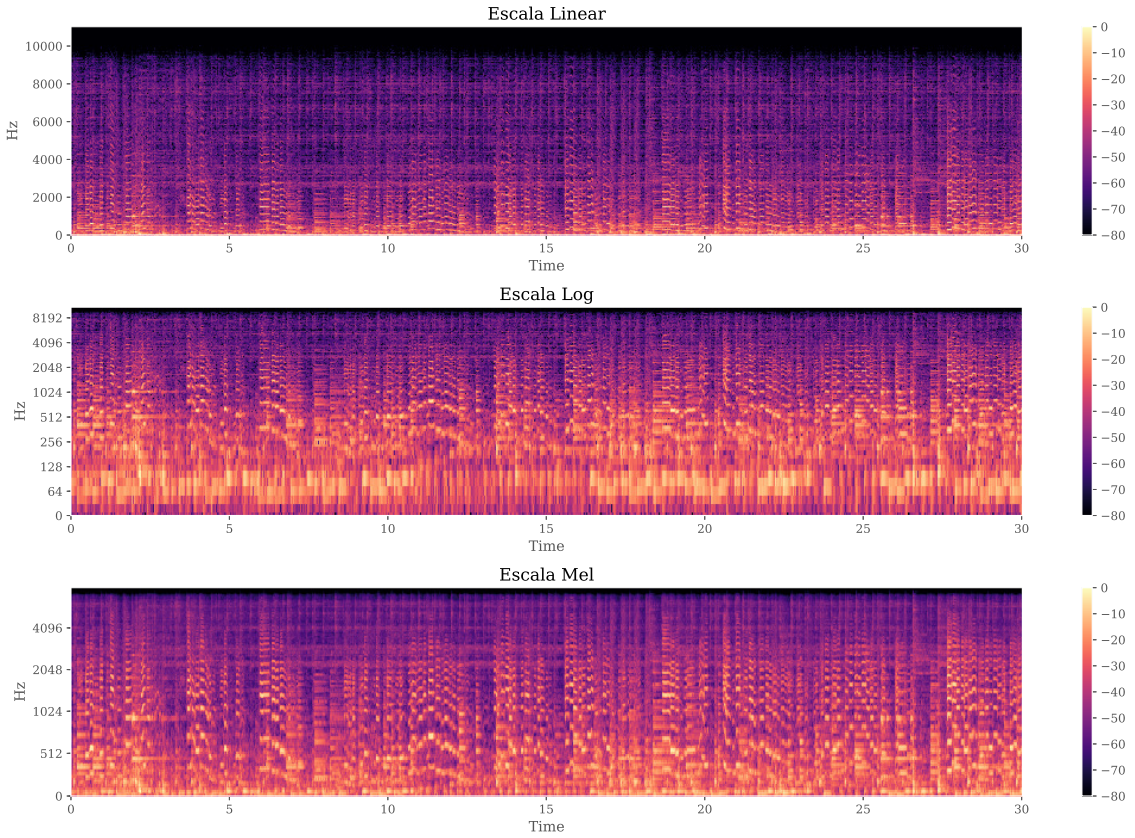


Figura 2.5: Diferentes escalas de frequência para um Espectrograma

2.3 Extração de Features

A extração de características (**features**) de um áudio é o processo de calcular representações numéricas compactas que caracterizam um sinal de áudio. O processo de escolha de quais features são mais adequadas para uma dada tarefa é o principal desafio nos sistemas clássicos de classificação e requerem o conhecimento de um especialista na área. Uma vez que definidos e extraídos, podemos utilizar as técnicas clássicas de machine learning para resolver nosso problema de classificação [9]. Neste trabalho iremos trabalhar com dois tipos de features de áudios, as características relacionadas ao timbre de uma música e as relacionadas ao *Tempo*.

2.3.1 Features de Textura do Timbre

As features baseadas em textura do timbre são características de baixo nível que medem a similaridade entre sons e fornecem uma interpretação semântica para o sinal (e.g. centroide, fluxo espectral, etc). As features a seguir são calculadas em pequenas janelas sobre o áudio, usualmente de 10 a 100 ms. Para resumir a informação em um único escalar, calculamos os seus momentos centrais dos vetores associadas a essas características.

Centroide Espectral

O centroide espectral pode ser pensado como o centro de gravidade da magnitude da STFT. Esta feature é comumente associada como o *brilho* do som, mas sua real informação é sobre o formato do espectro do nosso áudio e podemos interpretar altos valores (partes mais brilhantes) com as altas frequências. A equação que define o cálculo do centroide espectral é dada por:

$$C_t = \frac{\sum_n M_t[n] \times n}{\sum_n M_t[n]}$$

Onde $M_t[n]$ é a magnitude da transformada de Fourier na janela t e frequência n.

Rolloff Espectral

O Rolloff Espectral é uma outra medida da forma do espectro do áudio. Definimos essa medida como uma frequência de corte R_t à qual abaixo dela se concentra 85% da distribuição de magnitude do espectro. Matematicamente é dada por:

$$\sum_{n=0}^{R_t} M_t[n] = 0.85 \times \sum_n M_t[n]$$

Fluxo Espectral

Definimos como *Fluxo Espectral* a diferença quadrada entre as magnitudes normalizadas de espectros em janelas vizinhas. A propriedade mais importante relacionada ao fluxo espectral é que esta grandeza determina como o espectro muda localmente em um sinal. A equação para a determinação do fluxo espectral é dada por:

$$F_t = \sum_n (N_t[n] - N_{t-1}[n])^2$$

Onde N_t é a transformada de Fourier normalizada na magnitude na janela t.

RMSE

O Root-mean squared energy (**RMSE**) determina o quão *intenso* (**loud**) um sinal é em uma determinada janela. O RMSE de um sinal em uma janela é definido da seguinte forma:

$$\text{RMSE}_t = \sqrt{\frac{1}{N} \sum_n |x[n]|^2}$$

ZCR

O *Zero-crossing Rate* (**ZCR**), ou **taxa de cruzamentos por zeros**, é uma taxa que determina a quantidade de mudanças de sinal na magnitude de um áudio, que pode também ser visto como uma medida de ruidosidade.

$$Z_t = \frac{1}{2} \sum_n |sign(x[n]) - sign(x[n-1])|$$

MFCC

Mel Frequency Cepstral Coefficients, também conhecidos como **MFCC**, são features muito exploradas em processamento da voz mas que cada vez mais são utilizadas no campo de MIR. Os MFCCs são coeficientes que tem a capacidade de representar a amplitude do espectro da voz de forma compacta e cada etapa do seu cálculo é motivada por considerações de efeitos de percepção ou performance computacional. Abaixo apresentamos o algoritmo utilizado para cálculo dos coeficientes.

Algoritmo 1: Cálculo do MFCC

Result: MFCC

1. Obter o espectro através da STFT;
2. Aplicar a escala Mel;
3. Aplicar a Transformada Discreta de Cossenos;

A diferença do cálculo dos MFCCs para o Espectrograma da escala Mel está somente na última etapa do algoritmo, a aplicação da transformada discreta de cossenos. O objetivo dessa transformada é descorrelacionar os coeficientes encontrados através da diagonalização da matriz de autocorrelação, que também nos leva a uma representação mais compacta da distribuição de energia do sinal, de acordo com o número de MFCC que escolhermos tomar [15].

2.3.2 Features de Andamento

Por último, vamos investigar as características relacionadas ao *Tempo* (ou **andamento**) de uma música. Por definição, tempo é uma medida que determina o quão rápida ou lenta uma música é exibida. Usualmente essa grandeza é medida em BPM (*Batidas por minuto*). O conceito pode causar algumas confusões com o termo **ritmo**, que é o posicionamento de sons específicos no tempo, de forma repetitiva e regular [16]. Utilizaremos o tempo global de uma música como um dos descritores do áudio, apesar de saber que em alguns gêneros como Rock, o andamento varia muito em um intervalo de tempo. A ideia básica por trás do cálculo do tempo se dá a partir da agregação (em geral a média) de uma representação do áudio similar ao espectrograma, conhecida como tempograma, que determina a autocorrelação do sinal [17].

Capítulo 3

Aprendizado de Máquina

A teoria da evolução de Darwin é um dos maiores avanços científicos da história da humanidade. Seres vivos não são estáticos ou entidades imutáveis, mas sim mudam e evoluem constantemente. Sob essa ótica, na psicologia é utilizada uma definição funcional de aprendizado, onde diz-se que o aprendizado consiste de mudanças no comportamento de um organismo, que são o resultado de experiências e estímulos no ambiente no qual aquele organismo está inserido [18].

Este conceito pode ser estendido não somente para seres vivos mas também para uma máquina. Por uma perspectiva estatística e computacional, temos conceitos similares sobre o que é o aprendizado a partir de algoritmos.

3.1 Conceitos

Algoritmos de aprendizado de máquina (**Machine Learning**) são algoritmos capazes de aprender a partir de um conjunto de dados. Para entender o que significa aprendizado, vamos tomar como exemplo o problema de **aprendizado supervisionado**, onde temos associado ao conjunto de dados de entrada, um conjunto de dados de saída. Por aprender entendemos construir um modelo estatístico para predizer, ou estimar, uma saída a partir de entradas.

Uma definição amplamente aceita e difundida sobre aprendizado de máquina foi dada por Mitchell [19] e diz:

"Um algoritmo aprende de uma experiência E, em relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho nestas tarefas melhora com a experiência."

Nos próximos tópicos vamos explicar o que são essas métricas e como se adéquam sob a perspectiva do nosso problema.

3.1.1 Classificação

A tarefa que queremos abordar neste trabalho consiste em um problema de **classificação**, que é uma tarefa onde o computador deve dizer a qual classe uma dada observação (ou entrada) pertence. Por definição, o algoritmo deve realizar o mapeamento do conjunto de entradas em uma classe de saída através de uma função estimadora do tipo $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$. A variante deste modelo, que também será utilizada neste trabalho, é que a saída de mapeamento pode representar a probabilidade de um vetor x pertencer a cada uma das k classes [3]. Em nossas análises sempre partimos do pressuposto de que nossas observações são independentes e identicamente distribuídas.

Nossa tarefa consiste em dado um conjunto de entrada, que podem ser as features de cada áudio ou alguma representação pura do sinal, e queremos determinar uma função de estimação para classificar o gênero musical associado com essa entrada.

3.1.2 Desempenho

O desempenho P de um algoritmo é uma métrica que nos permite quantificar o quão acurado um modelo é na realização da tarefa T.

Como estamos trabalhando com um problema de classificação onde as classes são **balanceadas**, ou seja, a proporção de vezes que uma das k classes aparece no conjunto de dados é aproximadamente igual, podemos utilizar a **acurácia** como essa medida de desempenho. Por acurácia entendemos a proporção de acertos que o modelo produz. Uma métrica equivalente de desempenho é a **taxa de erro**, que é o complementar da acurácia.

3.1.3 Experiência

Dizemos que um algoritmo de aprendizado de máquina pode experimentar um *conjunto de dados* (**dataset**). Um dataset pode ser visto como uma matriz, onde as linhas representam observações e as colunas as features associadas com aquela entrada, onde as features do dataset também são chamadas de variáveis independentes. No caso de algoritmos de aprendizado supervisionado, para cada entrada do nosso dataset existe um rótulo (também conhecido por classe ou **label**), que chamamos de variável dependente.

Uma visão mais formal é que o aprendizado supervisionado envolve observar diversos exemplos de um vetor aleatório x e rótulos y, para aprender a predizer y a partir de x estimando $p(y|x)$.

Definimos como **treinamento** o ajuste da função de classificação a partir do conjunto de dados de entrada. Os dados de entrada de um algoritmo de Machine

Learning podem ser divididos entre uma parte com padrões e ruídos estocásticos de distribuição normal, dado por:

$$y = f(\boldsymbol{x}) + \epsilon \tag{3.1}$$

Nosso objetivo é ajustar uma função $\hat{y} = f(x;\theta)$ onde $\hat{y}$ seja o mais acurado possível, ou seja, o mais próximo de y o quanto puder. O que fazemos em geral é resolver um problema de otimização, onde tentamos ajustar os parâmetros θ de forma a minimizar essa diferença, que é modelada através de uma **função de custo**. Após a fase de treinamento, a capacidade do modelo de performar bem em um novo conjunto de dados é chamado de **generalização** e é o maior objetivo que buscamos em nossos algoritmos.

Quando ajustamos não somente a função mas também os ruídos associados aos dados, nosso algoritmo está em regime de **overfitting** e provavelmente não irá generalizar bem para novos dados de entrada. Uma forma de reduzir o erro de generalização é a utilização de técnicas de **Regularização**. Regularizadores são funções de penalização adicionadas à função de custo do modelo que tem como objetivo beneficiar funções mais simples sobre as mais complexas e assim evitar o overfitting.

3.2 Algoritmos Clássicos

O objetivo desta seção é explicar de forma superficial a teoria por trás de alguns modelos utilizados neste trabalho para realizar a classificação tendo como entrada as features dos áudios e saída os gêneros musicais. Uma descrição mais completa e detalhada sobre os algoritmos pode ser encontrada em [20][21].

3.2.1 Regressão Logística

A regressão logística, ou classificador de máxima entropia, é um modelo linear para classificação cujo objetivo é modelar a probabilidade a posteriori das K classes através de funções lineares em $\boldsymbol{x}$ e θ. Para facilitar o desenvolvimento, vamos considerar um problema dicotômico onde $y \in \{0, 1\}$. A ideia para um problema multiclasses como o nosso pode ser derivado de forma semelhante e pode ser encontrado em [20].

Seja $h_\theta(\boldsymbol{x}) = Pr(y = 1|\boldsymbol{x};\theta)$. Um mapeamento para essa probabilidade que respeita a condição de ter valores entre 0 e 1, em função linear dos parâmetros de entrada, é dada pela composição da função especial chamada **função sigmóide** com a função linear [22].

$$h_\theta(\boldsymbol{x}) = g(\theta^T \boldsymbol{x}) = \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \tag{3.2}$$

Uma convenção que adotamos é que o modelo linear em função dos parâmetros θ pode ser escrito como um produto escalar da forma $\theta^T \boldsymbol{x}$. Supondo que $x \in \mathbb{R}^n$, escrevemos: $\theta^T \boldsymbol{x} = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$. θ_0 é conhecido como intercepto do modelo. Outra convenção é que, como $h_\theta(\boldsymbol{x})$ representa a probabilidade de $\boldsymbol{x}$ pertencer a classe 1, definimos que se $h_\theta(\boldsymbol{x}) > 0.5$, $\boldsymbol{x}$ pertence a classe 1.

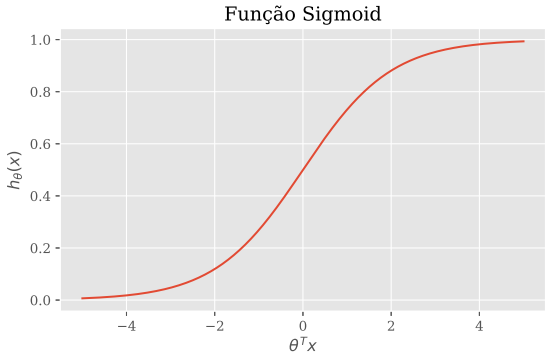


Figura 3.1: Gráfico da função sigmóide

Pela figura 3.1 podemos notar que $h_\theta(\boldsymbol{x})$ tende à 1 a medida que $\theta^T \boldsymbol{x} \to \infty$ e que $h_\theta(\boldsymbol{x})$ tende à 0, a medida que $\theta^T \boldsymbol{x} \to -\infty$.

A função de custo a ser otimizada (e calcularmos os melhores parâmetros θ do modelo) pode ser obtida através da **estimativa de máxima verossimilhança**. Vamos assumir que o evento de pertencer a classe 0 ou a classe 1 possua uma distribuição de Bernoulli.

$$\begin{aligned} Pr(y = 1|\boldsymbol{x}; \theta) &= h_\theta(\boldsymbol{x}), \\ Pr(y = 0|\boldsymbol{x}; \theta) &= 1 - h_\theta(\boldsymbol{x}). \end{aligned} \tag{3.3}$$

Combinando em uma única equação, podemos escrever que:

$$Pr(y|\boldsymbol{x}; \theta) = h_\theta(\boldsymbol{x})^y (1 - h_\theta(\boldsymbol{x}))^{1-y} \tag{3.4}$$

Neste trabalho vamos adotar a convenção de que um vetor com índice i na parte superior significa que estamos trabalhando com a i-ésima observação de um dataset.

Como definimos anteriormente, na construção desses conjuntos de dados partimos do pressuposto que as observações são independentes umas das outras. Portanto, podemos escrever a probabilidade associada para os vetores (x, y) como produto das probabilidades de cada observação, de forma que:

$$
\begin{aligned}
L(\theta) &= Pr(y|\boldsymbol{x};\theta) \\
&= \prod_{i=1}^{m} Pr(y^{(i)}|x^{(i)};\theta) \\
&= \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}}
\end{aligned}
\tag{3.5}
$$

Ao invés de maximizarmos esta função de verosimilhança, podemos utilizar o logaritmo dela que será mais fácil de manipular.

$$
\begin{aligned}
l(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)})(1-h_\theta(x^{(i)}))
\end{aligned}
\tag{3.6}
$$

A equação 3.6 é a nossa função de custo do modelo, e para obtermos os melhores valores de θ para a tarefa, devemos maximizar esta função. Um método bastante recorrente utilizado para este tipo de tarefa será discutido na seção 3.3.4.

3.2.2 Support Vector Machine

A Support Vector Machine (**SVM**) é um dos algoritmos mais populares para realizar tarefas de Machine Learning por seu alto poder de resolução de problemas. A principal ideia do algoritmo é construir um hiperplano, ou um conjunto de hiperplanos, em um espaço de alta ou infinita dimensão para separar as classes. Uma alta acurácia na solução do problema está relacionado com um hiperplano de maior margem (vetor normal ao hiperplano), já que intuitivamente quanto mais larga a margem, menor o erro de generalização do classificador.

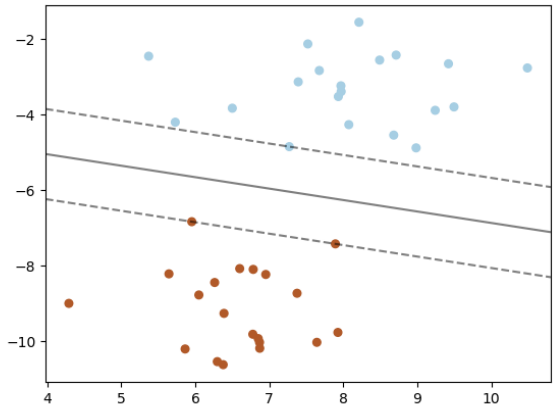


Figura 3.2: Processo de definição das margens da SVM [1].

Uma das vantagens de se utilizar a SVM é a possibilidade de estabelecer funções de separação não-lineares. Na SVM podemos utilizar **Kernels**, que são funções matemática utilizadas para computar a similaridade de um exemplo com seu mapeamento em um espaço de features, dado da seguinte forma:

$$x^{(i)} \in \mathbb{R}^n \Longrightarrow \begin{bmatrix} f_1^{(i)} = \text{Kernel}(x^{(i)}, x^{(1)}) \\ f_2^{(i)} = \text{Kernel}(x^{(i)}, x^{(2)}) \\ \vdots \\ f_m^{(i)} = \text{Kernel}(x^{(i)}, x^{(m)}) \end{bmatrix} \tag{3.7}$$

Neste trabalho utilizaremos um Kernel gaussiano, que é dado por uma **função gaussiana**. A função de custo da SVM será em torno do mapeamento $\theta^T f$ no espaço de features e não de uma simples combinação linear dos exemplos.

Os modelos que aqui foram apresentados são utilizados neste trabalho para estimar a função de classificação tendo como entrada as features dos áudios e saída os gêneros musicais. Outros modelos como **árvores de decisão** e **random forest** são utilizados com o objetivo de fornecer uma gama maior de classificadores e comparação de desempenho, mas não serão discutidos em profundidade por não agregarem valor ao objetivo final proposto neste trabalho.

Na próxima seção vamos introduzir uma técnica que nos permite utilizar como entrada uma representação mais simples do áudio (espectrograma), aprender progressivamente seus padrões para então realizar a classificação.

3.3 Redes Neurais Artificiais

A rede neural artificial (**RNA**) é a principal técnica deste trabalho. O objetivo de uma RNA é aproximar uma função f através de um mapeamento $y = \tilde{f}(\boldsymbol{x}; w, b)$ e aprender os melhores valores para os parâmetros w e b que resultam na melhor função aproximada. Nesta seção vamos entender o funcionamento deste algoritmo e como podemos utilizar para o reconhecimento de gêneros musicais. A nomenclatura $f(\boldsymbol{x}; w, b)$ é equivalente a $f(\boldsymbol{x}; \theta)$ que utilizamos na regressão logística, mas a primeira é mais comumente utilizada em redes neurais. Nessa nomenclatura, temos que $w = [\theta_1, ..., \theta_{n-1}]$ e $b = \theta_0$, onde w também é chamado de *peso* do modelo e b de viés (ou *bias*).

A concepção do modelo de Redes Neurais começou como um estudo de modelagem do sistema nervoso biológico dos seres vivos, mas desde então essa perspectiva mudou bastante. O elemento central do sistema nervoso responsável pela transmissão de impulsos nervosos é o **neurônio**. No sistema nervoso humano existem cerca de 86 bilhões dessas células. Cada neurônio recebe sinais de entrada através de seus

dendritos e transmite um sinal de saída para suas conexões, que são outros neurônios. O sinal de saída é transmitido através de uma estrutura chamada axônio, que se ramifica e possuí terminais que são responsáveis por realizarem as sinapses com os dendritos de suas conexões.

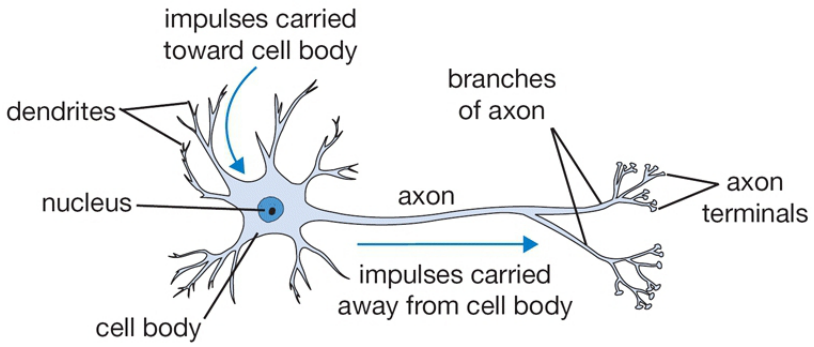


Figura 3.3: Modelo biológico para o Neurônio [2].

O modelo computacional não é tão diferente. Na figura 3.4 temos uma representação matemática aproximada para o neurônio biológico descrito na figura 3.3. Um sinal x_0 viaja do axônio de um neurônio, interage com os dendritos do receptor, de acordo com uma força sináptica w_0, resultando em w_0x_0. A ideia por trás dos modelos das RNAs é que a força sináptica (os parâmetros) são aprendidos através de exemplos e controlam a influência de um neurônio sobre outro [2].

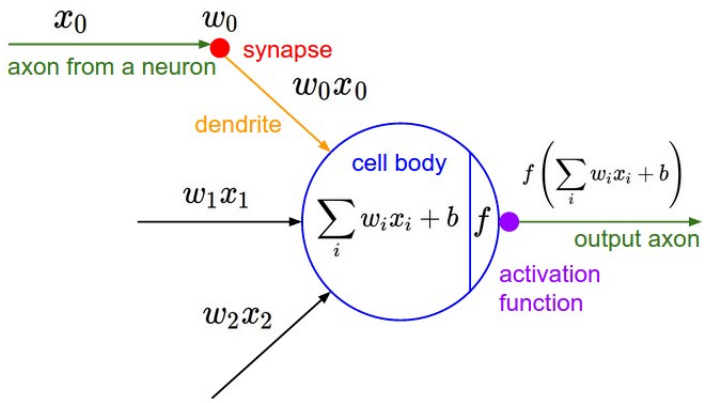


Figura 3.4: Modelo Artificial para o Neurônio [2].

No modelo computacional mais básico para o neurônio, os sinais carregados do dendrito para o corpo (também chamado de *soma* na biologia) são todos somados e disparam um novo sinal modelado através de uma **função de ativação**. Dedicaremos uma subseção para falar especificamente das funções de ativação, mas a sigmóide, a mesma que vimos na regressão logística, ainda é uma das escolhas mais populares. Este modelo para o neurônio artificial foi concebido por McCulloch e Pitts [23] em 1943, e até hoje é o mais aceito.

3.3.1 Conceitos

As redes neurais artificiais são o conjunto de neurônios artificiais conectados. Neste trabalho iremos trabalhar com um tipo de rede chamado *feedforward* pois os sinais só fluem em uma única direção na rede. Este tipo de rede também pode ser vista como um grafo acíclico direcionado (**DAG**) que nos permite modelar uma RNA através de composições de funções. Nas RNAs, os neurônios estão agrupados em **camadas**, onde neurônios de duas camadas adjacentes estão totalmente conectados par-a-par. A figura 3.5 mostra essa representação.

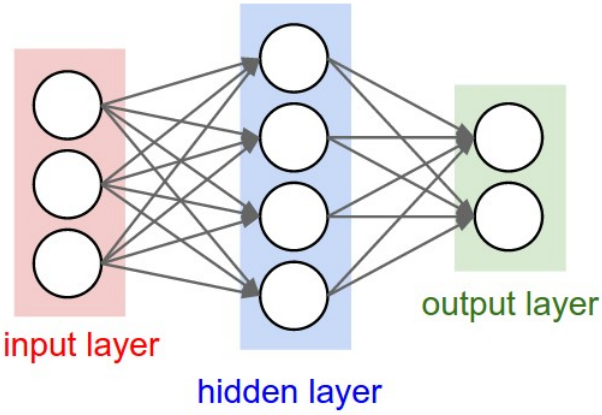


Figura 3.5: Rede Neural Artificial [2].

Usualmente a nomenclatura encontrada para esse tipo de estrutura das RNAs é: camada de entrada (**input layer**), camada oculta (**hidden layer**) e camada de saída (**output layer**). Quando existe mais de uma camada entre a camada de entrada e a camada de saída, dizemos que esse modelo é uma **rede neural profunda**.

Em um problema de classificação com K classes, é comum que a camada de saída contenha K neurônios e o valor numérico associado a saída de um desses neurônios corresponde à probabilidade da entrada $\boldsymbol{x}$ pertencer a tal classe. Essa ideia é uma generalização da regressão logística binária, que damos o nome de classificador **Softmax**. Como o esperado, o somatório dessas K probabilidades é igual a 1.

3.3.2 Funções de Ativação

As funções de ativação, também conhecidas como **não-linearidade** são funções matemáticas que recebem um único valor de entrada e produzem uma única saída, funções do tipo $y = f(x)$. Vamos discutir as 3 funções de ativações mais comuns na área de Redes Neurais.

Sigmóide

A função sigmóide foi largamente utilizada como função de ativação para as redes neurais por ser historicamente utilizada na regressão logística, como mostrado na equação 3.2. Hoje esta função de ativação não é tão utilizada, principalmente em redes neurais profundas, por ocasionar dificuldades durante o processo de otimização dos parâmetros w e b.

De acordo com a atualização dos parâmetros, o neurônio pode saturar em uma das duas regiões extremas da função (0 ou 1), onde seu gradiente é praticamente nulo, e assim não conseguir mais aprender com novos exemplos. Outro problema com esta a função é que ela não é centrada no zero e isso pode gerar instabilidade na dinâmica de atualização dos parâmetros do modelo.

O segundo problema foi resolvido pela função tangente hiperbólica como veremos abaixo.

Tangente Hiperbólica

A tangente hiperbólica (**tanh**) é uma alternativa à função sigmóide. Apesar de também poder ocasionar a saturação de um neurônio, a função é centrada no zero e na prática é mais estável durante o processo de otimização.

A tangente hiperbólica é dada pela equação 3.8 e podemos observar seu gráfico na figura 3.6.

$$\tanh(x) = 2\sigma(2x) - 1 \tag{3.8}$$

Onde $\sigma(x)$ é a função sigmóide.

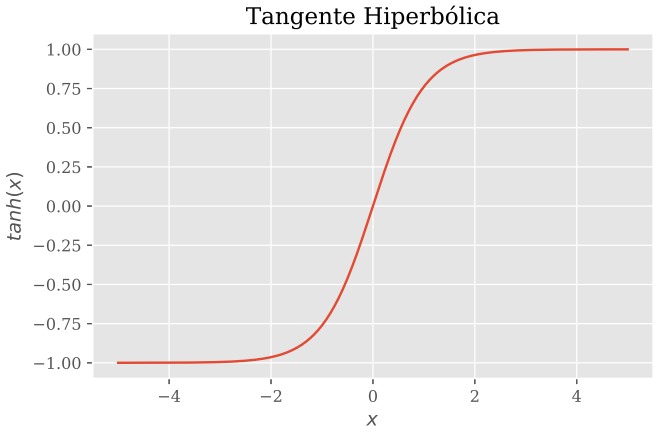


Figura 3.6: Função Tangente Hiperbólica

ReLU

A ReLU (*Rectified Linear Unit*) é provavelmente a função de ativação mais popular hoje. Esta ativação funciona como um limiar em zero. A equação 3.9 mostra a formulação matemática da função e a figura 3.7 mostra seu gráfico.

$$\text{relu}(x) = \max(0, x) \tag{3.9}$$

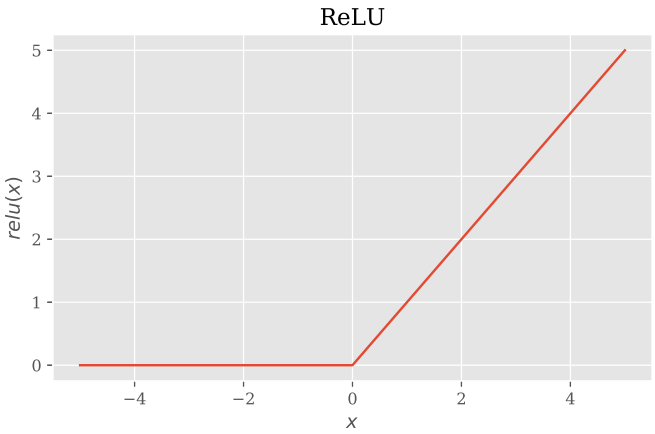


Figura 3.7: Representação gráfica da ReLU

A ReLU apresenta algumas vantagens se comparada com as outras funções apresentadas acima. A mais evidente é que a operação é mais simples de ser calculada. A outra é que a função não satura como a sigmóide ou a tangente hiperbólica, o que é uma consequência para a convergência mais rápida durante o processo de otimização.

3.3.3 Função de Custo

Da mesma forma que estudamos para as outras técnicas de classificação, a função de custo computa uma medida de diferença entre a classe real de uma entrada e a classe estimada pelo modelo. Em problemas de classificação de múltiplas classes também utilizamos a função de custo da regressão logística aqui nas redes neurais. No contexto das redes neurais normalmente utilizamos o termo **entropia cruzada** (*cross-entropy*) para descrever tal função de custo e representamos tal função por:

$$J(w, b) = -\frac{1}{m} l(\theta) \tag{3.10}$$

Onde $l(\theta)$ é a função de custo de entropia cruzada definida pela equação 3.6, m o número de entradas no conjunto de treinamento, $w = [\theta_1, ..., \theta_{n-1}]$ e $b = \theta_0$

3.3.4 Métodos de Otimização

O objetivo desta seção é explicar um dos algoritmos mais famosos para calcular os melhores valores de w e b da nossa função ajustada, de forma a minimizar o erro cometido pelo modelo. Este procedimento de ajuste de parâmetros é feito através do algoritmo conhecido como **método do gradiente** (ou *gradient descent*).

A função de custo $J(w, b)$ é uma função convexa e possuí um único máximo global. Para facilitar a intuição por trás do algoritmo, vamos considerar um exemplo de função no $\mathbb{R}^3$ através da figura 3.8.

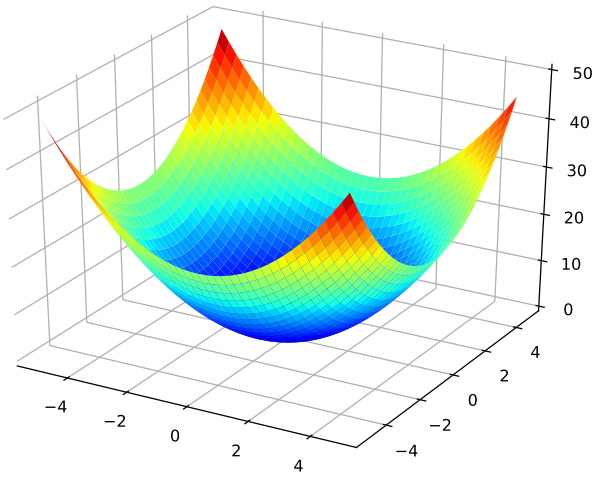


Figura 3.8: Representação gráfica de uma função convexa em $\mathbb{R}^3$

Em um ponto qualquer nesse gráfico, para chegarmos ao ponto de mínimo da função, podemos calcular e seguir a direção oposta em que o gradiente da função estiver apontando. O tamanho do passo que devemos dar nesta direção é dado por α e é chamado de **learning rate**. Para uma dada camada l da rede, podemos calcular a atualização dos seus parâmetros através da equação 3.11.

$$\begin{aligned} W^{[l]} &= W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}} \\ b^{[l]} &= b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}} \end{aligned} \tag{3.11}$$

Para utilizar este algoritmo, precisamos inicializar os pesos da rede neural antes de começar o processo de treinamento. Não podemos inicializar todos os pesos como zero pois não seria possível atualizar os valores no futuro através da equação 3.11. A inicialização mais utilizada é de escolher números aleatórios bem pequenos, com uma distribuição uniforme.

Os pesos são atualizados com base em todo o conjunto de dados de treinamento.

O ato de atualizar os pesos da rede neural com todos os dados disponíveis é chamado de **época**. Em geral esse processo é repetido por várias épocas até que não seja mais possível atualizar os pesos da rede neural ou esta comece a entrar em regime de overfitting.

A questão restante para finalizar o método de otimização é como calcular de forma eficiente os gradientes da função de custo. Veremos na próxima seção o **backpropagation** que é capaz de realizar tal tarefa.

Para resumir os tópicos aqui apresentados, para determinar os valores de W e b precisamos primeiro inicializar os pesos de forma adequada. Então, por um número de épocas determinadas (ou até um outro critério de convergência), repetimos o seguinte procedimento: Calcular o valor previsto $\hat{y}$ para cada exemplo do conjunto de treinamento, calcular o função de custo, calcular os gradientes e atualizar os pesos. Em pseudo-código:

Algoritmo 2: Gradient Descent

Result: W, b

repeat

    $\hat{y}$ = forward($\boldsymbol{x}$)
    loss = L($\hat{y}$, y)
    dW, db = backpropagation(loss)

    W = W - alpha*dW
    b = b - alpha*db

until *convergência*

Por fim é válido ressaltar que existem outras variantes do método do gradiente. No treinamento de redes neurais profundas é comum utilizarmos um conjunto de dados tão grande que podem não caber simultaneamente na memória do computador. Existem algoritmos que subdividem o dataset em pedaços menores, chamados de *mini-batch*, para então realizar o treinamento. Um algoritmo bem popular com essa característica é o **mini-batch stochastic gradient descent** [24]. Outra variante do método do gradiente é com relação à mudanças dinâmicas no learning rate, em contraste com a utilização de uma constante. Neste trabalho utilizaremos um algoritmo chamado **Adam** que implementa ambas dessas variantes e pode ser visto com mais detalhes em [25].

3.3.5 Backpropagation

O ato de computar uma saída $\hat{y}$ a partir de uma entrada $\boldsymbol{x}$ é chamada de **forward propagation**, pois a informação está fluindo da camada de entrada para a camada de saída. Uma vez tendo o valor de $\hat{y}$, podemos calcular a função de custo e calcular, da camada de saída para a camada de entrada, o gradiente da função de custo. O algoritmo para realizar esta segunda tarefa é chamado de **backpropagation**.

A principal ideia por trás do backprogation é computar, de maneira numérica e eficiente os gradientes da função de custo através de um grafo computacional e aplicando sucessivamente a regra da cadeia do cálculo diferencial.

Para entender o funcionamento do algoritmo, vamos considerar um exemplo mais simples, mas cuja ideia pode ser generalizada para outras arquiteturas de redes neurais. Para uma explicação mais aprofundada sobre o backpropagation recomenda-se a leitura de [3].

Seja uma rede neural com 3 neurônios na camada de entrada, dois na camada intermediária e 1 neurônio na saída, como apresentado na figura 3.9.

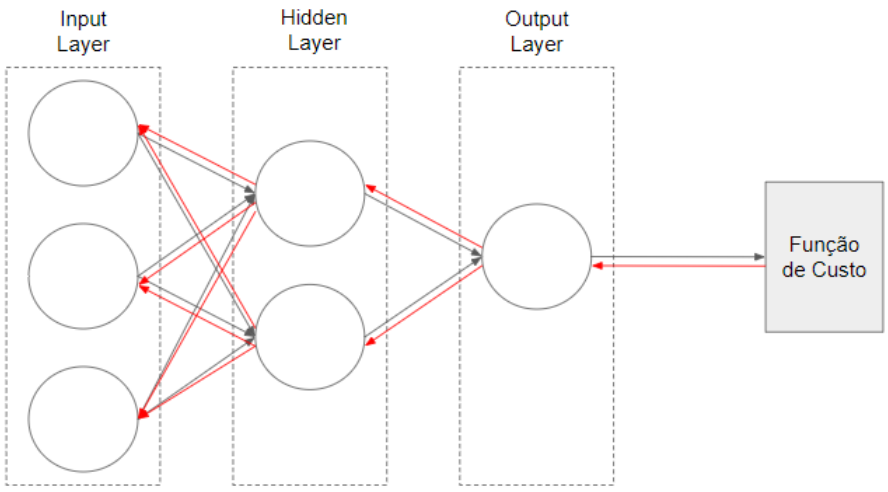


Figura 3.9: Rede neural simples para exemplificar o backpropagation

Para calcular o resultado $\hat{y}$ dessa rede, podemos compor as funções de forma:

$$\begin{aligned} z^{[1]} &= W^{[1]}\boldsymbol{x} + b^{[1]} \\ a^{[1]} &= g(z^{[1]}) \end{aligned} \tag{3.12}$$

Onde o índice superior indica a camada da rede que estamos nos referindo e g é a função de ativação do neurônio. De forma semelhante, para a camada de saída

temos:

$$\begin{aligned} z^{[2]} &= W^{[2]}\boldsymbol{a}^{[\mathbf{1}]} + b^{[2]} \\ a^{[2]} &= g(z^{[2]}) = \hat{y} \end{aligned} \tag{3.13}$$

Com o valor de $\hat{y}$ em mãos podemos calcular todos os gradientes associados, começando da camada de saída em direção à camada de entrada. Seja $\mathcal{L}(\hat{y}, y)$ a função de custo associado à rede neural. Os gradientes podem ser computados da seguinte forma:

$$\begin{aligned} da^{[2]} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[2]}} \\ dz^{[2]} &= \frac{\partial \mathcal{L}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} = da^{[2]} \frac{\partial g(z^{[2]})}{\partial z^{[2]}} \\ dW^{[2]} &= \frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial W^{[2]}} = dz^{[2]} a^{[1]T} \\ db^{[2]} &= \frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{\partial \mathcal{L}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b^{[2]}} = dz^{[2]} \end{aligned} \tag{3.14}$$

De forma semelhante, podemos calcular os gradientes associados à $W^{[1]}$ e $b^{[1]}$:

$$\begin{aligned} da^{[1]} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[1]}} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} = dz^{[2]} W^{[2]} \\ dz^{[1]} &= \frac{\partial \mathcal{L}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial z^{[1]}} = da^{[1]} \frac{\partial g(z^{[1]})}{\partial z^{[1]}} \\ dW^{[1]} &= \frac{\partial \mathcal{L}}{\partial W^{[1]}} = \frac{\partial \mathcal{L}}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial W^{[1]}} = dz^{[1]} \boldsymbol{x}^T \\ db^{[1]} &= \frac{\partial \mathcal{L}}{\partial b^{[1]}} = \frac{\partial \mathcal{L}}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial b^{[1]}} = dz^{[1]} \end{aligned} \tag{3.15}$$

Com os gradientes $dW^{[2]}$, $dW^{[1]}$, $db^{[2]}$ e $db^{[1]}$ em mãos, podemos utilizar o método do gradiente como mostrado na seção anterior para atualizar os parâmetros da rede neural.

3.3.6 Métodos de Regularização

Neste trabalho duas formas de regularização foram testadas nas redes neurais para aumentar a capacidade de generalização do modelo. São eles: **Regularização L2** e **Dropout**.

L2

A regularização L2 consiste em colocar um parâmetro de penalização na função de custo do modelo de tal forma que temos:

$$J_{\mathrm{L2}}(W, b) = J(W, b) + \frac{\lambda}{2} W^T W \tag{3.16}$$

Quanto maior o valor de λ, maior a penalização. A regularização L2, a cada iteração de atualização de pesos irá incluir uma penalidade dependendo de W, que aumentam o valor da função de custo $J(W, b)$ e incentiva que os valores dos parâmetros sejam pequenos em magnitude, que é uma forma de reduzir o overfitting.

Dropout

Dropout é uma técnica recente de regularização criada por Srivastava em [26]. A principal ideia é que este é um método que complementa os demais métodos de regularização. O dropout remove alguns neurônios (Processo também conhecido como *matar* o neurônio) de forma aleatória com uma probabilidade p durante a fase de treinamento, reduzindo a chance de um neurônio se adaptar muito a um dado de entrada (ou seja, entrar em regime de overfitting).

3.3.7 Redes Neurais Convolucionais

As redes neurais convolucionais (**CNNs**) foram introduzidas por Yann LeCun [27] com a proposta de processarem dados que possuam um certo tipo de topologia em grade com relação aos seus vizinhos. Por exemplo, imagens e espectrogramas podem ser vistas como uma grade 2D onde existe algum tipo de relação entre seus pixels; Um sinal puro de áudios também pode ser visto como uma grade 1D.

A característica principal de uma CNN é a utilização de uma operação matemática chamada **convolução**. A convolução é um operador linear que calcula a soma dos produtos de uma função x com uma segunda função w chamada de **kernel** (ou *filter*). O resultado dessa operação é conhecido como **feature map**. Matematicamente escrevemos:

$$s(t) = (x * w)(t) = \sum_a x(a) w(t - a) \tag{3.17}$$

Em geral, uma CNN é uma sequência de camadas, onde cada camada transforma um volume de ativações em outro volume. É comum utilizarmos três tipos de camadas em uma CNN: **Camadas convolucionais**, **camadas de Pooling** e camadas totalmente conectadas (**Fully-connected**). As junções dessas camadas compõem a arquitetura da rede neural.

A camada convolucional é onde aplicamos a operação de convolução ao volume de entrada através de um Kernel. O volume de entrada é um tensor que dependerá do tipo de dado sendo manipulado no trabalho. Para o caso de imagens, temos um tensor com três dimensões: Altura, largura e profundidade (canais). Para um áudio temos um tensor com duas dimensões, sendo uma a largura (Amplitude no tempo) e a profundidade (canais). O kernel também é um tensor, que são os parâmetros do nosso modelo e serão ajustados pelo algoritmo de aprendizado. Usualmente o kernel é espacialmente menor que o volume de entrada, mas possuí a mesma profundidade. Também é comum utilizarmos mais de um kernel para realizar a convolução com o volume de entrada da camada.

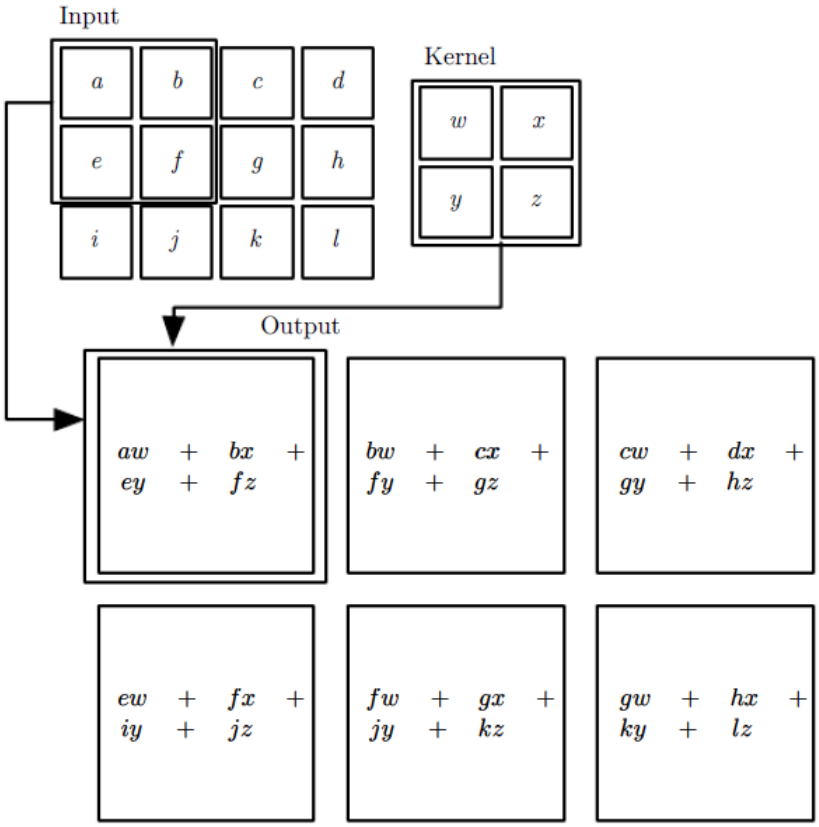


Figura 3.10: Exemplo de convolução para uma entrada 2D. Retirado de [3]

Durante a fase de *forward* de uma entrada, nós convoluímos o kernel com a entrada, deslizando a janela, como mostrado na figura 3.10. Durante a fase de backward, ajustamos os parâmetros do kernel. O kernel em geral se comporta como um extrator de features. Nas camadas mais iniciais da rede é comum notarmos que o kernel ajustou seus pesos de forma a se tornar um filtro detector de bordas, por exemplo. Já nas camadas mais finais encontramos filtros capazes de reconhecer features de mais alto nível. Por isso dizemos que CNNs são sistemas de classificação **end-to-end**, onde as camadas convolucionais são responsáveis por extraírem as features necessárias para descrever o volume de entrada e a camada totalmente

conectada no final é responsável por realizar a classificação.

A camada convolucional é uma representação mais compacta e com menos parâmetros para serem aprendidos do que em uma rede neural usual, devido as propriedades topológicas exploradas. Entretanto tempo alguns hiperparâmetros para serem ajustados nesse tipo de arquitetura, como o tamanho do kernel, a quantidade de kernels e o passo que a janela desliza (**stride**). Também é comum, para evitar que uma janela não possa convoluir até o final do volume por falta de espaço, inserir zero nas regiões das bordas, operação essa conhecida como **zero-padding**.

Após a camada convolucional é comum encontramos uma camada de não-linearidade, como vimos nas redes neurais tradicionais, onde aplicamos, em cada elemento do tensor de saída, uma função de ativação como a ReLU.

Já a camada de Pooling é uma camada que nos ajuda a reduzir a dimensão de um volume de saída de uma camada convolucional através do cálculo de estatísticas dos elementos vizinhos. Uma das operações mais comuns utilizadas é a de **maxpooling** onde calculamos o valor máximo dentro de uma vizinhança retangular. Outras métricas como média ou norma L2 também pode ser utilizada. Um exemplo de operação de maxpooling pode ser encontrado na figura 3.11. Novamente é possível definir hiperparâmetros para esse tipo de modelo, sendo eles: stride, zero-padding e o tamanho da janela do pooling.

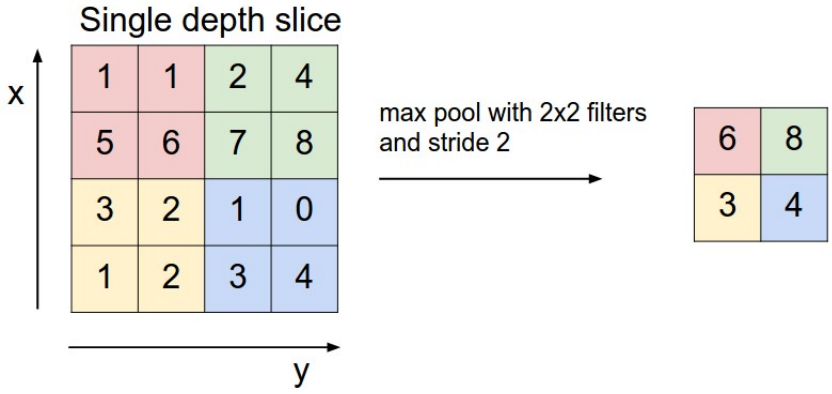


Figura 3.11: Exemplo de operação maxpooling [2]

Por último é comum utilizarmos camadas totalmente conectadas no final da rede para realizarem a tarefa de classificação. A saída dessa camada são os scores da rede com relação à cada classe do problema.

A figura 3.12 mostra um exemplo de arquitetura completa de uma CNN simples utilizada para classificação de imagens.

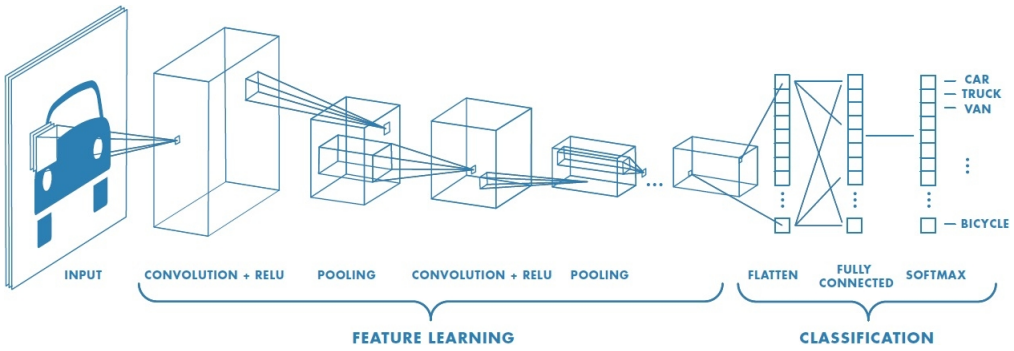


Figura 3.12: Arquitetura comum para uma CNN utilizada para reconhecimento de imagens. Retirado de [4].

Capítulo 4

Trabalhos Relacionados

O problema de reconhecimento de gêneros musicais não é uma tarefa simples nem para os seres humanos. Em um estudo sobre identificação de gêneros musicais [28], utilizando como amostra jovens universitários, Perrot mostrou que a acurácia de classificação para um problema de 10 gêneros musicais dada uma amostra de 3 segundos do áudio é de 70%. Mesmo para amostras de tempo maiores, a acurácia não mostrou melhoria significativa.

Um dos trabalhos mais clássicos da área de reconhecimento de gêneros musicais é o trabalho de Tzanetakis [9], onde o autor cria um dataset com 1000 músicas, distribuídas de forma balanceada em 10 gêneros musicais. Para realizar a classificação, são extraídas features de timbre, estrutura rítmica e de altura do conjunto de dados e então diversos classificadores são utilizados. O modelo com melhor desempenho é o **modelo de misturas de gaussianas**, com acurácia média de 61% com desvio-padrão de 4%.

Em 2003, utilizando o mesmo dataset criado por Tzanetakis, Li [29] utilizou novas features (conhecidas por "Daubechies Wavelet Coefficient Histograms", **DWCHs**) para realizar a tarefa de classificação. Então, utilizando uma SVM, Li reportou uma acurácia de 78.5% utilizando a técnica leave-one-out para validação do modelo.

Dentre os trabalhos que utilizam espectrogramas como imagens, podemos citar a contribuição de Costa em [30]. Técnicas de processamento de imagens são utilizadas no melspectrograma dos áudios, utilizando Padrões binários locais, e novamente uma SVM é utilizada como classificador. Neste trabalho outras bases de dados são utilizadas no processo de aprendizado e reconhecimento, como a ISMIR2004. A taxa de acurácia reporta chega aos 82% com desvio-padrão de 3,84%.

Alguns dos primeiros trabalhos de classificação de gêneros musicais utilizando redes neurais, utilizavam Deep Belief Networks para extrair features dos áudio e então utilizar um classificador para finalizar a tarefa. Em 2010, Hamel [31] propôs um sistema com essa arquitetura e obteve 77% de acurácia para o conjunto de dados do Tzanetakis.

Técnicas recentes utilizando Deep Learning também têm sido aplicadas ao reconhecimento de gêneros musicais. Em Lin Feng em [32] é proposta uma arquitetura de duas redes neurais, sendo uma delas uma CNN responsável por identificar padrões estruturais em uma janela de 3 segundos no áudio e padrões temporais nesta mesma janela, utilizando redes neurais recorrentes. No final uma rede neural fully-connected é utilizada como um classificador para determinar a que gênero musical aquela música pertence. Neste trabalho 3 conjuntos de dados diferentes são utilizados, sendo um deles o mesmo utilizado por Tzanetakis. A acurácia reportada no artigo é de 92%.

Capítulo 5

Metodologia

Nesta seção começamos a descrever a abordagem utilizada neste trabalho para resolver o problema de classificação de gêneros musicais, utilizando tanto a abordagem clássica de extração de features e utilização de um classificador, quanto a abordagem utilizando CNNs. Nosso objetivo é não somente comparar as acurácias dos métodos mas também compreender a vantagem e desvantagem de uma metodologia com relação à outra.

5.1 Conjuntos de Dados

Neste trabalho utilizamos dois conjuntos de dados para avaliar a performance de nossas metodologias. A primeira base de dados explorada foi a criada por Tzanetakis que se tornou uma referência na tarefa de classificação de gêneros musicais e que irá permitir que possamos comparar a nossa estratégia com a de outros autores que também utilizaram desta base.

A segunda base de dados é um dataset privado, onde foram coletadas amostras de músicas de uma aplicação de streaming. Apesar de não conseguirmos comparar diretamente os resultados obtidos nessa base de dados com outros trabalhos, podemos ter uma visão melhor de como os métodos que utilizam de Deep Learning se comportam com uma grande massa de dados.

5.1.1 GTZAN

O dataset conhecido como **GTZAN** foi construído por Tzanetakis e apresentado no seu artigo [9]. O conjunto de dados foi coletado entre anos 2000 e 2001, a partir de diversas fontes como CDs pessoais, rádio, gravações de microfone e outras variações.

O conjunto de dados consiste de 1000 arquivos de áudio no formato .au, tendo 30 segundos de duração cada. Os áudios estão distribuídos em 10 gêneros musicais, cada um contendo 100 músicas (ou seja, temos um problema balanceado). Cada

música possuí frequência de amostragem de $22050Hz$, um único canal (*mono*) e 16-bits. Este conjunto de dados está disponível na internet para download e utilização para fins acadêmicos. Os dez gêneros musicais presentes no conjunto de dados são: Metal, Disco, Clássica, Hiphop, Jazz, Country, Pop, Blues, Reggae e Rock.

5.1.2 Dataset privado

A segunda base de dados foi um dataset especial construído para este trabalho. Utilizando a *Web API* do Spotify, construímos uma ferramenta para pesquisar por músicas dado um conjunto de gêneros musicais e o número máximo de exemplos por categorias.

A ferramenta foi construída utilizando a linguagem de programação Javascript, através do interpretador NodeJS, onde podemos escrever código para rodar do lado do servidor e não somente no browser.

A ideia básica da aplicação é realizar múltiplos requests em paralelo para primeiro adquirir uma lista de músicas que pertencem a um gênero, limitado à 50 músicas por request pela API. Caso uma música apareça em mais de um gênero, escolhemos como sua categoria o primeiro resultado. Tendo a lista de músicas que serão usadas para compor o dataset, realizamos o download de uma prévia do áudio, que consiste de um arquivo *mp3* com 30 segundos do áudio, com taxa de amostragem de 22050 Hz e 16-bits. O download de cada prévia é feito de forma sequencial. Usando esta ferramenta, o dataset construído possuí 20000 arquivos, distribuídos sob as mesmas classes presentes no GTZAN, de forma balanceada, como podemos verificar na figura 5.1.

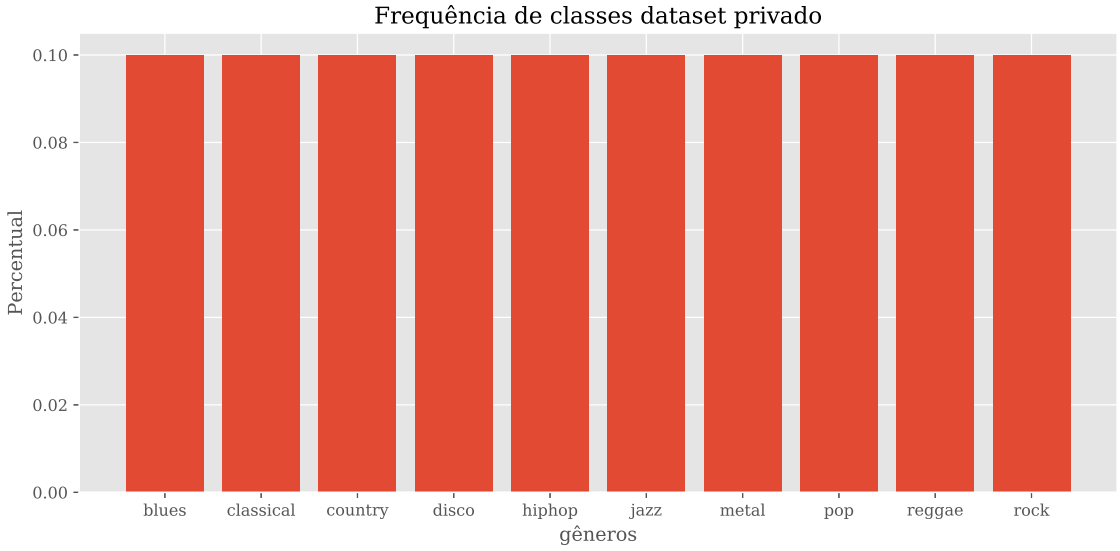


Figura 5.1: Contagem de elementos para cada classe do dataset privado.

5.2 Abordagem Clássica

A primeira abordagem discutida neste trabalho é a abordagem clássica que podemos subdividir em duas tarefas: Extração de features e classificação.

5.2.1 Construção das Features

Para cada arquivo de áudio foram extraídas um conjunto de métricas como discutido no capítulo 2. A ideia é extrair métricas a partir de pequenas janelas de áudio sobre todo o sinal, com tamanho de 1024 pontos (aproximadamente 50 milisegundos), para então calcular os momentos associados para cada uma dessas features, o que irá formar o nosso conjunto de dados. A única feature que é calculada sobre todo o domínio do áudio e por isso não determinamos os momentos é a feature de andamento.

As features de timbre utilizadas foram: Centróide, roloff espectral, fluxo espectral, RMSE, ZCR e 13 coeficientes MFCCs. A escolha de 13 coeficientes do MFCC se deu por três razões:

1. **Simplicidade**: Não queríamos um grande número de dimensões no modelo.
2. **Informação**: Por causa do DCT, grande parte da informação está presente nos primeiros coeficientes.
3. **Literatura**: Não existe uma regra para escolha do número de MFCCs, mas é comum encontrar um número entre 12 a 20 coeficientes em aplicações relacionadas à voz.

Um número maior de coeficientes foi testado e não houve ganho substancial de desempenho na tarefa.

Dessas features foram calculadas a média, o desvio-padrão, a curtose e obliquidade (skewness). A feature de andamento utilizada foi o tempo global. Ao final desse processo temos um total de $18 \times 4 + 1 = 73$ features, mais a variável dependente, que formam o nosso conjunto de dados. Para o GTZAN teremos um dataset com a forma 73 colunas e 1000 linhas, já para o dataset privado serão 73 colunas e 20000 linhas.

A linguagem de programação utilizada para extração das features, e também para a classificação e a abordagem por Deep Learning mais a frente, foi o Python, uma das linguagens mais populares na área de aprendizado de máquina. No Python encontramos um pacote chamado libROSA [33], utilizado para análise de áudio e música.

5.2.2 Classificação

Após a fase de extração de features, iniciamos o processo de classificação do dataset. A principal ideia é utilizar a técnica validação cruzada k-fold. Nessa técnicas subdividimos os dados em k subconjuntos, onde usamos $k-1$ conjuntos para treinamento e 1 para validação. O processo é repetido k vezes permitindo que cada subconjunto seja utilizado pelo menos uma vez como teste. Um importante detalhe é que neste trabalho a técnica foi utilizada de forma estratificada, ou seja, preservamos a distribuição original das classes de forma a não deixar o problema desbalanceado.

Para a validação dos modelos neste trabalho, utilizamos $k = 5$ folds. Para cada fold realizamos a normalização dos dados. É importante ressaltar que a média e desvio padrões utilizados para normalizar os dados de teste são os calculados para os dados de treinamento, evitando assim leaking de informação que poderia ocasionar uma falsa capacidade de generalização no modelo.

Os métodos de classificação utilizados nesta etapa do trabalho foram: regressão logística, árvore de decisão, random forest e SVM. A média e desvio padrão da acurácia para cada fold foi determinada.

Na regressão logística e na SVM foi utilizada regularização L2 nos modelos com o objetivo de diminuir o overfitting. Para a regressão logística utilizamos o parâmetro de regularização como $\lambda = 1$ e para a SVM $C = 2$, determinados de forma empírica através da otimização dos hiperparâmetros.

5.3 Abordagem por Deep Learning

A principal abordagem do trabalho consiste na utilização das CNNs. A ideia dessa abordagem é realizar a menor quantidade possível de pré-processamento nos dados e utilizar uma representação do áudio como entrada da rede. A abordagem utilizando deep learning também foi implementada utilizando o Python, com o framework para aprendizado profundo **Keras** e **Tensorflow**.

5.3.1 Representação

Como discutimos no capítulo 2, existem diferentes formas de se representar um áudio. O sinal no tempo, que é a representação mais usual, poderia ser diretamente o dado de entrada da rede neural. A única questão com essa abordagem é que, considerando um sinal de 30 segundos com 22050 Hz de taxa de amostragem, temos um vetor com 661500 entradas, que é um número bem grande e seria custoso utilizar essa representação.

Duas representações compactas do sinal puro são o espectrograma e o melspectrograma. Neste trabalho optamos por utilizar a representação através dos melspec-

trogramas pelo fato da escala Mel ser baseada na percepção da audição humana.

Na leitura dos arquivos, limitamos os vetores a possuírem um tamanho de 660000 pontos. Apesar de todos os arquivos terem aproximadamente 30 segundos de duração, devido à alguns milisegundos podem haver variações no número de pontos do arquivo original. O melspectrograma é calculado a partir dessa representação pura do sinal, onde utilizamos uma janela de tamanho 1024 pontos para a FFT do sinal, com hop-size de tamanho 512 (ou seja, 50% de sobreposição).

Dada a informação de que o ser humano consegue classificar o gênero musical com uma acurácia de 70% para um áudio de 3 segundos, como estudado por [28], decidimos também impor essa limitação à rede neural. Os melspectrogramas são divididos em janelas com tamanho de 3 segundos. Essas janelas possuem sobreposição de 50%, que é utilizado como uma técnica de ampliação dos dados. Uma das maiores dificuldades que encontramos ao treinar uma rede neural profunda em um conjunto de dados pequeno como o GTZAN é como lidar com o overfitting. Com essa ampliação dos dados foi possível diminuir este problema. Outras técnicas para redução do overfitting foram utilizadas, conforme descreveremos na seção de arquiteturas.

Ao tratar os melspectrogramas como imagens, também precisamos de uma dimensão para representar os canais. Para esta configuração o nosso conjunto de dados tem o seguinte formato: (*número elementos, altura, largura, canais*). No caso dos dados do GTZAN temos um conjunto de dados da forma $(19000, 128, 129, 1)$. Esses dados foram divididos utilizando a técnica de validação holdout, onde utilizamos 70% dos dados para treinamento e 30% para validação.

5.3.2 Arquiteturas

A escolha de uma arquitetura para uma CNN não é uma tarefa simples. Existem diversas combinações possíveis, algumas práticas comuns sobre a estrutura, mas nenhuma regra. O número de camadas, tamanho dos filtros e outras decisões são sempre hiperparâmetros que devemos ajustar em nosso modelo.

A ideia aqui é apresentar dois modelos com arquiteturas bem diferentes que foram testados neste trabalho. Para cada uma dessas arquiteturas, diferentes hiperparâmetros foram testados.

CNN 2D

A primeira arquitetura que realizamos experimentos foram com as CNN bidimensionais, tratando cada melspectrograma de entrada como uma imagem. A rede possuí 5 camadas convolucionais e uma única camada de saída totalmente conectada com 10 neurônios e o softmax para cálculo da probabilidade para cada classe. Um es-

quema da estrutura da rede pode ser visto na figura 5.2. Esta rede possuí um total de 171.594 parâmetros a serem ajustados.

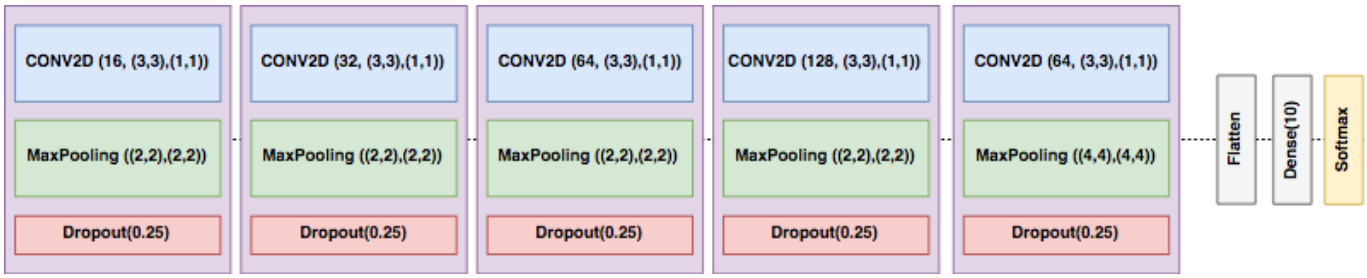


Figura 5.2: Arquitetura proposta para a CNN 2D

VGG16

Por último, utilizamos a ideia de replicar um modelo bem explorado na academia, o VGG16 [34]. Esta arquitetura é uma das mais famosas no conjunto das CNN, por ser uma arquitetura profunda e que obteve um ótimo desempenho no ImageNET [35]. A arquitetura do modelo pode ser vista na figura 5.3.

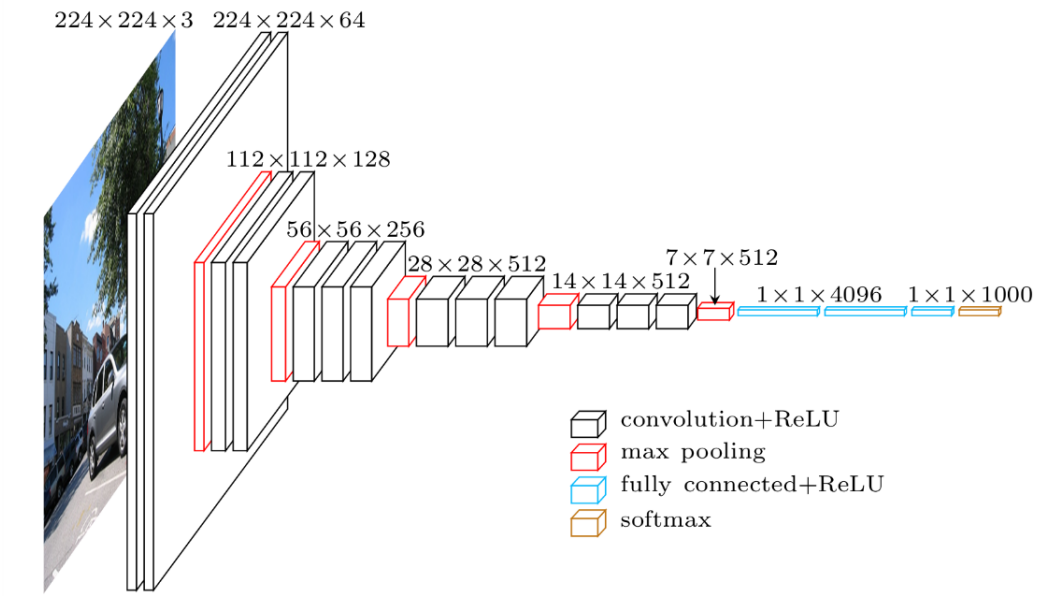


Figura 5.3: Arquitetura original do VGG16

Na prática, muitas vezes não treinamos um modelo do zero, com uma inicialização aleatória de pesos pois não possuímos um conjunto de dados suficientemente grande para treinar todos os pesos de forma a não ocorrer overfitting [2]. A ideia é utilizar um modelo que foi treinado em um conjunto de dados muito grande (como o ImageNET) e então utilizar as camadas convolucionais para extração de features

(processo conhecido como **transfer learning**) e remover as camadas totalmente conectadas do modelo original para treinarmos o nosso classificador com nossos pesos inicializados a partir dos originais da rede (conhecido como **fine-tuning**). É possível utilizar o processo de fine-tuning em toda a rede neural, até nas camadas convolucionais, mas é comum mantermos os pesos das primeiras camadas para evitar problemas de overfiting e pelo fato das primeiras camadas conterem somente filtros bem genéricos, como detectores de bordas, por exemplo.

No treinamento deste modelo mantivemos as primeiras 5 camadas convolucionais sem atualizar os pesos. As camadas restantes e a camada totalmente conectada que criamos, que possuí a forma (256, 10) neurônios, com a camada de saída Softmax, serão treinadas. Esta rede possuí um total de 16.702.090 parâmetros treináveis (e um total de 16.814.666).

5.3.3 Plataforma

Devido ao fato das operações matriciais serem tarefas altamente paralelizáveis em redes neurais, a utilização de uma plataforma com múltiplos núcleos gera um benefício muito grande de desempenho. A utilização de GPUs é um bom exemplo de tecnologia que nos permitiu o ganho de desempenho computacional nas redes neurais por possuírem um alto número de cores e implementarem de forma muito eficiente essas operações.

Neste trabalho utilizamos o serviço do Google Cloud Platform para alugar uma máquina com GPUs e tornar o processo de treinamento das redes muito mais ágil. Uma imagem padrão do Google para Deep Learning, com 52 GB de memória RAM e uma placa gráfica K80 possuí um custo médio de 1.2 dólares por hora e foi a escolhida neste trabalho por atender nossas demandas.

5.4 Comparações

Por fim, agora que vimos as duas abordagens utilizadas neste trabalho, podemos discutir sobre ambas. A abordagem clássica em geral é uma abordagem menos custosa computacionalmente do que a abordagem por deep learning, entretanto requer conhecimentos técnicos específicos de processamento de músicas.

A abordagem por deep learning possuí o ponto forte de que com apenas uma representação bruta do áudio (um melspectrograma), somos capazes de criar um sistema end-to-end para extrair as features através de operações de convolução e depois classificar com boa precisão.

Conforme veremos no capítulo 6, ambas as abordagens superam a acurácia humana para esta tarefa, mas os resultados para os modelos que utilizam CNNs são

melhores que os que utilizam a abordagem clássica, em geral.

Capítulo 6

Resultados

6.1 Experimentos no GTZAN

O primeiro experimento realizado no dataset GTZAN foi a abordagem clássica, como descrito no capítulo anterior. Na tabela 6.1 vemos a acurácia obtida pelos métodos utilizados neste trabalho.

Método	**Acurácia**	**Desvio-padrão**
Árvore de Decisão	0.502	0.03
Regressão Logística	0.700	0.013
Random Forest	0.708	0.032
SVM (RBF)	**0.762**	**0.034**

Tabela 6.1: Desempenho da abordagem clássica

O método que obteve o melhor desempenho nesta tarefa foi a SVM com kernel gaussiano (RBF). Classicamente a SVM é considerada um dos algoritmos mais poderosos para tarefas de aprendizado de máquina, principalmente quando estamos trabalhando com um conjunto de dados de alta dimensionalidade como é o nosso caso após a extração de todas as features. A matriz de confusão para a SVM pode ser vista na figura 6.1.

Já a abordagem utilizando CNNs, para ambas arquiteturas no dataset do GTZAN tivemos bons resultados como mostrado na tabela 6.2. Para garantir que não foi um resultado enviesado devido a separação dos dados, os modelos foram treinados e testados 3 vezes, de onde calculamos a média e o desvio-padrão.

Método	**Acurácia**	**Desvio-padrão**
CNN 2D	0.832	0.008
VGG16	**0.864**	**0.012**

Tabela 6.2: Desempenho da abordagem por Deep Learning

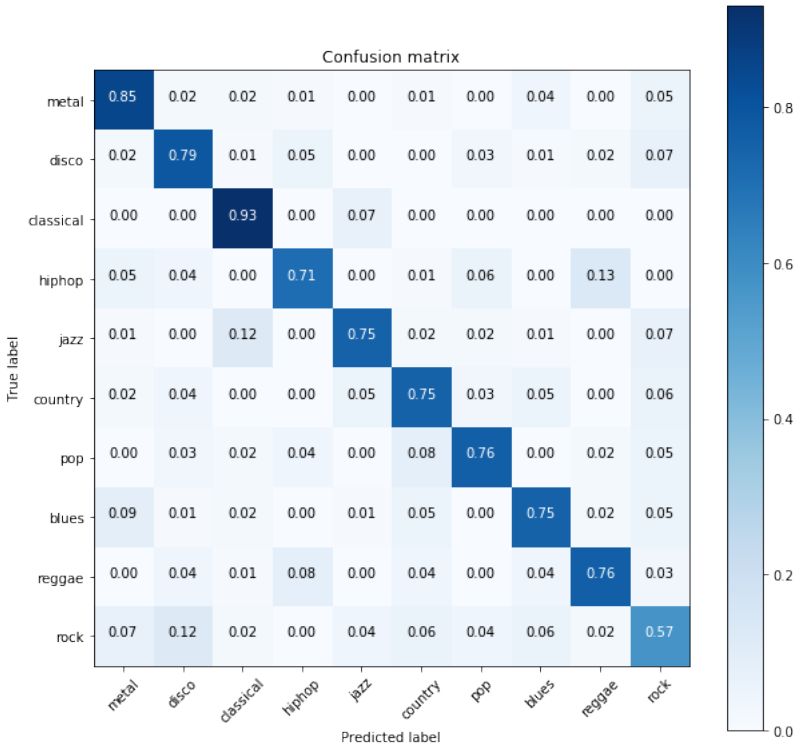


Figura 6.1: Matriz de confusão para a SVM, média dos 5-folds

Na figura 6.2 e na figura 6.3 encontramos os gráficos da função de custo e acurácia por época, para a CNN 2D e para a VGG16, respectivamente (Para a última execução do modelo). Neste dataset, para a CNN 2D, o modelo foi treinado por 50 épocas, onde cada época levou em média 15 segundos. Para a VGG16, treinada por 20 épocas, cada época levou em média 97 segundos.

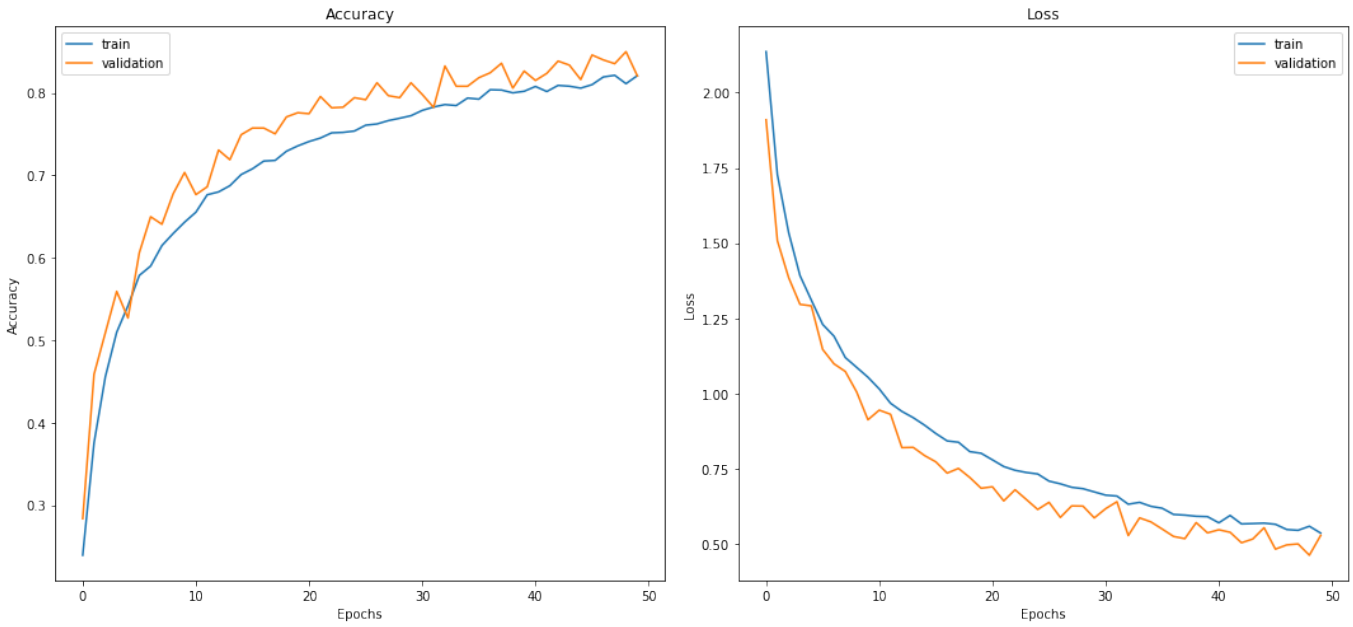


Figura 6.2: Acurácia e Função de custo por épocas para a CNN 2D no GTZAN

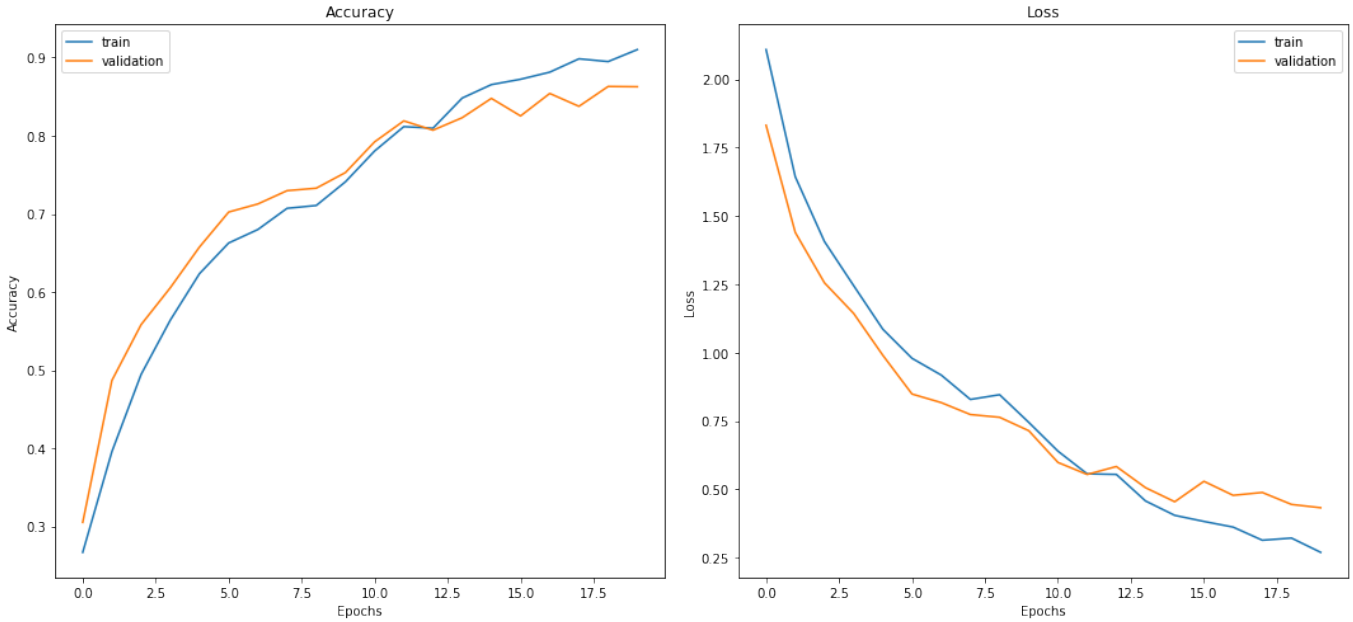


Figura 6.3: Acurácia e Função de custo por épocas para a VGG16 no GTZAN

Também podemos verificar a matriz de confusão para ambos os modelos nas imagens 6.4 e 6.5.

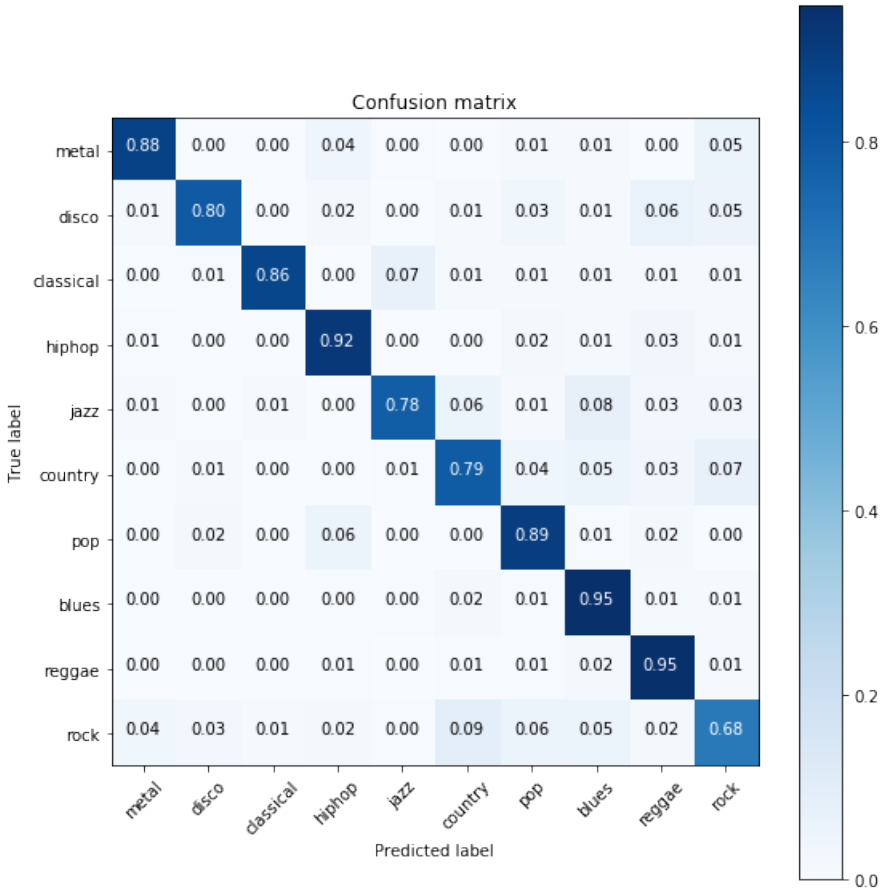


Figura 6.4: Matriz de confusão para a última execução do modelo CNN 2D

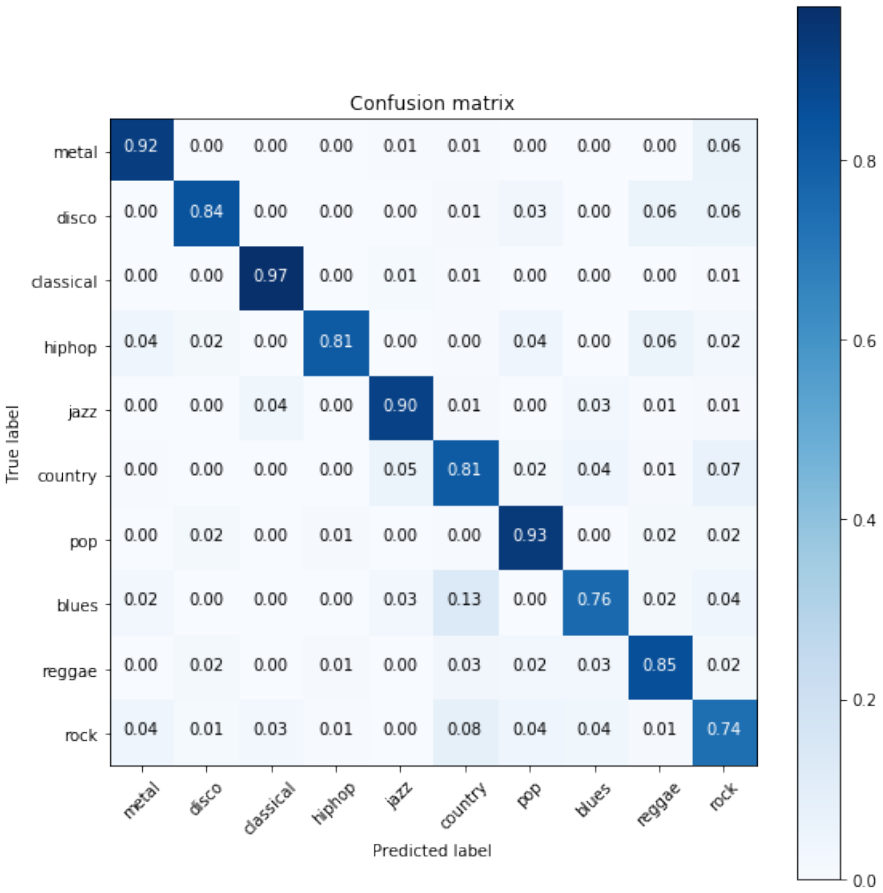


Figura 6.5: Matriz de confusão para a última execução do modelo VGG16

6.2 Experimentos no dataset privado

De forma análoga, primeiramente realizamos a abordagem clássica neste novo dataset. Na tabela 6.3 temos as acurácias obtidas pelos diferentes métodos utilizando validação cruzada com $k = 5$.

Método	**Acurácia**	**Desvio-padrão**
Árvore de Decisão	0.519	0.005
Regressão Logística	0.678	0.009
Random Forest	0.685	0.008
SVM (RBF)	**0.731**	**0.01**

Tabela 6.3: Desempenho da abordagem clássica no dataset privado

Novamente o método que obteve o melhor resultado foi a SVM. O desempenho do método quando comparado com a acurácia obtida no GTZAN pode ser explicado devido a uma maior variabilidade nos dados, pois temos agora uma amostra consideravelmente maior e gêneros musicais serem características subjetivas que mudam

todo o tempo. A matriz de confusão com uma média das execuções para o método de melhor acurácia pode ser visto na figura 6.6.

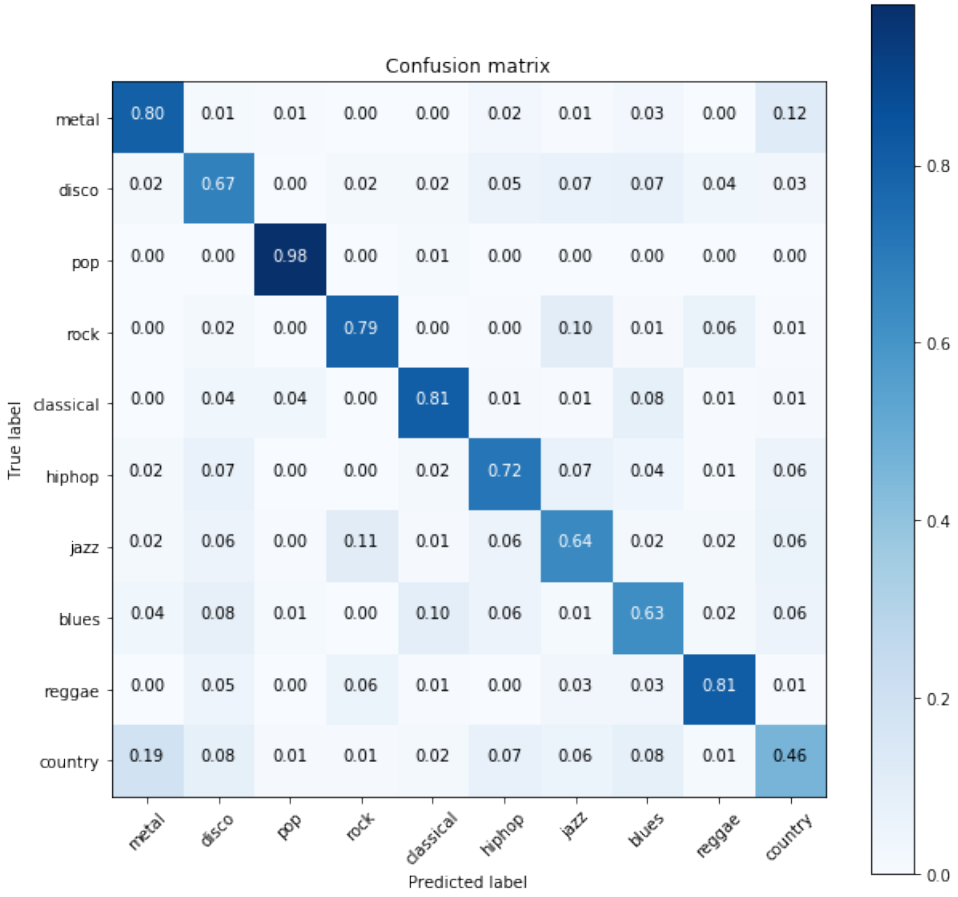


Figura 6.6: Matriz de confusão para a SVM no dataset privado

Também testamos os dois modelos de redes neurais convolucionais nessa base de dados. Por motivos de restrição no uso da plataforma, treinamos essas redes por uma quantidade similar de épocas, mas esses modelos poderiam ser treinados por mais tempo. Para a CNN 2D, o modelo foi treinado por 50 épocas, onde cada época levou em média 205 segundos. Já para a VGG16, treinada por 10 épocas, cada época levou em média 53 minutos. É interessante observar pelas curvas de acurácia e função de custo da CNN 2D na figura 6.7 e na VGG16, na figura 6.8), que seria possível treinar o modelo por um número de épocas maior sem entrar em regime de overfitting. Na tabela 6.4 podemos verificar as acurácias obtidas pelos modelos em 3 execuções.

Método	**Acurácia**	**Desvio-padrão**
CNN 2D	0.713	0.017
VGG16	0.801	0.013

Tabela 6.4: Desempenho da abordagem por deep learning no dataset privado

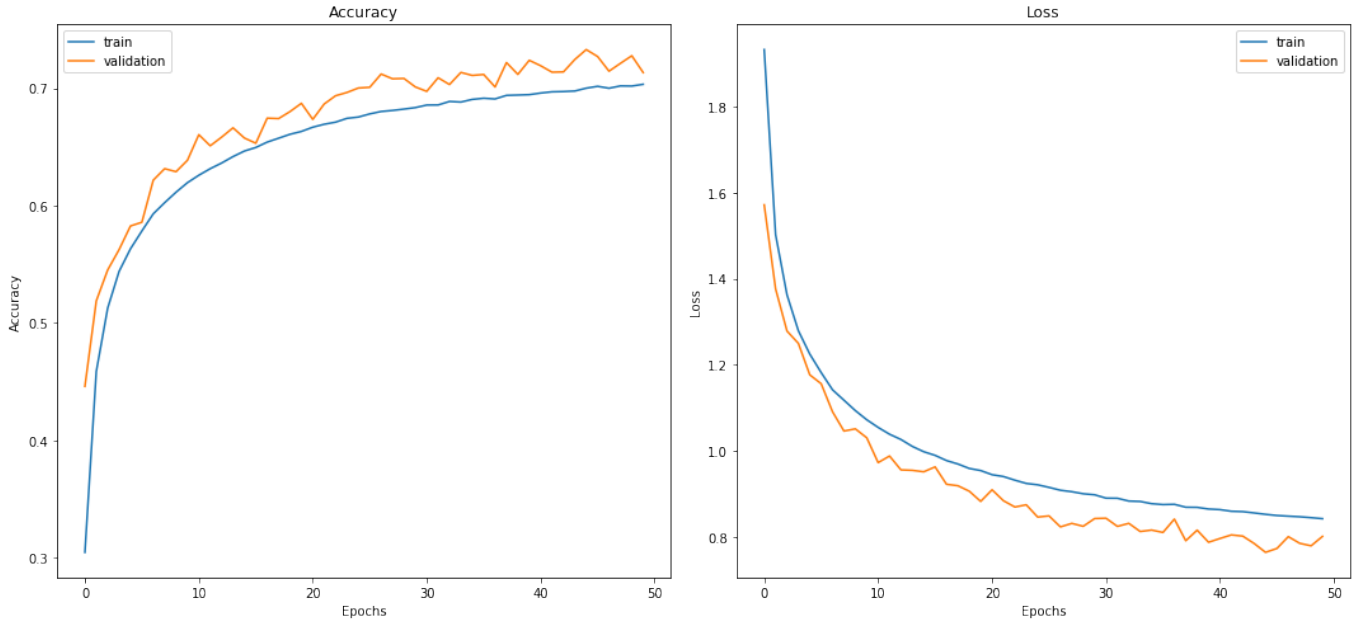


Figura 6.7: Acurácia e Função de custo para a CNN 2D no dataset privado

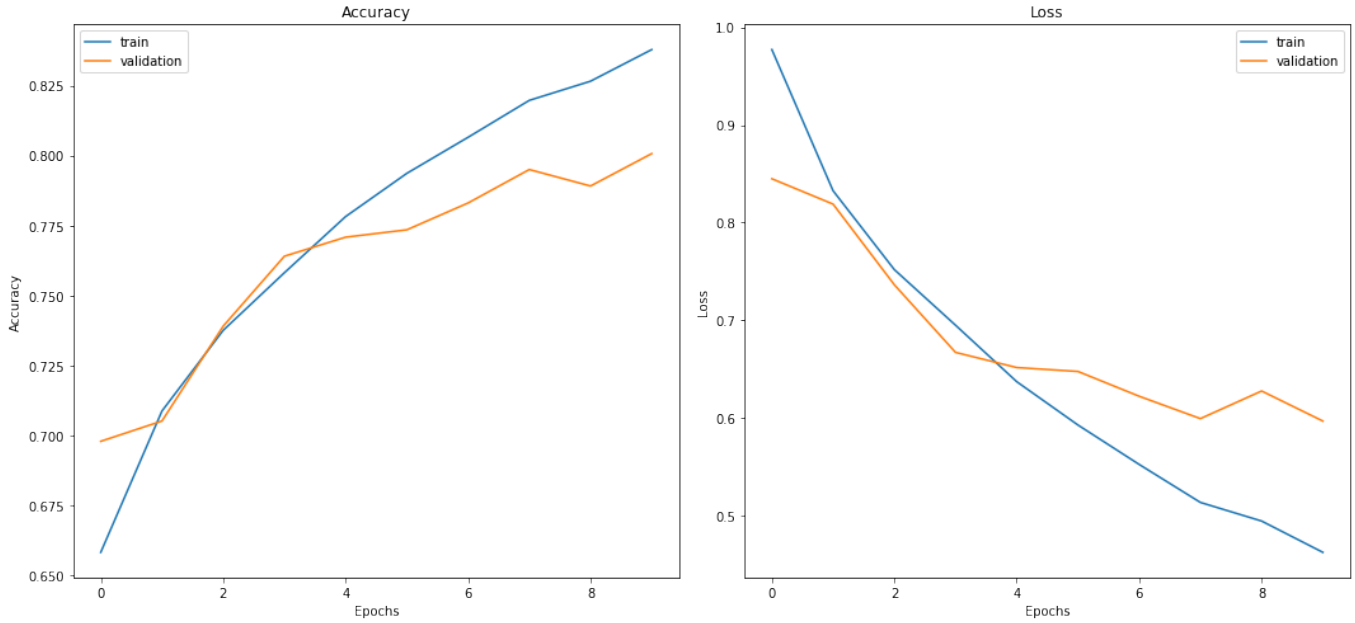


Figura 6.8: Acurácia e Função de custo para o VGG16 no dataset privado

Por último podemos verificar a matriz de confusão da última execução de cada modelo em 6.9 e 6.10.

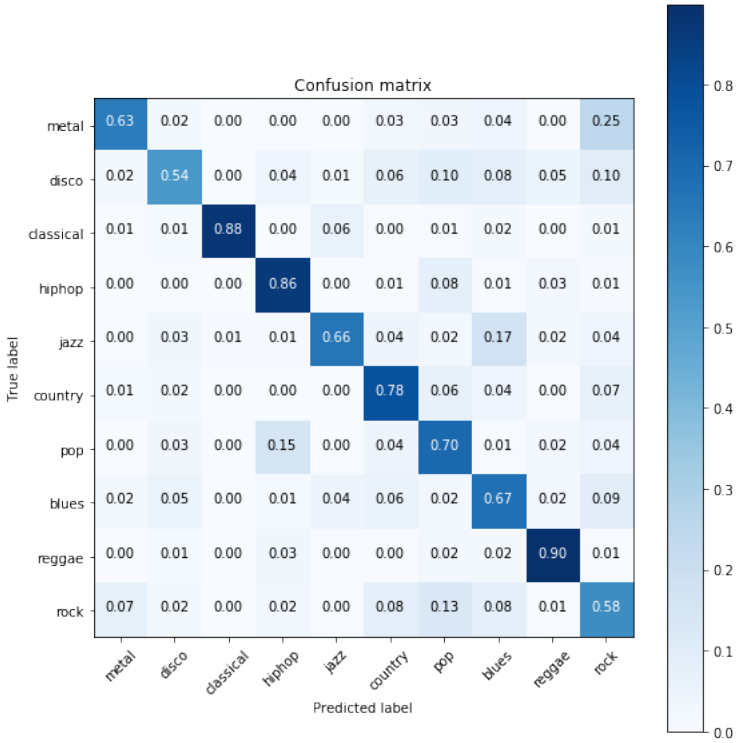


Figura 6.9: Matriz de confusão para a última execução do modelo CNN 2D no dataset privado

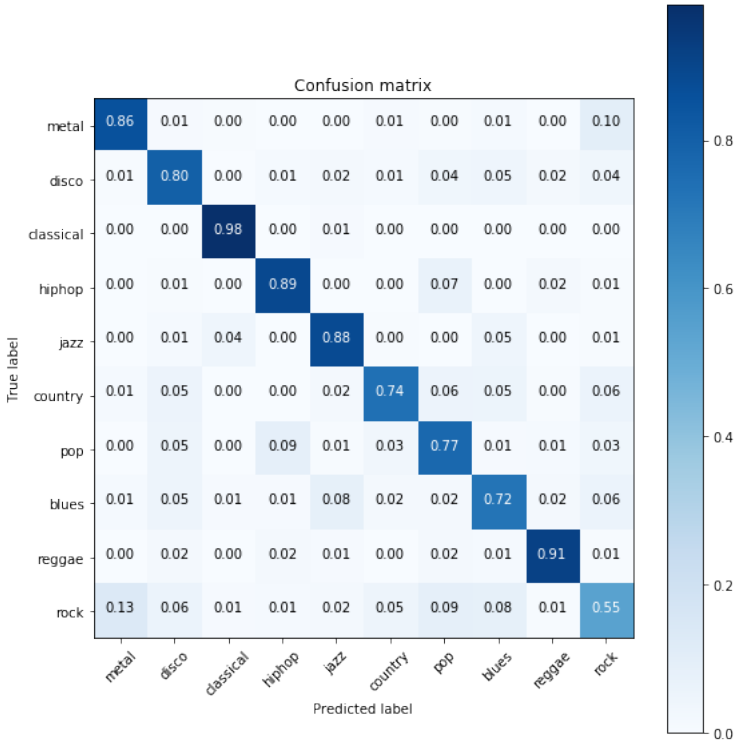


Figura 6.10: Matriz de confusão para a última execução do modelo VGG16 no dataset privado

Capítulo 7

Conclusão

Apresentamos nesse trabalho uma abordagem para a classificação de gêneros musicais usando as recentes técnicas de Deep Learning e comparamos com as abordagens mais clássicas de machine learning, não somente em aspectos de desempenho, mas também de facilidade de desenvolvimento e desempenho computacional.

Como discutido no capítulo 4, O desempenho humano para esta tarefa é de cerca de 70% para um áudio de 3 segundos. Mostramos neste trabalho que tanto a abordagem clássica quanto abordagens utilizando deep learning são capazes de superar essa acurácia.

A abordagem clássica de extração de features de forma manual e classificação também fornecem bons resultados, com o modelo da SVM atingindo uma acurácia de 76.2% no GTZAN e 73.1% no dataset privado. Entretanto essa abordagem requer uma expertise na área do problema, o que nem sempre é fácil ou possível.

A abordagem utilizando deep learning nos mostra que é possível criar um sistema end-to-end para realizar a tarefa, sem a necessidade de um pré-processamento no áudio por parte de um especialista. A acurácia também dos métodos utilizando esta abordagem é maior do que a abordagem clássica exposta (e maior que as referências que também utilizam a abordagem clássica). Nosso melhor modelo, utilizando transfer-learning na arquitetura VGG16 atingiu uma acurácia 86.3% no GTZAN. O melhor modelo para o dataset privado foi o VGG16, com acurácia média de 80.1%.

Como trabalhos futuros, seria interessante testar esta abordagem por Deep Learning para um conjunto maior de gêneros musicais, para que seja possível explorar a hierarquia entre gêneros em que existe uma maior sobreposição. Além disto, com as CNNs estamos apenas explorando as características estruturais do áudio e não trabalhando com a dimensão temporal. A utilização de redes neurais recorrentes em conjunto com as CNNs, como proposto em [32] poderia aumentar a acurácia do método. Obter um conjunto maior de dados e utilizar outras arquiteturas mais profundas também seria importante.

Referências Bibliográficas

[1] SCIKIT LEARN. "Support Vector Machines". . http://scikit-learn.org/stable/modules/svm.html, Aug. 2018. Acessado em Agosto de 2018.

[2] FEI-FEI LI ET AL. "CS231n: Convolutional Neural Networks for Visual Recognition". . http://cs231n.github.io/, Jan. 2018. Acessado em Janeiro de 2018.

[3] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] SOMIN WADHWA. "Building Your First ConvNet". . https://blog.floydhub.com/building-your-first-convnet/, Dez. 2017. Acessado em Agosto de 2018.

[5] BEN SISARIO AND KARL RUSSELL. "In Shift to Streaming, Music Business Has Lost Billions". . https://www.nytimes.com/2016/03/25/business/media/music-sales-remain-steady-but-lucrative-cd-sales-decline.html?mcubz=1, Mar. 2016. Acessado em Setembro de 2017.

[6] KEUNWOO CHO, GYÖRGY FAZEKAS, KYUNGHYUN CHO AND MARK SANDLER. "A Tutorial on Deep Learning for Music Information Retrieval". . https://arxiv.org/pdf/1709.04396.pdf, Sep. 2017.

[7] PASQUALE LOPS, MARCO DE GEMMIS AND GIOVANNI SEMERARO. "Content-based Recommender Systems: State of the Art and Trends". , Feb. 2009.

[8] TAO LI AND GEORGE TZANETAKIS. "Factors in automatic musical genre classification of audio signals". , Oct. 2003.

[9] GEORGE TZANETAKIS AND PERRY COOK. "Musical genre classification of audio signals". , Nov. 2002.

[10] NUSSENZVEIG, H. M. *Curso de Física Básica 2*. Editora Edgard Blücher, 1996.

[11] LERCH, A. *An Introduction to Audio Content Analysis Applications in Signal Processing and Music Informatics*. A JOHN WILEY SONS, 2012.

[12] BREGMAN, A. S. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[13] PACIFIC NORTHWEST SEISMIC NETWORK. "What is a Spectrogram". . `https://pnsn.org/spectrograms/what-is-a-spectrogram`.

[14] ALAN W BLACK. "Class notes from CMU rocesSpeech Psing - Fall 2011". . `http://www.speech.cs.cmu.edu/15-492/`.

[15] ULISSES DA ROCHA FIGUEIREDO. "Uma Abordagem Visual para Classificação de Gêneros Musicais Utilizando Pontos-Chave de um Espectrograma". , 2017.

[16] KARA ROGERS. "What's the Difference Between Tempo and Rhythm?" . `https://www.britannica.com/story/whats-the-difference-between-tempo-and-rhythm`. Acessado em Julho de 2018.

[17] MEINARD MÜLLER, FRANK KURTH AND PETER GROSCHE. "Cyclic tempogram - A mid-level tempo representation for music signals". , 2010.

[18] DE HOUWER, J., BARNES-HOLMES, D., MOORS, A. "What is learning? On the nature and merits of a functional definition of learning", *Psychonomic Bulletin & Review*, v. 20, n. 4, pp. 631–642, Aug 2013. ISSN: 1531-5320. doi: 10.3758/s13423-013-0386-3. Disponível em: <`https://doi.org/10.3758/s13423-013-0386-3`>.

[19] MITCHELL, T. M. *Machine Learning*. 1 ed. New York, NY, USA, McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

[20] TREVOR HASTIE, R. T., FRIEDMAN, J. *The Elements of Statistical Learning*. Springer, 2009.

[21] GARETH JAMES, DANIELA WITTEN, T. H., TIBSHIRANI, R. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017.

[22] DAN BONEH, ANDREW NG. "CS229: Machine Learning". . `http://cs229.stanford.edu/`, Aug. 2018. Acessado em Agosto de 2018.

[23] MCCULLOCH, W. S., PITTS, W. "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, v. 5, n. 4, pp. 115–133, 1943.

[24] RUDER, S. “An overview of gradient descent optimization algorithms”, *CoRR*, v. abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>.

[25] KINGMA, D. P., BA, J. “Adam: A Method for Stochastic Optimization”, *CoRR*, v. abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

[26] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. “Dropout: a simple way to prevent neural networks from overfitting”, *The Journal of Machine Learning Research*, v. 15, n. 1, pp. 1929–1958, 2014.

[27] LECUN, Y., BOTTOU, L., BENGIO, Y., et al. “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, v. 86, n. 11, pp. 2278–2324, 1998.

[28] PERROT, D. “Scanning the dial: An exploration of factors in the identification of musical style”, *Proc. of ICMPC 1999*, 1999.

[29] LI, T., OGIHARA, M., LI, Q. “A comparative study on content-based music genre classification”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 282–289. ACM, 2003.

[30] COSTA, Y. M., OTHERS. “Reconhecimento de gêneros musicais utilizando espectrogramas com combinaçao de classificadores”, .

[31] HAMEL, P., ECK, D. “Learning Features from Music Audio with Deep Belief Networks.” In: *ISMIR*, v. 10, pp. 339–344. Utrecht, The Netherlands, 2010.

[32] FENG, L., LIU, S., YAO, J. “Music Genre Classification with Paralleling Recurrent Convolutional Neural Network”, *CoRR*, v. abs/1712.08370, 2017. Disponível em: <http://arxiv.org/abs/1712.08370>.

[33] MCFEE, B., RAFFEL, C., LIANG, D., et al. “librosa: Audio and music signal analysis in python”. In: *Proceedings of the 14th python in science conference*, pp. 18–25, 2015.

[34] SIMONYAN, K., ZISSERMAN, A. “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *CoRR*, v. abs/1409.1556, 2014. Disponível em: <http://arxiv.org/abs/1409.1556>.

[35] DENG, J., DONG, W., SOCHER, R., et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*, 2009.