dissertação

**Resumo**

Palavras-chave:

**Abstract**

Keywords:

Lista de Quadros

Lista de Figuras

Lista de Tabelas

Lista de Siglas

Sumário

# Introdução

Inteligência artificial é um dos campos da ciência mais explorados da atualidade, sendo utilizada em uma gama crescente de aplicações que cobrem desde o entretenimento à saúde, passando por campos como segurança e políticas públicas. Sundar Pichai, CEO do Google, afirmou recentemente em um discurso no Fórum Econômico Mundial que a “inteligência artificial é provavelmente a coisa mais importante na qual a humanidade já trabalhou”, equiparando sua importância à eletricidade e ao fogo e declarando ainda que esta tecnologia é o coração do Google, uma das maiores empresas do mundo.(“Google Ceo Sundar Pichai: Artificial Intelligence More ’Profound Than Electricity or Fire’,” n.d.)

Nesse contexto mais amplo, redes neurais artificiais possuem um papel primordial, pois tem sido, em suas diversas encarnações e arquiteturas, o principal representante da inteligência artificial em diversas aplicações práticas.

Multidisciplinar desde o nascimento, o desenvolvimento das redes neurais artificiais pode ser remontado aos primeiros esforços para sistematização teórica da forma como o cérebro humano funciona, a partir dos trabalhos de Hermann von Helmholtz, Ernst Mach e Ivan Pavlov, na virada do século 19(Hagan et al. 1996).

## Problema: simulação de instrumentos acústicos (aplicação: bateria)

Um levantamento bibliográfico revela que trabalhos relacionados à utilização de redes neurais para a simulação de instrumentos musicais ainda são bastante escassos, a despeito do sucesso desta ferramenta em áreas direta ou indiretamente correlatas. Recentemente, a área de computer vision, por exemplo, tem recebido bastante atenção, com a aplicação e refinamento de novas arquiteturas e métodos gerando resultados inovadores em áreas como a geração procedural de imagens, colorização automática ou semi-automática de imagens em preto e branco, transferência de estilo entre outras.

Esses resultados estimulam a transposição de algumas dessas técnicas e insights para o caso da análise sonora, principalmente quando percebemos que o tratamento da representação do som em forma digital, que é comumente representada na forma de um vetor unidimensional de posições da onda sonora em relação ao tempo, é um caso específico da representação de imagens, que tem sido representadas como um vetor bi ou até tridimensional.

**XXX** adicionar imagem comparativa

Observa-se ainda que a representação sonora oferece, ao contrário das imagens, necessariamente uma correlação temporal bastante forte, que pode ser explorada por arquiteturas conhecidas, como as redes recorrentes, por exemplo.

## Questão de pesquisa

Busca-se, desta forma, identificar metodologia aplicável à codificação de instrumentos acústicos em uma rede neural permitindo que a mesma, depois de treinada, seja capaz de gerar os diversos sons produzidos pelo instrumento original de maneira realista, principalmente do ponto de vista da percepção humana.

A proposta inclui a identificação da arquitetura mais apropriada, assim como seus parâmetros, incluindo o número de camadas, neurônios, a mais apropriada forma de atualização dos pesos e eventuais formas de tratamento dos dados utilizados no treinamento.

## Escopo

## Estrutura

## Relevância

# Revisão Bibliográfica

## Redes Neurais Artificiais

Em 1943, o neurofisiologista Warren McCulloch e o matemático Walter Pitts foram responsáveis por formular o primeiro modelo matemático conhecido do cérebro humano(McCulloch and Pitts 1943), mostrando que topologias simples podem, em princípio, encarregar-se de operações aritméticas e lógicas(Yadav, Yadav, and Kumar 2015) complexas.

O primeiro passo para a utilização prática das redes neurais foi dado no final da década de 1950 por Frank Rosenblatt(Hagan et al. 1996), em sua proposta ao laboratório aeronáutico de Cornell de um autômato baseado em seu modelo simplificado de neurônio, o perceptron(Rosenblatt 1957).

Tomando como base também as teorias de McCulloch, Widrow desenvolveu a ADALINE (Adaptive Linear Neuron)(Widrow and Hoff 1960) bastante similar em estrutura ao modelo de Rosenblatt, e partilhando de suas limitações. O modelo de treinamento proposto, no entanto, era consideravelmente mais robusto, sendo utilizado até hoje.

A motivação final de Rosenblatt, inspirada nas teorias sobre o funcionamento do cérebro humano apresentadas por autores como Culbertson, Von Neumann e Ashby (Rosenblatt 1958) era a construção de uma máquina capaz de aprender à responder diretamente à estímulos físicos externos, como sinais luminosos.

Para tanto, tal máquina utilizaria como unidade fundamental o Perceptron, que funciona matematicamente como uma função aplicada à soma ponderada das entradas e dos viés dando origem a um classificador linear, capaz de atualizar seus pesos para aprender, através de exemplos de inputs e os correspondentes outputs desejáveis, a separar linearmente classes diferentes.

A despeito das limitações práticas do modelo proposto, como expostas de maneira um tanto pessimista por Minsky(Minski and Papert 1969), é interessante notar que muitos dos princípios presentes contemporaneamente no campo foram já introduzidos nos trabalho seminais de Rosenblatt e Widrow, notadamente o caráter estocástico das redes neurais, a necessidade de treiná-las com base em grandes conjuntos de dados, a característica de black box do modelo treinado. O último introduziu, como forma de treinamento, uma caso restrito do algoritmo de descida em gradiente.

O modelo de Rosenblat, tendo tido sua primeira implementação na forma de uma simulação em um computador IBM 704 (Bishop 2006), ao contrário da intenção inicial de utilizar hardware específico, inaugura também a prática da simulação neural.

Enquanto uma arquitetura baseada em uma única camada de Perceptrons apresenta severas limitações, o agrupamento sucessivo dessas camadas dá origem a uma topologia, conhecida como Multilayer Perceptron, capaz de atuar como um aproximador de uma função qualquer(Hornik 1991), limitada apenas pelo número de neurônios ocultos. e Desde que utilize uma função de ativação não-linear(Leshno et al. 1993) essa topologia é capaz de mapear qualquer conjunto de números finitos a qualquer outro com precisão arbitrária(Hornik, Stinchcombe, and White 1989) .

Um outro impedimento prático do trabalhos de Rosenblatt e Widrow foi a ausência de uma metodologia eficiente para a atualização dos pesos da rede, sobretudo envolvendo múltiplas camadas de neurônios; tal metodologia foi proposta por Werbos em 1974. Contudo, essa técnica permaneceu pouco conhecida na comunidade até ser redescoberta por Parker(Parker 1985) e, pouco tempo depois, Rumelhart(Rumelhart, Hinton, and Williams 1985) na segunda metade da década de 1980 (Mizutani, Dreyfus, and Nishio 2000) (Widrow and Lehr 1990). O algoritmo, que é conhecido como backpropagation, foi um dos responsáveis por reaquecer o interesse no campo(Hagan et al. 1996) na época, inaugurando sua fase atual, com o surgimento dos principais congressos sobre o assunto, como o IEEE International Conference on Neural Networks e periódicos, a exemplo do INNS Neural Networks, ao fim da década de 1980(Yadav, Yadav, and Kumar 2015).

Nos anos seguintes, observou-se uma profusão de novas arquiteturas, que foram aprofundando-se com a utilização de mais camadas, na medida dos avanços no hardware computacional. Outras formas de organizar camadas sucessivas foram também introduzidas, além de vários avanços nos algoritmos de treinamento.

## Arquiteturas Proeminentes

Aqui são introduzidas as arquiteturas mais relevantes para o tema proposto.

### Feed-Forward

Tem-se que a arquitetura básica é a chamada de Feed-Forward, que consiste de várias camadas sucessivas, que são totalmente ligadas entre si por meio de pesos. Os impulsos recebidos pelas camadas mais baixas fluem sucessivamente para as camadas posteriores, como sugere a nomenclatura desta arquitetura. São uma generalização do Multilayer Perceptron na medida em que utilizam em geral uma gama mais vasta de funções de ativação, muitas delas de forma sigmoidal, como a função logística and the hyperbolic tangent (Goldberg 2016)

### Redes Neurais Profundas

Com a melhoria do hardware, a topologia Feed-Forward foi ganhando um acréscimo de camadas, gerando as redes profundas (Deep Neural Networks). O aspecto mais importante dessa movimentação foi que as redes ganharam a habilidade de gera representações sucessivas, abstraindo diferentes aspectos dos dados em cada uma de suas camadas.

A grande vantagem desta topologia foi sua capacidade automática de extração de ‘features’, um trabalho que ficava a cargo dos pesquisadores anteriormente(Socher 2014). Em contrapartida, já forma como os dados são interpretados pela rede pode não ser facilmente inferida pelo pesquisador, e as redes podem assumir uma forte característica de caixa-preta.

### Redes Recorrentes

Em redes recorrentes os neuros são parcialmente alimentados com seus próprios estados anteriores, emulando um efeito similar à utilização de ligações entre neurons de uma camada abaixo com neurons de uma camada acima não adjacente (Veit, Wilber, and Belongie 2016).

Essa arquitetura foi proposta por Elman (Elman 1990) em 1990 com o propósito de capturar informações codificadas no encadeamento temporal os dados, e é bastante poderosa em várias aplicações, como modelos de previsão e classificação de informações(Xu, Auli, and Clark 2015), por exemplo.

### Redes Convolucionais

Trata-se de um tipo de arquitetura profunda, que é amplamente utilizado em problemas relacionados a imagens, atingindo resultados de ponta em várias áreas de visão de computadores (computer vision), como reconhecimento de objetos e rostos em imagens (Pang et al. 2017).

Esse tipo de arquitetura lida com o problema da alta dimensão de uma imagem substituindo camadas totalmente conectadas por camadas convolucionais, que varrem a imagem, movimentando-se em uma de suas dimensões um passo por vez cada vez (LeCun et al. 1998), e atualizando os pesos de acordo.

Esse procedimento permite a geração, nas camadas convolucionais da rede, de uma só representação para padrões que aparecem em diferentes pontos da imagem; tais representações são geralmente interpretadas nas camadas finais da rede, totalmente conectadas, de forma a gerar o resultado final.

### Generative Adversarial Networks

This technique/topology introduces the concept of one generative deep network creating content to be classified as artificially generated or not by another network (Goodfellow et al. 2014).

### Capsulenets

## Utilização de ANNs em áreas correlatas

### Speech

An year before, (Hinton et al. 2012) present an overview of the use of neural networks based approach in the field, while in this same year, (Graves, Mohamed, and Hinton 2013) produces state of the art results in the TIMIT phoneme recognition benchmark using a deep recurrent neural network, while (Maas, Hannun, and Ng 2013) points the superiority of rectifier nonlinearities over sigmodal activation functions in the task of continuous speech recognition.

With a hybrid architecture, combining a recurrent phonetic model and a deep neural network acoustic classifier, (Boulanger-Lewandowski et al. 2014) applies phone sequencing strategies to set new benchmarks in the TIMIT dataset, a technique that is proved by (Sak et al. 2015) to be superior to architectures like deep long short-Term memory recurrent neural networks and the widely used hidden Markov models.

(Sainath et al. 2015) investigates the optimization of convolutional nets hyperparameters, pooling and training strategies to applications in speech recognition tasks. (Zweig et al. 2017) and (Ying Zhang et al. 2017) investigates end-to-end systems, with the latter combining hierarchical convolutional nets with Connectionist Temporal Classification. In his work, (Yu Zhang, Chan, and Jaitly 2017) also explores end-to-end systems, via a very deep recurrent convolutional network employing NIN principles.

Speech synthesis state of the art, as seen in text-to-speech applications, for instance, is not yet achieved via end-to-end neural network approaches. Nonetheless, this area is being actively researched, and approaching production quality rapidly. (Zen and Sak 2015) tackles this task using unidirectional long short-term memory recurrent neural networks with a recurrent output layer, while (Wu and King 2016) further investigates this architecture, trying to discover the reasons for its effectiveness, and pinpoint wich factor are more relevant to que quality of the task, with the goal of offering a simplified topology.

### Music

Many language processing techniques are used in the music field; similarly, many image tasks can be translated to music, given a suitable representation for the input data, like spectrograms.

#### Classification

This task involves assigning tags, generally genre-related or emotion related, to musical pieces. (Costa, Oliveira, and Silla 2017) and (Choi, Fazekas, and Sandler 2016) tackles this task with a fully convolutional neural network fed, in the latter case, with music represented by a mel-spectrogram, while in (Choi et al. 2017) a recurrent architecture is also explored, to exploit the temporal correlation of the inputs.

#### Transcription

the common task in this field is to translate music parts, generally specific instruments, into a symbolic representation, like tablatures or music scores, for example. One of the first contributions to this field is seen in (Tuohy, Potter, and Center 2006) via the coupling of a network and a local heuristic hill-climber applied over the results, to generate tablatures from music.

With the use of a recurrent net, (Boulanger-Lewandowski, Bengio, and Vincent 2013) transcribes spectrograms of general musical parts into piano roll midi commands, while (Böck and Schedl 2012) offers a similar approach, restricted to polyphonic piano sound, as in the case of (Sigtia, Benetos, and Dixon 2016). The last work, however, uses different architectures for the acoustic, a simple network and the language, a recurrent network model. With the use of a bidirectional recurrent net fed with spectral representations, (Southall, Stables, and Hockman 2016), creates drum representations.

#### Generation

Here the aim is the inverse of that in the music transcription: given a representation, audio output is generated. One of the first works in the field is seen in(Stanley 2007), consisting of a compositional pattern producing network generating music on the fly based on user input. (Hutchings 2017) generates full drum parts based on a kick drum pattern, with a recurrent net, and investigates the quality of the results via an online survey.

### Image Compression

Despite beeing an important field in face of the communication era demands(Rehman, Sharif, and Raza 2014), the last review about the use of neural networks in image compression was made in 1999(Jiang 1999), while the last general review about the area dates from 2014(Rehman, Sharif, and Raza 2014).

(Ballé, Laparra, and Simoncelli 2016) proposes the use of stages of linear convolutional filters and non-linear activation functions, improving the Multiscale Structural Similarity for Image Quality Assessment(MS-SSIM) measure in all bitrates.

Superior measurements, with the same metrics, to standard compression methods such as JPEG and WebP, were obtained by (Johnston et al. 2017), using a recurrent architecture with SSIM loss function and adaptive bit allocation algorithm. With a modified loss function, (Theis et al. 2017) shows that autoencoders can achieve compressing results comparable with JPEG 2000 format.

(Toderici et al. 2016) also produces results better than standard codecs via different architectures based on recurrent neural networks, a binarizer, and a neural network for entropy coding.

Using a fuzzy neural network, (Wang and Gao 2015) achieves superior speed, robustness and quality in lossy image processing tasks. (Toderici et al. 2015) presents a progressive method, focused on reducing mobile phone data transfer, that allows arbitrary image quality depending on the quantity bits sent to the device. (Santurkar, Budden, and Shavit 2017) investigates the resilience of neural networks based compression, via a generative model capable of offering graceful degradation on the compressed images. Conceptual compressing is investigated by (Gregor et al. 2016), a technique that allows images to be retrieved from symbols. All the literature investigate deals with lossy compression.

### Image Synthesis

(Isola et al. 2016) presents a method to translate images via an adversarial network, generating images from outlines, for example, or providing automatic image colorization (Hwang and Zhou 2016), (Zhang, Isola, and Efros 2016), (Larsson, Maire, and Shakhnarovich 2016) and (Iizuka, Simo-Serra, and Ishikawa 2016), the last focusing on automatic image colorization via a convolutional network, based on the extraction of local and global features, learned in the supervised training process.

Also using adversarial nets, (Frans 2017) introduces control to the process of generating coloured images from sketches, with the use of colour maps fed into the , a concept that is further explored in (Sangkloy et al. 2016), where user interaction and a feed-forward architecture enables real-time colorization of images via the input of colour clues via scribbles in arbitrary areas of the image.

(Gatys, Ecker, and Bethge 2016) tackles the problem of style transfer: extracting features of an image and applying to another one, without changing the semantics of the latter, with the application of a convolutional network.

(Kulkarni et al. 2015), on the other hand, proposes a method that enables a convolutional-deconvolutional network to disentangle the features extracted from images, allowing the manual generation of images in different positions and lighting conditions via the tweak of variables fed to the network. Also investigating image manipulation, by means of adversarial nets, (Zhu et al. 2016) attempts to learn features directly from raw data.

(Oord, Kalchbrenner, and Kavukcuoglu 2016) proposes a deep recurrent topology, with improved residual connections, capable of reconstructing occluded images, that could be also used in image compression tasks.

(Theis and Bethge 2015) investigates a recurrent architecture composed of multi-dimensional long short term memory units in the context of modelling image distributions.

### Video

Problems in this area can be understood as a general case of aforementioned image-related tasks, with the added complexity of taking advantage of temporal correlations and a much higher data dimensionality. Motivated by this, (Karpathy et al. 2014) investigates approaches capable of extending convolutional neural networks in order to enable them to take advantage of temporal information in the inputted data. In a similar attempt, extending facial expression recognition to videos, (Khorrami et al. 2016) merges convolutional and recurrent networks, measuring the relative relevance of each one in the final results. Tackling the curse of dimensionality, (Yang, Krompass, and Tresp 2017) presents the tensor-train concept, enabling the transfer of improvements of other architectures to high dimensional sequential data. (He et al. 2015) uses a deep bidirectional long short-term memory recurrent neural net to set benchmarks in the recognition of emotions, via audio and video processing.

## O estado da arte da síntese sonora

## ANN Frameworks

A área de machine learning encontra-se muito ativa atualmente, tanto no âmbito da pesquisa quanto no empresarial, com várias grandes empresas de tecnologia incorporando essa tecnologia em suas competências essenciais. Dessa forma, assistimos à uma proliferação de ferramentas e frameworks focados em diferentes aspectos da área, muitos deles desenvolvidos ou endossados por essas empresas, como Microsoft, Google e Amazon. Somando-se a isso o fato de que, em maior ou menor grau, todos eles apresentam uma curva de aprendizado significativa, fica evidente que uma comparação direta de todos, ou mesmo da maioria das ferramentas disponíveis, torna-se impraticável em um horizonte de tempo razoável.

A literatura disponível sobre o tema é bastante escassa, talvez pela volatilidade do tema.

## Algoritmos de Otimização

## Transformada Discreta de Fourier

A transformada de Fourier, em sua forma contínua, é um tipo de transformada integral, como abaixo:

Tendo sido formulada por Fourier, enquanto investigava o fenômeno da transferência de calor **xxx**, trata-se de uma das ferramentas mais utilizadas na investigação de sistemas físicos (Lyons 2011), sendo de especial importância no campo de processamento de sinais **xxx**.

Esta ferramenta permite que a descrição de um sinal , em relação ao tempo, seja transformada em uma representação, deste mesmo sinal, no domínio da frequência, na forma ; e são denominados par de Fourier.

Com o advento dos computadores, e a emergência de sinais digitais, uma versão discreta foi formulada **xxx**, e pode ser definida como abaixo:

onde e são sequências de N números complexos. Utilizando a fórmula de Euler, , podemos alterar a definição acima, da forma exponencial, para a trigonométrica, reorganizando o expoente de forma a evidenciar algumas características importantes, na forma abaixo.

A forma acima torna mais fácil uma interpretação geométrica do algoritmo. Para cada valor de é feita a multiplicação, elemento a elemento, das N posições do impulso com N posições de um onda em forma de cosseno (), pura, de “frequência” m, na parte real da equação. Analogamente, em sua parte imaginária, o mesmo processo ocorre: cada posição medida do impulso também é multiplicada pela posição equivalente, desta vez, de uma onda em formato de seno(), com “frequência” m.

Note-se que esta “frequência”, e por isso a insistência nas aspas, refere-se a quantos ciclos das ondas () e () acima definidas estão contidos no espaço dos N pontos utilizados na transformada e não pode, per-se, ser relacionada à frequência física da onda.

Para o caso de sinais que, no domínio do tempo, consistem de uma sequência finita de números reais, que são os considerados neste trabalho, pode-se, à luz da definição acima, derivar uma interessante e útil propriedade do algoritmo, que adiantamos: A descrição do sinal no domínio da frequência é simétrica, de forma que , com $X^\* $ denotando o conjugado complexo de . Dessa forma, sendo par, precisamos de apenas de termos de para descrever completamente o pulso no domínio da frequência, enquanto são suficientes caso seja ímpar.

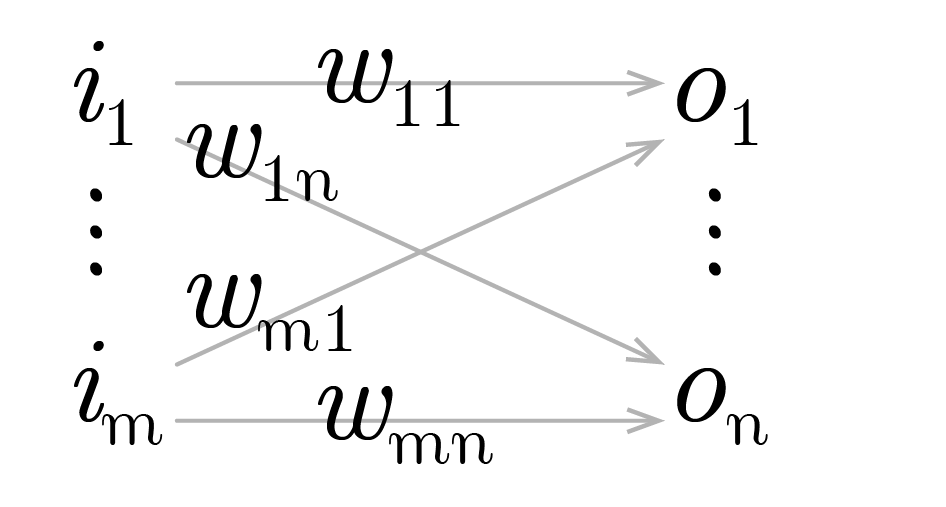
O código em Python abaixo, amplamente comentado, serve para elucidar o exposto. Cabe observar que implementações mais eficientes, como o algoritmo Fast Fourier Transform, são utilizadas na prática, se utilizam da simetria da transformada para economizar operações **xxx**

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.fftpack import fft  
  
# Discrete Fourier Transform | Transformada Discreta de Fourier  
def DFT(x):  
 N = x.shape[0]  
 # cria um vetor de N entradas complexas, preenchido com zeros  
 X = np.zeros(N, dtype='complex64')  
 n = np.array([p for p in range(N)]) # n = {0,1,2,...,N-1}  
 # Versão de "alta resolução" de n, usada para o gráfico apenas  
 Hn = np.linspace(0, N-1, 100\*N) # Hn ={0,...,1,...,2,...,N-1}  
 for m in range(N): # m E {0,1,2,...,N-1}  
 C\_Si = np.exp(-2 \* np.pi \* m \* n / N \* 1j) # C(m) - S(m) i  
 # Versão de "alta resolução" de C\_Si, usada para o gráfico apenas  
 HC\_Si = np.exp(-2 \* np.pi \* m \* Hn / N \* 1j)  
 X[m] = sum(np.multiply(x, C\_Si)) # multiplicação termo a termo  
 # plotando:  
 plt.figure(1)  
 plt.suptitle('M =' + str(m), fontsize=16)  
 plt.subplot(211)  
 plt.plot(n,x[n].real, 'k:.', label='x[n]')  
 plt.plot(n,C\_Si.real, 'k--o', label='C[n]')  
 plt.plot(Hn,HC\_Si.real, 'k-',label='C[n] ideal')  
 plt.ylabel('Real', fontsize=14, color='k')   
 plt.subplot(212)  
 plt.plot(n,x[n].imag, 'k:.', label='x[n]')  
 plt.plot(n,C\_Si.imag, 'k--o', label='C[n]')  
 plt.plot(Hn,HC\_Si.imag, 'k-',label='C[n] ideal')  
 plt.ylabel('Imaginário', fontsize=14, color='k')  
 plt.legend()  
 plt.savefig('0' + str(m) + '.png', dpi=150)  
 plt.close('all')  
 #calcula o erro em relação ao algoritmo da biblioteca Scipy  
 error = sum((X-fft(x))\*\*2)  
 print(round(error,5))  
 # retorna a parte não redundante da transformada  
 return X[0 : int(N/2+1)] if (N % 2 == 0) else X[0 : int((N+1)/2)]  
  
def s(t):  
 return np.cos(2 \* np.pi \* t)  
  
t = np.linspace(0, 1, 8) # 8 pontos entre 0 e 1  
x = s(t) # aplica a função ponto a ponto  
X = DFT(x)  
  
# plotando:  
plt.figure(1)  
plt.subplot(211)  
plt.ylabel('Real', fontsize=14, color='k')  
plt.stem(X.real,linefmt='k--',markerfmt='ko', basefmt='k--')  
plt.subplot(212)  
plt.stem(X.imag,linefmt='k--',markerfmt='ks', basefmt='k--')  
plt.ylabel('Imaginário', fontsize=14, color='k')  
plt.savefig('transform.png', dpi=150)  
plt.close('all')

# Metodologia

## Nomenclatura

Tendo em vista certa variação observada na literatura, convém formalizar a nomenclatura utilizadas neste trabalho. Define-se uma camada de uma rede neural artificial conforme o diagrama abaixo:



1. representação de uma camada de uma rede

Onde:

é um vetor de inputs à rede, e um vetor de alvos, de forma que para cada um dos ventores de input há um vetor de alvo correspondente, com a mesma dimensão da saída da camada;

é uma matriz representando os pesos das ligações entre os neurônios de entrada e saída, de forma que seria o peso da ligação entre o iésimo neurônio de entrada e o jésimo neurônio de saída.

representa cada elemento da multiplicação entre o vetor de entrada e os pesos da camada, antes da aplicação, elemento à elemento, da função de ativação;

é um vetor representando as saídas finais da camada;

representa os erros individuais da rede para cada uma das saída motivadas por cada um dos inputs. Cabe notar que é a forma mais usual de computo dos erros individuais; define-se ainda o escalar como a soma dos erros individuais para cada um dos vetores de erros.

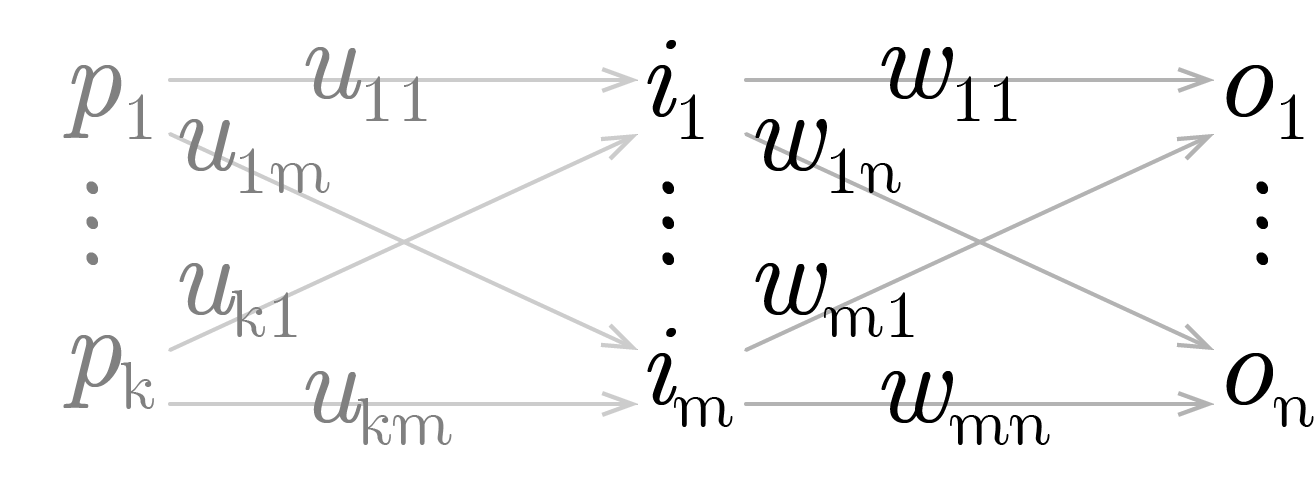
De posse das definições acima, somos capazes de definir de forma compacta, em notação matricial, o movimento de predição (forward pass) de uma camada, como será visto. E, por extensão, de uma rede, na medida em que estas podem ser trivialmente obtidas pela justaposição de um número arbitrários de camadas. O mesmo pensamento, de forma simétrica, presta-se à definição do movimento que atualizará os pesos (backward pass). Dessa forma, derivaremos a seguir a forma básica do algoritmo de backpropagation. Além de introduzir o referencial matemático, isso permitirá que as extensões desse algoritmo, como Adam e Adagrad, utilizadas adiante, possam ser mais rigorosamente compreendidas.

## Backpropagation

Tem-se que o gradiente do tensor de erros em relação a um peso arbitrário pode ser escrito da forma abaixo, com aplicação direta da regra da cadeia, e observando a seguinte notação para uma função qualquer : :

A matriz de incrementos para cada um dos pesos, a cada iteração, pode ser definida como abaixo, com a adição de uma taxa de aprendizado com sinal negativo. Assim é, uma vez que o gradiente acima aponta para sentido de maior crescimento dos erros no espaço dos pesos: minimizar os erros implica, portanto, em mover os pesos em sentido oposto.

Resta definir a forma dos erros nas entradas da rede, permitindo assim que um número arbitrário de camadas sejam conectadas e treinadas. Para tanto, convém considerar uma camada anterior à rede em tela, conceitual, com a seguinte forma e notação:



1. Duas camadas sobrepostas

E, de forma análoga:

Comparando com é fácil observar que o somatório na primeira parte da última expressão corresponde a derivada do erro na primeira equação. Podemos definir, da seguinte forma, o tensor do erro propagado:

Fixamos, dessa forma, todos os termos necessários à uma arquitetura modular: para camadas ocultas, para as quais não há a que se falar em aferirmos diretamente o erro, o mesmo é propagado a partir das camadas posteriores. Em forma vetorial, pode-se sintetizar o algoritmo como abaixo, observando a conveniência de introduzirmos o tensor para evitar redundância nos cálculos, e a notação denotando o produto de Hadamard (elemento a elemento) entre dois tensores:

## Samples Utilizados

Para a busca da coleção de samples a serem utilizados para o treinamento da rede, algumas características são essenciais. Em primeiro lugar, uma licença permissiva, que não limite a utilização, modificação e posterior divulgação dos resultados obtidos. Além disso, todas as peças de um kit ordinário devem estar presentes, e idealmente, organizadas de maneira inteligível. É essencial também que, para cada peça, samples representando várias dinâmicas tenha sido gravadas, com especial atenção à intensidade com que cada peça é golpeada (velocity).

Com isso em mente, procedeu-se à busca, em sites e blogs especializados, de indicações sobre trabalhos disponíveis que pudessem enquadrar-se nas condições citadas. Dois em especial foram selecionados: o primeiro é um esforço de disponibilizar, através da plataforma Github, uma coleção open source de sons de bateria. Mais detalhes podem ser encontrados em https://github.com/crabacus/the-open-source-drumkit. Trata-se de uma coleção de samples em formato .wav, separadas em pastas e nomeadas de acordo com a intensidade do golpe, e as peças específicas. Com uma média de 10 articulações por peça. O segundo é um esforço para a elaboração de um instrumento virtual de bateria, como descrito em https://www.drumgizmo.org e apresenta samples gravadas a partir de 5 kits. O número de articulações é bem maior, girando em torno de 20 por kit, e as gravações apresentam múltiplos microfones, com pelo menos um microfone por peça e são, em geral, de melhor qualidade. Além disso, a maioria dos kits adere à licença Creative Commons Attribution 4.0 International que permite o livre uso e adaptação do material disponibilizado em trabalhos derivados.

## Qualidade mínima necessária

## Feed Forward totalmente conectada

O presente trabalho pretende seguir, em sua metodologia, uma gradação crescente da complexidade das arquiteturas. Dessa forma, é natural que a investigação tenha início com uma análise da arquitetura Feed-Forward, com redes compostas apenas de um número arbitrário de camadas totalmente conectadas.

Em primeiro lugar, procedeu-se ao tratamento dos samples: foram escolhidos, para esta investigação inicial, 5 tambores (os 4 tons disponíveis, e o bumbo esquerdo) disponibilizados no kit Aasimonster, de cada um dos quais 20 dinâmicas (velocity) foram escolhidas. Do total de 100 samples, cada um com 16 canais (1 por microfone utilizado na gravação) foram extraídos os canais pertinentes à cada peça, resultando em 100 samples em formato .wav, com um canal por sample. As dinâmicas para cada peça foram normalizadas em grupos de 4, de forma a emular 4 samples independentes para 5 níveis diferentes de intensidade por peça. Cada um dos samples foi truncado, para possuir exatamente um segundo de duração, e um efeito fade-off foi aplicado aos samples que estendiam-se além desse intervalo.

Esse conjunto de samples, de forma bastante direta, representou os alvos da rede neural, na forma de um vetor com dimensões 100 x 44100, cada uma de suas linhas consistindo em uma representação digital da onda sonora de cada sample. A conversão foi efetuada utilizando o módulo Pywave da linguagem Python, e cada vetor foi normalizado no espaço [-1,1], e a maior amplitude guardada, de forma a ser reaplicada, depois, aos resultados da rede treinada.

Exemplificando, temos que o sample 154 corresponde ao quarto sample referente à primeira peça (o tom mais agudo), golpeada com intensidade 5. Assim é que os samples 321, 322, 323 e 324, por exemplo, correspondem às quatro variações do mesmo evento: cada um deles representa a terceira peça golpeada com intensidade 2. Os dados de entrada simplesmente refletem essa nomenclatura em um intervalo [0,1].

Nomenclatura utilizada

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Peça | Peça Norm. | Intensidade | Intensidade Norm. | Sample # | Sample # Norm. |
| 1 | 0.00 | 1 | 0.00 | 1 | 0.00 |
| 2 | 0.25 | 2 | 0.25 | 2 | 0.33 |
| 3 | 0.50 | 3 | 0.50 | 3 | 0.66 |
| 4 | 0.75 | 4 | 0.75 | 4 | 1.00 |
| 5 | 1.00 | 5 | 1.00 | - | - |

Observa-se, então, que a lógica “convencional” da utilização de redes neurais foi invertida, no melhor do nosso conhecimento, pela primeira vez na literatura: busca-se, de pronto, treinar a rede para a síntese, a partir de inputs arbitrários, de dados predefinidos. Não estamos, por exemplo, como é costumeiro, classificando as diferentes ondas. O que pretende-se é forçar uma associação de cada uma das características da onda à cada campo do input. Esse rationale tem algumas implicações importantes: critérios de validação, embora possíveis de construir, perdem bastante do seu significado e foram deixados de lado em favor de um julgamento subjetivo da qualidade dos samples gerados. Ademais, a habilidade de generalização da rede, embora desejável, não é sempre crucial, ao menos no caso específico das peças discretas de um kit de bateria; para as dinâmicas de cada peça, contudo, sim.

### 01 Algoritmos de Otimização

Utilizando os 100 samples processados como descrito, buscou-se investigar os hiperparâmetros mais adequados; a plataforma escolhida foi o Keras, como acima descrito. Em primeiro lugar, é conveniente investigar o melhor algoritmo de atualização dos pesos: embora, conceitualmente, mesmo a versão pura da descida em gradiente, utilizando uma taxa de aprendizado adequada, leve à eventual convergência da rede, na prática a velocidade dessa convergência varia consideravelmente dependendo do método utilizado. A literatura disponível, embora esclarecedora, não é unanime ao apontar uma técnica universal, de forma que uma investigação empírica faz-se necessária, e é descrita a seguir.

testou-se, em uma rede com uma camada oculta composta por 100 neurons, em minibatches de 100 inputs e durante 500 épocas os seguintes algoritmos de otimização: descida em gradiente (standard gradient descent - sgd) com taxa de aprendizado (learning rate - lr) de 0,01, o algoritmo experimental RMSProp, os otimizadores Adam, Adagrad, Adadelta, Adamax e Nadam. A função de ativação utilizada em todas as camadas foi a tangente hiperbólica (tanh), e a função de perda foi a média dos quadrados dos erros. Foi tomado cuidado para que todos os testes iniciassem em condições pseudo-randomicas idênticas.

Sgd e Adadelta ofereceram resultados bastante semelhantes, e aquém dos outros. Todos os outros convergiram de forma semelhante, sendo o Adagrad o mais rápido de todos, e Nadam o mais eficaz em diminuição da função de perda (loss).

A comparação por tempo relativo, disponível a partir do Tensorboard, permite visualizar a superioridade do algoritmo Nadam: com 2,57 minutos de treino, o tempo em que o treinamento foi concluído com o algoritmo Adagrad, ele proporcionou o menor valor da função objetivo.

Convém investigar, dessa forma, o comportamento da rede treinada com ambos os algoritmos, já que o parâmetro mais importante é a rapidez com que a rede alcança um nível de erro capaz de gerar samples com qualidade satisfatória. Após essa comparação exploratória, mais profunda, poderemos identificar tanto o valor limite da função de perda, quanto o tempo em que a rede mais eficiente leva para alcançá-lo, utilizando essas informações como parâmetros para investigações posteriores, inclusive de outras arquiteturas.

### Grid search para topologia e parâmetros

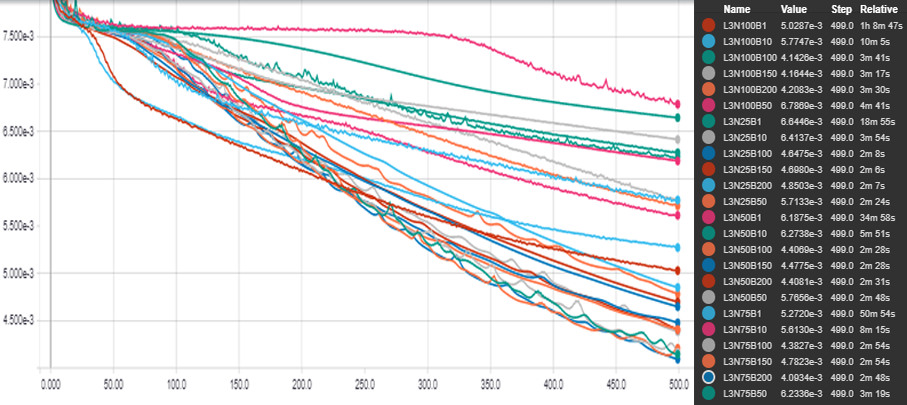
Fixado os potenciais algoritmos de otimização, procedeu-se a um grid search, para investigar o número de camadas ocultas, respectivos neurons e o efeitos dos batch sizes na rede, tanto em relação ao tempo de treinamento quanto à velocidade (e limite) de convergência.

#### Adagrad

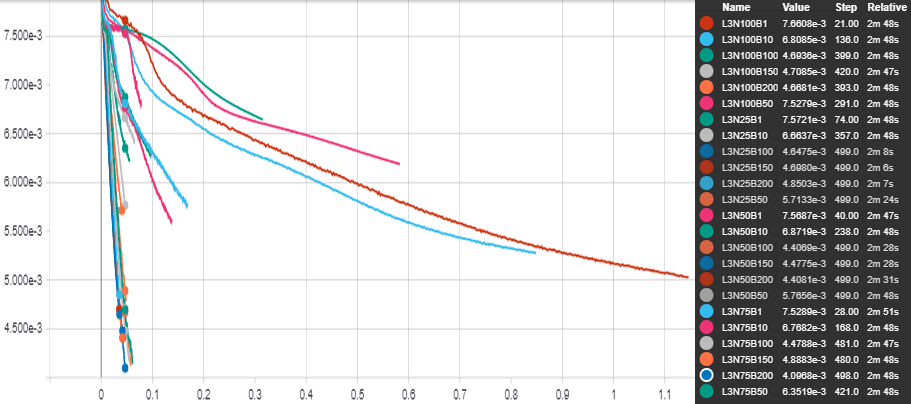
Testou-se batches de 1, 10, 50, 100, 150 e 200 samples para redes com 25,50,75 e 100 neurônios em cada camada oculta, em um intervalo de 0 a 3 camadas ocultas, gerando um total de 82 redes distintas, que foram treinadas durantes 500 épocas.

Para uma rede sem camadas ocultas, um batch igual a 1 foi o mais efetivo, em 500 épocas, porém o mais lento. batches de 100 e 50 foram os mais rápidos. O impacto no valor final da perda, no entanto, foi desprezível para todos os tamanhos de batches. para um camada oculta com 25 neurons, batches de 100 e 150 neurons ofereceram resultados idênticos, sendo os mais rápidos e mais eficientes.

A partir desse ponto, batches de 100, 150 e 200 começaram a apresentar comportamentos semelhantes. Cabe notar que batches unitários parecem piorar muito o tempo de execução na medida em que o número de neurônios aumenta. temos abaixo os melhores resultados por batch e seus respectivos tempos. Todos eles encontram-se nas redes com 3 camadas ocultas.



1. Adagrad: Loss x Step - 3 camadas ocultas



1. Adagrad: Loss x Time - 3 camadas ocultas

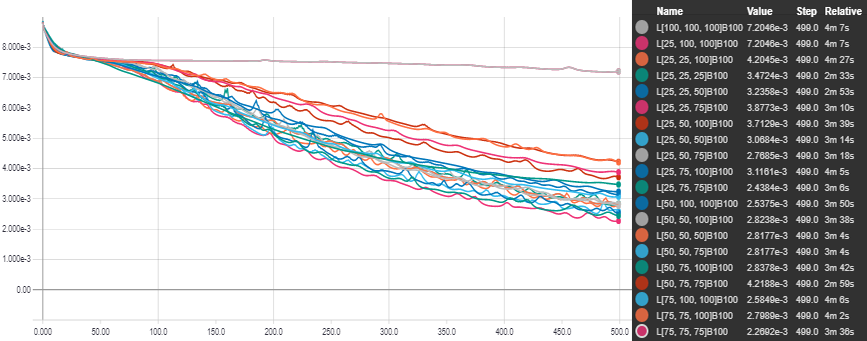
Menores valores da perda após 500 iterações

|  |  |  |  |
| --- | --- | --- | --- |
| Batches | neurônios por camada oculta | Loss (e-3) | tempo |
| 200 | 75 | 4,0934 | 2m 48s |
| 100 | 100 | 4,1426 | 3m 41s |
| 150 | 100 | 4,1644 | 3m 17s |
| 1 | 50 | 4,4069 | 2m 28s |
| 10 | 75 | 5,6130 | 8m 15s |
| 50 | 25 | 5,7133 | 2m 24s |

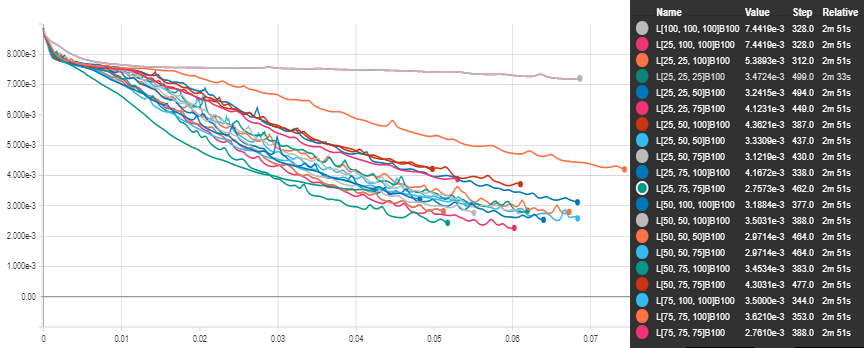
Nesse ponto fica evidente que uma arquitetura formada por 3 camadas ocultas de 75 neurons é suficiente para aprender a representar ondas, e a mais eficiente sob o otimizador Adagrad. O gráfico da função de perda em função do tempo de treinamento torna essa visualização mais fácil.

#### Nadam

Para essa etapa batches de 10, 50 e 100 samples foram utilizados; dispensou-se batches unitários, pelo comportamento ineficiente identificado, e os batches acima de 100, já que não introduziram melhorias significativas. Foram testadas, para cada um desses batches, redes com todas as combinações entre 25, 50, 75 e 100 neurônios, até o limite de 3 camadas ocultas. Os resultados são apresentados abaixo, para as arquiteturas com 3 camadas.



1. Nadam: Loss x Step - 3 camadas ocultas



1. Nadam: Loss x Time - 3 camadas ocultas

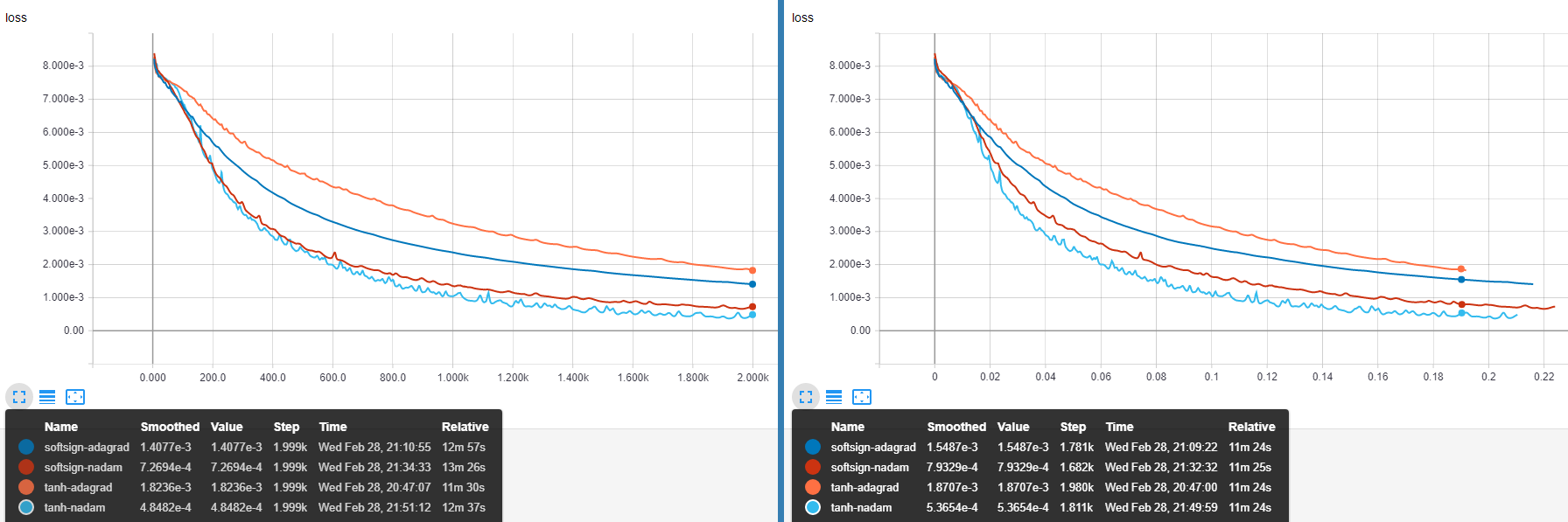
Menores valores da perda após 500 iterações

|  |  |  |  |
| --- | --- | --- | --- |
| Batches | neurônios por camada oculta | Loss (e-3) | tempo |
| 10 | 75 100 100 | 0.44632 | 16m 59 s |
| 10 | 50 50 100 | 1.5559 | 15m |
| 100 | 25 75 75 | 2,4384 | 3m 6 s |
| 50 | 75 75 75 | 2.2692 | 3m 36 s |
| 100 | 75 75 75 | 2.2692 | 3m 38 s |
| 50 | 75 100 | 4.9025 | 5m 4s |

Uma rede feedforward com 3 camadas de 75 neurons parece oferecer a arquitetura ideal, com ambos os otimizadores.

#### Ativações e Otimizadores

A performance dos dois, para 2000 steps, é comparada abaixo; também é comparado o impacto da função de ativação softsign, em relação à tanh utilizada até agora.



1. Comparação Ativações e Otimizadores

Podemos ver que a combinação do otimizador Adagrad com a função de ativação tanh é a mais eficiente.

### Baseline e Análise dos resultados preliminares

Os testes acima proporcionam um insight numérico sobre o comportamento geral da arquitetura feedforward na resolução do problema em tela. Faz-se necessário, contudo, uma análise dos samples gerados pelas redes acima, de forma a entender, do ponto de vista sensorial, sua qualidade.

Além disso, é necessário entender a capacidade de generalização das redes investigadas anteriormente. Tomando como métrica a função de perda final, podemos selecionar algumas redes para investigar os samples por elas gerados.

Analisando a rede com maior função de perda após o treinamento no caso a rede sem camadas ocultas treinada com Adagrad em batches de 100, a rede escolhida como baseline (75 neurons em cada uma de 3 camadas ocultas, otimizador Nadam, batches de 100) e a rede com a menor perda após o treinamento: a rede com 75, 100 e 100 neurons em suas camadas ocultas, treinada com Nadam em batches de 10, desenvolveremos alguma intuição acerca da relação entre a função de perda e a qualidade percebida dos samples gerados. Os dados são sintetizados abaixo:

Redes para análise de Samples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N° | Camadas | Batch | Otimizador | Valor final da função de perda |
| 01 | 75 100 100 | 10 | Nadam | 4.4632 E-4 |
| 02 | 75 75 75 | 100 | Nadam | 2.2692 E-3 |
| 03 | 0 | 100 | Adagrad | 7.6421 E-3 |

## Redes Recorrentes

## Redes Convolucionais

## Capsulenets

## Redução de Dimensionalidade dos Dados

# Resultados

# Conclusão

pandoc -o Dissert.docx Dissert.md –bibliography=bib.bib –toc

# Referências

Ballé, Johannes, Valero Laparra, and Eero P Simoncelli. 2016. “End-to-End Optimized Image Compression.” *arXiv Preprint arXiv:1611.01704*.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. springer.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent. 2013. “High-Dimensional Sequence Transduction.” In *Acoustics, Speech and Signal Processing (Icassp), 2013 Ieee International Conference on*, 3178–82. IEEE.

Boulanger-Lewandowski, Nicolas, Jasha Droppo, Mike Seltzer, and Dong Yu. 2014. “Phone Sequence Modeling with Recurrent Neural Networks.” In *Acoustics, Speech and Signal Processing (Icassp), 2014 Ieee International Conference on*, 5417–21. IEEE.

Böck, Sebastian, and Markus Schedl. 2012. “Polyphonic Piano Note Transcription with Recurrent Neural Networks.” In *Acoustics, Speech and Signal Processing (Icassp), 2012 Ieee International Conference on*, 121–24. IEEE.

Choi, Keunwoo, George Fazekas, and Mark Sandler. 2016. “Automatic Tagging Using Deep Convolutional Neural Networks.” *arXiv Preprint arXiv:1606.00298*.

Choi, Keunwoo, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. “Convolutional Recurrent Neural Networks for Music Classification.” In *Acoustics, Speech and Signal Processing (Icassp), 2017 Ieee International Conference on*, 2392–6. IEEE.

Costa, Yandre MG, Luiz S Oliveira, and Carlos N Silla. 2017. “An Evaluation of Convolutional Neural Networks for Music Classification Using Spectrograms.” *Applied Soft Computing* 52. Elsevier:28–38.

Elman, Jeffrey L. 1990. “Finding Structure in Time.” *Cognitive Science* 14 (2). Wiley Online Library:179–211.

Frans, Kevin. 2017. “Outline Colorization Through Tandem Adversarial Networks.” *arXiv Preprint arXiv:1704.08834*.

Gatys, Leon A, Alexander S Ecker, and Matthias Bethge. 2016. “Image Style Transfer Using Convolutional Neural Networks.” In *Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition*, 2414–23.

Goldberg, Yoav. 2016. “A Primer on Neural Network Models for Natural Language Processing.” *J. Artif. Intell. Res.(JAIR)* 57:345–420.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Nets.” In *Advances in Neural Information Processing Systems*, 2672–80.

“Google Ceo Sundar Pichai: Artificial Intelligence More ’Profound Than Electricity or Fire’.” n.d. *CNNMoney*. Cable News Network. <http://money.cnn.com/2018/01/24/technology/sundar-pichai-google-ai-artificial-intelligence/index.html>.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. “Speech Recognition with Deep Recurrent Neural Networks.” In *Acoustics, Speech and Signal Processing (Icassp), 2013 Ieee International Conference on*, 6645–9. IEEE.

Gregor, Karol, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. 2016. “Towards Conceptual Compression.” In *Advances in Neural Information Processing Systems*, 3549–57.

Hagan, Martin T, Howard B Demuth, Mark H Beale, and others. 1996. *Neural Network Design*. Vol. 20. Pws Pub. Boston.

He, Lang, Dongmei Jiang, Le Yang, Ercheng Pei, Peng Wu, and Hichem Sahli. 2015. “Multimodal Affective Dimension Prediction Using Deep Bidirectional Long Short-Term Memory Recurrent Neural Networks.” In *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, 73–80. ACM.

Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, et al. 2012. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups.” *IEEE Signal Processing Magazine* 29 (6). IEEE:82–97.

Hornik, Kurt. 1991. “Approximation Capabilities of Multilayer Feedforward Networks.” *Neural Networks* 4 (2). Elsevier:251–57.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. “Multilayer Feedforward Networks Are Universal Approximators.” *Neural Networks* 2 (5). Elsevier:359–66.

Hutchings, P. 2017. “Talking Drums: Generating Drum Grooves with Neural Networks.” *arXiv Preprint arXiv:1706.09558*.

Hwang, Jeff, and You Zhou. 2016. “Image Colorization with Deep Convolutional Neural Networks.”

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. “Let There Be Color!: Joint End-to-End Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification.” *ACM Transactions on Graphics (TOG)* 35 (4). ACM:110.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. “Image-to-Image Translation with Conditional Adversarial Networks.” *arXiv Preprint arXiv:1611.07004*.

Jiang, J. 1999. “Image Compression with Neural Networks–a Survey.” *Signal Processing: Image Communication* 14 (9). Elsevier:737–60.

Johnston, Nick, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. 2017. “Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks.” *arXiv Preprint arXiv:1703.10114*.

Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. “Large-Scale Video Classification with Convolutional Neural Networks.” In *Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition*, 1725–32.

Khorrami, Pooya, Tom Le Paine, Kevin Brady, Charlie Dagli, and Thomas S Huang. 2016. “How Deep Neural Networks Can Improve Emotion Recognition on Video Data.” In *Image Processing (Icip), 2016 Ieee International Conference on*, 619–23. IEEE.

Kulkarni, Tejas D, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. “Deep Convolutional Inverse Graphics Network.” In *Advances in Neural Information Processing Systems*, 2539–47.

Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. 2016. “Learning Representations for Automatic Colorization.” In *European Conference on Computer Vision*, 577–93. Springer.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. “Gradient-Based Learning Applied to Document Recognition.” *Proceedings of the IEEE* 86 (11). IEEE:2278–2324.

Leshno, Moshe, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. 1993. “Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function.” *Neural Networks* 6 (6). Elsevier:861–67.

Lyons, Richard G. 2011. *Understanding Digital Signal Processing, 3/E*. Pearson Education India.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng. 2013. “Rectifier Nonlinearities Improve Neural Network Acoustic Models.” In *Proc. ICML*. Vol. 30. 1.

McCulloch, Warren S, and Walter Pitts. 1943. “A Logical Calculus of the Ideas Immanent in Nervous Activity.” *The Bulletin of Mathematical Biophysics* 5 (4). Springer:115–33.

Minski, Marvin L, and Seymour A Papert. 1969. “Perceptrons: An Introduction to Computational Geometry.” *MA: MIT Press, Cambridge*.

Mizutani, Eiji, Stuart E Dreyfus, and Kenichi Nishio. 2000. “On Derivation of Mlp Backpropagation from the Kelley-Bryson Optimal-Control Gradient Formula and Its Application.” In *Neural Networks, 2000. IJCNN 2000, Proceedings of the Ieee-Inns-Enns International Joint Conference on*, 2:167–72. IEEE.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. “Pixel Recurrent Neural Networks.” *arXiv Preprint arXiv:1601.06759*.

Pang, Yanwei, Manli Sun, Xiaoheng Jiang, and Xuelong Li. 2017. “Convolution in Convolution for Network in Network.” *IEEE Transactions on Neural Networks and Learning Systems*. IEEE.

Parker, David B. 1985. “Learning Logic.”

Rehman, Mehwish, Muhammad Sharif, and Mudassar Raza. 2014. “Image Compression: A Survey.” *Research Journal of Applied Sciences, Engineering and Technology* 7 (4). Maxwell Science Publishing:656–72.

Rosenblatt, Frank. 1957. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory.

———. 1958. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review* 65 (6). American Psychological Association:386.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1985. “Learning Internal Representations by Error Propagation.” California Univ San Diego La Jolla Inst for Cognitive Science.

Sainath, Tara N, Brian Kingsbury, George Saon, Hagen Soltau, Abdel-rahman Mohamed, George Dahl, and Bhuvana Ramabhadran. 2015. “Deep Convolutional Neural Networks for Large-Scale Speech Tasks.” *Neural Networks* 64. Elsevier:39–48.

Sak, Haşim, Andrew Senior, Kanishka Rao, and Françoise Beaufays. 2015. “Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition.” *arXiv Preprint arXiv:1507.06947*.

Sangkloy, Patsorn, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2016. “Scribbler: Controlling Deep Image Synthesis with Sketch and Color.” *arXiv Preprint arXiv:1612.00835*.

Santurkar, Shibani, David Budden, and Nir Shavit. 2017. “Generative Compression.” *arXiv Preprint arXiv:1703.01467*.

Sigtia, Siddharth, Emmanouil Benetos, and Simon Dixon. 2016. “An End-to-End Neural Network for Polyphonic Piano Music Transcription.” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24 (5). IEEE Press:927–39.

Socher, Richard. 2014. “Recursive Deep Learning for Natural Language Processing and Computer Vision.” Citeseer.

Southall, Carl, Ryan Stables, and Jason Hockman. 2016. “Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks.” In *ISMIR*, 591–97.

Stanley, Kenneth O. 2007. “Compositional Pattern Producing Networks: A Novel Abstraction of Development.” *Genetic Programming and Evolvable Machines* 8 (2). Springer:131–62.

Theis, Lucas, and Matthias Bethge. 2015. “Generative Image Modeling Using Spatial Lstms.” In *Advances in Neural Information Processing Systems*, 1927–35.

Theis, Lucas, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. “Lossy Image Compression with Compressive Autoencoders.” *arXiv Preprint arXiv:1703.00395*.

Toderici, George, Sean M O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. 2015. “Variable Rate Image Compression with Recurrent Neural Networks.” *arXiv Preprint arXiv:1511.06085*.

Toderici, George, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. 2016. “Full Resolution Image Compression with Recurrent Neural Networks.” *arXiv Preprint arXiv:1608.05148*.

Tuohy, Daniel R, Walter D Potter, and Artificial Intelligence Center. 2006. “An Evolved Neural Network/Hc Hybrid for Tablature Creation in Ga-Based Guitar Arranging.” In *ICMC*.

Veit, Andreas, Michael J Wilber, and Serge Belongie. 2016. “Residual Networks Behave Like Ensembles of Relatively Shallow Networks.” In *Advances in Neural Information Processing Systems*, 550–58.

Wang, Bo, and Yubin Gao. 2015. “An Image Compression Scheme Based on Fuzzy Neural Network.” *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 13 (1):137–45.

Widrow, Bernard, and Marcian E Hoff. 1960. “Adaptive Switching Circuits.” STANFORD UNIV CA STANFORD ELECTRONICS LABS.

Widrow, Bernard, and Michael A Lehr. 1990. “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation.” *Proceedings of the IEEE* 78 (9). IEEE:1415–42.

Wu, Zhizheng, and Simon King. 2016. “Investigating Gated Recurrent Networks for Speech Synthesis.” In *Acoustics, Speech and Signal Processing (Icassp), 2016 Ieee International Conference on*, 5140–4. IEEE.

Xu, Wenduan, Michael Auli, and Stephen Clark. 2015. “CCG Supertagging with a Recurrent Neural Network.” In *ACL (2)*, 250–55.

Yadav, Neha, Anupam Yadav, and Manoj Kumar. 2015. An Introduction to Neural Network Methods for Differential Equations. Springer.

Yang, Yinchong, Denis Krompass, and Volker Tresp. 2017. “Tensor-Train Recurrent Neural Networks for Video Classification.” <https://arxiv.org/pdf/1707.01786.pdf>.

Zen, Heiga, and Haşim Sak. 2015. “Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis.” In *Acoustics, Speech and Signal Processing (Icassp), 2015 Ieee International Conference on*, 4470–4. IEEE.

Zhang, Richard, Phillip Isola, and Alexei A Efros. 2016. “Colorful Image Colorization.” In *European Conference on Computer Vision*, 649–66. Springer.

Zhang, Ying, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. 2017. “Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks.” *arXiv Preprint arXiv:1701.02720*.

Zhang, Yu, William Chan, and Navdeep Jaitly. 2017. “Very Deep Convolutional Networks for End-to-End Speech Recognition.” In *Acoustics, Speech and Signal Processing (Icassp), 2017 Ieee International Conference on*, 4845–9. IEEE.

Zhu, Jun-Yan, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. 2016. “Generative Visual Manipulation on the Natural Image Manifold.” In *European Conference on Computer Vision*, 597–613. Springer.

Zweig, Geoffrey, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke. 2017. “Advances in All-Neural Speech Recognition.” In *Acoustics, Speech and Signal Processing (Icassp), 2017 Ieee International Conference on*, 4805–9. IEEE.