

# Composição de Partituras Automatizadas Através de *Machine Learning*

*Automated Music Sheet Composing with Machine Learning*

André Igor Gallacci<sup>1</sup>

João Augusto Silva Lêdo<sup>2</sup>

James Clauton da Silva<sup>3</sup>

José Vital Ferraz Leão<sup>4</sup>

Leonardo César Bottaro<sup>5</sup>

Renato de Aguiar Teixeira Mendes<sup>6</sup>

## RESUMO

A partitura é uma linguagem musical padronizada, por um contexto histórico de representação universal. A partir desse contexto, a escrita de partituras utiliza-se da percepção musical e estudos específicos sobre a notação musical, ou seja sinais, símbolos e abreviações que compõem uma determinada partitura. Considerando que demanda um árduo processo de escrita manualmente; A fim de acelerar e automatizar o processo de criação de partituras, além de auxiliar a aprendizagem do manuseio de instrumentos musicais, desta forma é proposta uma ferramenta computacional que receba tais sonoridades musicais e através do emprego de inteligência artificial em *Machine Learning*, seja capaz de interpretar tais sonoridades musicais e fazer a escrita da partitura. Para tal realiza-se um levantamento bibliográfico focado em teoria musical, redes neurais e processamento digital de sinais e implementa-se em *JavaScript* e *Python* para o recebimento, processamento e apresentação do resultado. Os resultados obtidos são encorajadores, porém algumas limitações encontradas.

**Palavras-Chave:** Partitura, Música, Tempo Real, Inteligência Artificial, Aprendizagem de máquina.

---

<sup>1</sup> Acadêmico de Engenharia da Computação do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

<sup>2</sup> Acadêmico de Engenharia da Computação do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

<sup>3</sup> Professor Doutor do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

<sup>4</sup> Professor Mestre do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

<sup>5</sup> Acadêmico de Engenharia da Computação do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

<sup>6</sup> Professor Doutor do Centro Universitário Católico Salesiano Auxilium – Araçatuba-SP.

## ABSTRACT

The Music Sheet is a standard music language, with a universal representation backed by a historical context. From this context, the music notation utilizes the musical perception and specific research, meaning the use of symbols and abbreviations to compose a music score. Considering the complexity of writing such scores by hand, with the intent of accelerating, automating the process, and helping the learning stage, we propose a computer toolset that will receive, through a microphone, the sound of musical instruments, and then, by using machine learning, the computer should be able to correctly detect the music notes and then write them down as classic music notation. To this purpose, we got several references, including music theory, deep neural networks and digital signal processing, implemented in JavaScript and Python, to receive, process, and show the results. The results we obtained are encouraging, though some limitations were found.

**Key-Words:** Music Sheet, Music, Real Time, Artificial Intelligence, Machine Learning.

## INTRODUÇÃO

Mediante a um levantamento efetuado desde o fim do ano de 2016 e meados de 2017, constatou-se que no mercado, não existem quaisquer ferramentas de automação de composição de partitura que implementem *Machine Learning* para processar os dados recebidos das sonoridades em tempo real. Tal fato ocasiona lentidão no processo de criação da partitura e em muitos casos imprecisão na composição de tal.

Em razão da escassez no mercado de ferramentas de modo geral que utilizem *Machine Learning*, portanto um mercado praticamente inexplorado, surge a oportunidade de explorar os benefícios atuais que a inteligência artificial fornece na implementação de tais algoritmos. Estes algoritmos impulsionam a aplicação em questão adicionando a ela credibilidade e uma maior exatidão, pelo fato de se tratar de um aprendizado de máquina no qual ameniza-se possíveis erros existentes em uma aplicação convencional que não utilize *Machine Learning*, simplesmente aumentando a gama de aprendizado que a máquina irá receber.

A ideia principal do trabalho é suprir a escassez existente no mercado desse tipo de produto e propor uma óptica inovadora da maneira de se programar as futuras aplicações utilizando-se de métodos de *Machine Learning* e inteligência artificial, como boas práticas de futuras aplicações de qualidade no mercado.

O objetivo é propor com excelência e exatidão, um software interativo com uma interface

gráfica simples e funcional que esteja conectada ao *TensorFlow*, o qual esteja recebendo adequadamente informações matriciais de um FFT devidamente programado para receber tais sonoridades musicais.

## **METODOLOGIA**

A implementação do projeto demanda um conjunto de ferramentas, para que possa viabilizar e se tornar possível sua concretização.

Neste trabalho é realizado um levantamento bibliográfico, com foco em teoria musical, redes neurais e ferramentas de processamento digital de sinais e a implementação da programação responsável por receber, processar e apresentar os dados relacionados a formação da partitura.

O software recebe através de um leitor de FFT desenvolvido em *JavaScript* os sons captados pelo microfone do computador e os processa com base no treinamento feito previamente no *Tensorflow* desenvolvido em *Python* sob MIDIs das músicas utilizadas no *training set*, e posteriormente gerando uma saída binária de 88 posições que significam as 88 possíveis notas, cuja quais são analisadas pelo *front-end* do software que apresenta para o usuário suas notas visuais em uma partitura.

### **Captação de dados:**

Inicialmente a captação do som, será feita pelos microfones do dispositivo que está a rodar a aplicação, o som deve ser captado e tratado através de um software desenvolvido em *JavaScript* o qual implementa a transformada de Fourier que irá transformar um sinal analógico em digital. Para fazer isso, a Transformada de Fourier analisa matematicamente a onda sonora e a transforma em espectros de um sinal discreto e, por conseguinte em amostras, para que o sinal analógico possa ser tratado digitalmente. O software responsável pela digitalização do som denomina-se FFT (*Fast Fourier Transform*), ele está sendo implementado em *JavaScript* pois é a linguagem de programação da qual é desenvolvido o software todo. O software, portanto, tem sua entrada definida, que é o som, e sua saída é uma matriz de 1024 amostras do som, a cada 1024 amostras, será gerado uma nova matriz, que inicialmente alimentará o *TensorFlow*, cujo qual é responsável pelo tratamento inteligente da aplicação.

### ***Machine Learning:***

#### **Treinamento:**

O *TensorFlow*, é uma API do Google de código aberto que trabalha em seu interior com os métodos de *Deep-Learning*, atualmente propagado como um método extremamente promissor

na área de Inteligência Artificial, a linguagem de programação e estruturação do *TensorFlow* para adequar seu sistema ao software em questão será o Python.

Em posse dessas matrizes, o *TensorFlow* trabalha em dois momentos distintos, o momento de seu treinamento, para que ele possa ter uma extrema eficácia na análise do som, e o momento em que ele estará de fato analisando as sonoridades.

No primeiro momento será disponibilizado ao FFT sonoridades distintas, em seguida, será disponibilizado as notas musicais uma a uma em todas as suas oitavas, posteriormente esses dados serão retroalimentados incessantemente ao *TensorFlow* para que possa ser treinado devidamente;

Uma vez configurado com os parâmetros de reconhecimento, em posse desses dados, A biblioteca irá minimizar em cada ciclo de sua retroalimentação sua margem de erro que é medido de 0 à 1; O treinamento só termina quando a margem estiver entre 0,99 e 1.

### **Execução:**

Finalizado a primeira etapa de treinamento, segue para a segunda etapa, a de seu funcionamento, na qual o *TensorFlow*, recebe em tempo real as matrizes geradas pelo FFT através das funções da API de alto nível chamada Keras, o *TensorFlow* interpreta essas matrizes, em tempo real e a partir disso dará como saída números significativos que correspondem internamente no programa a uma nota da partitura.

### **Interface Visual da partitura:**

Posteriormente é chegada a fase visual do projeto, na qual irá disponibilizar a interface de interação entre usuário e o software, ela é desenvolvida em *JavaScript*. A parte visual irá receber os números significativos enviados pelo *TensorFlow* e os atribuir a seus respectivos símbolos na partitura, criando um ambiente visual amistoso aos olhos de um músico, ao enxergar tal partitura eletrônica, que é executada em tempo real

O motivo de se utilizar *JavaScript* como linguagem de programação oficial do projeto, é a possibilidade de se criar um projeto voltado aos navegadores web, dos quais nos permite um acesso nativo ao WebSocket e uma portabilidade adequada quanto a multiplicidade de dispositivos, dos quais se pode rodar tais aplicações. Isso cria uma possibilidade para futuras implementações, como a adequação do software para um serviço online, alocado em um servidor dedicado.

### ***VexFlow:***

É uma biblioteca *Open-source* criada em *Javascript* e disponibilizada gratuitamente no *gitHub* para a exibição de diversos tipos de notação musical, incluindo partituras convencionais.

### ***Keras:***

É uma API de redes neurais artificiais de alto nível escrita em Python que é capaz de executar sobre o *TensorFlow* e tem a função de auxiliar o desenvolvimento no *TensorFlow* pois viabiliza uma melhor performance na construção do grafo da rede neural e diminui a quantidade de código *boilerplate* necessário.

### ***Tensorflow:***

*TensorFlow* é uma ferramenta desenvolvida pelo Google com o intuito de impulsionar novas tecnologias e tendências no ramo da inteligência artificial para o consumo final, cujo qual tem a função de estimular e ajudar a criação de aplicações com uma aposta no futuro utilizando-se mais do que nunca aplicações práticas de inteligência artificial. Essa ferramenta está pautada em estudos de *Deep-Learning* cuja qual se acredita ser o método mais promissor de se implementar uma inteligência artificial atualmente.

### ***Python:***

O Python é uma linguagem poderosíssima de programação, nela existem diversas ferramentas matemáticas e um método inteligente de alocação de variáveis na memória o qual impulsiona e viabiliza implementações de inteligência artificial. A ferramenta *TensorFlow* está contida como uma biblioteca da linguagem Python para ser utilizada na criação dessas aplicações inteligentes.

### ***JavaScript:***

O *JavaScript* é a linguagem eleita para se desenvolver toda a aplicação em questão, ela é uma linguagem com enfoque em aplicações web, e por conta disso está sendo implementada no projeto, por ela nos fornecer a desejada portabilidade, cuja qual permite executar a aplicação em qualquer sistema operacional e dispositivo que suportem um navegador moderno, assim portanto aumentando a gama de consumidores, pois o *JavaScript* nos possibilita chegar a qualquer consumidor final que utilize qualquer sistema operacional em seus dispositivos

### ***Machine Learning:***

É a área de estudo na qual é dedicado em construir algoritmos que permitam que a máquina consiga aprender com os seus erros e tenha a capacidade de prever informações sem que sejam explicitamente programadas. (SAMUEL, 1959)

### ***WebSocket:***

O protocolo *WebSocket* permite a comunicação bidirecional entre um cliente executando código não-confiável em um ambiente controlado a um host remoto que optou por receber as comunicações de tal código. O protocolo consiste na abertura de um *handshake* seguido de um frame básico de mensagens, composto sobre uma camada TCP. O objetivo da tecnologia é providenciar um mecanismo para aplicações baseado em navegadores que necessitam de comunicação bidirecional com servidores para que não dependam em abrir várias conexões HTTP. (MELNIKOV, 2011)

### **Inteligência Artificial:**

Segmento de estudo que tem como propósito criar dispositivos dos quais sejam capazes de assimilar a cognição humana. (MCCARTHY, 1956)

### ***Deep Learning:***

É um segmento de pesquisa de *Machine Learning* cujo qual tem como objetivo convergir o *Machine Learning* para seu objetivo principal que é a inteligência artificial consequentemente indo além do aprendizado de máquina *shallow*. Utilizando-se de métodos matemáticos e múltiplas camadas das quais vão se aprofundando e abstraindo a informação, dando origem ao nome de *Deep Learning*. (Dechter, 1986)

### **Transformada de Fourier:**

O método de cálculo da transformada discreta de Fourier a partir da expressão

**Figura 1:**

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi n \frac{k}{N}} = \sum_{K=0}^{N-1} f_k W^{kn} . n = 0.....N - 1$$

Transformada de Fourier

Utiliza  $N^2$  produtos entre números complexos e  $N(N-1)$  somas, possuindo assim

complexidade computacional  $O(N \log N)$  que está sendo implementado em *JavaScript*.

### **Escopo do trabalho:**

Está sendo desenvolvido em *JavaScript* o FFT o qual recebe as informações do microfone e criando uma tabela com esses dados, na etapa de treinamento da rede neural criada através do *TensorFlow*, é apontado como entrada (X) essa tabela feita pelo FFT, e como saída desejada (Y) um MIDI de como deveria ser a música que foi captada pelo FFT, para que assim a rede neural possa ir atualizando seus pesos para poder identificar após a etapa de treino corretamente o que está sendo reproduzido assim como foi feito nos exemplos do MNist cujo qual o Google utiliza-se nos seus reconhecimentos de imagem e voz. Após a etapa de treinamento, tem-se a rede em si funcionando e identificando corretamente as sonoridades e devolvendo as saídas corretas para os dados que irão compor a partitura.

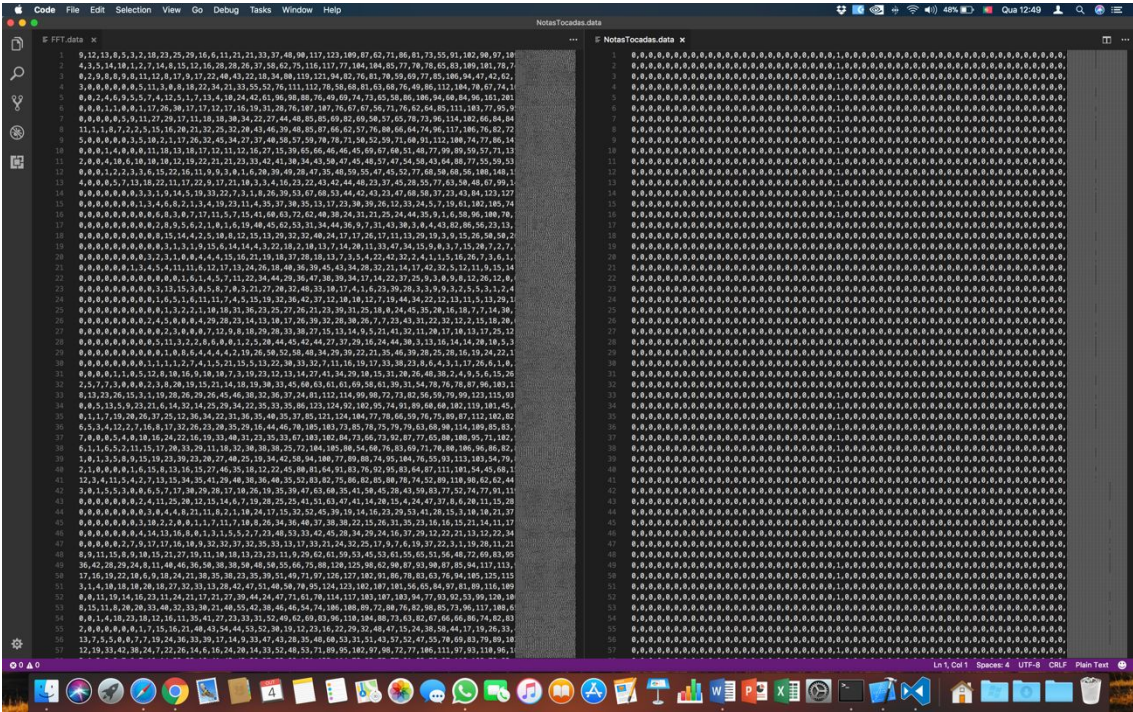
A partir do Keras, é criado o processo de construção da rede neural feita pelo *TensorFlow* o qual funciona como uma biblioteca, que recebe os dados de entrada e saída desejada dentro do contexto de redes neurais supervisionadas, inicia-se os pesos, define-se o *bias*, monta o modelo matemático da rede, calcula-se o custo para corrigir a perda e a insere para otimizar o processo de atualização de pesos para que possa ter uma maior precisão de acertos, inicia-se a sessão criada pelo *TensorFlow* cujo qual irá construir o grafo da rede neural ou como é definido pelo *TensorFlow*, os “*Tensors*”, através de funções do Keras, e após criada a estrutura da rede Neural é iniciada a etapa de treinamento.

Após a etapa de treinamento, a rede neural é salva, é implementado apenas a sua chamada para processar as informações das quais ela recebe em tempo real os pacotes de som gerados pelo FFT os quais chegam em *Python* para serem processados pela rede neural através de um *WebSocket*. A partir disso a rede retorna vetores de 88 posições de zeros e uns ao *WebSocket*, indicando as 88 possíveis notas existentes, que poderão estar contidos em cada pacote de dados analisados pelo *TensorFlow*. Em posse desses dados o ambiente visual desenvolvido tal como os leitores de MIDI e o FFT em *JavaScript*, recebe os vetores de 88 posições e os interpreta corretamente de acordo com as notas que estão sendo tocadas sequencialmente e é montada a sua estrutura visual de uma partitura utilizando a biblioteca VexFlow.

# TESTES DAS TECNOLOGIAS

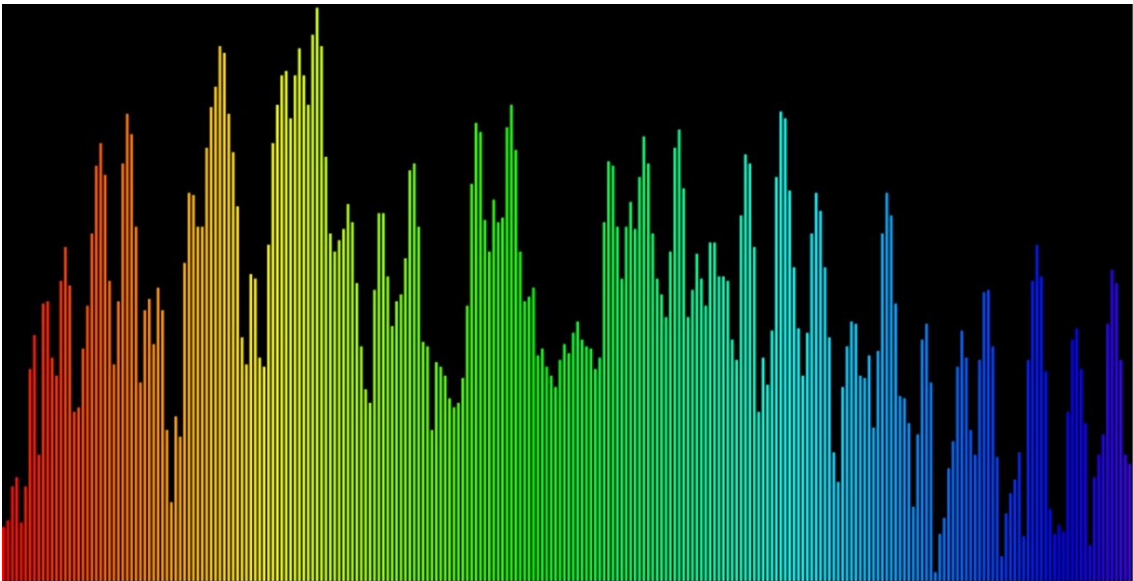
## FFT:

Figura 2:



Entrada gerada pelo FFT e a saída desejada.

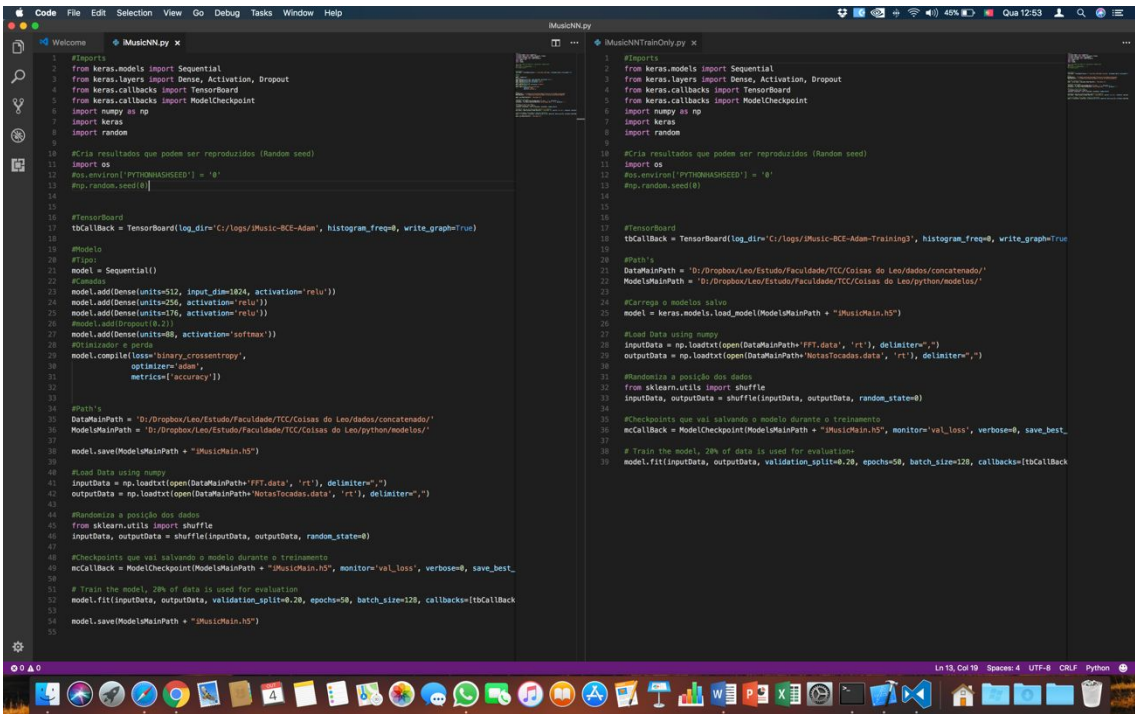
Figura 3:



Amostra.

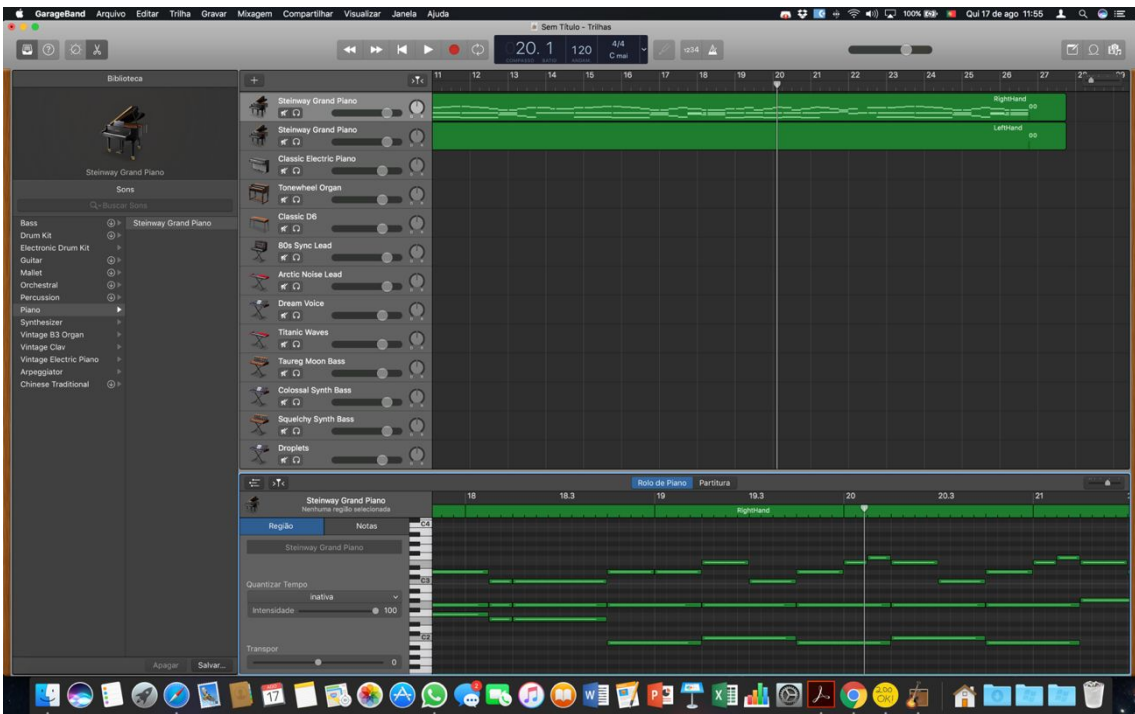


Figura 4:



Treinamento da rede Neural.

Figura 5:



Leitura do MID em espaçamentos.

Figura 6:



Leitura do MIDI convertido em sua partitura correta.

## CONCLUSÃO

O objetivo apresentado foi parcialmente atingido. A Rede Neural produzida é capaz de identificar notas individuais e suas respectivas pausas. O ambiente gráfico é capaz exibir a representação do que foi captado e identificado pela rede.

A principal limitação do projeto é a inability de reconhecer múltiplas notas simultaneamente. Por conta da dificuldade presente na baixa resolução do FFT, a rede neural não foi capaz de abstrair corretamente o que compõe uma nota, sendo assim, a rede resultante desta pesquisa consegue somente reconhecer notas singulares.

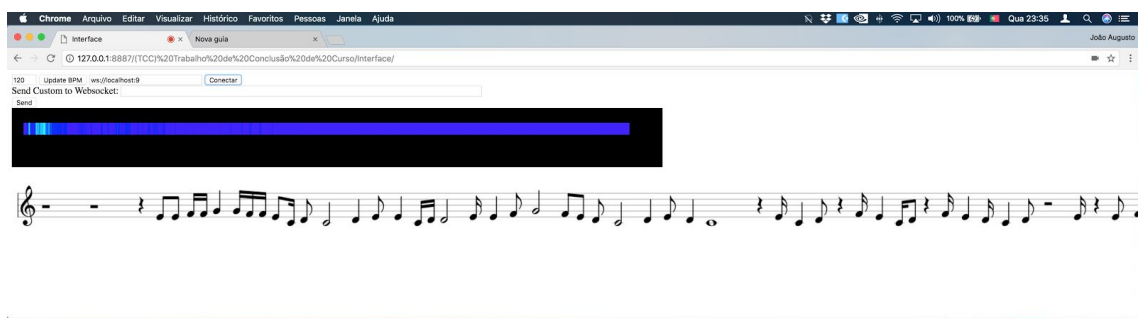
O uso do FFT antes da Rede Neural pode ser o principal responsável por essa inability. O uso das sonoridades diretamente na rede, sem um pré-processamento, junto de um método espacial para reconhecimento dos sons, provavelmente otimizaria o reconhecimento, pois o fator limitador é o próprio FFT.

A coleta de mais dados para o treinamento também é essencial para o aperfeiçoamento para a Rede Neural. Juntamente com mais variedades de instrumentos, ambientes, microfones, e caixas de som, providenciando assim maior qualidade e variedade de dados que representam o mesmo conceito.

Dado que o foco desta pesquisa é a composição de partituras automatizadas e não somente de *Machine Learning*, mais recursos deverão ser investidos na pesquisa da Rede Neural detectora de notas.

Todo o código fonte e recursos utilizados no projeto estão disponíveis em: <https://github.com/leocb/Composicao-Automatica-de-Partituras-com-Machine-Learning>.

Figura 7:



Resultado final obtido com a rede neural, exibido na interface gráfica do projeto.

## REFERÊNCIAS

CATALOGAÇÃO DE MÚSICA IMPRESSA.

<<http://www.abinia.org/catalogadores/56-181-1-PB.pdf> > Acessado em 4 de outubro de 2017.

DEEP LEARNING ...MOVING BEYOND SHALLOW MACHINE LEARNING SINCE 2006.

<<http://deeplearning.net>> Acessado em 4 de outubro de 2017.

ICMC-USP. APRENDIZADO DE MÁQUINA.

<[http://www2.ic.uff.br/~kdmile/MachineLearning\\_Andre.pdf](http://www2.ic.uff.br/~kdmile/MachineLearning_Andre.pdf)> Acessado em 4 de outubro de 2017.

IME-USP. TRANSFORMADA DE FOURIER: FUNDAMENTOS MATEMÁTICOS, IMPLEMENTAÇÃO E APLICAÇÕES MUSICAIS.

<[https://www.ime.usp.br/~kon/MAC5900/seminarios/seminario\\_Jorge.pdf](https://www.ime.usp.br/~kon/MAC5900/seminarios/seminario_Jorge.pdf)>

Acessado em 11 de maio de 2017.

JAVASCRIPT. *START CREATING THE FUTURE NOW.* <<https://www.javascript.com>> Acessado em 29 de abril de 2017.

MIDI.js. <<https://galactic.ink/midi-js/>> Acessado em 15 de agosto de 2017.

P5.js. <<https://p5js.org/>> Acessado em 05 de abril de 2017.

PYTHON. *PYTHON IS A PROGRAMMING LANGUAGE THAT LETS YOU WORK QUICKLY AND INTEGRATE SYSTEMS MORE EFFECTIVELY.* <<https://www.python.org>> Acessado em 3 de maio de 2017.

TENSORFLOW. *AN OPEN-SOURCE SOFTWARE LIRARY FOR MACHINE INTELIGENCE.*

<<https://www.tensorflow.org>> Acessado em 3 de maio de 2017.

TORNADO. <<http://www.tornadoweb.org>> Acessado em 19 de julho de 2017.

KERAS: *THE PYTHON DEEP LEARNING LIBRARY.* <<https://keras.io>> Acessado em 29 de setembro de 2017.

THE WEBSOCKET PROTOCOL < <https://tools.ietf.org/html/rfc6455> > Acessado em 27 de junho de 2017.

WEBSOCKET. *THIS IS WEBSOCKET.ORG.* <<https://www.websocket.org>> Acessado em 29 de abril de 2017.