

UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Marcos Pedro Ferreira Leal Silva

Detecção de Instrumentos Musicais com Redes Neurais
Profundas

Niterói-RJ

2017

MARCOS PEDRO FERREIRA LEAL SILVA

Detecção de Instrumentos Musicais com Redes Neurais Profundas

Trabalho submetido ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Prof. Aline Marins Paes Carvalho

Niterói-RJ

2017

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

S586 Silva, Marcos Pedro Ferreira Leal
Detecção de instrumentos musicais com redes neurais profundas /
Marcos Pedro Ferreira Leal Silva. – Niterói, RJ : [s.n.], 2017.
54 f.

Projeto Final (Bacharelado em Ciência da Computação) –
Universidade Federal Fluminense, 2017.
Orientador: Aline Marins Paes Carvalho.

1. Rede neural. 2. Aprendizado de máquina. 3. Processamento de
sinais. 4. Instrumentação musical. I. Título.

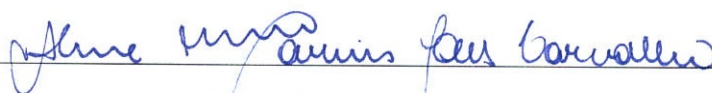
CDD 006.3

MARCOS PEDRO FERREIRA LEAL SILVA

Detecção de Instrumentos Musicais com Redes Neurais Profundas

Trabalho submetido ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Aprovado por:



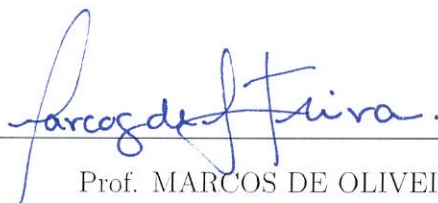
Prof. ALINE MARINS PAES CARVALHO, D.Sc. - Orientador

UFF



Prof. LUIS ANTONIO BRASIL KOWADA , D.Sc.

UFF



Prof. MARCOS DE OLIVEIRA LAGE FERREIRA , D.Sc.

UFF

Niterói-RJ

2017

*“Never been last,
but I’ve never been first
No I may not be the best,
but I’m far from the worst“*

.

(Avicii, Trouble)

Agradecimentos

Os agradecimentos principais são direcionados à Aline M. Paes que não podia ter se mostrado melhor orientadora e professora, minha mãe que sempre buscou o melhor para mim, Robledo Cabral e Vinícius Mendes que solucionaram inúmeros problemas em videoconferências. Um agradecimento especial a todos aqueles que contribuíram para a produção desse trabalho acadêmico diretamente com revisões e sugestões ou indiretamente com distrações e momentos divertidos.

Agradecimentos especiais são direcionados ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela oportunidade oferecida através do programa Ciências sem Fronteiras.

Resumo

Abordamos nesse trabalho sobre como classificar um instrumento dominante em um trecho de música usando redes neurais convolucionais. Introduzimos alguns conceitos de processamento de sinais digitais, abordamos o funcionamento da Transformada de Fourier e sua utilização para geração de espectrogramas que são a entrada do nosso algoritmo. Introduzimos o conceito de aprendizado de máquina e discutimos sobre a evolução das redes neurais e algumas construções famosas utilizadas no trabalho. Apresentamos três redes famosas utilizadas para a classificação (LeNet, AlexNet e GoogLeNet) e suas devidas arquiteturas e tarefas originais. Confrontamos as redes treinadas com diferentes conjuntos de dados gerados a partir de espectrogramas variando os parâmetros de tempo (100ms e 500ms) e escala de cor da potência do espectrograma (escala de cinza e escala de cores). Apresentamos os testes mais relevantes e comentamos o comportamento de cada rede e exibimos alguns de seus resultados. Comentamos sobre as escolhas de outros trabalhos da literatura e por fim resumimos nossas conclusões e apontamos trabalhos futuros para melhoria dos resultados. **Palavras-chave:** Aprendizado de Máquina, redes Neurais, processamento de sinais

Abstract

We will approach this work on how to classify a dominant instrument in a piece of music using convolutional neural networks. We introduce some digital signal processing concepts, we discuss the operation of the Fourier Transform and its use to generate spectrograms that are input to our algorithm. We introduce the concept of machine learning and discuss the evolution of neural networks and some famous architectures used in the work. We present three famous networks used for classification (LeNet, AlexNet and GoogLeNet) and their original architectures and tasks. We compare the trained networks with different data sets generated from spectrograms by varying the time scale (100ms and 500ms) and color spectrogram power scale (gray scale and color scale). We present the most relevant tests and comment on the behavior of each network and show some of its results. We commented on the choices of other papers in the literature and finally summarized our conclusions and pointed out future work to improve results.

Keywords: machine learning, neural networks, signal processing

Sumário

Resumo	vi
Abstract	vii
Lista de Figuras	xi
Lista de Tabelas	xii
1 Introdução	1
1.1 Motivação	1
1.2 Objetivos	2
1.3 Solução Proposta	3
1.4 Escopo	3
1.5 Organização do Texto	4
I Embasamento Teórico	5
2 Processamento de Sinais	6
2.1 Amostragem	6
2.2 Transformada de Fourier	7
2.2.1 Vazamento Espectral e Funções Janela	11
2.3 Representações de Áudio	13
2.3.1 Espectrogramas	13
3 Deep Learning	16
3.1 Aprendizado de Máquina	16
3.2 Redes Neurais	19

	ix
3.3 Deep Learning	22
3.3.1 LeNet	25
3.3.2 AlexNet	26
3.3.3 GoogLeNet	28
II Abordagem e Entradas	31
4 Metodologia Experimental	32
4.1 Amostras	32
4.2 Redes Neurais	35
5 Trabalhos Relacionados	40
6 Resultados	42
III Conclusões	48
7 Conclusões	49
7.1 Trabalhos Futuros	50

Lista de Figuras

2.1	Exemplos de <i>alias</i> com frequências de amostragem baixas	6
2.2	Funções Cosseno	9
2.3	Magnitude do resultado da Transformada de Fourier	10
2.4	Forma de Onda e DFT de um Dó 3 de um Piano	10
2.5	Secção sem Vazamento Espectral	11
2.6	Distorção dado um Vazamento Espectral	11
2.7	Funções Janela	12
2.8	Sobreposição de Janelas para Reconstrução	12
2.9	Espectrograma de Dó 3 de um Piano	13
2.10	Espectrograma de um sinal senoidal com três janelas de tempo	14
2.11	Comparações de quatro funções janela em um Dó 3 de um Piano	15
3.1	Modelo de Aprendizado de Máquina	17
3.2	Overfitting	18
3.3	Modelo de neurônio proposto por McCulloch & Pitts (1943)	19
3.4	Dados gerados a partir de portas OR, AND e XOR	20
3.5	Funcionamento de uma camada de convolução	23
3.6	Detecção de Bordas com Convolução	23
3.7	Camada de Pooling	24
3.8	Multilayer Perceptron	25
3.9	LeNet5	25
3.10	MNIST	26
3.11	AlexNet	27
3.12	Resultados do Teste da AlexNet	28
3.13	GoogLeNet e o sistema Inception	28
3.14	Bloco Inception com Redução de Dimensionalidade	29

6.1	Teste com Fur Elise (Piano) 100ms de Amostragem	44
6.2	Teste com Fur Elise (Piano) 500ms de Amostragem	45
6.3	Matrizes de Confusão da AlexNet	46

Lista de Tabelas

3.1	Valores Extraídos do Iris Dataset	18
4.1	Amostras Utilizadas	36
6.1	Acurácias Obtidas por cada Rede, a partir dos dados em Escala de Cinza .	45
6.2	Acurácia dos dados com Escala de Cor	47

Capítulo 1

Introdução

1.1 Motivação

Imagine estar andando na rua e ouvir uma música que lhe agrada bastante mas não se sabe o nome. Rapidamente você usa seu celular para captar os últimos segundos da música. Felizmente você foi capaz de encontrar a música mas ao escutar a música na internet descobre que a versão que você ouviu foi um cover, uma regravação de outro músico com algum instrumento específico. Sem ter certeza de qual instrumento estava tocando na versão que você ouviu sua busca se torna exaustiva por ter que ouvir diferentes versões de diferentes músicos.

Imagine agora um cientista forense da polícia federal analisando uma gravação de áudio quando uma música começa a tocar e atrapalhar a transcrição do áudio. Seria ideal para o policial ser capaz de detectar exatamente a versão da música tocada mas, como nem sempre isso é possível, o trabalho dele seria muito melhorado se ao menos ele fosse capaz de detectar os instrumentos utilizados para poder filtrar as frequências de cada um posteriormente e assim obter um áudio mais claro.

Essas são situações onde saber o instrumento tocado podia auxiliar a resolver o problema. Existe muito esforço na detecção de músicas por si com grandes grupos como SoundHound e Shazaam mas a vasta quantidade de regravações de músicos independentes torna o trabalho de ter todas as versões em seus bancos de músicas uma tarefa quase impossível. Mesmo grandes empresas de upload de áudio como YouTube e Soundcloud poderiam se beneficiar caso detectassem automaticamente uma versão regravada para organizar seus resultados baseados em instrumentos. Essa atividade poderia ser facilmente

executada por um perito ou músico com alguma experiência mas a quantidade de esforço necessária para tal tarefa assim como o retorno para as empresas em forma de produto que possam oferecer aos consumidores finais superam o preço e esforço que um algoritmo computacional poderia solucionar.

Nesse trabalho apresentamos uma possível solução para classificação de instrumentos musicais em uma dada janela de tempo. A solução apresentada aqui pode ser utilizado para otimizar buscas de músicas por softwares já existentes como as empresas citadas anteriormente ou servir de ferramental para outras áreas.

1.2 Objetivos

Nosso objetivo nesse trabalho é treinar um algoritmo de aprendizado de máquina, no caso redes neurais de convolução, para que seja capaz de classificar um instrumento dominante em um dado espaço de tempo. Nosso esforço é obter um modelo que se aproxime de uma possível solução matemática genérica suficiente para abordar o problema das mais diversas fontes e com os mais diversos instrumentos.

Trabalhos relacionados sugeriram soluções matemáticas utilizando-se propriedades estereofônicas da gravação [Barbedo e Tzanetakis 2011] (limitando gravemente o número de instrumentos que podem ser classificados) mas apresentaram resultados bastante limitados para serem aplicados cotidianamente. Soluções puramente matemáticas são extremamente frágeis a ruídos e mudança de fontes de áudios de gravação e meio de reprodução. Tendo isso em vista diversas pesquisas foram levantadas com algoritmos de aprendizado de máquina [Agostini, Longari e Pollastri 2003] [Herrera-Boyer P. Geoffroy 2003] usando estratégias de representação da música semelhantes às descritas nesse trabalho. Percebemos que esses trabalhos foram fortemente influenciados por algum conhecimento especialista e resultaram em soluções atreladas ao domínio [Herrera-Boyer P. Geoffroy 2003]. A preferência por um certo grupo de instrumentos ou a construção de um banco de dados com um instrumento tocando sozinho por amostras geram resultados ótimos em ambiente controlado mas nem sempre indicam um sistema genérico suficiente a ser aplicado em diversas situações. Exemplificamos essa situação com a adoção de dois conjuntos de dados: um com instrumentos tocados separadamente em ambiente controlado e outro com instrumentos dominantes em um certo tempo. Após a adoção de uma representação

para o áudio não adotamos qualquer medida fruto da observação especialista que pudesse otimizar o desempenho pois nosso intuito é demonstrar como o aprendizado de máquina com redes neurais convolucionais profundas apresenta excelentes resultados sem demais informações sobre o domínio.

1.3 Solução Proposta

Discorreremos ao longo desse trabalho sobre a utilização de espectrogramas com 100 e 500ms de janela de tempo para a geração de conjuntos de imagens que é utilizado como entrada para o treinamento de algumas redes neurais conhecidas que se utilizam de deep learning. Tratamos sobre como os parâmetros de tempo e diferentes parâmetros escolhidos influenciam o espectrograma. Introduzimos as redes neurais utilizadas (LeNet [LeCun Y. 1989], a AlexNet [Krizhevsky Ilya Sutskever 2012] e a GoogLeNet [Szegedy et al. 2014]) e explicamos a arquitetura delas e seus componentes utilizados.

Treinamos a rede para detectar diversos instrumentos avaliando seus respectivos espectrogramas e analisamos o desempenho da rede com dois parâmetros: um conjunto de espectrogramas de validação¹ e um conjunto de espectrogramas de músicas selecionadas². Verificamos os resultados com matrizes de confusão e histogramas na análise de trechos reais de música quando alimentados a essa rede. Apresentamos os testes mais relevantes nos resultados para então abordarmos as conclusões obtidas desses experimentos.

1.4 Escopo

Escolhemos nesse trabalho apenas algumas formas de representação de dados musicais com imagens, no caso espectrogramas e comparamos os resultados entre eles dada a grande quantidade de parâmetros possíveis para a especificação dos mesmos. Existem diversas outras formas de representação de dados musicais que não são abordadas nesse trabalho.

Trabalhamos com três redes neurais convolucionais já conhecidas e construídas para outras atividades. Não desejamos expor uma nova arquitetura ou modelo nesse trabalho

¹O subconjunto de validação consiste em entradas que não foram alimentados para a rede no momento de treinamento mas que são um subconjunto do conjunto de treinamento

²A esse conjunto de entradas genéricos damos o nome de conjunto teste

apenas constatar como o mesmo modelo pode ser reaproveitado para diferentes aplicações com o treinamento correto do mesmo.

1.5 Organização do Texto

Introduzimos na parte I o embasamento teórico básico para a compreensão do texto. São tratados nessa parte a questão da amostragem de sinais contínuos, suas frequências ideais para armazenagem, a transformada de Fourier e suas aplicações como o espectrograma. Discorremos sobre alguns dos parâmetros mais relevantes ao espectrograma como janela de tempo utilizada e funções janela para redução do vazamento espectral para a extração de informação útil do sinal trabalhado. Introduzimos aqui o conceito geral de aprendizado de máquina e redes neurais, exemplificando com as três redes utilizadas no projeto.

Na parte II tratamos dos procedimentos dos nossos experimentos detalhando nosso conjunto de dados e o conjunto de espectrogramas gerado para cada experimento. Contrastamos nossas escolhas para cada parâmetro com a literatura clássica e apontamos os objetivos de nossos experimentos. Tratamos separadamente do resultado de cada experimento e apresentamos os índices de acurácia e erro para a determinação da melhor rede e das melhores entradas. Definimos ainda como selecionaremos o melhor dos nossos experimentos para comparar os resultados com um segundo passo adicionando um SVM à rede para a classificação dos instrumentos.

Na parte III tratamos das conclusões obtidas com os experimentos realizados e discorremos sobre possíveis alterações para melhores resultados. Abordamos ainda possíveis modificações que podem ser efetuadas em trabalhos futuros para melhora e comparação dos resultados obtidos.

Parte I

Embasamento Teórico

Capítulo 2

Processamento de Sinais

O mundo como vemos consiste de diversos valores contínuos sendo constantemente avaliados pelos mais diversos sensores presentes no nosso corpo. Entretanto, a atual arquitetura nos nossos computadores não nos permite armazenar explicitamente uma entrada contínua de dados através de nenhum sensor. Por esse simples motivo todo sinal contínuo deve passar por um processo de amostragem para permitir que alguma computação seja efetuada.

2.1 Amostragem

Amostragem em processamento de sinais consiste em tomar um sinal contínuo em um número específico de amostras a cada período de tempo, de modo que a discretização desse sinal seja possível.

Entretanto devemos tomar cuidado ao escolher a frequência a qual a amostragem é feita para evitar efeitos indesejados como o *aliasing* ou o descarte de informações importantes.

Podemos ver na Figura 2.1 os efeitos de três frequência de amostragem muito baixa. No caso A, a senoide é amostrada uma vez

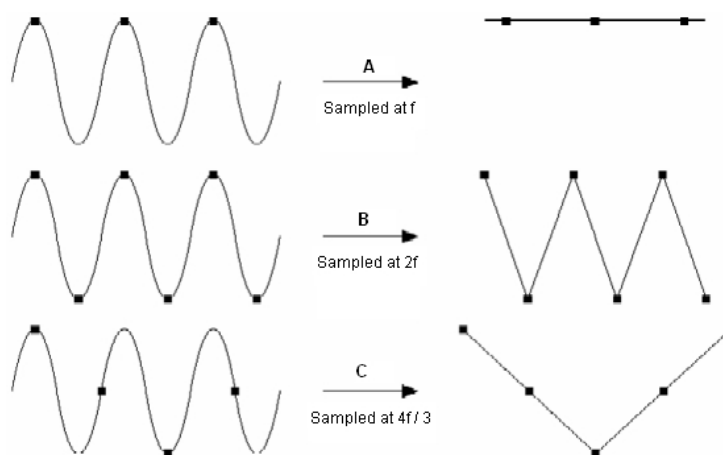


Figura 2.1: Exemplos de *alias* com frequências de amostragem baixas

a cada ciclo da onda, de modo que a onda amostrada parece constante. No caso B, as amostras foram tomadas na metade da frequência da onda resultando num sinal serrilhado e no caso C o valor de $3/4$ da frequência gerou o sinal em V ao lado. O *alias* é esse efeito que altera a aparência da onda amostrada que ocorre quando a frequência de amostragem não foi bem calculada. Uma boa frequência de amostragem deve ser tal de modo que a onda amostrada se assemelhe o suficiente com a onda que a gerou.

Som em particular é uma onda mecânica longitudinal com propagação circuncêntrica. A maior parte dos sons são composições de senóides puras cuja frequência é medida em hertz e potência em decibéis. Ambas as medidas são utilizadas no Sistema Internacional de medidas (SI). É sabido que seres humanos tem capacidade de ouvir frequências na faixa entre 20Hz e 20000Hz [Rossing 2007] [Rosen 2011]. Esse limite nos permite estabelecer uma frequência de amostragem de áudio eficiente tomando como base o teorema de Nyquist que determina que para amostrar um sinal sem perder nenhuma informação é necessário capturar o sinal a pelo menos duas vezes a frequência mais alta que se deseja capturar [Nyquist 2002] [Geerts e Sansen 2002]. Dado esse enunciado, frequências de pelo menos 40000Hz (40kHz) são suficientes para capturar áudio sem que o ouvido humano seja capaz de determinar nenhuma diferença entre o sinal capturado e o sinal original.

É comum entretanto encontrar outros valores como 44100Hz (44.1kHz) ou 48000Hz (48kHz) por algumas razões matemáticas e eletrônicas. O valor de amostragem de 48kHz é um valor múltiplo inteiro de 8kHz e 16kHz que já eram valores consagrados na indústria telefônica facilitando assim a integração entre circuitos. O valor de amostragem de 44.1kHz é herdado das primeiras gravações de áudio feitas com adaptações das gravadoras de vídeo, dado que as fitas cassete eram a única forma de armazenar tamanha quantidade de informação a época. Para atingir a maior quantidade de dispositivos adotou-se uma frequência que fosse múltiplo comum dos padrões de vídeo mais populares, o NTSC e o PAL [Watkinson 2000].

2.2 Transformada de Fourier

A transformada de Fourier [Fourier 1822] é uma função matemática que pode decompor qualquer sinal em um somatório infinito de senóides de diferentes amplitudes e

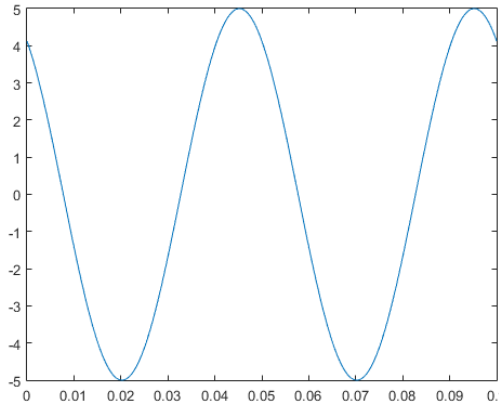
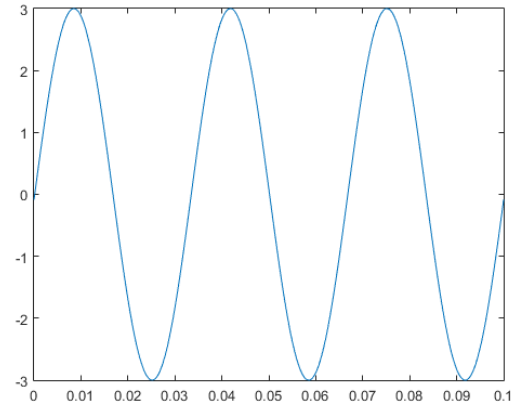
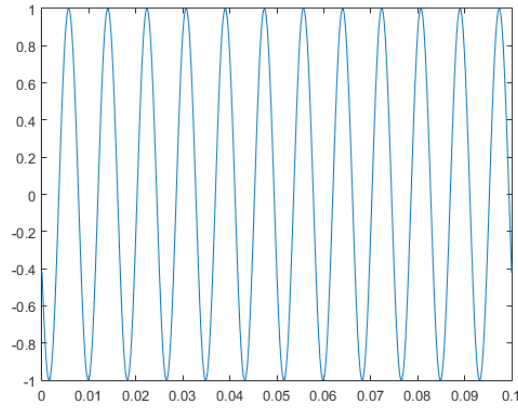
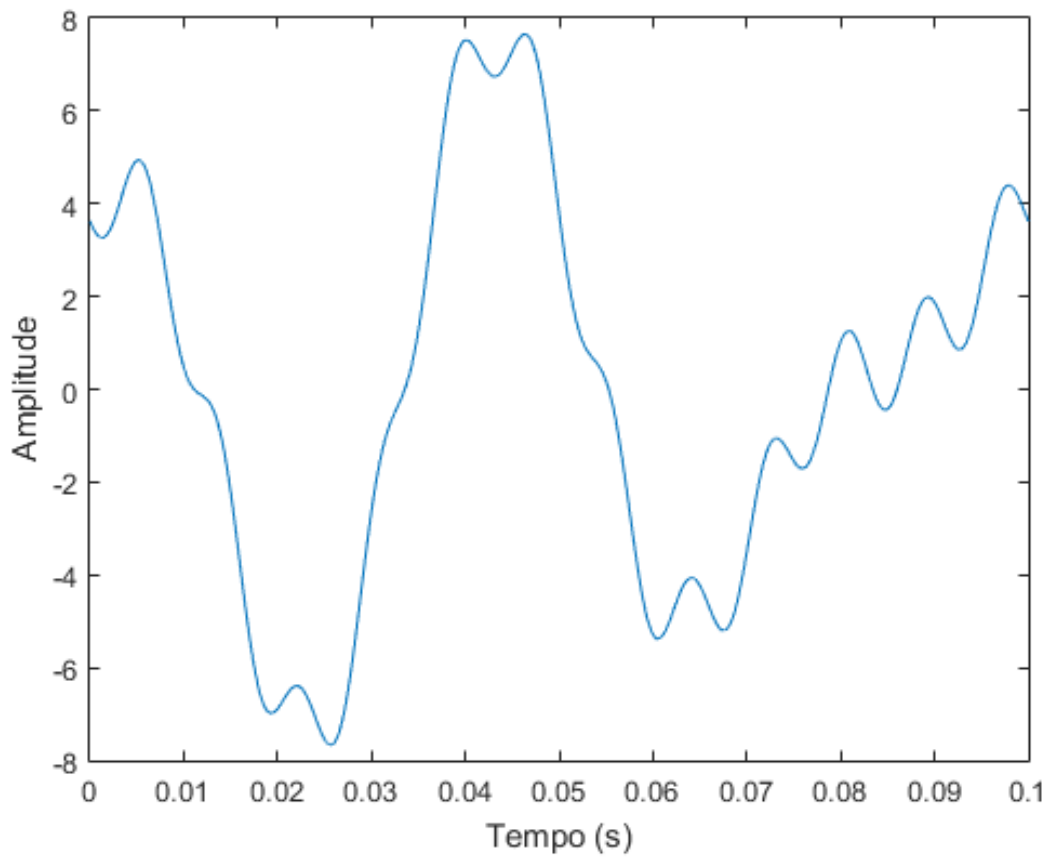
frequências. A transformada de Fourier é também chamada de representação do domínio de frequência do sinal já que ela quebra a onda em suas frequências múltiplas também chamadas de harmônicas. A transformada de Fourier é uma função que possui inversa, logo a partir da soma de todas as harmônicas geradas é possível gerar a onda original sem perda alguma.

O cálculo da transformada de Fourier nos apresenta uma soma infinita de harmônicos como já previamente estabelecido e isso é um problema para nossa atual arquitetura de computadores que suporta apenas valores discretos. Para isso, quando aplicamos a transformada a um sinal discretizado numa escala temporal, como por exemplo uma música amostrada a 44.1kHz, podemos definir uma somatória discreta da transformada através da construção 2.1.

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp\left(-j \frac{2\pi}{N} kn\right), k = 0, \dots, N - 1 \quad (2.1)$$

Transformada de Fourier Discreta (DFT) para um sinal k com N amostras

Para ilustrar o funcionamento da Transformada de Fourier tomaremos 3 sinais exibidos nas figuras 2.2a, 2.2b e 2.2c cada qual com amplitude, frequência e defasagem inicial próprias que serão somados em um sinal complexo visto em 2.2d. Aplicamos então a DFT nesse último sinal com uma amostragem de 44.1kHz por ser um valor de amostragem comum para áudio. A DFT nos retorna um valor complexo para cada amostra do sinal. Os valores reais da DFT correspondem a magnitude de cada frequência dos sinais que compõe o sinal original e as valores imaginários correspondem ao ângulos de defasagem de cada uma dessas frequências. Analisando a magnitude do sinal na imagem 2.3, podemos perceber claramente os três sinais que compõem nossa onda. Cada frequência específica do sinal que compôs a onda gerou um pico com amplitude proporcional a onda que foi gerado. O sinal com frequência de 120Hz possui menor amplitude e o sinal de 20Hz a maior amplitude. A defasagem inicial de cada sinal é armazenada na parte imaginária do sinal. Cada um dos picos possui valores 0.6, -1.6 e 2.0 para os valores de 20Hz, 30Hz e 120Hz respectivamente expressando a defasagem que confere com as equações descritas em cada onda.

(a) 20Hz: $f(t) = 5\cos(2\pi 20 + 0.6)$ (b) 30Hz: $f(t) = 3\cos(2\pi 30 - 1.6)$ (c) 120Hz: $f(t) = \cos(2\pi 120 + 2.0)$ 

(d) Soma das Ondas

Figura 2.2: Funções Cosseno

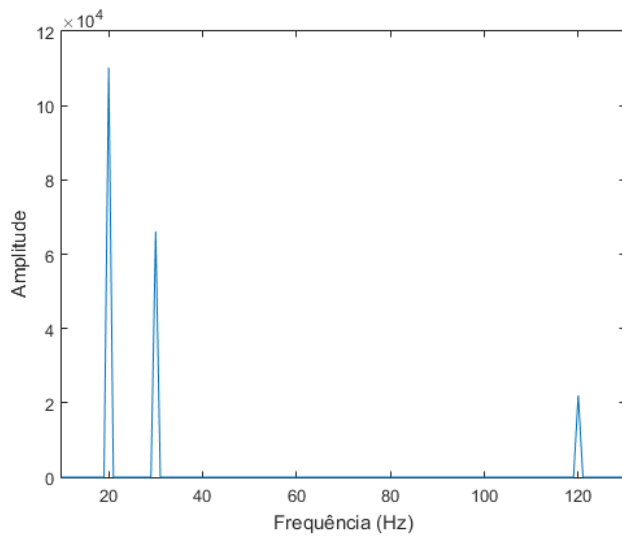
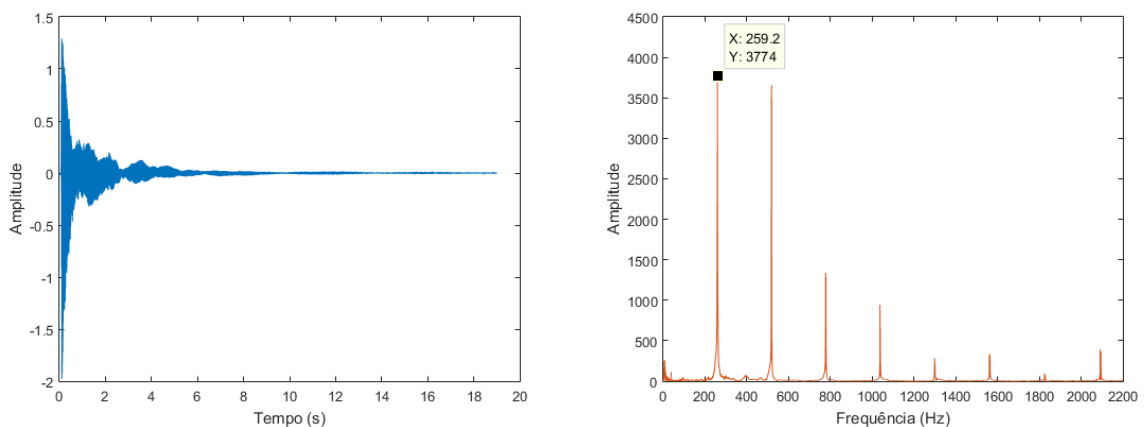


Figura 2.3: Magnitude do resultado da Transformada de Fourier

A maior amplitude da magnitude da DFT é chamada frequência base e é esse valor que determina qual nota musical um dado instrumento está tocando. Podemos ver, por exemplo, na figura 2.4b que a maior amplitude se encontra em 259Hz, que corresponde aproximadamente ao Dó Central no piano (C4) [Yeh C. 2010] [Lots S. 2008]. Analisando diferentes instrumentos tocando uma

mesma nota musical podemos perceber uma mesma frequência base mas diferentes amplitudes para as harmônicas. Essas características determinam o *timbre* e são as principais características para permitir a diferenciação única de cada instrumento entre diferentes grupos. Dependendo de quão grande sejam essas variações é possível determinar diferentes tipos de instrumentos de um mesmo tipo (como violinos e violoncelos) ou mesmo o mesmo instrumento com diferentes tipos de afinação.



(a) Forma de Onda de um Dó 3 em um piano

(b) Magnitude da DFT e harmônicos

Figura 2.4: Forma de Onda e DFT de um Dó 3 de um Piano

2.2.1 Vazamento Espectral e Funções Janela

A transformada de Fourier, no entanto, deve ser cuidadosamente interpretada. A DFT assume que o sinal de entrada é periódico, o que nem sempre ocorre, e mesmo sinais que são originalmente periódicos, como uma onda senoidal, podem apresentar comportamentos inesperados caso a janela de tempo analisada não corresponda a frequência exata (ou algum múltiplo inteiro) da

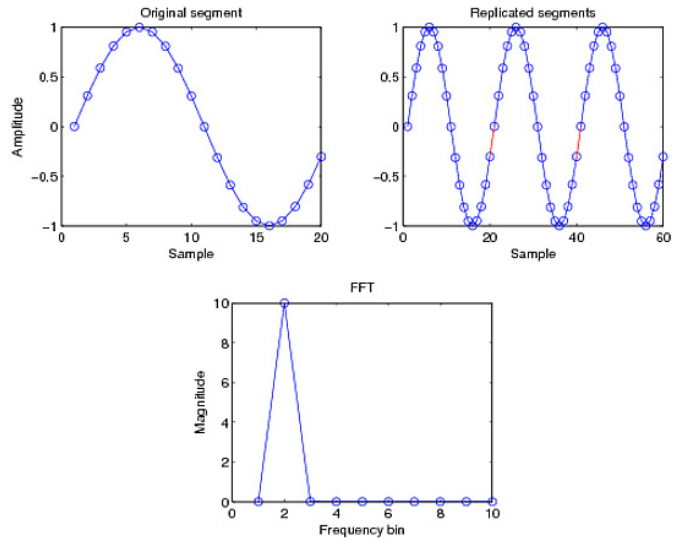


Figura 2.5: Secção sem Vazamento Espectral

onda. Podemos ver nas figuras 2.5 e 2.6 na imagem a esquerda o segmento seccionado, na imagem a direita o segmento replicado várias vezes com o segmento em vermelho explicitando a interpolação entre os segmentos inicial e final do segmento seccionado e na parte de baixo o efeito no resultado da DFT. O vazamento espectral ocorre especialmente nos extremos da onda não periódica que quando replicados não reconstruem a onda original. Para reduzir esse tipo de efeito são utilizadas funções específicas antes da secção ser calculada na DFT, as funções janela.

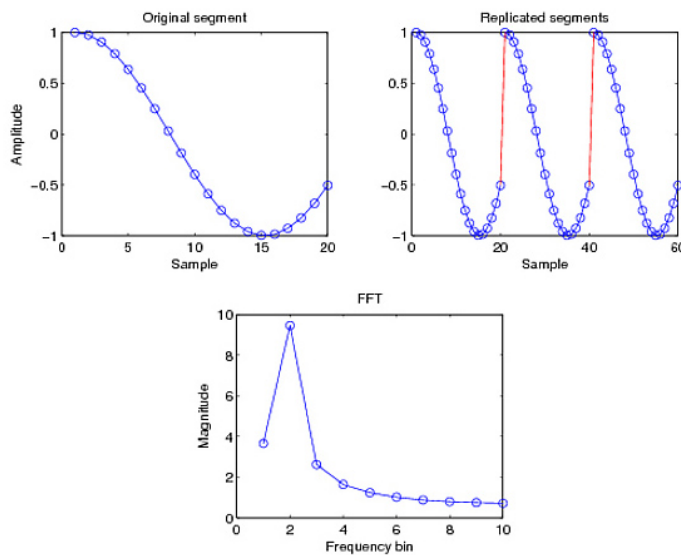


Figura 2.6: Distorção dado um Vazamento Espectral

As funções janela tem como intuito reduzir a possível diferença de valores entre diferentes extremos seccionados de uma dada onda. Uma função janela nada mais é do que uma sequência de valores que é multiplicado pelo sinal que se deseja seccionar e efetuar operações para somar posteriormente com a menor perda possível de conteúdo. Existem diversas funções janelas famosas, cada

uma com aplicações distintas. Das mais famosas podemos citar a janela retângulo, Hamming, Hanning e Black-Harris, que podem ser vistas na imagem 2.7. Podemos perceber que as funções janela são simétricas e tendem a valores próximos a zero nos extremos. Essa tendência a zero é o fator principal para redução do vazamento espectral e é o que garante a maioria essa aparência de sino.

Cada função tem propriedades e aplicações espectrais específicas com as diferentes configurações de pico de e limites laterais para o intervalo em que a janela está definida [Harris 1978]. A utilização de uma função janela resulta na perda de informação dos extremos da seção analisada. Para solucionar isso, utilizamos janelas de tempo com passos que gerem uma sobreposição suficiente para reconstruir a onda suficientemente bem para a aplicação desejada. Podemos ver um exemplo na imagem 2.8 onde tomamos janelas Hamming num espaço de 40 unidades e usamos sobreposição de 50%, resultando em três janelas. Somando as três temos a parte debaixo da mesma figura aonde podemos ver que os extremos ainda possuem perda de informação mas as ondas sobrepostas se aproximam bastante dos valores originais da onda. Quanto maior a sobreposição das seções, menor a diferença da onda original pra onda final calculada desviando o problema do vazamento espectral.

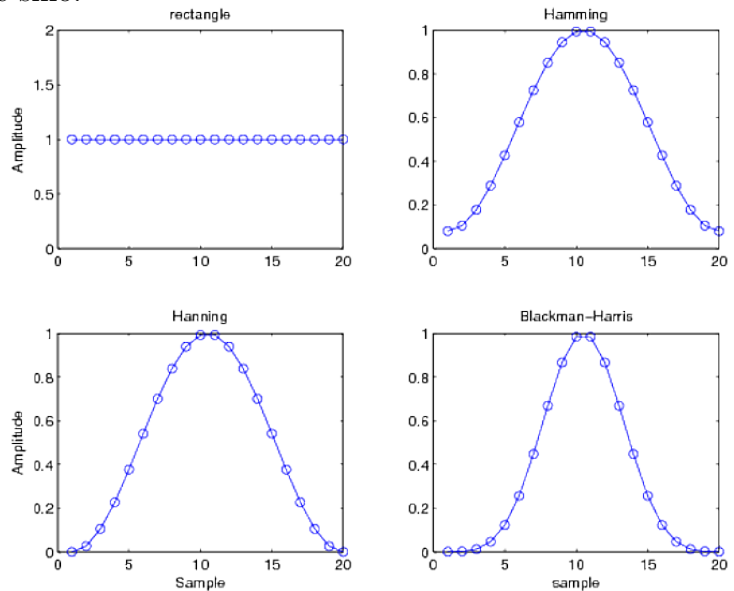


Figura 2.7: Funções Janela

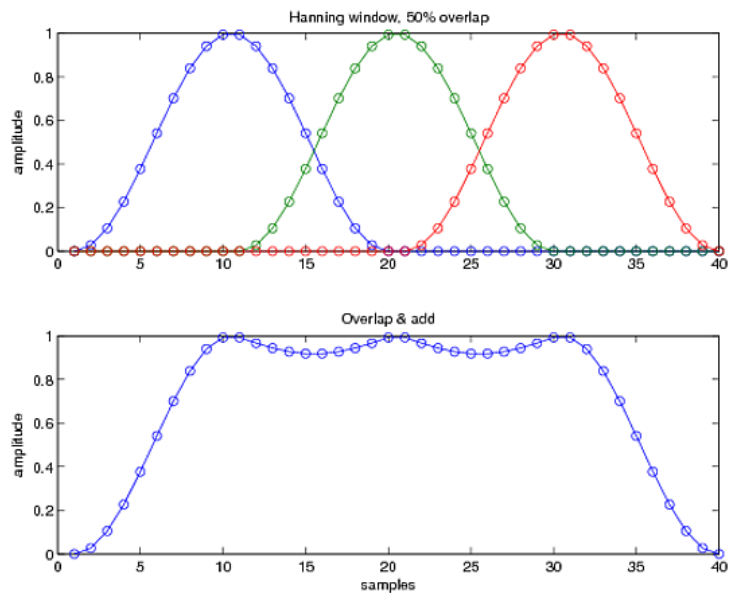


Figura 2.8: Sobreposição de Janelas para Reconstrução

Quando maior a sobreposição das seções, menor a diferença da onda original pra onda final calculada desviando o problema do vazamento espectral.

2.3 Representações de Áudio

Existem diversas características particulares que podem ser extraídas de áudio através dos mais diferentes processos. Adotamos nesse trabalho espectrogramas como forma de representar áudios por ser uma forma genérica, ou seja, não é direcionada a algum tipo específico de fonte ou de ambiente.

2.3.1 Espectrogramas

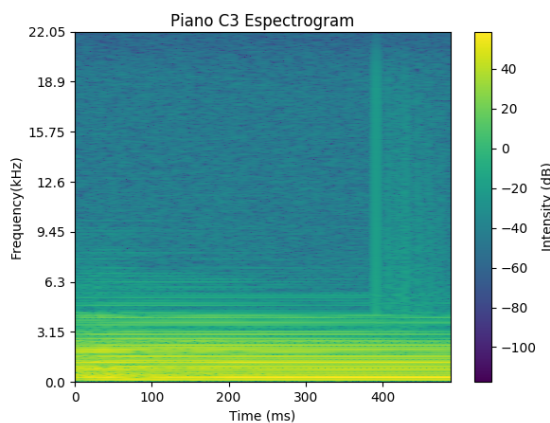


Figura 2.9: Espectrograma de Dó 3 de um Piano

de suas componentes, ou harmônicos, separadamente. Espectrogramas são uma abordagem popular para reduzir a dimensionalidade de sons dado a grande quantidade de amostras. Para ilustrar essa magnitude tomemos uma música de 3 minutos de duração com amostragem de 44.1kHz temos um total de $3min * 60s * 44100amostras/s = 7.938.000amostras$.

Um espectrograma é uma representação em três dimensões de uma janela de tempo de um certo sinal, como podemos ver na figura 2.9. No eixo horizontal representamos o tempo, no vertical as frequências produzidas e a intensidade da potência de cada frequência como a intensidade da cor conforme a barra de cor na legenda.

Existem diversos parâmetros para a extração de um espectrograma que forneça informações úteis como a escolha de uma função janela boa para a aplicação desejada, a escolha da largura da janela e sua sobreposição ou a decisão de qual escala de frequência deve ser utilizada. Existe um *trade-off* entre todos os parâmetros usados em um espectrograma. Podemos ver o impacto na resolução conforme a janela de tempo escolhida

Um espectrograma é construído em cima dos valores de cada uma das magnitudes de uma Transformada de Fourier Discreta em uma certa janela de tempo, resultando então em uma matriz que é representada graficamente como uma imagem.

A Transformada de Fourier decompõe uma onda em todas as suas componentes e é uma ferramenta poderosa para analisar sinais complexos e analisar cada uma

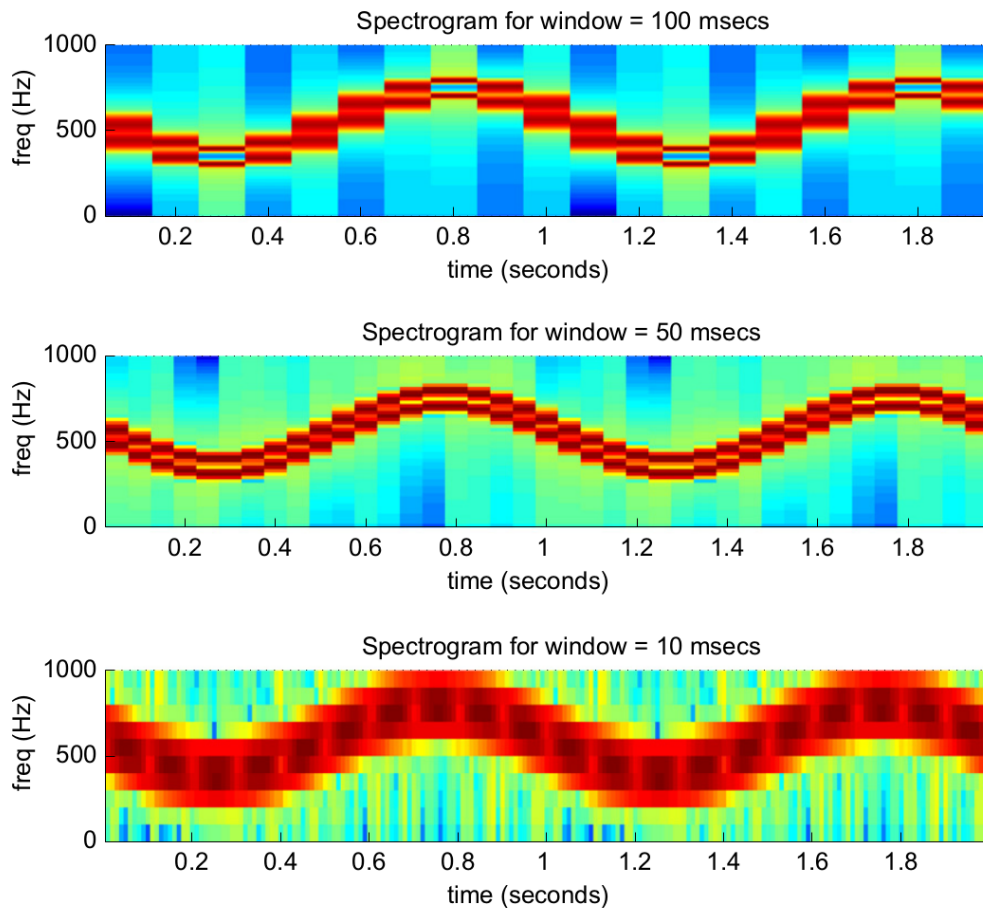


Figura 2.10: Espectrograma de um sinal senoidal com três janelas de tempo.¹

na figura 2.10. Na figura temos um sinal de 500Hz cuja frequência varia ao longo do tempo acompanhando a forma senoidal conforme podemos perceber nas três imagens. Na primeira janela a resolução do tempo é ruim, isso é, temos dificuldade de acompanhar a variação da frequência do sinal na escala de tempo resultando em vários degraus entre cada transformada. Na terceira imagem a resolução da frequência é ruim pois não conseguimos acompanhar a variação da frequência com precisão no eixo vertical. O melhor resultado obtido ocorre no segundo espectrograma.

Podemos observar ainda a influência de diferentes funções de janela utilizadas na figura 2.11. Os melhores parâmetros variam com particularidades do sinal de entrada. Idealmente, um bom espectrograma é capaz de representar a variação da frequência numa janela específica de tempo.

¹Reimpresso de "Introduction to Audio Analysis: A MATLAB Approach", 1a ed., Giannakopoulos, T., Pikrakis, A., Copyright (2014), com permissão da Elsevier.

Window Functions Comparison on Piano C3

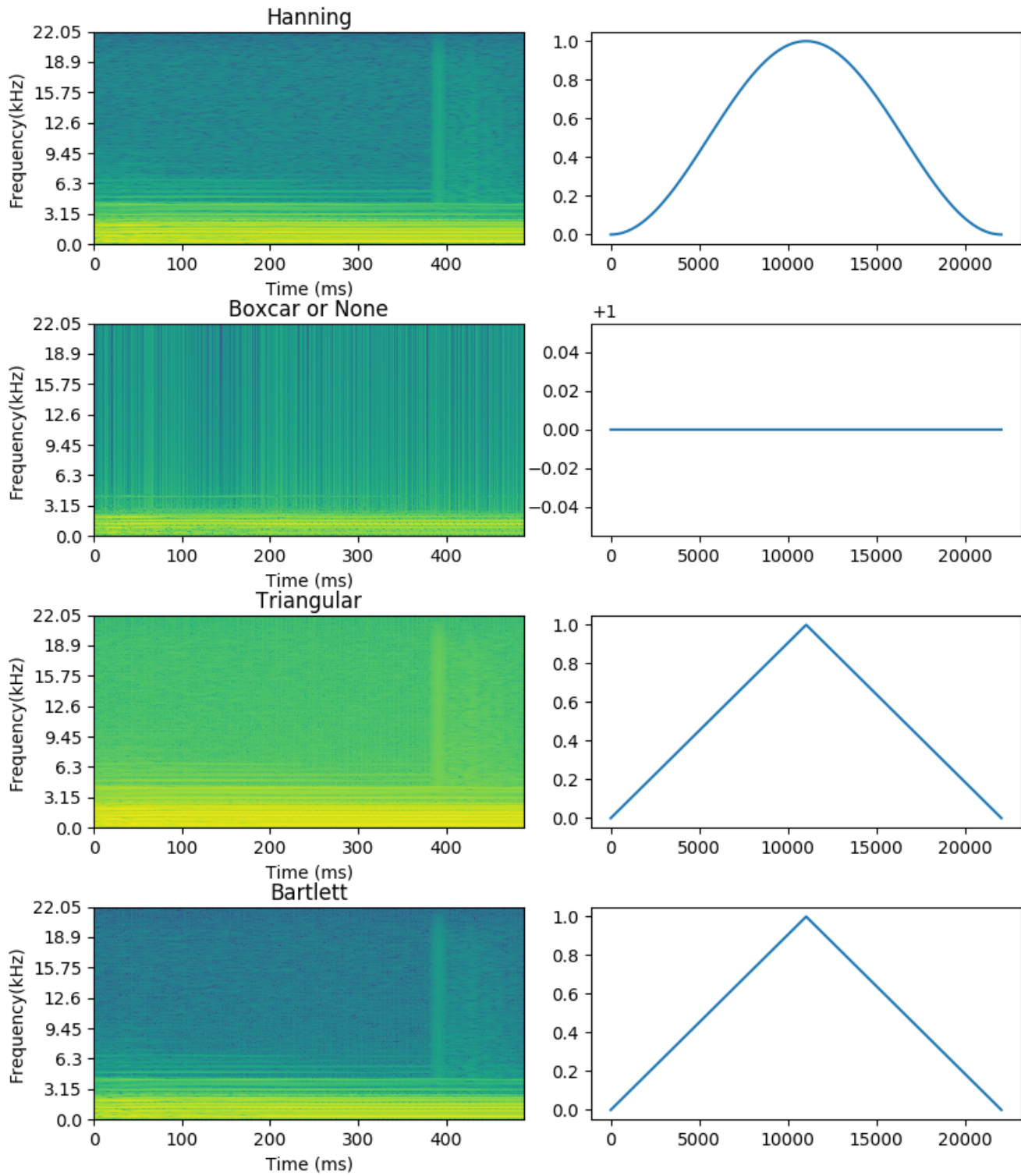


Figura 2.11: Comparações de quatro funções janela em um Dó 3 de um Piano

Capítulo 3

Deep Learning

3.1 Aprendizado de Máquina

Se tomarmos uma figura de uma árvore, por exemplo e perguntarmos a uma criança o que há na figura, provavelmente teremos uma resposta correta, ainda que a criança nunca tenha visto aquela árvore específica antes. Se perguntarmos a um adulto o que é uma árvore, no entanto, teremos uma definição muito mais difusa como resposta. Nosso aprendizado não se deu observando todas as árvores do mundo, nem tampouco definindo precisamente o que está presente em uma árvore. Nosso aprendizado foi apenas fruto da observação de árvores.

A descoberta de padrões permitiu a humanidade obter diversos sucessos ao longo de sua trajetória como a o movimentos dos astros por Kepler ou as leis da física por Newton. A evolução do reconhecimento de padrões associado com o nascimento da computação resultou no aprendizado de máquina como subcampo da inteligência artificial.

”Aprendizado de máquina dá aos computadores a habilidade de aprender sem serem explicitamente programados”(SAMUEL, A., 1959) através da construção de agentes que sejam capazes de aprender automaticamente baseado em suas próprias experiências efetuando a atividade sucessivamente tentando diferentes opções que podem levá-lo ao melhor resultado ou em observações de outros.

Uma forma de representação comum de um conjunto de dados é uma tabela onde cada coluna representa uma característica (*feature*) fornecida para a solução do nosso problema e cada linha expressa um conjunto de dados obtidos em uma amostra. Essas características podem assumir valores numéricos (inteiros ou reais) ou nominal (sendo esse

parte de um conjunto contável). Existem diversas técnicas para tratar cada tipo de valor e convertê-los para uma escala ideal ao algoritmo que se deseja utilizar caso seja necessário.

Podemos observar na figura 3.1 o esquema básico de um modelo para aprendizagem. No aprendizado de máquina desejamos obter um mapeamento das características fornecidas para uma certa atividade. Para isso, fornecemos um conjunto de dados que é a base de experiência a qual desejamos treinar o agente, representado pelo *Training Set* no modelo, que é fornecido ao algoritmo de aprendizado (*Learning Algorithm*). Esse algoritmo tenta encontrar uma aproximação da função real que originou os dados de entrada, chamamos essa aproximação de hipótese (h no modelo). Essa hipótese é refinada dado os diversos exemplos fornecidos. Após algumas iterações do problema uma hipótese final é alcançada e esse é o modelo fornecido como resposta. Tomando esse modelo h podemos para um novo conjunto de dados \mathbf{X} prever um dado valor Y , como no exemplo X representa a área de uma residência e Y o preço predito para a mesma.

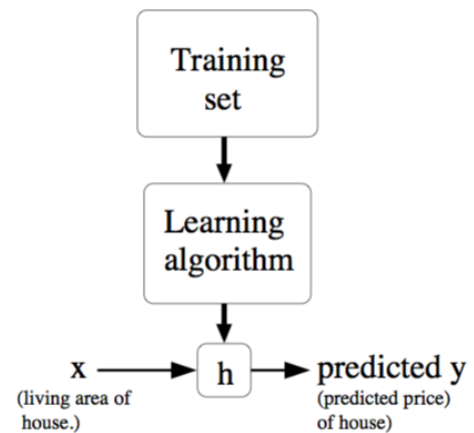


Figura 3.1: Modelo de Aprendizado de Máquina

Existem três tipos principais de aprendizado baseados no retorno que é concedido ao agente: não supervisionado, por reforço e supervisionado. No aprendizado não supervisionado o agente busca padrões sem uma avaliação contínua do modelo. A ideia básica desse tipo de aprendizado é reunir informações potencialmente semelhantes em alguma característica. No aprendizado por reforço o agente possui uma série de recompensas ou punições para expressar seu desempenho e aprender a partir das observações das mesmas. No aprendizado supervisionado o agente sabe a entrada e a saída de um certo conjunto de dados e a partir dos mesmos obtêm uma aproximação da função que gerou o conjunto de dados fornecidos.

Quando a classe do problema possui um domínio discreto chamamos o problema de aprendizado de *classificação*. Quando a classe do problema possui um domínio contínuo o problema é chamado de *regressão*

Para ilustrar esse esquema utilizaremos o conjunto de dados de Iris [FISHER 1936] que consiste em quatro medidas extraídas dos comprimentos e larguras das sépalas e

pétalas de três diferentes espécies de Iris, como podemos ver nos exemplos extraídos na tabela 3.1. Nesse exemplo, as *features* (características) são os comprimentos anotados em cm e a classe é a espécie da planta. O conjunto de exemplos de treinamento consiste nas diversas medidas extraídas que compõe o conjunto de dados. A hipótese final é capaz de, dado um conjunto de medidas novas, classificar corretamente uma nova planta.

Sépala		Pétala		Espécie
L	W	L	W	
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
7.6	3.0	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica

Tabela 3.1: Iris Dataset

L - Length (Comprimento)

W - Width (Largura)

os dados que temos o desempenho do agente.

Podemos avaliar diferentes modelos com diferentes conjuntos de treino, validação e teste para obter o melhor resultado (*crossed validation*). Caso percebamos que o conjunto de teste está mapeando os dados de treino podemos encerrar o treinamento (*early stopping*) para ter um modelo mais genérico, como indicado na imagem 3.2.

Existem cenários no qual não sabemos determinar as informações relevantes para o problema que queremos solucionar. Nesse caso podemos consultar um especialista do assunto para averiguar quais seriam as possíveis características relevantes para o problema ou tentar buscar resultados baseados em técnicas de aprendizado de representações onde o modelo descobre sozinho quais as características são relevantes. O aprendizado de representações é largamente utilizado atualmente com

O algoritmo pode não aprender uma boa hipótese por diversos motivos como poucos exemplos para se basear, treinamento com muitas iterações podendo levar ou a um mapeamento direto da informação de treinamento (chamado *overfitting* por se ajustar em excesso ao modelo) ou a uma função que não representa bem o modelo por falta de ajuste dos exemplos fornecidos (chamado *underfitting* já que não se adequou ao conjunto de dados). Em todas essas situações o agente encontrará dificuldade quando for apresentado a outras entradas.

Para evitar esse tipo de situação podemos dividir o conjunto de treinamento em diferentes subconjuntos de treinamento, validação e teste para poder medir com

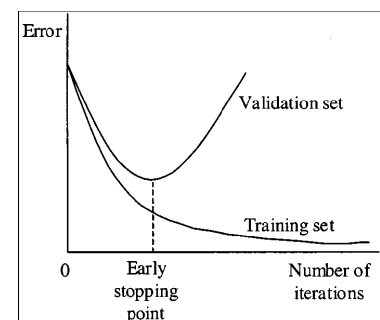


Figura 3.2: Overfitting ¹

¹Retirado de <http://neuron.csie.ntust.edu.tw/homework/94/neuron/Homework3/M9409204/discuss.htm>

redes neurais e deep learning.

3.2 Redes Neurais

A ideia de redes neurais se origina na tentativa de imitar o comportamento do cérebro humano, porém representando cada neurônio separadamente. Em 1943 McCulloch e Pitts[McCulloch e Pitts 1943] introduziram o modelo matemático de um neurônio, ou *perceptron*, como pode ser visto na figura 3.3. Nesse modelo temos diversas entradas, cada uma com um respectivo peso, sendo somadas por uma função de soma que é então avaliada por uma função de transferência para uma dada saída desse neurônio.

Uma função de soma toma as entradas e a partir de alguma soma (simples ou ponderada) retorna um valor de soma das entradas. Funções de soma podem ser simplesmente somas das entradas, somas dos quadrados, dos cubos, dos logaritmos naturais, entre várias outras.

Uma função de transferência recebe como entrada o valor da função soma e a partir de uma ativação qualquer passa adiante um valor na saída do neurônio. Das funções de transferência mais famosas podemos citar a Sigmoid e a Retificadora, descritas pelas equações 3.1 e 3.2, respectivamente. Um neurônio que utiliza uma função retificadora como função de transferência é conhecido como ReLU (*Rectified Linear Unit*)

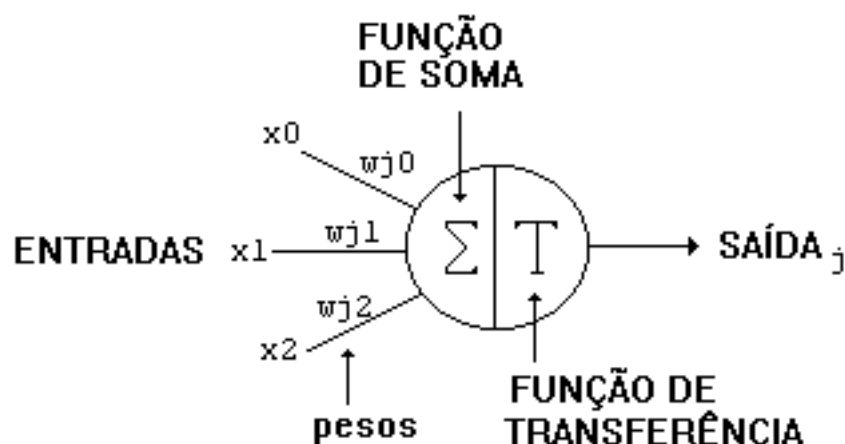


Figura 3.3: Modelo de neurônio proposto por McCulloch & Pitts (1943) ²

²Retirado de <http://redesneuraisartificiais.blogspot.com.br/2010/10/o-primeiro-modelo-de-um-neuronio-criado.html>

$$S(t) = \frac{1}{1 + e^{-t}} \quad (3.1)$$

$$f(x) = \max(0, x) \quad (3.2)$$

Tendo em vista o modelo de aprendizado discutido anteriormente já ilustrado pela imagem 3.1 temos que a hipótese que esse modelo deve aprender são os pesos ótimos representados por w_j na figura 3.3. Existem algumas técnicas para obter esses valores ótimos tomando muitas vezes valores aleatórios ou fixos para preencher o estado inicial. Um dos algoritmos mais simples é do *delta rule* que dado uma entrada calcula os erros entre a saída resultante e a saída desejada e usa isso para criar um ajuste nos pesos, representado na Equação 3.3.

$$\Delta = w - w_{old} = -\eta \frac{\partial E}{\partial w} = \eta \delta x \quad (3.3)$$

Mesmo apresentando ótimo desempenho para certos domínios demonstrou-se alguns anos depois que esse modelo só funciona com dados linearmente separáveis, isso é, dados que sejam separáveis através de uma reta. Como podemos ver na imagem 3.4 não é possível traçar uma reta que separe todos os pontos quando avaliamos as saídas da porta XOR, embora seja possível com as funções AND e OR.

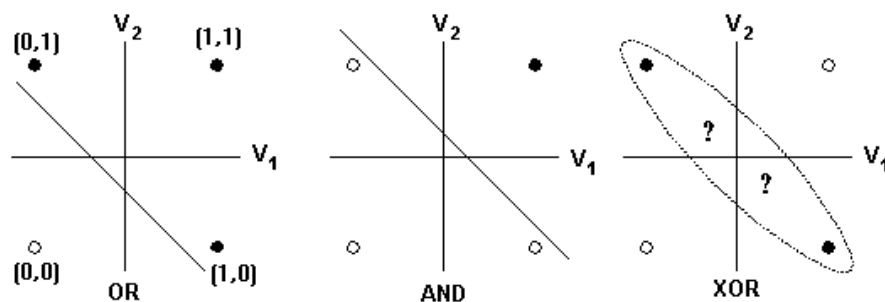


Figura 3.4: Dados gerados a partir de portas OR, AND e XOR ³

Para evitar que o modelo seja capaz somente de prever dados linearmente separáveis podemos interligar os neurônios em camadas. A medida que adicionamos neurônios e camadas seremos capazes de prever dados mais complexos, uma vez que aumentamos a

³Retirado de <http://ecee.colorado.edu/~ecen4831/lectures/NNet3.html>

dimensionalidade possível para o modelo encontrar possível planos que separem corretamente os dados. Mas ao mesmo tempo aumentamos a complexidade de treinamento pois aumentam os números de pesos a serem atualizados e com isso o tempo demandado para o mesmo.

$$\Delta w_{ji} = \alpha(t_j - y_j) - g'(h_j)x_i \quad (3.4)$$

O *delta rule* (Equação 3.4) é um caso especial de outro mais genérico e mais amplamente utilizado o *Backpropagation*. Esse algoritmo foi introduzido em 1970 mas ganhou popularidade em 1986 [Rumelhart D. E. 1986] quando apontou-se que era mais eficiente para treinar redes do que os algoritmos presentes até a época. O Backpropagation recebe uma entrada e toma a saída obtida com os pesos atuais comparando-a com o resultado esperado, mas usando uma função de perda (*loss function*) que gera um valor de erro para a camada anterior que se propagada até a camada de entrada. A taxa que define quanto esse erro irá alterar os valores dos pesos já computados é chamada de taxa de aprendizado. Taxas de aprendizado muito grandes tendem a oscilar muito e não estabilizar em um valor ideal por isso costumam ser valores pequenos. Usualmente uma única taxa de aprendizado pode demorar muito ou mesmo ser incapaz de encontrar os pesos ótimos para os neurônios. Para contornar esse tipo de problema é usual usar taxas de aprendizado que diminuam a medida que o treinamento toma andamento. Essa taxa variável é chamada de *Momentum*.

Observando a Equação 3.4 temos que o α representa a taxa de aprendizado, o trecho $(t_j - y_j)$ representa o erro entre as saídas esperadas (t_j) e a saídas obtidas (y_j). O $g'(x)$ representa a função de ativação do neurônio que recebe h_j que é a soma dos pesos das entradas do neurônio a partir de uma entrada x_i .

Introduzindo-se uma analogia para o funcionamento do algoritmo podemos imaginar que estamos em uma montanha e desejamos descer ao ponto mais profundo do vale dessa montanha. O caminho não nos é claro por culpa de uma névoa então devemos tomar a decisão com base no pouco que podemos ver. Intuitivamente buscamos o caminho com maior inclinação para podermos descer e buscar novamente a maior inclinação para continuarmos nossa descida. A taxa de aprendizado condiz com o tempo que andamos em linha reta entre uma avaliação e outra. Caso tomássemos utilizássemos tempos variáveis para reavaliar nossa caminhada estaríamos usando o mesmo princípio do *momentum* descrito anteriormente.

Atualmente existem diversas variações baseadas nesse algoritmo como o SGD (*Stochastic gradient descent*) que computa o gradiente de maneira incremental com aproximação estocástica.

As redes neurais tem seu treinamento medido em épocas (*epochs*). Uma época explicita que todo o conjunto de dados foi avaliado ao menos uma vez para influenciar nos pesos da rede. é comum que os pesos se estabilizem em uma faixa após algumas épocas indicando que uma possível estagnação do aprendizado naquele ponto.

Dado que já existiam outros algoritmos mais eficientes na classificação de modelos linearmente separáveis, as redes neurais perderam sua popularidade à época. Anos depois com o aparecimento de diversos tipos de neurônios e camadas com transformações não lineares eficientes foi possível montar redes com centenas de camadas e generalizar entradas com alto nível de abstração automaticamente [LeCun, Bengio e Hinton 2015] [Deng e Yu 2014].

3.3 Deep Learning

Até muito recentemente o treinamento de vários neurônios era um processo muito lento para gerar resultados em tempo hábil, por ser extremamente custoso para a CPU. Com o desenvolvimento das GPUs, voltadas especialmente para computação gráfica e fruto do investimento na indústria de jogos, tornou-se possível treinar grandes redes em tempo conveniente, graças a vetorização da matemática necessária aplicada nos núcleos especializados em operações matriciais. Atualmente, existem linguagens específicas para lidar com os núcleos das GPUs assim como um constante desenvolvimento de bibliotecas e ferramentas específicas das empresas, buscando auxiliar pesquisadores e entusiastas com o treinamento de redes neurais.

Além dos neurônios previamente apresentados podemos ter outras estruturas matemáticas compondo a rede neural como as **camadas de convolução** e as **camadas de pooling**. Essas camadas são frequentes em aplicações de visão computacional como detecção de objetos e segmentação. Uma camada de convolução toma como entrada uma matriz, que usualmente está representando uma imagem, e aplica um filtro de dimensão $N \times N$. Para cada posição possível desse filtro na imagem original é calculada a soma do valor do pixel original multiplicado pelo valor respectivo do filtro, resultando então em uma

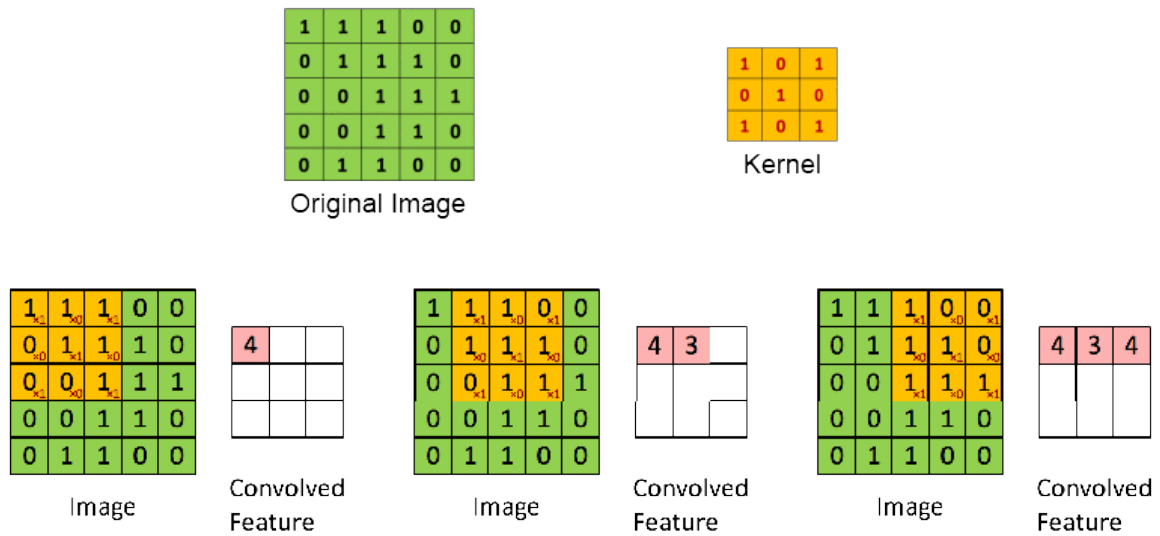


Figura 3.5: Funcionamento de uma camada de convolução ⁴

nova matriz. Essa operação acaba por reduzir a dimensão da matriz original em alguns pixels sendo essa redução proporcional ao tamanho do kernel utilizado. Essa operação é extremamente útil quando analisamos imagens, pois esperamos que imagens possuam valores estacionários em dados blocos, isso é, tomando um certo bloco pequeno o suficiente esperamos valores próximos o suficiente, identificando um certo padrão da informação representada na imagem. Podemos visualizar o processo na imagem 3.5. Diferentes tipos de kernel resultam em diferentes aplicações como detecção de bordas como na imagem 3.6, desfocagem (*blur*) e nitidez (*sharpening*).

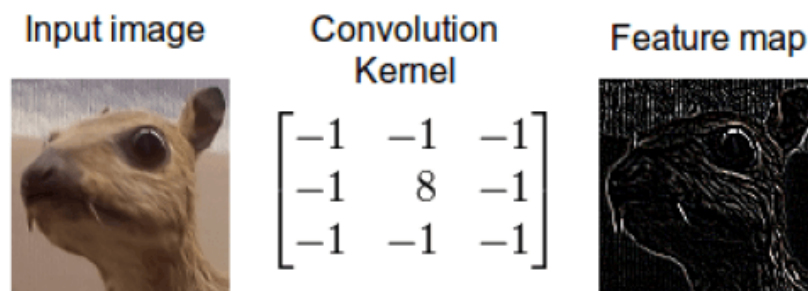


Figura 3.6: Detecção de Bordas com Convolução ⁵

⁴Retirado de <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

⁵Retirado de <http://timdettmers.com/2015/03/26/convolution-deep-learning/>

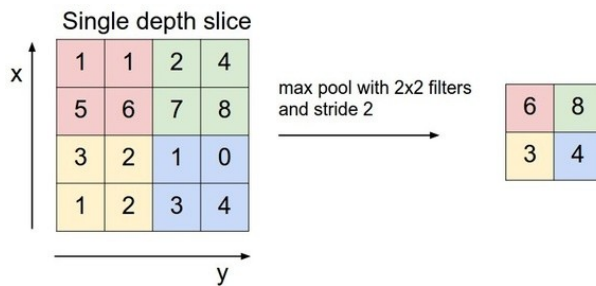


Figura 3.7: Camada de Pooling

Uma camada de **pooling** por sua vez reduz a dimensão da matriz de entrada num processo similar a camada de convolução. Usualmente, as operações nessa camada são de máximo ou mínimo daí um filtro $N \times N$ toma o maior (ou o menor) valor da matriz original e replica no resultado.

Essa camada é útil pois o descarte agressivo de pequenas informações reduz o overfitting [Srivastava Geoffrey Hinton 2014]. As camadas **convolucionais** são utilizadas em conjunto com camadas de pooling para reduzir o número de características aprendidas, uma vez que é esperado que certas propriedades da imagem se repitam.

Uma camada **completamente conectada** configura uma camada onde todos os neurônios da entrada estão conectados com todos os neurônios da saída [Priddy K. L. 2005]. Essa camada é equivalente a um MLP (*Multilayer Perceptron*) ou seja, vários neurônios em camada como podemos ver na Figura 3.8. A rede da figura apresenta dois neurônios de entrada ligados a 4 neurônios ligados a uma camada intermediária à saída. A essas camadas intermediárias dá-se o nome de camadas ocultas. E essa camada está ligada a 3 neurônios de saída. Repare que os neurônios da camada intermediária possuem uma função de ativação não linear e os da camada de saída são lineares apenas ilustrando que diferentes tipos de neurônio podem ser usados na construção de uma camada completamente conectada.

Uma camada de **softmax** costuma estar na saída da rede e ter o número de saídas com o mesmo número de classes que se deseja classificar. Essa camada recebe o vetor de entrada e através de uma função logística comprime as entradas num vetor de saída com o mesmo comprimento das classes expressando a probabilidade de cada classe dever ser classificada [Bishop 2006] [Priddy K. L. 2005].

Para ilustrar topologias possíveis de redes utilizadas em Deep Learning selecionamos três redes renomadas no reconhecimento de imagens.

⁶Retirado de <http://users.ics.aalto.fi/ahonkela/dippa/node41.html>

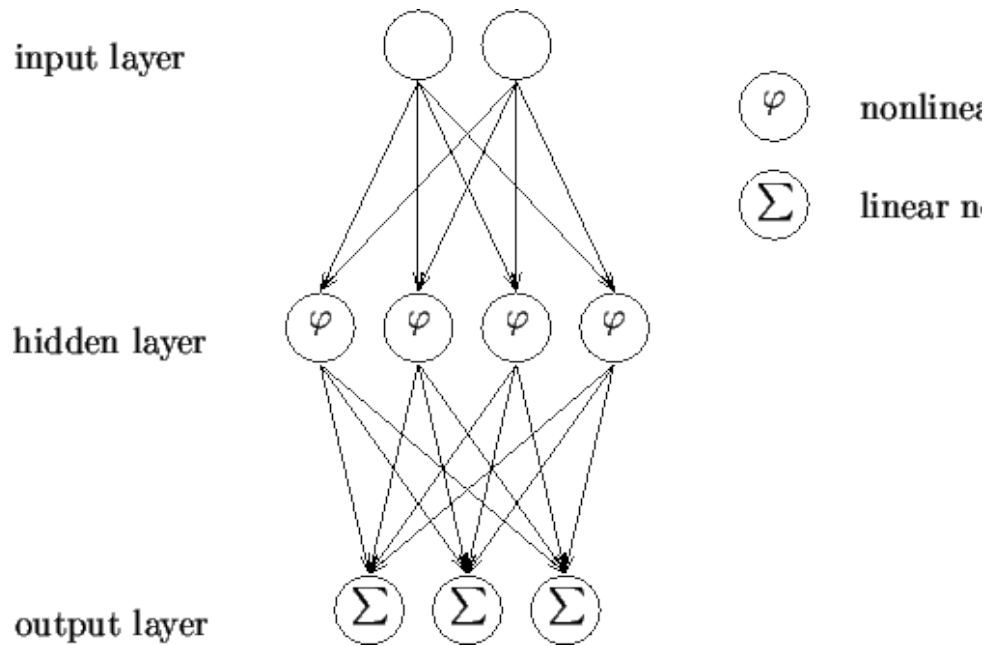


Figura 3.8: Multilayer Perceptron ⁶

3.3.1 LeNet

A rede conhecida como LeNet [LeCun Y. 1989] foi introduzida em 1989, objetivando classificar dígitos escritos a mão presentes no banco de dados MNIST. Essa rede possui cinco camadas sendo duas de convolução, conforme podemos ver na figura 3.9

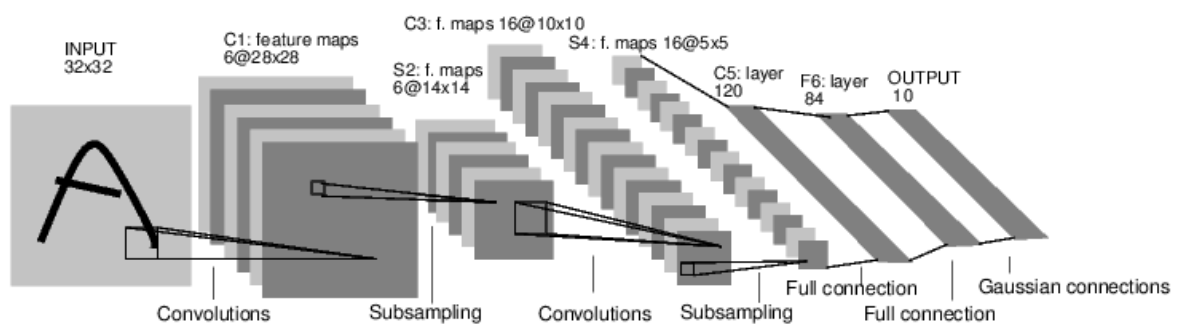


Figura 3.9: LeNet5

Podemos ver na figura 3.10 um pequeno conjunto dos dados usados para o treinamento da rede. Esses dígitos foram extraídos de cartas e pacotes enviados pelos correios e digitalizados em imagens monocromáticas de 28x28 pixels. O banco de dados possui 60000 exemplos no conjunto de treinamento e 10000 exemplos no conjunto de testes.

Em seu trabalho [LeCun Y. 1989] LeCun discorre sobre como a adição de camadas numa rede neural reduziu a taxa de erros de 12.0% para uma rede com uma única camada

para 3.8% com uma rede de 2 camadas, 2.45% com 3 camadas e como a rede com convolução apresentou erros de 1.7%, com a inserção de apenas uma camada de convolução, 1.1% para uma rede com 4 camadas e por fim 0.8% com a LeNet-5, que é a rede apresentada na figura 3.9.

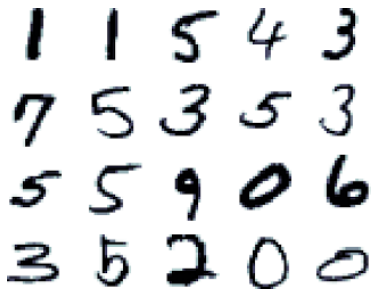


Figura 3.10: MNIST

A LeNet foi umas das primeiras redes a apresentar ótimos resultados com camadas de convolução, representadas por C e camadas de subsampling (que são parecidas com pooling mas ao invés de tomarem o maior valor tomam a média dos valores), representadas por S e por fim as camadas completamente conectadas representadas por F respectivamente na figura 3.9.

Os mapas de features são frutos de diferentes tipos de kernel usados para extrair diferentes características entre uma camada e outra. A camada $C1$ possui 6 mapas de features de tamanho 5×5 . Como a entrada é aumentada para 32×32 para garantir que apenas os 28 pixels centrais sejam avaliados pela rede temos que nessa primeira camada cada mapa de feature, ou seja, cada kernel gera uma matriz de 28×28 como indicado logo abaixo do $C1$ na imagem 3.9. A camada $S2$ por sua vez também possui 6 mapas de feature mas com tamanho 14×14 resultante da avaliação 2×2 dos mapas da camada $C1$. As camadas $C3$ e $S4$ possuem comportamento similar com a quantidade de mapas de features utilizados e seus tamanhos abaixo do nome de cada camada na imagem 3.9. A camada $C5$ é uma camada de convolução mas possui mapa de feature de tamanho 5×5 e, uma vez que a camada $S4$ provê essa mesma dimensão de entrada, resulta numa conexão completa entre as camadas. A camada $C5$ foi marcada como convolução para permitir uma maleabilidade da rede em se adaptar a entradas de outros tamanhos sem que a arquitetura fosse alterada, sendo necessário somente alterar o valor dos mapas de feature. Por fim a camadas $F6$ é uma camada completamente conectada que se liga a uma saída softmax explicitando a probabilidade de cada um dos números de ser o valor real para a entrada.

3.3.2 AlexNet

A AlexNet foi desenhada para competir na ILSVRC⁷ de 2010 que é uma famosa competição anual onde os participantes são desafiados a classificar diferentes imagens

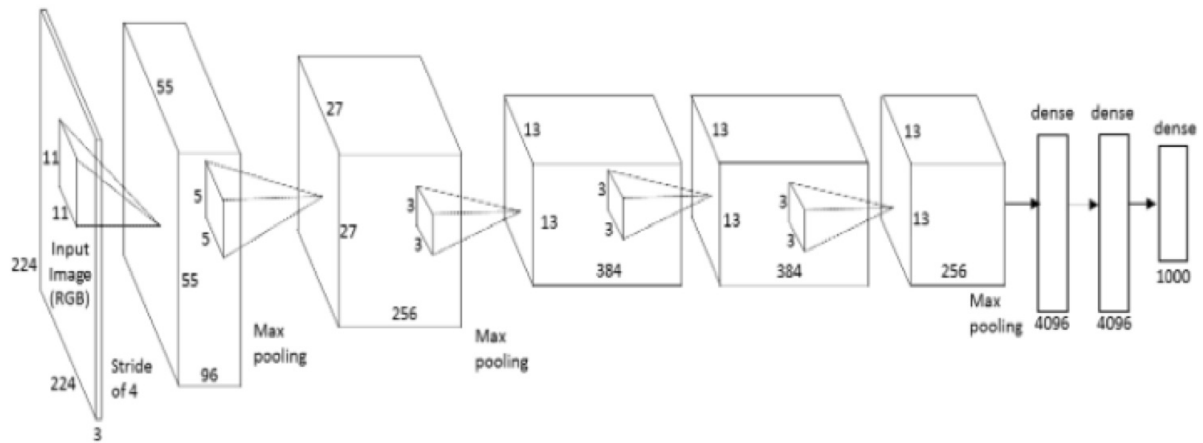


Figura 3.11: AlexNet

em várias categorias. A competição de 2010 teve 1.2 milhões de imagens de entrada com 1000 classes diferentes para serem determinadas. A AlexNet emprega oito camadas sendo cinco camadas de convolução, (algumas seguidas por camadas de *max-pooling*) e três camadas completamente conectadas no final da rede incluindo a *softmax* com 1000 saídas, respectivas a cada uma das classes.

A primeira camada recebe uma imagem colorida e cada canal de cor RGB é tratado separadamente. As entradas da rede são imagens de 256x256 que são recortadas em imagens menores de 224x224 em diferentes áreas da imagem para serem fornecidas ao treinamento. Quando vai executar uma classificação o pedaço central é recortado para não haver discrepância entre diferentes execuções. A primeira camada de convolução possui mapa de feature de tamanho 11 com passo de 4 resultando em 96 saídas de 55x55 conforme a imagem 3.11 aponta. Na segunda camada de convolução os 96 mapas de feature da camada anterior são avaliados com mapas de feature de 5x5 resultando em 256 novos mapas de feature mas esses passam por uma camada de max pooling nesse mesmo bloco, como indicado na figura. As camadas seguintes possuem comportamento similar com blocos com pooling quando indicado até chegarem nas camadas completamente conectadas com 4096 entradas cada e por fim uma camada de softmax com 4096 entradas e 1000 saídas.

A rede apresentou ótimo desempenho na competição exatamente pelo emprego das camadas de convolução, que carregam consigo premissas consideradas válidas quando tratamos de imagens, reduzindo a taxa de erro a 15.3%.

⁷<http://image-net.org/challenges/LSVRC/2010/>

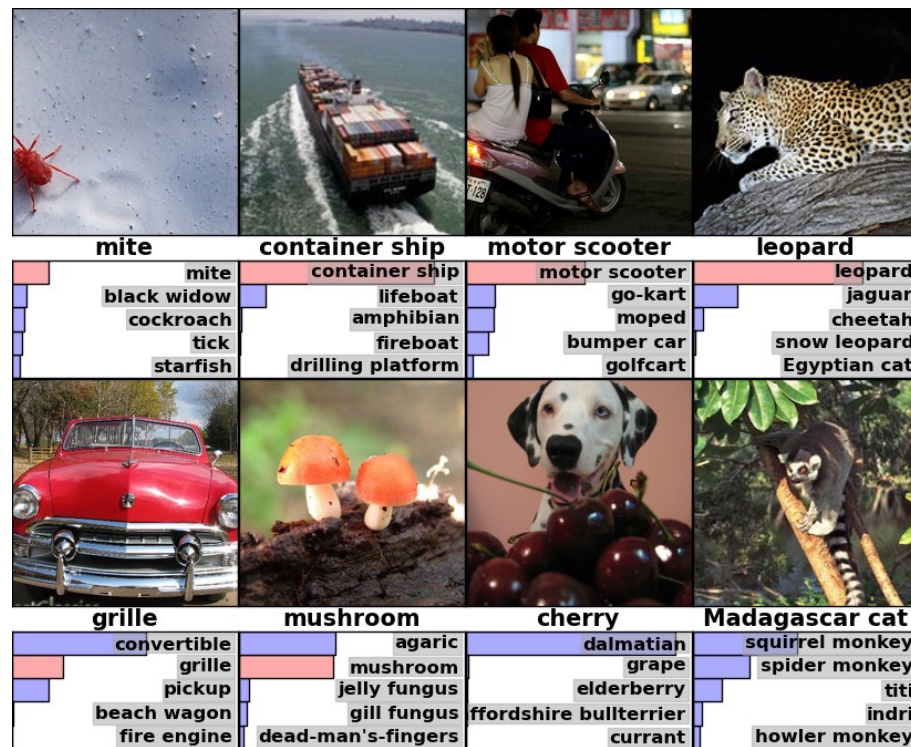


Figura 3.12: Resultados do Teste da AlexNet

3.3.3 GoogLeNet

A rede conhecida como GoogLeNet [Szegedy et al. 2014] foi introduzida em 2014 durante o ImageNet Challenge, onde o objetivo era classificar corretamente o maior número de classes de imagens apresentadas na competição. Essa rede possui 22 camadas, fruto do sistema chamado *Inception*, onde a cada bloco destacado na figura 3.13 repete-se a mesma arquitetura de camadas de convolução e uma camada de pooling.

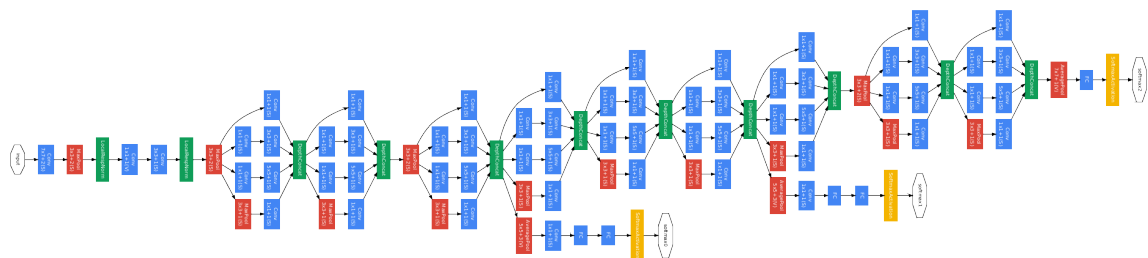


Figura 3.13: GoogLeNet e o sistema Inception

Um módulo do *Inception* consiste em blocos de convoluções com filtros 1x1, 3x3 e 5x5 e uma camada de max-pooling 3x3. Para a competição conforme a figura 3.14. O primeiro bloco a ser pensado foi um bloco que com concoluções 1x1, 3x3, 5x5 e um max pooling de 3x3 paralelos cujas saídas se encontrassem numa concatenação de filtros. A esse primeiro bloco chamou-se *naive* por ser a ideia mais simples. Essa configuração pode ser proibitivamente cara quando vários módulos são inseridos na rede com vários mapas de feature especialmente por causa da convolução 5x5. A união das saídas das camadas de convolução com a de pooling leva a um aumento das saídas a cada estágio causando um número muito grande com apenas alguns estágios.

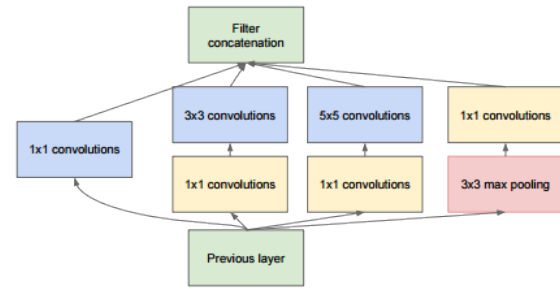


Figura 3.14: Bloco Inception com Redução de Dimensionalidade

Para contornar isso inseriu-se uma camada de convolução 1x1 antes das outras convoluções e após o pooling conforme a figura 3.14 para reduzir a dimensão de maneira a controlar o rápido crescimento de saídas a cada estágio.

Por motivos de eficiência estrutural decidiu-se utilizar camadas de convolução tradicionais no início da rede e utilizar os blocos Inception acoplados a essas camadas. Essa necessidade apenas reflete uma pequena deficiência na implementação corrente não sendo um fator proibitivo para construção.

Resultados semelhantes foram obtidos com o sistema *Inception* sendo usado em profundidade ou em largura. A arquitetura publicada expressa o sistema em profundidade por ter apresentado resultado ligeiramente melhor no desafio de 2014. A rede apresenta 3 camadas de softmax (no 4º, 7º e 9º grupos Inception respectivamente) para reportar a precisão da classificação em outras camadas antes da convolução final.

Podemos perceber que com o poder computacional rapidamente adquirido através dos anos foi possível evoluir rapidamente da classificação de 10 dígitos escritos a mão em imagens monocromáticas de 28x28 pixels para a classificação de 1000 classes de objetos em imagens coloridas de aproximadamente 256x256 pixels.

Para diversas aplicações, redes neurais são uma ótima escolha por serem um modelo geralmente mais compacto do que as outras soluções disponíveis e, conseqüentemente, mais rápido para inferir resultados após seu treinamento do que outros algoritmos de

aprendizado de máquina. Entretanto, essa compacticidade e rapidez vêm com um preço que é o alto custo computacional do treinamento da rede que cresce com a quantidade de camadas e neurônios inseridos assim como cresce também a capacidade da rede abstrair sozinha as representações das features que são relevantes para tal atividade.

Parte II

Abordagem e Entradas

Capítulo 4

Metodologia Experimental

Nesse capítulo explicamos com detalhes que permitam reprodutibilidade a geração de nossas entradas com os espectrogramas assim como os parâmetros para as redes utilizadas. As ferramentas utilizadas também são percorridas de modo que os resultados podem ser verificados em experimentos posteriores.

4.1 Amostras

Dado um sinal de áudio que contenha música desejamos obter quais são os possíveis instrumentos utilizados na sua construção. Para isso tomamos seções da música e determinamos o instrumento dominante de cada uma das seções. Utilizamos o conjunto de dados fornecido pela Sonatica Philharmonica [Orchestra 2009] da qual utilizamos 23 instrumentos frequentemente presente em músicas clássicas e também o conjunto de dados fornecido pelo IRMAS [Barcelona 2012] com 11 categorias de instrumentos. Uma descrição dos instrumentos presentes em cada dataset pode ser vista na tabela 4.1. Fica claro analisando a Tabela que o conjunto de dados do IRMAS é mais rico pelo fato de oferecer uma quantidade maior de amostras. Os dados do Sonatica são instrumentos tocando uma única nota por um período de tempo e os dados do IRMAS representam o instrumento dominante de certa melodia. Cada amostra consiste num arquivo de áudio digital WAV com 44.1kHz de taxa de amostragem que é o padrão para armazenamento digital de áudio puro e não comprimido. É esperado de antemão que o conjunto de dados com menor número de classes e maior número de amostras expresse um resultado mais generalizado. Podemos antecipar também que o esforço necessário para treinar um

modelo somente com os dados do Sonatica será dezenas de vezes menor do que para os dados do IRMAS.

A partir de cada arquivo WAV geramos diferentes espectrogramas para analisar diferentes comportamentos e tentar obter o melhor resultado. Como dito anteriormente, um espectrograma é um gráfico de três dimensões onde adotamos a dimensão horizontal como tempo, a vertical como frequência e a terceira, representada pela intensidade da cor, indicando a potência do sinal em decibéis. Adotar uma representação com tons de cinza para a potência reduz consideravelmente a escala de valores possíveis se compararmos com uma escala de cores, pois uma escala de cinza utiliza até 256 valores e uma escala de cores adota até 16.777.216 valores. Gostaríamos de verificar se essa compressão da informação traz muitas perdas na análise dos nossos modelos. Por isso, para cada experimento, geramos um conjunto de espectrogramas coloridos e outro em escala de cinza.

Para a geração dos espectrogramas, definimos a janela de tempo com o valor de 500ms em contraste a outras pesquisas que definiram valores menores na faixa de 10 a 50ms [Uzan e Wolf 2015] [Abdel-Hamid et al. 2012] [Sainath Ron J. Weiss 2015]. Acreditamos que valores maiores sejam capazes de capturar uma expressão melhor do instrumento dominante em uma música, uma vez que ocorre com frequência a sobreposição de instrumentos para composição das mesmas. Uma janela de tempo muito grande reduz a possibilidade de dominância de certos instrumentos que apareçam apenas em trechos da melodia. Uma janela muito pequena, por sua vez, aumenta consideravelmente a quantidade de espectrogramas gerados e conseqüentemente o tempo de treinamento, mas ainda assim não garantindo uma precisão maior. Um limite maior de tempo permite o modelo generalizar características temporais que não seriam visíveis, dado que suas entradas são temporalmente independentes, ou seja, não existe correlação entre uma amostra x_i e uma amostra x_{i+1} , onde i indica uma sequência de tempo na melodia. Não existe na literatura um acordo sobre qual janela de tempo usar entre as mais diferentes tarefas de classificação musical. Alguns usam janelas tão grandes quanto a própria melodia para obter de maneira muito mais resumida informações sobre a música enquanto outros buscam extrair de janelas de tempo consideravelmente menores alguma informação relevante. Isso leva nossa escolha intermediária a ser baseada apenas em uma intuição baseado nos outros valores escolhidos.

Cada espectrograma representa um exemplo utilizado na nossa rede. A utilização

de imagens como entrada não é a tradicional utilizada no aprendizado de máquina conforme vimos com o conjunto de dados do Iris mas podemos abstrair uma imagem como uma representação matricial que pode ser vista como um longo vetor de características se necessário. Podemos observar a quantidade de exemplos de cada modelo na tabela 4.1. A escala de cor adotada (cinza ou colorido) não influencia na quantidade de espectrogramas gerados, o único aspecto influente na quantidade de amostras é a janela de tempo utilizada. É importante observar que a tabela 4.1 não implica na conversão direta de espectrogramas gerados na mesma pois nem todos os arquivos que compõem a base de dados são múltiplos das janelas de 500ms e 100ms, implicando em pedaços da amostra descartados. Além disso, os arquivos de áudio possuem dois canais de gravação, possibilitando a extração do espectrograma de cada canal separadamente conforme . Cada espectrograma corresponde a um instante de tempo completamente independente pois não foi utilizada nenhuma sobreposição de tempo entre diferentes amostras, ou seja, o passo entre amostras corresponde exatamente ao tamanho da janela de tempo utilizada. Suponha que a primeira amostra corresponda a janela de tempo de 0 a 500ms do canal 1. Essa independência indica que a amostra seguinte, por exemplo, será na janela de tempo de 500ms a 1000ms e não algum intervalo diferente como 400ms a 900ms.

Devemos observar que quaisquer modificações nos parâmetros do espectrograma geram um novo conjunto de dados que tem que ser reaprendido pelo modelo. Um conjunto de dados mais complexo (seja pela dimensão do conjunto ou pela informação contida em cada exemplo) implica também num tempo de treinamento maior o que foi um fator limitante para a escolha de conjuntos de dados para treinamento nesse experimento.

Em nossos experimentos treinamos inicialmente as redes com quatro conjuntos de dados tanto para os dados da Sonatica quanto do IRMAS:

- Espectrogramas de 500ms Coloridos
- Espectrogramas de 500ms Preto & Branco
- Espectrogramas de 100ms Coloridos
- Espectrogramas de 100ms Preto & Branco

Utilizamos para a geração dos espectrogramas um script python em conjunto com a biblioteca *matplotlib*. Dos parâmetros alterados da chamada padrão temos:

- o valor NFFT que representa a quantidade de pontos utilizados em cada transformada. Valores potência de 2 apresentam melhor desempenho devido a implementação do algoritmo e por isso o valor 1024 foi utilizado por apresentar o melhor custo benefício entre qualidade do espectrograma e tempo para seu cálculo.
- o valor F_s que explicita a frequência de amostragem do sinal. Essa frequência é obtida diretamente do arquivo de áudio carregado que nos nossos exemplos é de 44.1kHz.
- o valor *noverlap* que indica a quantidade de pontos de sobreposição entre os blocos de cada transformada. Assim como o NFFT esse valor possui melhor performance se valores com potência de 2 forem utilizados. Adotamos para esse parâmetro o valor de 512, fruto do melhor resultado visual.

O conjunto de validação toma 25% do conjunto de amostras para avaliar o desenvolvimento da rede periodicamente. O conjunto de teste consiste em músicas oferecidas em conjunto com o banco de dados IRMAS. Os instrumentos que não estão presentes nos dois conjuntos de dados foram testados com músicas onde o instrumento era o único tocado para avaliar se o modelo seria capaz de classificar o mesmo. O comprimento médio das músicas utilizadas no conjunto de testes foi de 4 minutos resultando em aproximadamente 2400 amostras de teste para 100ms e 480 amostras de teste de 500ms para cada.

4.2 Redes Neurais

Utilizamos nessa monografia as três redes neurais discutidas anteriormente: LeNet, AlexNet e GoogLeNet. A rede LeNet foi modificada para receber entradas com as mesmas dimensões das outras redes, ou seja, imagens de 256x256 pixels menor que a entrada de 28x28 a qual foi originalmente projetada. Essa modificação acarreta numa maior quantidade de operações nas camadas de convolução que foram aumentadas para comportar esse aumento. A rede em si não sofreu modificações estruturais somente os tamanhos dos kernels foram alterado de maneira compatível com o novo tamanho de entrada para manter a última camada de convolução com a conexão completa como discutido na seção 3.3.1. A rede LeNet não é capaz de trabalhar com imagens coloridas corretamente pois foi

Tabela 4.1: Amostras Utilizadas - A coluna Seg representa as amostras em segundos e as colunas de 500ms e 100ms os espectrogramas gerados para cada janela de tempo respectivamente

Instrumento	Sonatica			Instrumento	IRMAS		
	Seg	500ms	100ms		Seg	500ms	100ms
AltoFlute	43	68	413	Organ	2045	2046	14322
Trombones	73	82	676	Clarinet	1514	1515	8148
GrandPiano	312	541	3043	Acoustic Guitar	1910	1911	13377
Violin	331	431	3090	Electric Guitar	2279	2280	15690
CorAnglais	34	57	337	Flute	1352	1353	9471
Trumpet	156	213	1429	Voice	2333	2334	16338
Basses	124	154	1154	Saxophone	1877	1878	13146
Violas	88	116	825	Cello	1163	1164	10605
Horn	101	128	949	Violin	1739	1740	12180
Celli	101	112	918	Piano	2162	2163	15141
Piccolo	49	85	481	Trumpet	1730	1731	12117
Tuba	82	105	768				
BassClarinet	36	53	350				
TenorTrombone	31	44	293				
Clarinet	104	169	1005				
Cello	88	155	863				
Contrabassoon	28	38	263				
Percussion	74	946	3160				
Oboe	38	236	708				
Harp	135	92	351				
Bassoon	371	454	1316				
Chorus	335	672	3647				
Flute	161	498	1538				
BassTrombone	44	138	426				

originalmente desenhada para receber caracteres manuscritos monocromáticos e por esse motivo os experimentos realizados com espectrogramas com escalas de cor (que utilizam 3 camadas na entrada RGB) não puderam ser efetuados na mesma.

A AlexNet e a GoogLeNet foram utilizadas conforme arquitetura as descrições originais. Todas as redes foram treinadas por 20 épocas e foram validadas a cada uma das épocas. Recordando que uma época é a quantidade de iterações necessárias para que todas as entradas sejam avaliadas e possam influenciar nos pesos da rede ao menos uma vez. Todas as redes utilizaram a abordagem SGD (*Stochastic gradient descent*) como função de atualização de pesos, com três taxas durante o mesmo aprendizado: 0.01, 0.001 e 0.0001. Cada uma permanece por um terço do tempo de treinamento especificado. No caso como especificamos 20 épocas de treinamento cada um é utilizado por aproximadamente 6,67 épocas. Essas três taxas de aprendizado permitem que a rede aprenda pesos que façam sentido para o domínio mais rapidamente nos primeiros passos e refine continuamente os mesmos nos últimos passos.

Para o treinamento de cada rede utilizamos como arcabouço de implementação das redes o Caffe [Jia et al. 2014] em conjunto com a interface gráfica do DIGITS[NVIDIA] fornecida pela NVIDIA, com uma GPU GTX 980M 4GB, um processador i7 4860HQ @ 3.5Ghz e 32GB de RAM. O Caffe foi escolhido como arcabouço dentre os vários disponíveis pois foi uma das primeiras e mais famosas implementações com suporte a paralelização com GPUs resultando, em uma ótima velocidade de treinamento e em um grande grupo de usuários para auxiliar no aprendizado. O Caffe trabalha com a descrição das redes em meta arquivos, que descrevem cada camada e os respectivos pesos e taxas de treinamento que serão utilizados.

Essa modularidade permite fazer alterações facilmente nas redes e testar diferentes configurações sem a necessidade de alteração de código algum, apenas do arquivo de configuração que pode ser facilmente duplicado. Podemos ver um exemplo do arquivo de configuração da primeira camada de convolução da AlexNet no Listing 5.1 aonde destacamos as camadas que se conectam acima e abaixo e os parâmetros de convolução.

O Caffe oferece a possibilidade de salvar o modelo treinado em um arquivo com extensão *.caffemodel* a cada certa quantidade de iterações, para ser utilizado em outros lugares ou mesmo servir de base como re-treino em uma etapa de refinamento posterior. Esse arquivo pode ser utilizado facilmente com a ligação oferecida pelo Caffe para Python

em várias aplicações. Nesse arquivo são salvos todos os pesos dos neurônios treinados até a iteração a qual ele foi salvo. O treinamento não utilizou nenhum conjunto de parâmetros iniciais, ou seja, eles foram inicializados de forma aleatória.

Listing 4.1: Descrição da camada 1 de Convolução da AlexNet no Caffe

```

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}

```

Adotamos as mesmas entradas das arquiteturas originais para a AlexNet (que recebe 128 imagens coloridas por lote) e para a LeNet (que recebe sessenta e quatro (64) imagens em escala de cinza por lote). A GoogLeNet originalmente recebia trinta e duas (32) imagens por lote, mas por limitações de memória da GPU reduzimos esse valor para dez (10) imagens por iteração. A necessidade de se alterar a quantidade de imagens a cada lote na GoogLeNet não afeta o desempenho da mesma, só aumenta o tempo de treinamento.

O DIGITS é uma interface gráfica para o Caffe desenvolvido pela NVIDIA que facilita a gerência dos conjuntos de dados, criação das redes e até mesmo dos resultados obtidos. A interface permite um acesso muito mais simples aos dados e às redes, pois possui a chamada de diversas funções do Caffe por detrás da interface gráfica. Essa interface é acessada através de um browser e fica disponível num servidor web local que pode ser facilmente configurado para ficar disponível na internet, facilitando a utilização remota por diversos usuários a um cluster, por exemplo.

Capítulo 5

Trabalhos Relacionados

Um dos primeiros trabalhos na área obtiveram bons resultados com pequenas redes neurais completamente conectadas [Kaminsky Materka 1995] com a seleção de quatro instrumentos bastante distintos (piano, marimba, acordeão e percussão), a aplicação de conhecimentos especialistas (como filtragem de frequências) e um número extremamente reduzido de amostras (10 amostras para cada). As acurácias desse trabalho alcançaram valores de 97% a 100%.

Houve uma rápida evolução para utilização de novas arquiteturas de rede neural como mapas auto organizados e rede neural com atraso de tempo [Cemgil A. T. 1998] [Toiviainen P. 1998] [Feiten B. 1994]. Novamente os resultados apresentados foram encorajadores por apresentar acurácias acima de 97% mas novamente o conjunto de dados extremamente reduzido (apenas 40 amostras para 10 classes de instrumentos em com variação de uma oitava de frequência entre as notas selecionadas) [Cemgil A. T. 1998]. Os resultados com mapas auto organizados foram comparados com julgamentos humanos e obtiveram boa aprovação.

Para a tarefa de classificação de instrumentos outras técnicas apresentaram bons resultados [Agostini, Longari e Pollastri 2003] [Barbedo e Tzanetakis 2011] [Lots S. 2008] [Yeh C. 2010]. Entretanto essas técnicas apresentaram sempre a presença de um conhecimento especialista ou mesmo a comparação de diferentes características apontadas pelo especialista para determinação da melhor para a tarefa. Técnicas semelhantes são utilizadas em reconhecimento de voz e tratamento de outros sinais de áudio, mais uma vez fortemente atrelado ao domínio devido a necessidade da presença do conhecimento especialista.

Já desde 1998 a LeNet apresentou excelentes resultados na classificação de dígitos escritos a mão em entradas bastante reduzidas e em escala de cinza [LeCun Y. 1989]. Mas mesmo com a existência da tecnologia seu treinamento não era simples sendo apenas com a facilidade do acesso às GPUs e a otimização do desempenho das redes neurais convolucionais que sua utilização voltou a ser popular em 2012 com o aparecimento da AlexNet [Krizhevsky Ilya Sutskever 2012] e da GoogLeNet [Szegedy et al. 2014] em 2014. A utilização de redes neurais recorrentes foi feita com sucesso recentemente em 2014 [Sigtia S. 2014] utilizando uma técnica similar a utilizada na compreensão de fonemas para reconhecimento de fala. Utilizando uma estratégia similar [Li, Chan e Chun 2010] treinou redes neurais de convolução para classificar instrumentos mas utilizou como entrada outra representação gráfica do som, o MFCC, e construiu uma rede própria utilizando 3 camadas de convolução para a classificação de apenas 10 instrumentos.

Percebemos em toda a literatura uma influência muito forte da opinião do especialista mesmo após a utilização de redes neurais de convolução com *deep learning* para aprendizado das representações, o que seria um passo para a remoção da necessidade do especialista para obtenção de bons resultados.

Capítulo 6

Resultados

Analisamos os conjuntos conforme a descrição do último capítulo e organizamos nesse capítulo as acurácias, resultados dos testes e indicações sobre os melhores resultados. Devemos ser cautelosos ao apontar o melhor resultado de cada experimento. Por exemplo, o resultado com acurácia mais alta não indica necessariamente que o resultado é o melhor, como veremos a frente. O melhor resultado é aquele que dadas diferentes entradas das mais diversas fontes ainda é capaz de identificar o instrumento predominante corretamente, em outras palavras, o melhor resultado é aquele que consegue generalizar melhor a tarefa alvo.

Podemos perceber observando a tabela 6.1 e 6.2 que várias redes obtiveram acurácias bastante altas quando confrontadas com seus respectivos conjuntos de validação.

Podemos perceber um leve aumento de acurácia quando observamos as redes treinadas com janela de 100ms. A diferença não é tão grande a ponto de ser crucial pois as redes são inicializadas com diferentes pesos e é comum uma pequena diferença entre as mesmas arquiteturas. Quando observamos os resultados de dados gerados com escala de cor podemos observar um aumento geral das acurácias de todas as redes. A rede LeNet não pôde ser utilizada nesse treinamento pois sua entrada não comporta imagens coloridas.

Mas ao avaliarmos o resultado com o conjunto de testes gerado a partir das melodias marcadas do IRMAS fomos confrontados com um problema. Todas as redes treinadas com o conjunto de dados do Sonatica apresentaram uma tendência muito forte a supor que todas as entradas eram Percussão. Isso fica ainda mais claro quando observamos os testes, como na Figura 6.1. Na Figura podemos observar as classificações das cerca de

800 amostras marcadas como Piano gerados a partir da música Fur Elise - Beethoven representados pela barra vermelha e indicada como *Ground Truth*. O primeiro conjunto de barras marcado com Sonatica 100ms P&B representa as redes treinadas com o respectivo conjunto de dados (sendo P&B a indicação da utilização da escala de cinza) e cada barra de cor respectiva indica os instrumentos que foram classificados mais vezes. No caso todas as amostras que eram esperadas como Piano foram classificadas incorretamente como Percussão pelas três redes. O conjunto de barras seguinte representam as redes treinadas com o conjunto de dados IRMAS a 100ms P&B e podemos observar que as três redes obtiveram bons resultados nessa classificação uma vez que as três foram capazes de apontar corretamente o instrumento predominantes em suas classificações. O conjunto seguinte de barras só possui duas barras pois foi treinado com o Sonatica 100ms Colorido e a LeNet não oferece suporte a treinamento com cores. Observamos que a adição de cores já foi um fator importante para a remoção da tendência observada na classificação de todas as amostras como percussão mas ainda assim as amostras foram classificadas incorretamente como Trombone Grave. Por fim o último conjunto de dados treinado com IRMAS a 100ms e cores obteve bons resultados pois apresentou um alto número de classificações corretas. O conjunto de dados do Sonatica, por ser menor e sem ruído externo além do próprio instrumento, apresenta resultados melhores se avaliarmos somente a acurácia confrontando-o com dados controlados. Entretanto ao avaliarmos diferentes tipos de fontes como as músicas separadas do .

Observando as matrizes de confusão da AlexNet nas Figuras 6.3a e 6.3b onde a Figura 6.3a exibe a matriz confusão da AlexNet treinada com 500ms e cores com os dados do IRMAS confrontada com os dados de treinamento também do IRMAS e a Figura 6.3b que representa a matriz confusão dos dados da AlexNet treinada com 100ms e cores com os dados do Sonatica testando os .

Avaliamos então os dados de instrumento de percussão e descobrimos ser na realidade um subconjunto de diversos tipos distintos de instrumentos de percussão conforme uma observação mais profunda do conjunto [Orchestra 2009]. A quantidade de amostras de Percussão corresponde a 11,4% das amostras do banco sendo a segunda classe majoritária do conjunto de dados perdendo apenas para o piano. Como nosso experimento desejava classificar somente o instrumento e não uma hierarquia de instrumentos, nós fomos confrontados com duas opções: remover a classe de percussão ou tornar cada sub-

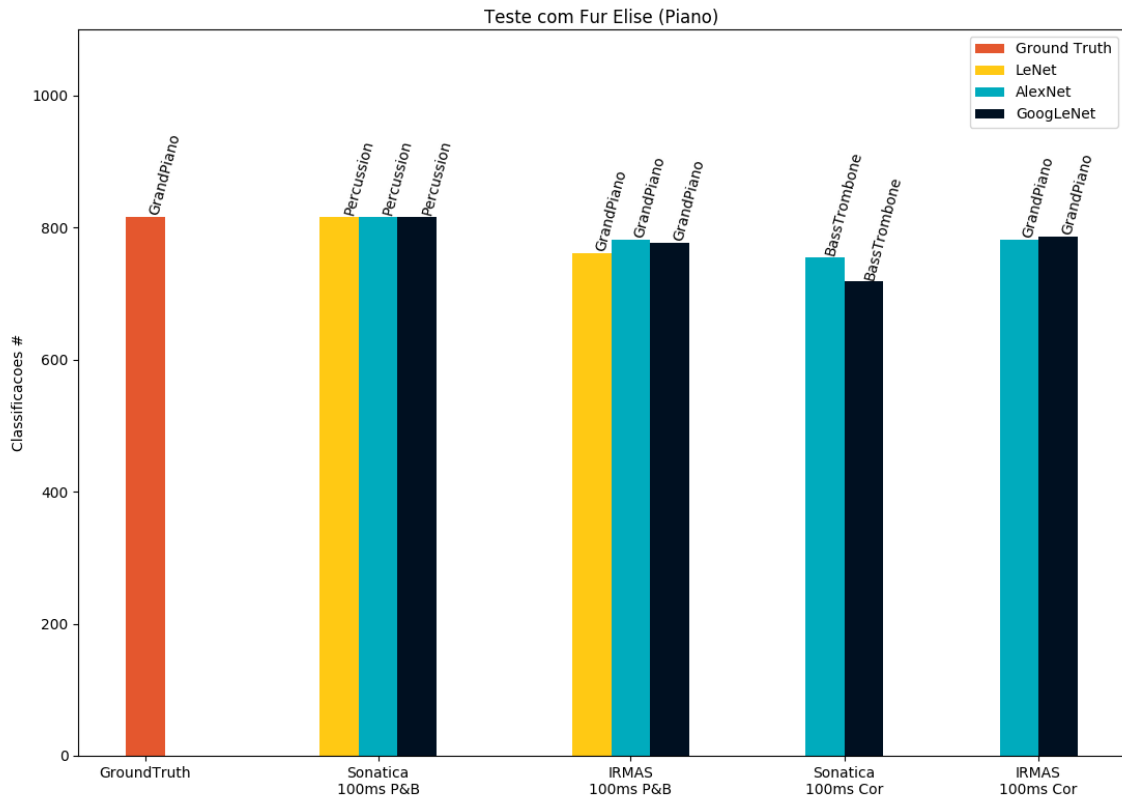


Figura 6.1: Teste com Fur Elise (Piano) 100ms de Amostragem

conjunto uma nova classe para então reavaliar as redes. Tornar cada subconjunto uma nova classe não se mostrou uma boa ideia, pois cada um deles era composto de um a cinco arquivos de áudios que gerariam poucas amostras para suas respectivas classes e isso nos levou a remoção dessa classe em nossa classificação. A remoção dessa classe entretanto não trouxe melhora grande na tarefa de classificação do instrumento predominante.

Percebemos que em várias situações o comportamento da AlexNet e da GoogLeNet são parecidos. As duas apresentaram acurácias semelhantes com pequenas variações e mesmo resultados semelhantes como observado na Figura 6.1 e 6.2. Devemos ressaltar que a arquitetura da GoogLeNet é várias vezes mais profunda que a AlexNet e isso acarretou um tempo consideravelmente maior no treinamento. Tanto a AlexNet quanto a GoogLeNet apresentaram ótima performance quando apresentadas aos dados do IRMAS

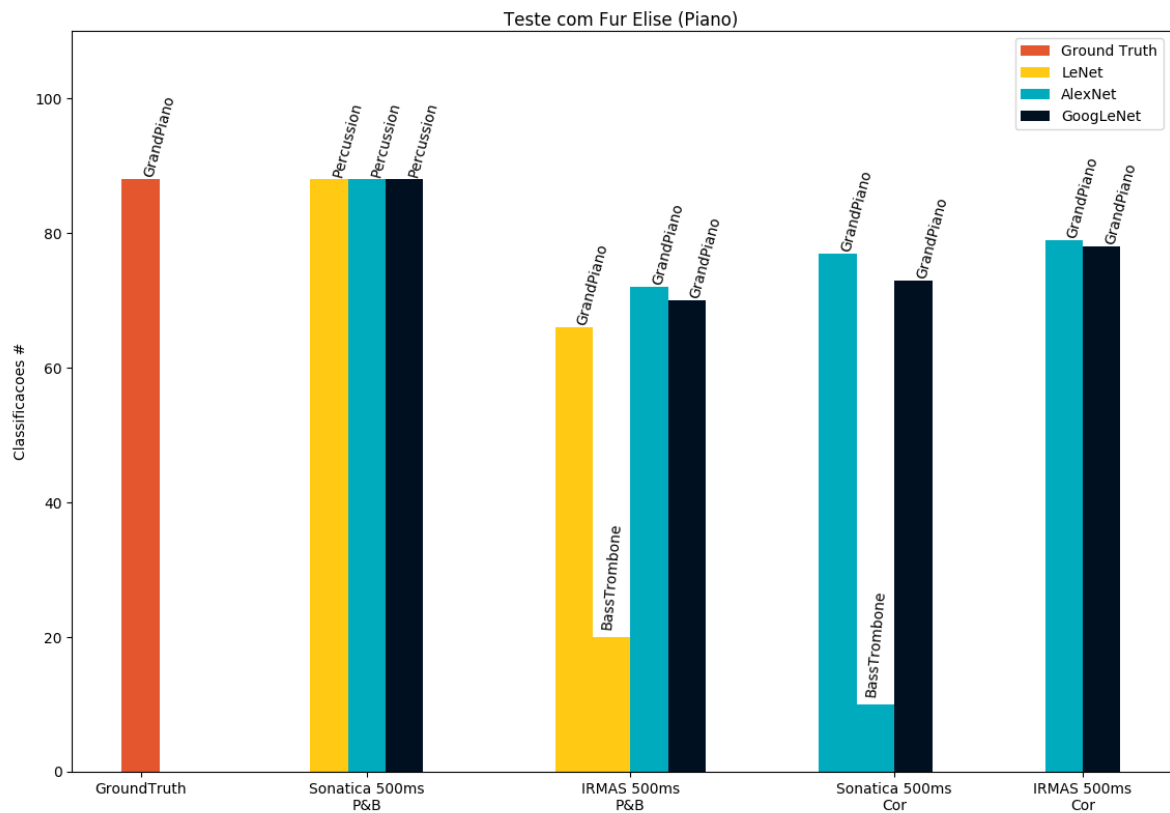
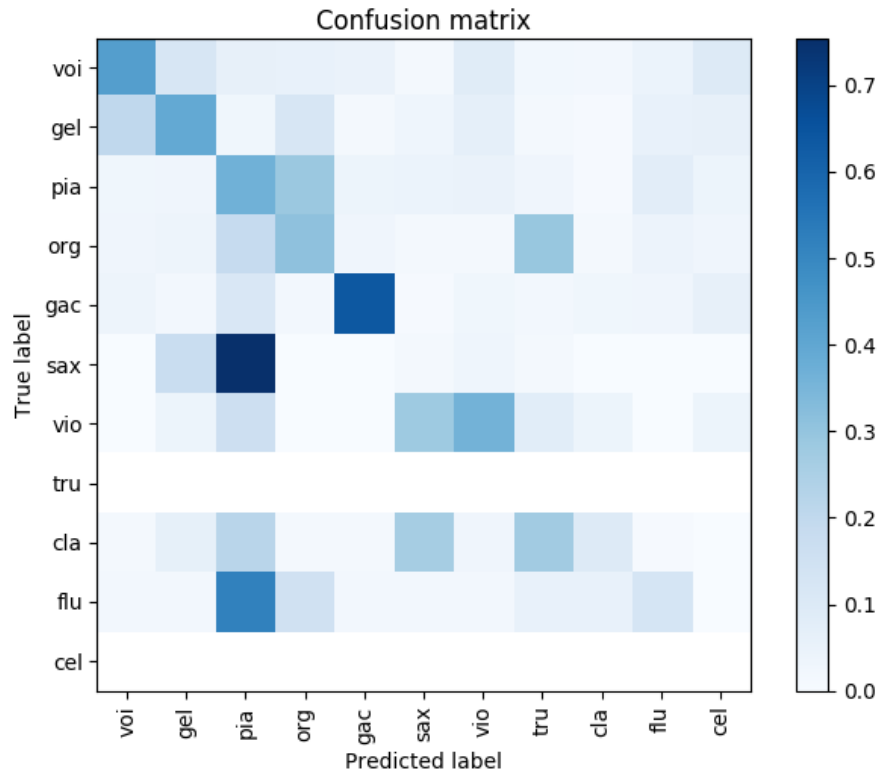


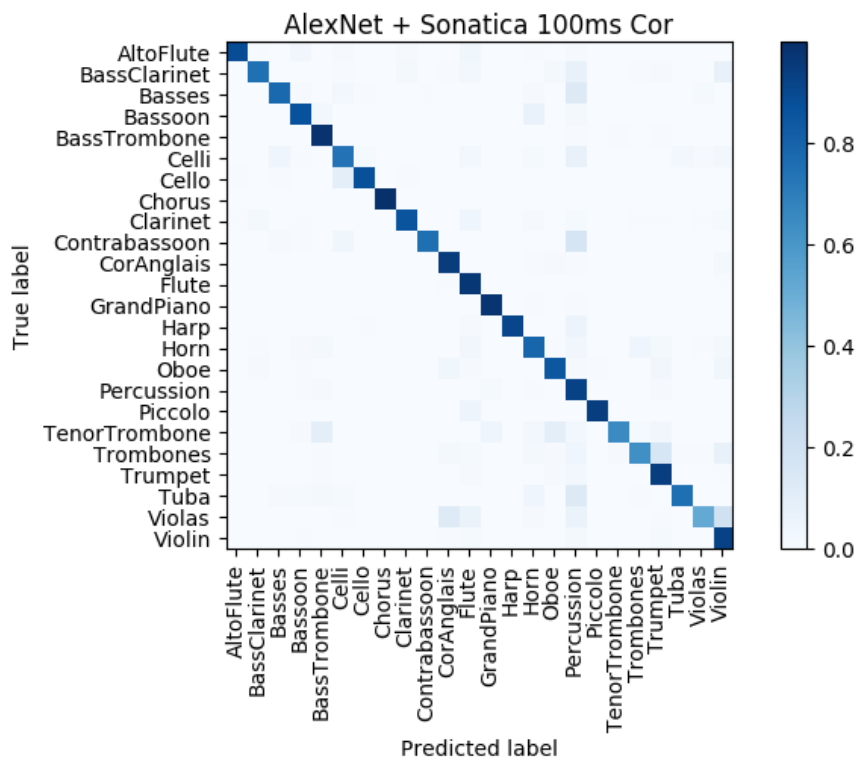
Figura 6.2: Teste com Fur Elise (Piano) 500ms de Amostragem

Tabela 6.1: Acurácias Obtidas por cada Rede, a partir dos dados em Escala de Cinza

	LeNet	AlexNet	GoogLeNet
Sonatica 100ms	89,33%	76,05%	78,43%
Sonatica 100ms s/ Percussão	97,97%	91,44%	92,60%
IRMAS 100ms	61,54%	69,07%	70,02%
Sonatica 500ms	82,59%	66,16%	80,09%
Sonatica 500ms s/ Percussão	92,63%	92,15%	91,09%
IRMAS 500ms	54,72%	57,81%	57,52%



(a) AlexNet treinada com IRMAS, 500ms de Janela em Cinza



(b) AlexNet treinada com IRMAS, 500ms de Janela em Cinza

Figura 6.3: Matrizes de Confusão da AlexNet

Tabela 6.2: Acurácia dos dados com Escala de Cor

	AlexNet	GoogLeNet
Sonatica 100ms	90,76%	92,36%
Sonatica 100ms s/ Percussao	92,20%	93,33%
IRMAS 100ms	72,22%	71,38%
Sonatica 500ms	93,02%	90,80%
Sonatica 500ms s/ Percussão	96,05%	95,63%
IRMAS 500ms	76,56%	77,35%

Parte III

Conclusões

Capítulo 7

Conclusões

Apresentamos nesse trabalho uma possível técnica para a classificação de instrumentos musicais. Fomos confrontados com diversas possibilidades de entradas e diversos parâmetros para cada uma como diferentes janelas de tempo. A partir da escolha de alguns parâmetros baseados na literatura geramos diferentes conjuntos de dados que usamos como entrada. Também escolhemos redes neurais convolucionais como modelo a ser experimentado nesse trabalho escolhemos arquiteturas renomadas como a LeNet, AlexNet e GoogLeNet para verificar seu comportamento em um domínio diferente do que foram originalmente desenhados. Após treinarmos as redes comparamos os resultados apontados a partir do conjunto de testes composto por diferentes melodias.

Notamos que o conjunto de dados do Sonatica apresentou-se muito simples e não representou uma boa amostra para o aprendizado da representação dos instrumentos, ao contrário do IRMAS que, com uma quantidade considerável de amostras apresentou ótimos resultados.

Concluimos que a compressão muito grande da escala de cor representando a escala de potência é extremamente prejudicial ao experimento. Uma arquitetura como a LeNet que só dê suporte a entradas muito simples não foi capaz de identificar as características necessárias para a representação levando a dificuldade de classificação em ambientes mais genéricos. Um ambiente genérico configura um conjunto sem obrigações fortes, diferentemente dos dados de treinamento que precisam estar padronizados. Gravações dos mais diversos tipos de dispositivos, ambientes e instrumentos com diferentes afinações compõem dados mais genéricos que gravações controladas sem ruído de fundo, por exemplo.

Observamos um comportamento muito bom das redes neurais com camadas de

convolução para a tarefa de classificação de instrumentos, ainda que consideravelmente diferente da sua utilização mais conhecida, que é o reconhecimento de objetos como no ILS-VRC. Observamos também que a adição em excesso de camadas de convolução não trouxe melhoras significativas para o processo de aprendizagem como observado comparando-se a AlexNet e a GoogLeNet, apenas aumentando a complexidade do processo de treinamento e, conseqüentemente, o tempo necessário para o treinamento dos mesmos.

7.1 Trabalhos Futuros

Existem diversas modificações nesse trabalho que podem apresentar resultados interessantes para trabalhos futuros. Nossa entrada utilizou espectrogramas com janelas de 100 e 500ms com as especificações citadas na Seção 4.1 mas diferentes janelas de tempo, tanto mais curtas quanto mais longas podem expressar diferentes comportamentos na detecção de instrumentos. Podemos alterar também as funções janela utilizadas para variar o vazamento espectral e observar o comportamento do aprendizado nas redes.

Além do espectrograma existem outras diversas formas de se trabalhar com representações de áudio. Trabalhar com a entrada crua, isso é, valor numérico que foi tomado em cada amostra de tempo é uma ótima maneira de se extrair o máximo possível de informações dos dados uma vez que não seria efetuada nenhuma compressão. Entretanto, seria necessário um modelo que fosse capaz de suportar essa grande quantidade de amostras como discutido na Seção 2.3.1. Existem também outras representações gráficas como o MFCC ou entropia do sinal que podem ser facilmente adaptadas à mesma estratégia discutida nesse trabalho.

A detecção de vários instrumentos em músicas pode se utilizar também de diferentes meios de gravação com um vetor de microfones para facilitar a extração e classificação de cada um com maior precisão [Yeh C. 2010] tanto com a mesma técnica como com variantes.

A utilização de redes neurais convolucionais nesse trabalho apresentou ótimo resultado, mas como apontado as amostras não possuem sequência temporal. Caso modificássemos nosso modelo para redes neurais recorrentes [Sak Senior 2011] seríamos capazes de capturar a dependência temporal e observar melhor alguma relação entre amostras sequenciais.

A presença de um especialista pode trazer a tona características que facilitem a tarefa de classificação. Existem na literatura de processamento de sinais digitais várias características clássicas para áudio como energia, taxa de *zero crossing*, entropia da energia, fluxo espectral, vetor de croma entre outros[GIANNAKOPOULOS 2014]. Essas características podem ser agrupadas em forma de tabela e comparadas com outros algoritmos de aprendizado de máquina. Esse trabalho em particular focou em extrair informação sem o conhecimento do especialista mas é natural que a presença do mesmo seria um excelente fator para melhoria dos resultados.

Existe uma técnica em aprendizado de máquina chamada *Ensemble* que consiste em misturar mais de uma técnica para obter bons resultados. Uma das técnicas pensadas para esse projeto foi, por exemplo, tomar a última camada completamente conectada da AlexNet e conectar as saídas dos neurônios a algum outro classificador. Isso poderia trazer melhorias pois diferentes técnicas apresentam diferentes problemas e vantagens que podem ser contornados ou diminuídos com a utilização de mais de uma técnica.

Referências Bibliográficas

- [Abdel-Hamid et al. 2012]ABDEL-HAMID, O. et al. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2012. p. 4277–4280. ISSN 1520-6149.
- [Agostini, Longari e Pollastri 2003]AGOSTINI, G.; LONGARI, M.; POLLASTRI, E. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, Hindawi Publishing Corp., New York, NY, United States, v. 2003, p. 5–14, jan. 2003. ISSN 1110-8657. Disponível em: <<http://dx.doi.org/10.1155/S1110865703210118>>.
- [Barbedo e Tzanetakis 2011]BARBEDO, J. G. A.; TZANETAKIS, G. Musical instrument classification using individual partials. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 19, n. 1, p. 111–122, Jan 2011. ISSN 1558-7916.
- [Barcelona 2012]BARCELONA, M. T. G. U. P. F. *IRMAS: A DATASET FOR INSTRUMENT RECOGNITION IN MUSICAL AUDIO SIGNALS*. 2012. Disponível em: <<http://www.mtg.upf.edu/download/datasets/irmas>>.
- [Bishop 2006]BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387310738.
- [Cemgil A. T. 1998]CEMGIL A. T., G. F. Classification of musical instrument sounds using neural networks. *CiteSeer*, jan 1998.
- [Deng e Yu 2014]DENG, L.; YU, D. Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, v. 7, n. 3–4, p. 197–387, 2014. ISSN 1932-8346. Disponível em: <<http://dx.doi.org/10.1561/20000000039>>.

- [Feiten B. 1994]FEITEN B., G. S. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, dec 1994.
- [FISHER 1936]FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, Blackwell Publishing Ltd, v. 7, n. 2, p. 179–188, 1936. ISSN 2050-1439. Disponível em: <<http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x>>.
- [Fourier 1822]FOURIER, J. J. *Théorie analytique de la chaleur*. [S.l.]: Paris: Firmin Didot, père et fils, 1822.
- [Geerts e Sansen 2002]GEERTS, M. S. Y.; SANSEN, W. *Design of multi-bit delta-sigma A/D converters*. [S.l.]: Springer, 2002. ISBN 1-4020-7078-0.
- [Gelfand 2011]GELFAND, S. *Essentials of Audiology*. [S.l.]: Thieme, 2011. P. 87.
- [GIANNAKOPOULOS 2014]GIANNAKOPOULOS, A. P. T. *Introduction to AUDIO ANALYSIS: A MATLAB Approach*. [S.l.]: Academic Press, 2014.
- [Harris 1978]HARRIS, F. J. On the use of windows for harmonic analysis with discrete fourier transform. *Proceedings of the IEEE*, v. 66, p. 51–83, jan 1978.
- [Herrera-Boyer P. Geoffroy 2003]HERRERA-BOYER P. GEOFFROY, P. S. D. Automatic classification of musical instrument sounds. *Journal of New Music Research*, v. 32, n. 1, p. 3–21, 2003. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1076/jnmr.32.1.3.16798>>.
- [Humphrey, Bello e LeCun 2013]HUMPHREY, E.; BELLO, J.; LECUN, Y. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, Springer US, v. 41, n. 3, p. 461–481, 2013. ISSN 0925-9902. Disponível em: <<http://dx.doi.org/10.1007/s10844-013-0248-5>>.
- [Jia et al. 2014]JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [Kaminsky Materka 1995]KAMINSKY MATERKA, A. I. Automatic source identification of monophonic musical instrument sounds. *Neural Networks, 1995. Proceedings., IEEE International Conference*, dec 1995.

- [Kohonen 1995]KOHONEN, T. Self-organizing maps. 2nd ed. *Computer Music Journal*, jan 1995.
- [Krizhevsky Ilya Sutskever 2012]KRIZHEVSKY ILYA SUTSKEVER, G. E. H. A. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.
- [Lartillot 2007]LARTILLOT, P. T. O. *A Matlab Toolbox for Musical Feature Extraction From Audio*. [S.l.]: Elsevier, 2007.
- [Le et al. 2012]LE, Q. et al. Building high-level features using large scale unsupervised learning. In: *International Conference in Machine Learning*. [S.l.: s.n.], 2012.
- [LeCun, Bengio e Hinton 2015]LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 521, n. 7553, p. 436–444, May 2015. ISSN 0028-0836. Insight. Disponível em: <<http://dx.doi.org/10.1038/nature14539>>.
- [LeCun Y. 1989]LECUN Y., B. J. S. B. D. D. H. R. E. H. W. H. L. D. J. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, p. 541–551, 1989.
- [Lee et al. 2009]LEE, H. et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009. (ICML '09), p. 609–616. ISBN 978-1-60558-516-1. Disponível em: <<http://doi.acm.org/10.1145/1553374.1553453>>.
- [Lee Yan Largman e Ng. 2009]LEE YAN LARGMAN, P. P. H.; NG., A. Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. *NIPS*, v. 1, 2009.
- [Li, Chan e Chun 2010]LI, T. L.; CHAN, A. B.; CHUN, A. Automatic musical pattern feature extraction using convolutional neural network. In: *Proc. Int. Conf. Data Mining and Applications*. [S.l.: s.n.], 2010.
- [Lots S. 2008]LOTS S., L. S. Perception of musical consonance and dissonance: an outcome of neural synchronization. *Journal of The Royal Society Interface*, The Royal Society, v. 5, n. 29, p. 1429–1434, 2008. ISSN 1742-5689. Disponível em: <<http://rsif.royalsocietypublishing.org/content/5/29/1429>>.

- [McCulloch e Pitts 1943]MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, n. 4, p. 115–133, 1943. ISSN 1522-9602. Disponível em: <<http://dx.doi.org/10.1007/BF02478259>>.
- [Minsky 1969]MINSKY, M. S. P. An introduction to computational geometry. *MIT Press*, 1969.
- [NVIDIA]NVIDIA. *NVIDIA Deep Learning GPU Training System (DIGITS)*. Disponível em: <<https://github.com/NVIDIA/DIGITS>>.
- [Nyquist 2002]NYQUIST, H. Certain topics in telegraph transmission theory. *Reprint as classic paper in: Proc. IEEE*, v. 90, p. 617–644, 2002.
- [Orchestra 2009]ORCHESTRA, P. *SOUND SAMPLES*. 2009. Disponível em: <<http://www.philharmonia.co.uk>>.
- [Picone 1996]PICONE, J. *FUNDAMENTALS OF SPEECH RECOGNITION: A SHORT COURSE*. [S.l.]: INSTITUTE FOR SIGNAL AND INFORMATION PROCESSING, 1996. Department of Electrical and Computer Engineering, Mississippi State University.
- [Priddy K. L. 2005]PRIDDY K. L., K. P. E. *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68)*. [S.l.]: SPIE- International Society for Optical Engineering, 2005. ISBN 0819459879.
- [Rabiner e Juang 1993]RABINER, L. R.; JUANG, B. H. *Fundamentals of speech recognition*. United states ed. PTR Prentice Hall, 1993. Paperback. ISBN 0130151572. Disponível em: <<http://www.worldcat.org/isbn/0130151572>>.
- [Rosen 2011]ROSEN, S. *Signals and Systems for Speech and Hearing (2nd ed.)*. [S.l.]: BRILL, 2011.
- [Rossing 2007]ROSSING, T. *Springer Handbook of Acoustics*. [S.l.]: Springer, 2007. P. 747, 748. ISBN 978-0387304465.
- [Rumelhart D. E. 1986]RUMELHART D. E., H. G. E. W. R. J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, oct 1986.
- [Sainath Ron J. Weiss 2015]SAINATH RON J. WEISS, A. S. K. W. W. O. V. T. N. Learning the speech front-end with raw waveform cldnns. *Interspeech 2015*, 2015.

- [Sak Senior 2011]SAK SENIOR, A. W. B. F. H. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proc. Interspeech*, p. 338–342, sep 2011.
- [Sigtia S. 2014]SIGTIA S., B. E. C. S. W. T. G. A. d. . D. S. An rnn-based music language model for improving automatic music transcription. *15th International Society for Music Information Retrieval Conference (ISMIR)*, oct 2014.
- [Srivastava Geoffrey Hinton 2014]SRIVASTAVA GEOFFREY HINTON, A. K. I. S. R. S. N. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15 (2014), 2014.
- [Szegedy et al. 2014]SZEGEDY, C. et al. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. Disponível em: <<http://arxiv.org/abs/1409.4842>>.
- [Szegedy et al. 2014]SZEGEDY, C. et al. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. Disponível em: <<http://arxiv.org/abs/1409.4842>>.
- [Toiviainen P. 1998]TOIVIAINEN P., T. M. L. J. H. M. Timbre similarity: Convergence of neural, behavioral, and computational approaches. *Music Perception*, dec 1998.
- [Uzan e Wolf 2015]UZAN, L.; WOLF, L. I know that voice: Identifying the voice actor behind the voice. In: *2015 International Conference on Biometrics (ICB)*. [S.l.: s.n.], 2015. p. 46–51. ISSN 2376-4201.
- [Watkinson 2000]WATKINSON, J. *The Art of Digital Audio*. second. [S.l.]: Focal Press, 2000. Pg. 104, ISBN 978-0240515878.
- [Yeh C. 2010]YEH C., R. A. R. X. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 18, n. 6, p. 1116–1126, Aug 2010. ISSN 1558-7916.