U.PORTO

MESTRADO
MULTIMÉDIA - ESPECIALIZAÇÃO EM MÚSICA INTERACTIVA E DESIGN DE SOM

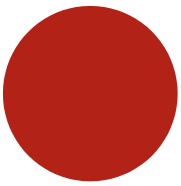
Otimização Automática para o Controlo Intuitivo de Parâmetros de Síntese Sonora

Jorge Pandeirada

M
2016

FACULDADES PARTICIPANTES:

FACULDADE DE ENGENHARIA
FACULDADE DE BELAS ARTES
FACULDADE DE CIÊNCIAS
FACULDADE DE ECONOMIA
FACULDADE DE LETRAS



Otimização Automática para o Controlo Intuitivo de Parâmetros de Síntese Sonora

Jorge Pandeirada

Mestrado em Multimédia da Universidade do Porto

Aprovado em provas públicas pelo Júri:

Presidente: Professor Doutor Rui Rodrigues

Vogal Externo: Professor Doutor Dimitris Andrikopoulos

Orientador: Professor Doutor Rui Penha

Resumo

Explorar as capacidades tímbricas de um algoritmo de síntese de som no seu todo, pode ser um processo difícil devido à quantidade de informação que daí pode ser recuperada. Criar todas as possibilidades de combinações entre apenas dois parâmetros de um algoritmo de síntese pode refletir-se em algumas dezenas ou até centenas de combinações possíveis, mas esse valor cresce exponencialmente à medida que são acrescentados parâmetros. Se a informação presente nas combinações de parâmetros for analisada com recurso a descritores de som, podemos com alguma facilidade descrever as potencialidades tímbricas e espectrais do algoritmo de síntese em questão, dentro do espaço tímbrico que pode ser caracterizado pelos descritores escolhidos.

Este trabalho estuda a possibilidade de otimizar e controlar automaticamente um conjunto de parâmetros de sintetizadores virtuais, alterando apenas as características relativas a cada descritor. Serão usadas redes neuronais para mapear as combinações de parâmetros aos descritores, transformando-os num controlo de alto nível dos parâmetros de síntese, que se processa de forma automática, e sobretudo intuitiva. Por exemplo, será possível incrementar o 'brilho', ou retirar 'conteúdo harmónico', ao som resultante de um algoritmo de síntese sonora.

O trabalho prático que serve de suporte a esta dissertação foi realizado em Max/MSP, para gestão de dados e geração de som, e Wekinator, que permitiu o uso das Redes Neuronais para o mapeamento entre descritores e parâmetros.

Abstract

Exploring the timbral capabilities of a sound synthesis algorithm as a whole, can be a difficult process due to the amount of information that can be retrieved. Creating all possible combinations of two parameters of a synthesis algorithm can reflect on a few tens or even hundreds of possible combinations, but this value grows exponentially as parameters are added. If the information present in the parameter combinations is analyzed using sound descriptors, we can relatively easily describe the timbre and spectral capabilities of the synthesis algorithm in study, within the timbral space in which each descriptor is inserted.

This work studies the possibility of automatically optimize and control a set of parameters of a virtual synthesizer changing only the features for each descriptor. They will be used neural networks to map combinations of parameters to descriptors, making them a high level control of the synthesis parameters, automatically processed, and particularly intuitive. For example, it will be possible to extend the 'brightness' or decrease 'harmonic content', to the resulting sound from a sound synthesis algorithm.

The practical work which support this thesis was carried out in Max/MSP for data management and sound generation; and Wekinator, which allowed the use of Neural Networks for mapping between descriptors and parameters.

Agradecimentos

A realização desta dissertação de mestrado contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade, e aos quais estou bastante grato.

Aos professores e orientadores Rui Penha e Gilberto Bernardes, pela ajuda e orientação tanto na escolha do tema, como nas várias questões técnicas no decorrer do projeto. Representam um exemplo profissional que quero seguir.

Ao Matthew Davies pela preciosa e pronta ajuda em Matlab.

Aos estimados colegas Abel Neto, Manuel Brásio e Urbano Ferreira, pelo companheirismo e amizade ao longo deste processo.

À minha grande amiga Cristiana Graça, pela paciência e pelas palavras de incentivo.

Aos meus pais, Elzira e Jorge, cujo apoio foi incondicional e pelo modelo de coragem e superação que representam para mim, espero um dia conseguir tamanhas qualidades. Mas em especial ao meu avô, João Maria "Paquete", que partiu entretanto, mas viveu o suficiente para me ensinar valores de vida importantíssimos que me vão seguir para sempre, e sem nunca se aperceber disso. A ele dedico este trabalho!

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

A otimização automática para o controlo intuitivo de parâmetros de síntese sonora tem inúmeras vantagens do ponto de vista da criação e interpretação musical que vive essencialmente deste meio. Tal otimização pode ser conseguida com recurso à exploração do espaço sonoro do algoritmo de síntese em questão. Este pode ser um trabalho exigente e demorado dada a dimensão deste espaço. A exploração automática do espaço sonoro de um sintetizador representa um modo viável de analisar e avaliar as capacidades do mesmo.

O objetivo do presente estudo é explorar o tema da otimização automática sobre os parâmetros de quatro sintetizadores virtuais distintos de um ponto de vista intuitivo. Não será uma nova metodologia face às anteriormente propostas, mas antes procurar-se-á propor uma ferramenta adequada para a integração numa prática de composição e performance de música eletroacústica com o Max/MSP. Tal modelo terá como auxilio descritores sonoros, normalmente usados na disciplina de Recuperação de Informação Musical.

1.1. Contexto/Enquadramento/Motivação

No ambiente de trabalho da composição de Música Eletrónica na qual me insiro, é por vezes difícil alcançar o som desejado ou imaginado com a utilização de um sintetizador virtual. O número de combinações de parâmetros é enorme e as opções sonoras acrescem com o número de parâmetros presentes no algoritmo de síntese. A dificuldade aumenta se houver uma escassez de conhecimento quanto às técnicas de síntese e o seu

funcionamento. O controlo de um conjunto de parâmetros de um sintetizador de modo a aproximar o som resultante de um som dado é uma tarefa complicada, sobretudo quando não há uma grande familiaridade com o sintetizador em questão. Um algoritmo, por sua vez, seria facilmente capaz de criar uma 'população' de conjuntos de parâmetros, e no fim do seu processo de escolha e interpolação, oferecer uma serie de hipóteses com probabilidades idênticas de satisfazer a resolução do problema. É principalmente desta temática que surge o interesse pela otimização de parâmetros.

O tema do trabalho partiu ainda de dois pontos de interesse pessoal: (1) a necessidade de criação de um sistema que facilitasse o processo de composição tímbrica, e (2) um mesmo sistema que encurtasse a distância entre um som imaginado e o seu equivalente real. Dado o meu envolvimento na área da composição musical, o recurso à síntese sonora é constante. Também por estar envolvido nesta área, percebo que a quantidade de opções que um sintetizador oferece ao utilizador é tão vasta, que por vezes se torna difícil criar um timbre desejado, pois existe um sem número de alternativas. Estas alternativas mesmo não representando o resultado que se procura, podem resolver o problema em questão e até dar um rumo diferente ao processo de composição.

A otimização de parâmetros com recurso a Algoritmos de otimização automática pareceu desde logo uma solução viável pela quantidade de informação disponível, mas o número de investigações feitas na área é vasto, e houve desde inicio a vontade de realizar um trabalho que de algum modo tivesse uma contribuição para o estudo da otimização de parâmetros de síntese sonora.

O modelo que apresento vai de encontro à ideia de que se pode tornar fácil a utilização de um sintetizador virtual e de forma intuitiva.

Feita a contextualização, a Tabela 1 tem representada a principal ideia que serve como base e enquadramento para este trabalho. O lado esquerdo representa o processo normal da análise de descritores, em que estes aparecem naturalmente em último. À direita está o método que queremos alcançar, em que os descritores de áudio representam um controlo de alto nível para os parâmetros de síntese.

Tabela 1 – Processo que serve de base para o trabalho prático. O lado esquerdo representa o processo natural da análise de descritores. À direita está o método que queremos alcançar, em que os descritores de áudio representam um controlo de alto nível para os parâmetros de síntese.

1.2. Projeto

Como foi acima descrito, as duas principais ideias para a escolha de tema e projeto, foi (1) a necessidade de criação de um sistema que facilitasse o processo de composição tímbrica, e (2) um mesmo sistema que encurtasse a distância entre um som imaginado e seu equivalente real. Desta forma, o projeto teria sempre que ver com a geração de som e/ou a sua manipulação, e fazia sentido que girasse em torno da síntese sonora.

O projeto parte do princípio que as Redes Neuronais podem ser utilizadas para a otimização automática de parâmetros de síntese sonora. Até à data foram realizadas poucas experiências com este recurso porque os Algoritmos Genéticos se mostram mais eficazes. Na sua tese de doutoramento, Yee-King dedica um capítulo à comparação de várias técnicas algorítmicas e as suas capacidades de realizar buscas dentro do espaço tímbrico de um algoritmo de som. Entre 'Genetic Algorithms', 'Hill Climber', 'Data Diven Search' e 'Neural Networks', que foram as que tiveram piores resultados (Yee-King, 2011). A seguinte tabela mostra o desempenho médio por algoritmo e por sintetizador. Cada algoritmo teve como objetivo encontrar uma

correspondência tímbrica a 20 sons dados, dentro da base de dados de sons que criou. Os dados são distribuídos na tabela por pontuação, pelo que o melhor desempenho sintetizador/algoritmo aparece no topo.

Optimiser	Synthesizer	Mean	Standard deviation	SD %	error
GeneticAlgorithmSO	InterFMOS	1.63e+13	7.97+13	490.16	1.96e-08
GeneticAlgorithmSO	BasicFMOS	1171254.21	2975605.86	254.05	3.59e-05
GeneticAlgorithmSO	BasicSubOS	294349.66	425419.75	144.53	0.0001
HillClimberBestOption	BasicFMOS	157637.41	688455.7	436.73	0.0003
HillClimberBestOption	BasicSubOS	53977.4	178831.26	331.31	0.0008
DataDrivenNearestNeighbour	BasicFMOS	40842.15	41508.45	101.63	0.001
HillClimberBestOption	InterFMOS	5090.84	20429.7	401.3	0.01
DataDrivenNearestNeighbour	InterFMOS	3389.37	19505.18	575.48	0.01
DataDrivenNearestNeighbour	BasicSubOS	885.78	1569.31	177.17	0.05
HillClimberBestOption	GrowableOS	192.3	243.88	126.82	0.22
GeneticAlgorithmSO	InterSubOS	98.76	113.81	115.24	0.43
GeneticAlgorithmSO	GrowableOS	80.39	82.85	103.07	0.52
DataDrivenNearestNeighbour	InterSubOS	57.56	30.68	53.31	0.73
HillClimberBestOption	InterSubOS	44.43	15.7	35.34	0.95
FFNeuralNetSO	BasicSubOS	22.87	15.63	68.32	1.84
FFNeuralNetSO	InterSubOS	14.57	3.54	24.3	2.88
FFNeuralNetSO	InterFMOS	7.13	2.92	41.03	5.89
FFNeuralNetSO	BasicFMOS	5.04	2.09	41.35	8.33

Tabela 2 – Desempenho por sintetizador/algoritmo (Yee-King, 2011)

Embora os Algoritmos Genéticos possam representar uma vantagem na sua eficácia, devido aos cálculos complexos que levam à seleção de um ou vários 'indivíduos' capazes de resolver um problema, carecem na rapidez do processo. As Redes Neuronais podem constituir uma vantagem devido ao seu desempenho em tempo real. É um sistema capaz de aprender com os eventos passados, e assim que este 'treino' é feito, o fluxo de informação é mais rápido e eficaz.

O projeto desta dissertação é a implementação de Redes Neuronais na otimização de parâmetros de síntese de som, com o auxilio de descritores de análise sonora. Os descritores, como será explanado mais à frente, oferecem uma quantidade de valores introduzidos numa determinada escala, resultantes de uma análise sonora. Como o nome indica, os descritores sonoros caracterizam um som e oferecem uma representação numérica para determinada característica. Os seus nomes normalmente adjetivam o som, de forma a que seja percetível e de fácil categorização.

1.3. Problema(s), Hipótese(s) e Objetivos de Investigação

Um dos principais problemas em que se centra a primeira parte do trabalho, que consumiu bastante tempo de aprendizagem e estudo, é a geração de um grande número de combinações de parâmetros de síntese, que servirá de base ao treino de algoritmos neuronais, assim como uma análise de descritores que advém das combinações. Para este problema será encontrada uma solução para que assim o trabalho possa prosseguir.

Questões secundárias que podem ser levantadas neste trabalho são:

1) É possível a criação de um sistema de otimização automática de parâmetros, que ofereça uma componente intuitiva, de rápida resposta, passível de ser usada em tempo real?
2) Poderão Algoritmos Regressivos, ou Redes Neuronais suportar tecnicamente esta ideia, e dar o contributo necessário para a sua realização?
3) Quais as vantagens das Redes Neuronais para os Algoritmos Genéticos, visto que os últimos são mais eficazes nos resultados?

1.4. Metodologias de Investigação

Existem três fortes componentes presentes no tema: Descritores de áudio, otimização automática de parâmetros de síntese sonora, e timbre. Dessa forma, todas serão abrangidas na revisão bibliográfica, assim como outros temas que levaram à realização prática do projeto.

A parte prática do trabalho, foi criada em Max/MSP, linguagem de programação que me acompanha desde o primeiro interesse em computação musical, e Wekinator, um programa que põe à disposição do utilizador vários algoritmos relativos com Machine Learning, tema que também será abordado na revisão bibliográfica.

1.5. Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos. Os problemas apresentados respeitarão uma ordem cronológica de

procedimento, ou seja, a resolução do Problema 1 leva ao Problema 2, em diante até à implementação prática do projeto.
No Capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados. De seguida, os problemas relativos ao trabalho prático serão apresentados: no Capítulo 3, é feito um estudo sobre o problema das combinações de parâmetros de síntese. No Capítulo 4, são apresentados os sintetizadores e os descritores a usar, e discutidas as capacidades tímbricas de ambos. O Capítulo 5, detém a implementação prática do trabalho que serve de suporte a este documento. O último capítulo da dissertação são as conclusões e trabalho futuro.

2. Revisão Bibliográfica

Um problema que principalmente os profissionais da área do som normalmente se deparam é a dificuldade de descrever com precisão um som gravado, e por esta razão atribuem aos mesmos designações metafóricas para representar as características tímbricas do som produzido por determinado instrumento, por exemplo: "empty/full, round/pointed, resonant/dull, dark/light, narrow/broad, poor/rich" (Bernardes, 2014). Estes descritores têm como objetivo a representação e descrição morfológica escrita de um determinado som para facilitar a compreensão de futuros leitores. "Mas com que certeza podem afirmar que quem lê a descrição sonora irá interpretá-la da mesma forma que os primeiros?"[1] (Abdullah, et al., 2015) Há uma grande probabilidade da mesma palavra/descritor ser interpretado de formas diferentes.

2.1. Introdução

A avaliação (ou categorização) de um som, é normalmente feita depois da sua execução. Na maioria dos casos, só depois de ouvir um som, se é capaz de o categorizar. Este trabalho explorará principalmente a ideia da possibilidade de alterar a espectromorfologia de um som antes de ser gerado, oferecendo como parâmetros do mesmo género dos acima descritos, e que ao longo do trabalho serão chamados Descritores. Para tal, só será possível a geração e alteração do "objeto sonoro" (Schaeffer, 1966) com o auxilio da síntese de som.

[1] Do inglês: "But to what extent will the reader understand and perceive the sound based on the writer's explanation?" Página 1

O exercício de criação tímbrica com recurso a síntese sonora, requer a calibração e otimização dos parâmetros e técnicas da mesma. Tal tarefa pode ter um processo difícil e demorado, principalmente se não houver uma ligação intuitiva entre os parâmetros e o som produzido. Em muitos casos, a própria interface destes mecanismos representa um obstáculo, principalmente para utilizadores menos treinados.

O presente capítulo destina-se à introdução e contextualização do projeto e dissertação final de Mestrado, começando pela revisão do Estado da Arte do mesmo.

2.2. Espectromorfologia e Timbre

A Espectromorfologia é uma proposta de Denis Smalley, preocupa-se em analisar a morfologia, ou seja, a alteração ao longo do tempo das características espectrais de um som (Smalley, 1997).

Smalley (1994) em *Defining Timbre - Refining Timbre* começa por definir o timbre como o atributo que confere identidade especromorfológica. Aponta para a difícil tarefa de definição de um som, de expandir as noções até à data assumidas de timbre, e para o problema de tornar standard as definições de identidade de um som. Mais tarde Lasse Thoresen (2001/4) denota a necessidade do desenvolvimento de uma tecnologia (e léxico) que ajude a descrever a "fenomenologia" da música em termos experimentais". Esta abordagem fenomenológica do timbre foi iniciada por Schaeffer e mais tarde continuada por Smalley.

Rob Weale no Glossário de Termos *"EARS"*, define a espectromorfologia como "uma ferramenta para analisar e descrever a experiência auditiva"[1] (…) "as duas partes do termo apontam para a interação entre o espectro de um som (espectro-) e a forma como o mesmo muda no tempo (-morfologia). A espectro- não pode viver sem a -morfologia e vice-versa: algo tem que ser moldado, e essa forma tem que ter conteúdo sónico"[2].

Mohd Hassan Abdullah et al. (2015), investiga a possibilidade de usar visualização de timbre a partir de espectrogramas com a finalidade de

[1] Tradução do inglês: "(…) tools for describing and analysing listening experience."
[2] Tradução do inglês: "The two parts of the term refer to the interaction between sound spectra (spectro-) and the ways they change and are shaped through time (-morphology). The spectro- cannot exist without the -morphology and vice versa: something has to be shaped, and a shape must have sonic content."

reconhecer instrumentos tradicionais Malaios. Neste artigo são levantadas algumas questões já apresentadas no introito deste documento, relativos aos descritores que, por exemplo, Etnomusicólogos usam para categorizar alguns sons que gravam, relativos com o trabalho que executam. Na sua maioria são descritos de formas metafóricas para que mais tarde possam ser reconhecidos. O problema que se impõe tem que ver com o facto de estes descritores poderem ser interpretado de formas diferentes, por pessoas diferentes.

Nos instrumentos tradicionais malaios, não existe um sistema fixo de afinação, mas um músico intérprete experiente nesta área sabe precisar o "som aceitável"[1] para cada instrumento, e é descrito como "alto (kuat), penetrante (gemersik), aguçado (tajam), e tenso (tegang)"[2]. Mas como é possível compreender e precisar o som de um instrumento a partir destes descritores sem um domínio e uma imersão na cultura e na origem?

Também englobando o estudo dos instrumentos tradicionais malaios, um outro artigo de Andrew Blackburn e Jean Penny (2015), tem como objetivo encontrar modelos de notação que possam auxiliar no desenvolvimento futuro da notação de timbre musical. O estudo tem três focos: Etnomusicologia, Music-Mixte (encontro entre musica eletrónica e música instrumental), e música eletroacústica.

2.3. Descritores

A categorização de som tem um papel importante nas diferentes áreas assim como para os profissionais que dele dependem enquanto ferramenta de trabalho, e a adopção de termos metafóricos que descrevam um determinado som acaba por ser importante, principalmente em termos de comunicação. Vejamos o exemplo de um profissional de Field Recording que acabou de gravar vários sons do mesmo ambiente, mas que soam todos de forma diferente. Na próxima etapa do trabalho terá que disponibilizar as suas gravações a um Sound Designer. Desta forma é preciso associar um rótulo a cada som, para que o segundo interveniente perceba as intenções do primeiro, e a verdade é que, na comunicação entre estes profissionais, e sem conhecimento prévio, a generalidade deles consegue entender um determinado som como 'brilhante', 'baixo', 'rugoso', 'claro', 'escuro', entre

[1] Tradução do inglês: "acceptable sound", página 2
[2] Tradução do inglês: "(...)(*kuat*) loud, (*gemersik*) penetrating, (*tajam*) sharp and (*tegang*) taut.", página 2

muitos outros. Mas aqui estamos a falar apenas ao nível da comunicação e perceção humana. Os descritores de som tornam-se mais importantes e abrangentes para outras áreas de estudo a partir do momento em que podem ser calculados e computados. Por exemplo, "o brilho de um som pode ser extraído pelo descritor de áudio *Spectral Centroid*, que mede o centro de massa da representação do domínio do tempo de um sinal de áudio, e exprime o brilho do som apenas em um valor" (Bernardes, 2014). Desta forma, aquilo que até aqui era do domínio da perceção, torna-se agora calculável e descritível dentro de uma escala de valores.

Para este trabalho, os descritores terão um papel fulcral, na medida em que servirão de base de análise descritiva de um número de sons gerados por um algoritmo de síntese. Os valores desta análise serão de seguida associados aos parâmetros de síntese do som analisado.

2.4. Otimização de Parâmetros

Dentro do campo da otimização de parâmetros, foram já realizados alguns trabalhos que se assemelham com este projeto, usando como ferramenta principal Algoritmos Genéticos. Resumido de forma simplista, o Algoritmo começa por criar uma *geração* de indivíduos. Cada indivíduo contém uma série de parâmetros criados aleatoriamente. É feita uma avaliação para perceber o número de indivíduos que podem ser 'aproveitados' na próxima iteração. De seguida são feitas mutações entre os que ficaram para criar novos indivíduos com maiores taxas de sucesso. Por aí em diante até perceber qual o indivíduo que contém o conjunto de parâmetros que mais se assemelha ao resultado que é proposto.

Um trabalho pioneiro que desencadeou todo o interesse nesta área foi de Yee-King et. al. (2008), e é precisamente um sistema que com o auxilio de Algoritmos Genéticos faz a otimização de parâmetros de um qualquer sintetizador virtual, com base num determinado ficheiro de áudio. O objetivo é que o sintetizador produza um som o mais parecido possível com o som alvo. Com a otimização automática de parâmetros surge do seguinte problema: "Como os sintetizadores modernos se tornam mais capazes e as subjacentes arquiteturas de síntese mais obscuras, a tarefa de programa-los para produzir um som desejado torna-se mais demorado e complexo"[1]. Como

[1] Traduzido do inglês: "As modern synthesizers become more capable and the underlying synthesis architectures more obscure, the task of programming them to produce a desired sound becomes more time consuming and complex." Página 1.

solução para este problema surge *Synthbot*, um assistente de composição, pensado para poupar tempo e trabalho. Depois de finalizado, o programa foi submetido a dois testes, um técnico, destinado a avaliar a performance dos algoritmos envolvidos; e um teste prático, para o qual foram convidados 10 participantes com experiência em síntese sonora, a "competir"[2] com o programa, com o objetivo de programar dois sons com base noutros tantos. O primeiro derivava de um instrumento de nota fixa, e o segundo foi gerado com o algoritmo de síntese que tinham à disposição para realizar o teste. No segundo teste, *SynthBot* superou músicos com experiencia na área, e concluiu-o em cerca de metade do tempo.

É uma ideia interessante porque num estado ideal, este programa pode quase imitar um som dado, e de seguida o utilizador é livre de alterar toda a base de síntese, e a partir daí criar um som único. Contudo, um dos objetivos do meu trabalho enfrenta o problema de uma utilização ágil, intuitiva, e em tempo real. Desta forma, posso concluir que a abordagem de *SynthBot* não representa uma solução para esse problema, já que todo o processo algorítmico tem que ser repetido para cada som novo.

Matthieu Macret tem também trabalhos de algum interesse para este projeto. No artigo Macret et. al. (2014), os autores questionam-se sobre qual o conjunto de parâmetros de entrada (e a relação entre eles) que um sintetizador deve ter para chegar a um determinado som dado previamente. Também existe o recurso a Algoritmos Genéticos para a busca de resultados numa base de dados previamente criada. Neste caso não é usado um sintetizador virtual externo, toda a síntese é feita em patches no programa Pure Data, o que desde logo representa uma vantagem já que o utilizador pode experimentar, interagir com os patches. É usado o algoritmo inspirado no trabalho de Garcia (2001), para a geração automática de técnicas de síntese a partir das respetivas formulas, aplicando os componentes que servem de base à síntese sonora, como osciladores, operadores de adição/multiplicação, filtros, etc. Segundo Macret et. al., cada experiência, destinada à otimização automática de parâmetros para um som dado, resultou em cerca de 5 horas.

Anteriormente ao trabalho realizador em Pure Data, Macret et al. (2012) em implementam a calibração e otimização de síntese para sons harmónicos, usando exclusivamente uma variante da técnica de *Frequency Modulation*, por argumentarem que até à data ainda não havia sido plenamente estudada

[2] Traduzido do Inglês: "compete", em Peeters et. al. (2008), página 1

e implementada para este fim. Ainda sobre o trabalho de Macret, realizou uma dissertação nos mesmo moldes dos trabalhos anteriores, sobre a otimização de um sintetizador digital físico, o OP-1 da *Teenage Engineering*, também este com o recurso a algoritmos genéticos (Macret, 2013).

Acima foram referidos alguns dos trabalhos mais importantes na calibragem de sintetizadores virtuais com o recurso a Algoritmos Genéticos. Um problema comum dos referidos trabalhos é o tempo necessário até haver uma resposta sónica por parte do sistema. Sendo a criação de um sistema de uso intuitivo e ágil um dos principais objetivos do meu trabalho, os exemplos acima referidos não representam uma solução viável para solucionar esse problema. Dessa forma serão apresentados na próxima secção conceitos como Machine Learning e Redes Neuronais. Estas técnicas algorítmicas estão na base da minha proposta de dissertação: um sistema computacional dirigido à otimização automática e intuitiva de parâmetros de síntese sonora.

2.5.Machine Learning

Nesta secção serão apresentados os conceitos e métodos de Machine Learning, assim como as Redes Neuronais. Esta temática surge após a otimização automática de parâmetros, pois contém a solução para o problema anteriormente apresentado. É importante enfatizar o facto de esta não ser a minha área de estudo, e servirá como introdução e recurso ao programa Wekinator que coloca à disposição todos os métodos e algoritmos aqui apresentados.

Machine Learning é um método de análise de dados que automatiza a construção de modelos analíticos. São usados algoritmos que 'aprendem' iterativamente a partir de uma base de dados. Este método oferece a possibilidade a um computador de encontrar informação escondida sem que seja explicitamente programado para tal. O comportamento e funcionamento iterativo relativo ao Machine Learning é importante porque o mesmo algoritmo tem a capacidade de se adaptar a diferentes meios, à medida que é exposto a novos dados, e consequentemente usa a faculdade de aprendizagem com base em computações anteriores para gerar novos e melhores resultados, de confiança acrescida.

Os vários algoritmos usados nos modelos de Machine Learning têm hoje em dia a sua presença assegurada nas nossas vidas, e existe uma crescente utilização dos mesmos, não só pelas suas potencialidades práticas, mas

também devido ao desenvolvimento tecnológico. Estes dois fatores fazem com que aquilo que entendemos hoje como Machine Learning se diferencie do mesmo termo no passado. A capacidade de automaticamente aplicar cálculos matemáticos complexos em grandes bases de dados, iteração após iteração, e cada vez mais rapidamente é algo recentemente desenvolvido. A aptidão para detetar informações importantes em estruturas de escala alargada é um dos principais desafios destes sistemas, e Collins (2008) aborda o uso de Machine Learning durante uma improvisação musical. Kotsiantis (2007), explora as várias técnicas de Machine Learning que funcionam de forma supervisionada como *Data Mining*, *Decision Trees*, *Neural Networks*, entre outros, concluindo que a principal questão quando lidamos com Machine Learning não se prende com "qual o algoritmo de aprendizagem que é superior aos restantes, mas sim em que condições um método em particular pode superar significativamente os outros num determinado problema"[1].

O interesse em volta de modelos como Machine Learning deve-se também ao desenvolvimento de uma tecnologia cada vez mais acessível, possante e eficaz, que possibilita a rápida e automática produção de modelos capazes de analisar dados maiores e mais complexos, e da mesma forma oferecer uma respostas rápida e com resultados precisos, principalmente em tempo real e sem a necessidade da intervenção humana.

> "So if you've got large volumes of rapidly-changing data, keep your human analysts, but make them much more productive with machine learning tools. In this age of fast-moving data streams, you need fast-moving modeling streams to keep up."
>
> Thomas H. Davenport, The Wall Street Journal

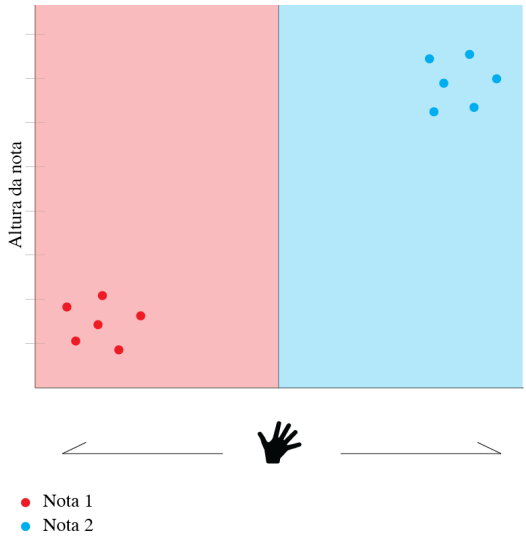
2.5.1.Métodos de Machine Learning

Como referido anteriormente, Machine Learning usa algoritmos destinados especificamente analisar bases de dados, que por sua vez oferecem uma resposta rápida e concisa de acordo com as preferências do utilizador. Mas

[1] Traduzido do inglês: "(...) is not whether a learning algorithm is superior to others, but under witch conditions a particular method can significantly outperform others on a given application problem", página 21.

existem métodos mais comuns do que outros no uso de Machine Learning, e que diferem no tipo de interação que o utilizador tem com o sistema. Os dois mais usados são possivelmente *Supervised Learning* e *Unsupervised Learning. Semi-Supervised* e *Reinforcement Learning* são outras duas tecnologias usadas não tão regularmente e com contextos específicos.

Supervised Learning é o nome dado ao modelo que usa algoritmos treinados com exemplos 'rotulados', ou seja, na fase de treino este algoritmo recebe valores de entrada para os quais são conhecidos previamente as respetivas saídas. Existe, portanto, uma relação direta entre o valor de entrada e a respetiva saída. Existem dois tipos de Supervised Learning: Classificação e Regressão, comummente referidos como *Classification* e



Regression respetivamente.

Figura 1.0 - Exemplo de Classificação

O uso da Classificação é usado em problemas que requerem uma saída de valor discreto, ou seja, tentamos mapear um valor de entrada com diferentes categorias de saída. Na Figura 1 está representado o seguinte exemplo: um sistema com uma câmara que segue o movimento da mão do utilizador. Foram adicionados ao algoritmo alguns exemplos que representam valores de saída mais tarde enviados para um oscilador. Dessa forma podemos ver os pontos vermelhos e azuis como duas notas separadas em altura. Para efeitos representativos, imaginemos que a Nota 1 tem o valor de 440Hz e a Nota 2 tem o valor de 660Hz. O propósito deste sistema é controlar a

frequência do oscilador com a localização horizontal da mão do utilizador. No caso da Classificação, e depois de

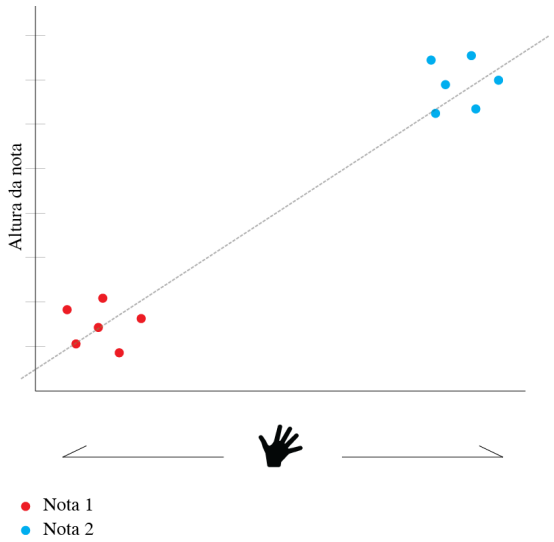


Figura 1.1 Linear Regression

'treinados' os exemplos, o algoritmo define as áreas em que cada nota vai atuar. Não existe outro valor a ser enviado para o oscilador. Se a mão for captada no lado esquerdo será tocada a Nota 1, e pelo contrario for localizada do lado direito é tocada a Nota 2.

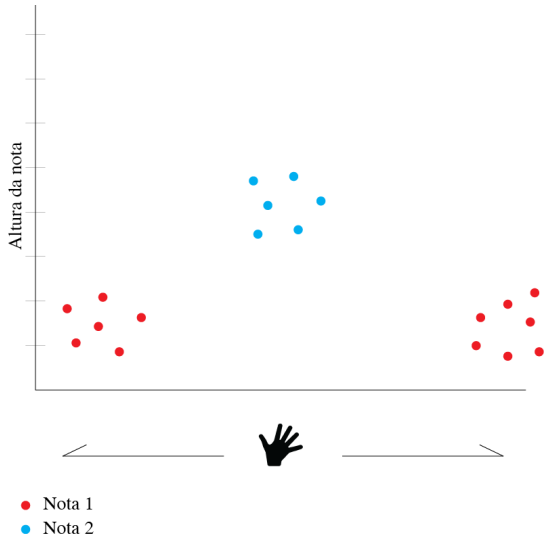


Figura 1.2 Polynomial Regression

Usando a Regressão, o processo utilizado tenta 'adivinhar' o resultado do input dado, com valores de saída contínuos, querendo com isto dizer que

estaremos a tentar mapear valores de entrada de natureza variável a valores de saída do mesmo tipo, dentro de uma determinada extensão. Na Figura 1.1, e pegando no mesmo exemplo dado anteriormente, o algoritmo comporta-se de forma diferente na medida em que ao invés de tocar apenas as notas definidas para cada espaço calculado, cria um mapeamento linear para todos os valores de saída possíveis que se encontram dentro e fora dos exemplos previamente dados.

Podemos chamar ao ultimo exemplo de algoritmo de Regressão Linear, pois o mapeamento entre os valores de entrada e saída é feito linearmente. Mas e se o utilizador pretender que a mesma nota se encontre em dois espaços distintos? A Figura 1.2 mostra-nos esse mesmo problema.

Desta forma precisamos de outro tipo de algoritmo de Regressão chamado Polinomial. Um algoritmo de Regressão Polinomial de primeira ordem encarrega-se de resolver o mapeamento de forma correta quando temos problemas deste tipo. A Figura 1.3 representa uma possível solução para este problema

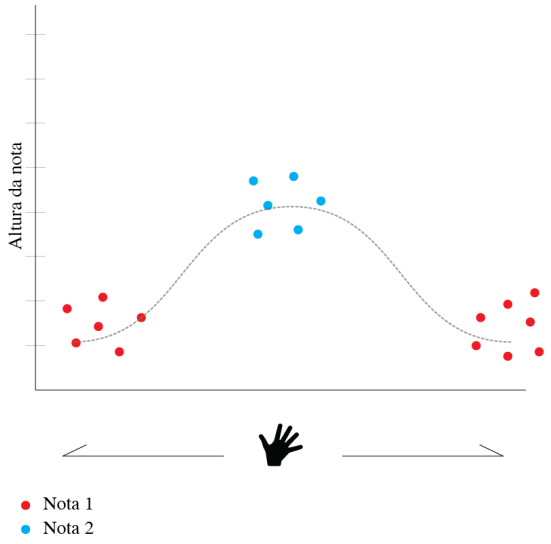


Figura 1.3 Polynomial Regression Primeira Ordem

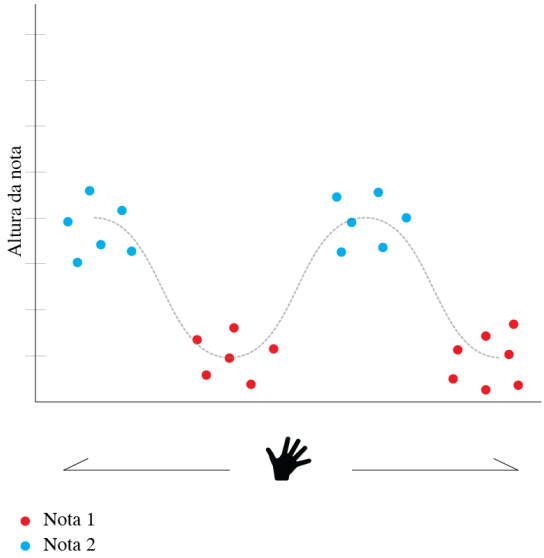


Figura 1.4 Polynomial Regression Segunda Ordem

Ainda assim este género de regressão precisa de ser calibrada previamente de forma responder da melhor forma. O anterior exemplo é relativo a uma Regressão Polinomial de primeira ordem porque o polinómio criado teve apenas que resolver uma interrupção no espaço da Nota 1. Mas se houver uma nova interrupção a ordem do polinómio terá que ser alterada.

Unsupervised Learning atua de forma diferente perante os dados fornecidos. Permite-nos abordar problemas com pouco (ou nenhum) conhecimento tanto dos resultados que serão gerados, como do efeito que as variáveis terão no resultado final. Principalmente não existe um rótulo para cada input como no primeiro caso. Vejamos o exemplo de um problema relativo a reconhecimento facial: podemos afirmar com certeza que um algoritmo só por si não tem a capacidade de perceber e ter consciência do que é uma face dentro de uma base de dados de imagens, mas a partir do agrupamento de dados (*clustering*) tem a capacidade de depreender que a imagem de uma face é diferente de uma paisagem, que por sua vez é diferente de um casa, por exemplo.

2.5.2.Redes Neuronais

Uma outra técnica que pode ser usada para este fim são as Redes Neuronais.

A aprendizagem deste tema deveu-se em parte a *"Nature of Code"* e *"Neural Networks and Deep Learning"*, dois websites de grande importância para o estudo deste tema já que explanam da melhor forma todo o procedimento. Em *"Neural Networks and Deep Learning"*, depois de todo o enredo teórico, o autor leva a cabo a realização de um sistema baseado nas Redes Neuronais, capazes de ler e interpretar letras manuscritas, depois de basear a sua aprendizagem numa base de dados com 60,000 imagens contendo letras manuscritas.

Estas redes são precisamente inspiradas nas homónimas presentes no nosso cérebro. São redes ou camadas de 'indivíduos' (chamados *perceptrons*), presentes na Figura 2, que recebem um número finito de entradas, correm operações matemáticas simples com essas mesmas entradas, e de seguida geram um valor de saída, que segue para a camada seguinte. Cada *perceptron* envia a sua saída produzido para todos os *perceptrons* da camada seguinte, e assim sucessivamente, até que haja um resultado final.

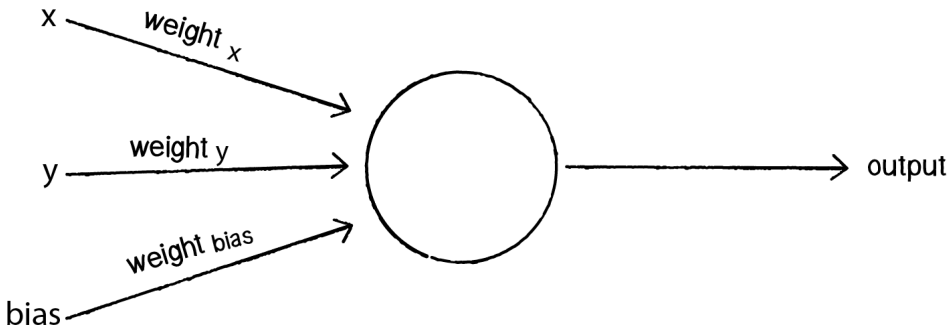
As Redes Neuronais são adaptativas, o que quer dizer que se adaptam ao ambiente em que se encontram. Acontece da seguinte forma: a cada entrada do *perceptron* é multiplicado um peso. O peso pode ser visto como uma probabilidade. No final da rede, o resultado é analisado, e se houver espaço para melhoria (que geralmente existe) todos os pesos são reavaliados e redefinidos, de modo a que o resultado final seja o ideal.

No caso especifico do meu trabalho, as Redes Neuronais fariam o mapeamento ou a tradução entre os descritores apresentados ao utilizador e os vários parâmetros do sintetizador.
Esta é uma técnica pouco usada para a otimização de parâmetros de síntese, porque a Rede pode ter um consumo de tempo acima do aceitável na sua primeira fase de aprendizagem. Mas assim que este processo é terminado responde com uma fluidez impar, pois o processo de computação que é levado a cabo é muito simples. Este ponto pode ser proveitoso ao nível da performance ao vivo, em que o sistema tem uma resposta quase instantânea ao impulso do interprete. Outra possibilidade é a calibragem de acordo com áudio de entrada (seja pré-gravado ou tocado ao vivo). Se este input for analisado segundo os descritores usados, em teoria, o sistema pode facilmente aproximar o resultado final do sintetizador ao áudio analisado, aproximando este trabalho com os acima descritos, mas com o uso de Redes Neuronais ao invés de Algoritmos Genéticos.

Dentro da área musical existem pouco exemplos semelhantes ao que quero realizar. Em Lee et. al. (1992) os autores apresentam aquela que é a primeira

tentativa de realizar um modelo de controlo de síntese usando técnicas de Machine Learning. Os autores apresentam o problema da transformação de gestos na performance em parâmetros de síntese. Já em Wessel et. al. (1998) foi aplicado o modelo anterior, e a Rede Neuronal teve como entrada os valores dos descritores *Fundamental Frequency* e *Loudness*. O objetivo era fazer uma estimativa de amplitudes e frequências de parciais de som.

Fora da área da otimização automática de parâmetros, foram feitos alguns trabalhos com o recurso a imagem como é o caso de *Neuroklang: Real-time Timbre Control using Neural Networks de Visda Goudarzi*. Neuroklang é uma interface baseada em imagem captada por uma câmara, e usa as Redes Neuronais para interpretar os gestos realizados pelo utilizador e transporta-los para uma realidade tímbrica. O processamento de imagem é feito em Processing, e o som provem de um sintetizador programado no ambiente ChucK. Um trabalho semelhante, mas anterior ao acima explanado, é de Fels et. al. (1998). Com o auxilio de luvas preparadas com vários tipos de sensores, este sistema faz o mapeamento entre os gestos levados a cabo pelo utilizador e os controlos de um sintetizador de voz. A ideia principal era fazer a tradução da linguagem gestual. Argumentando a importância deste sistema, os autores dão o exemplo da comunicação difícil entre uma pessoa



com deficiência auditiva, e uma outra invisual.

Figura 2 – Perceptron (Fonte: *"Nature of Code"*)

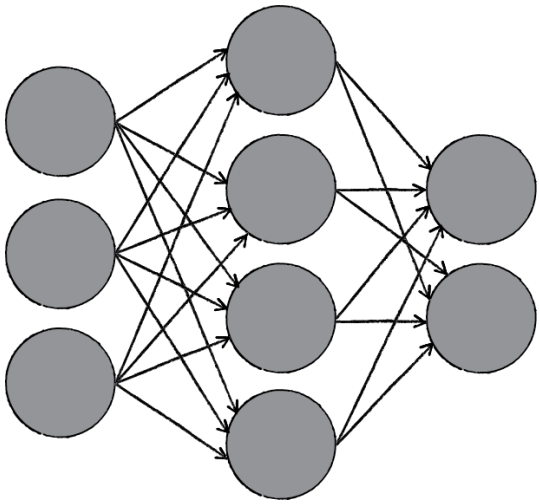


Figura 2.1 - Rede Neuronal (Fonte: *"Nature of Code"*)

2.6. Recursos Técnicos

2.6.1. Max/MSP

Max começou a ser criado por Miller Puckette nos anos 80, no IRCAM, na altura pelo nome de Patcher. Em 1990 foi lançada uma versão comercial pela Opcode Systems, desenvolvido e modificado por David Zicarelli. Em 1999, a empresa de Zicarelli (Cycling'74) acabou por abraçar o projeto depois da Opcode Systems o ter largado. Desde então tem sido desenvolvido e comercializado pela mesma empresa.

Max (nome genérico) é um ambiente de programação gráfica, usado normalmente para música e multimédia. A programação é feita por blocos que podem ser ligados entre si via *patch cables*. Cada bloco contém uma função, assim como várias entradas e saídas, dependendo da função que executa e os argumentos que recebe. Uma janela de programação, ao invés de conter código escrito, conta com módulos ligados entre si, quase como um sintetizador modular, em que cada módulo tem uma tarefa especifica, e o utilizador (programador, no caso do Max) decide qual o caminho que o sinal deve seguir, seguindo um conjunto de regras básicas da síntese.

2.6.1.1. Descriptors~

Descriptors~ é um objeto de Max/MSP que faz parte de uma biblioteca de objetos externos criado por Alexander Harker chamado AHarker Externals. Este objeto pode ser usado para calcular um grande número de descritores

de áudio a partir de um excerto de som armazenado num buffer~ em MAX. Oferece a capacidade de calculo necessária para produzir informações úteis sobre uma dada amostra.

Desta biblioteca foram usados os seguintes descritores harmónicos:

- *Energy*
- *Brightness*
- *Harmonic Ratio*
- *Roughness*
- *Inharmonicity*
- *Spread*
- *Skewness*

2.6.1.2. Zsa.Descriptors

Zsa.Descriptors é uma biblioteca para Max/MSP de análise de descritores de som em tempo real desenvolvido por Mikhail Malt e Emmanuel Jourdan (2008). Esta biblioteca foi estudada, mas não teve uso porque para além de se destinarem à analise em tempo real, apenas uma pequena parte é de cariz harmónico. Por uma questão de conveniência apenas as duas restantes bibliotecas aqui descritas foram utilizadas.

2.6.2. Wekinator

Wekinator é um programa grátis e de fonte aberta, que permite o uso de técnicas de Machine Learning associadas à performance ao vivo. Segundo os autores, em Fiebrink et. al. (2010), este sistema "enfatiza uma interação mais rica entre homem e computador na conceção de sistemas Machine Learning"[1]. A título de exemplo, o Wekinator permite a qualquer utilizador desenvolver os próprios instrumentos e controlá-los de forma intuitiva e interativa com recurso a vários algoritmos, encarregados de computar a resposta mais rápida às ações humanas.

[1] Do Ingês: "(...)which emphasizes a richer human-computer interaction in the design of machine learning systems."

Wekinator usa algoritmos de Machine Learning, já aqui explicados, que têm como particularidade a capacidade de 'aprender' com os eventos anteriores, e dessa forma adequar-se aos novos meios

2.6.3. Open Sound Control

Open Sound Control, ou OSC, é um protocolo de comunicação para sistemas em rede, desenvolvido pela CNMAT. Foi criado com a intenção de partilhar em rede dados relativos à performance musical, como gestos, parâmetros e sequências musicais entre instrumentos, computadores ou outros aparelhos multimédia. É por isso muitas vezes utilizado em recurso ao MIDI, já que as suas principais vantagens são a agilidade na partilha de dados, assim como a resolução das mensagens.

Para este trabalho será usado o OSC para partilha de informação entre o MAX e o Wekinator.

2.6.4. MatLab

MatLab é uma linguagem de programação criada pela MathWorks, destinada principalmente à computação numérica e gráfica. Como o próprio nome sugere, MatLab dirige-se não só à programação de matrizes, mas também à álgebra linear e análise numérica. Atualmente, a capacidade do MatLab estende-se para além de um "Laboratório de Matrizes", integra a capacidade de realizar cálculos, desenvolver algoritmos, analisar, explorar e visualizar dados, desenvolver gráficos de engenharia e científicos, entre outras potencialidades. Criado no final da década de 70, este é tanto um ambiente de trabalho como uma linguagem de programação, e foi desenhado para resolver problemas numericamente.

Um dos aspetos de maior interesse no MatLab é o facto da linguagem permitir a criação das próprias ferramentas e algoritmos, possibilitando a sua reutilização. Desta forma tornou-se possível a criação de Toolboxes, que são nada mais do que bibliotecas destinadas à resolução de determinados problemas, criadas na sua maioria pelos próprios utilizadores do programa, provenientes principalmente da área de investigação cientifica.

Desta forma, MatLab torna-se numa ferramenta de trabalho essencial principalmente para alguns setores da engenharia e da ciência, não só pela extensão das suas capacidades, mas também porque permite solucionar

variados problemas computacionais num intervalo de tempo inferior a outras linguagens de programação.

2.6.4.1. The Timbre Toolbox

Neste trabalho, foi usado o ambiente e a linguagem de programação MatLab para analisar com base numa biblioteca de descritores um conjunto de excertos de áudio, gerados pelos sintetizadores virtuais em estudo. A biblioteca (ou Toolbox) utilizada para este fim e tem como nome *The Timbre ToolBox*. Em Peeters (2011), os autores apresentam um conjunto de algoritmos da sua autoria, destinados à analise de descritores de áudio, ou timbre. Neste artigo, referem-se ao termo "timbre" como algo que "engloba um conjunto de atributos auditivos presentes em eventos sonoros, assim como altura, intensidade, duração e posição espacial".

Esta 'Toolbox' analisa todos os ficheiros de áudio dentro de uma determinada pasta, e para cada excerto fornece um ficheiro de texto com a análise de todos os descritores ao seu dispor. De seguida, os ficheiros são analisados e os descritores a usar são reunidos num só ficheiro.

Os descritores de 'Timbre Toolbox' utilizados foram:

- *Energy*
- *Noisiness*
- *Inharmonicity*
- *Spread*
- *Skewness*

Em Peeters et. al. (2011), os autores fazem uma breve descrição de cada descritor e o respetivo cálculo ou algoritmo usado. Note-se que no caso do descritor *Noise Energy*, por exemplo, o valor que advém da análise é proporcionalmente inversa ao valor de *Harmonic energy*. Enquanto a primeira é obtida analisando a energia que não corresponde a nenhum sistema de série de harmónicos, a segunda é calculada somando os harmónicos 'explicados' presentes na amostra de áudio. Por outras palavras, transposto à realidade deste projeto, à medida que o valor de *Noise energy* aumenta, *Harmonic energy* diminui, e vive-versa. *Noisiness*, aqui utilizado, representa o rácio de *Noise energy* para a energia total da amostra. Representando uma característica sonora já bastante estudada, *Noisiness* teve já várias designações: "Pierre Scheaffer designa-o como massa; Denis Smalley usa

tipologia espectral; e Lasse Thoresen chama-o de largura espectral"[1] Bernardes (2014).

2.7. Resumo e Conclusões

Neste capítulo foram abordadas algumas temáticas que irão ser usadas no trabalho que se segue. Foram também respondidas algumas questões colocadas inicialmente.

Depois do estudo e comparação entre Algoritmos Genéticos e Redes Neuronais, percebe-se que as segundas são as indicadas para o meu trabalho, já que têm a capacidade de, depois de treinadas, reagir a qualquer estimulo com uma resposta rápida e intuitiva. O sistema que advirá deste trabalho terá decerto uma componente de performance, já que sistema terá a capacidade de interagir em tempo real com o utilizador. Para além do mais, as Redes Neuronais estão tecnicamente facilitadas, e postas à disposição pelo programa Wekinator, fazendo com que o meu foco se vire para a exploração da capacidade de intuição do sistema final.

De seguida irá ser apresentado o primeiro problema a solucionar: a criação das combinações de parâmetros de síntese que servirão de base para o treino neuronal. Este foi o problema que consumiu mais tempo de estudo, dadas as dimensões que pode tomar.

[1] Do Ingês: "Pierre Schaeffer designates it as mass; Denis Smalley uses spectral typology; and Lasse Thoresen names it spectral width." em Bernardes (2014) página 53.

3. Criação do Espaço de Síntese Sonora

O mais completo processo para explorar o espaço sonoro de um algoritmo de síntese sonora, é possivelmente com o recurso ao maior número de possibilidades, se não todas, que o mesmo está apto a produzir. É possível a obtenção desta informação, gerando todas as combinações possíveis de parâmetros que um sintetizador oferece. Note-se que a possibilidade de gerar todas as combinações reais é remota, dada a infinidade de números irracionais que se podem retirar de um simples incremento entre 0.0 e 0.1, por exemplo.

Gerar todas as combinações de parâmetros possíveis para um dado sintetizador pode ser o maior problema neste campo devido ao número de possibilidades e combinações que podem ser realizadas, dependendo, no entanto, do número total de parâmetros associados à operação.

$$N^o\ de\ combinações = N^o\ de\ valores^{N^o\ de\ parametros}$$

Por exemplo, para um sintetizador com 10 parâmetros em que cada um assume 11 valores entre 0.0 e 1.0 (com um incremento de 0.1) existem 11^10 possibilidades, um resultado que se aproxima das 26 mil milhões (ou biliões) de combinações. Numa primeira abordagem a este problema, houveram várias tentativas de solução, mas rapidamente se percebeu que só forneceriam material para corroborar o argumento relativo à dificuldade deste processo.

Neste capítulo irei explicar as várias tentativas para a solução do problema relativo à criação do maior número de possibilidades dentro de uma

determinada escala, em que cada variável assume um total de 11 valores. Neste capítulo, a referência para 'total de combinações possíveis' assumirá os mesmos contornos do exemplo anterior em que cada parâmetro representa 1 em 11 valores situados entre 0.0 e 1.0 [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 e 1.0]

Concluirei o capítulo com a dedução dos vários factos, prós e contras desta abordagem, e a descrição da solução tomada para a continuidade do projeto.

3.1. Seleção de Algoritmos de Síntese

Antes da criação de combinações de parâmetros, foram escolhidos os sintetizadores a usar no trabalho prático. A escolha dos algoritmos de síntese teve os seguintes critérios:

1) Programa gratuito, para uma livre utilização;
2) Sistema com funcionamento em multiplataforma (prioritariamente MacOSX e Windows);
3) Algoritmo de síntese curto com recurso a poucos parâmetros. Quanto menos parâmetros estiverem envolvidos, maior será o foco e capacidade de aprendizagem do sistema sobre os restantes.

Estes foram os três instrumentos virtuais escolhidos que melhor satisfizeram os critérios anteriores:

- Elec7ro - TAL
- NoiseMaker - TAL
- U-No - TAL

Contudo, os sintetizadores escolhidos fazem parte da mesma marca, e após alguma experiência e audição com os mesmos chegou-se à conclusão que seria preciso pelo menos um algoritmo de síntese diferente, e que destoasse das principais preocupações tímbricas dos presentes: emular sintetizadores analógicos. Foi selecionado o sintetizador virtual Twist, programado com um tipo de síntese chamado *Sound Morphing*, e que auditivamente se assemelha à síntese por modelos físicos. Contudo, embora seja o sintetizador com menos parâmetros disponíveis, é o único que não é gratuito. A seleção deste sintetizador assenta no facto de utilizar modelos de síntese deferentes dos sintetizadores, o que faz com que o resultado sónico seja diferente, e principalmente pelo número reduzido que parâmetros de que dispõe.

3.2. Criação de combinações de valores

Como referido anteriormente, aquele que parece ser o processo teoricamente mais simples, é provavelmente o que representa maior dificuldade na execução: a criação de um mapa com todas as combinações de parâmetros possíveis que um sintetizador pode executar. Desde logo aparecem dois grandes problemas: espaço e tempo.

Nesta secção serão analisados os vários testes realizados, a uma escala reduzida, com o intuito de perceber qual a abordagem a seguir posteriormente, para o mesmo efeito, e à escala normal. Ou seja, sendo que será preciso um mapa com todas as combinações para sintetizadores com um máximo de 16 parâmetros, serão realizados testes para apenas 10 parâmetros. No final será feita uma conclusão em análise aos vários testes e à sua importância para a continuidade do trabalho.

3.2.1. Fase de testes

À medida que são acrescentadas variáveis (ou parâmetros) e os valores que os mesmos podem assumir, o número de combinações aumenta exponencialmente. Este aumento está representado na Tabela 3, onde se percebe a discrepância no número de combinações para os mesmos parâmetros, sendo que cada um pode tomar um diferente número de valores. A terceira e a quarta colunas, apresentam uma queda abrupta na quantidade de combinações que podem ser obtidas, com a simples diferença de um valor por parâmetro.

1	2	3	4	5
Sintetizador	**Número de Parâmetros a ser usados**	**Número de combinações para 11 valores**	**Número de combinações para 10 valores**	**Número de combinações para 3 valores**
Elec7ro-II	16	$4{,}594972986^{16}$	1^{16}	43 046 721
NoiseMaker	16	$4{,}594972986^{16}$	1^{16}	43 046 721
Twist	10	25 937 424 601	10 000 000 000	59 049
U-No	17	$5{,}054470285^{17}$	1^{17}	129 140 163

Tabela 3 - Comparação de combinações por número de valores

O mapa de combinações tem que ser formatado de forma a poder ser interpretado corretamente pelo próximo estágio, ou seja, terá que ser gravado em formato .txt para ser mais tarde interpretado pelo objecto "coll" do MAX/MSP.

Foram realizadas várias abordagens, com o intuito de perceber qual o sistema (ou linguagem de programação) mais conveniente para a geração de tamanha quantidade de informação. Na sua maioria, os testes foram feitos para apenas 10 parâmetros, sendo que cada um toma 11 valores entre 0.0 e 1.0. Os testes cujos valores se diferenciem dos anteriormente referidos serão devidamente descritos. O processo utilizado foi também igual para todos os sistemas, aproximando-se mais uma vez do exemplo dos ponteiros do relógio: uma sequência de contadores cujo incremento de cada um depende unicamente do alcance total do anterior, ou seja, quando o Contador 1 chega ao seu máximo (1.0) reinicia a sua contagem e é incrementado um valor (0.1) no Contador 2, e assim sucessivamente até que todas as combinações sejam abrangidas.

Cada um dos Testes que aqui será descrito está presente no Anexo Digital como o respetivo nome e número de teste.

3.2.1.1. Teste nº1

Os primeiros testes foram realizados em MAX/MSP, mas este revelou-se lento para a geração de uma quantidade tão alargada de informação. Como referido anteriormente, a estrutura com que cada alínea foi gerada obedecia a uma formatação específica para que o ficheiro final pudesse ser lido e interpretado na seguinte fase do projeto, desta forma:

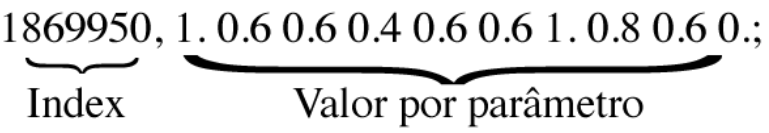


Figura 3 - Exemplo de Justificação de mensagem, Teste nº 1

A Figura 3 está associada ao Teste nº 1 em que se reduziu o número de valores por parâmetro para 6, ou seja: 0.0, 0.2, 0.4, 0.6, 0.8 e 1.0.

O teste anterior mostrou-se pouco funcional porque todas as alíneas (combinações) resultariam na ocupação de um grande espaço em disco. O número total de combinações (alíneas) ocuparia uma grande quantidade de

espaço. A título de exemplo, este teste resultou num ficheiro de texto contendo 2929686 alíneas, de 137,4Mb, sendo à medida que se acrescentassem novos parâmetros ao sistema, o espaço utilizado aumentaria exponencialmente.

No Anexo Digital referente a este teste, encontra-se tanto o ficheiro de texto final contendo combinações para 10 parâmetros, assim como o patch preparado para criar combinações do mesmo tipo, mas apenas para 5 parâmetros.

3.2.1.2. Teste nº2

Surgiu de seguida a necessidade de comprimir informação, de maneira a ocupar menos espaço em disco. O seguinte teste teve precisamente essa preocupação como prioridade. Foram geradas todas as combinações possíveis para apenas 5 parâmetros com 11 valores entre 0.0 e 1.0 e guardadas num ficheiro de texto. De seguida, o mesmo ficheiro foi lido por vários contadores retirando apenas o índice de cada alínea. Desta forma uma combinação de 10 parâmetros seria representada apenas dois números (índices) do mesmo ficheiro, como é descrito na Figura 3.1.
Esta abordagem mostrou-se mais ágil visto que o poder de computação necessário para levar a cabo o processo foi menor comparativamente ao teste anterior.

Combinação pretendida - 1. 0.6 0.6 0.4 0.6 0.6 1. 0.8 0.6 0.;
Índice 1 - 188400, 1. 0.6 0.6 0.4 0.6;
Índice 2 - 130120, 0.6 1. 0.8 0.6 0.;
Combinação final - 188400 130120

Figura 3.1 - Exemplo de Justificação de mensagem, Teste nº 2

3.2.1.3. Teste nº3

Foi testado em terceiro lugar o Microsoft Excel, com um código Macro na linguagem de programação VBA (Visual Basic for Applications). Esta terceira abordagem surgiu como recurso ao MAX/MSP, visto que nos primeiros testes

se revelou lento comparativamente à quantidade de informação a ser gerada. Desta forma houve a necessidade de experimentar diferentes linguagens de programação e modos de processamento. O Microsoft Excel mostrou-se mais rápido comparativamente ao MAX/MSP, a geração de dados é muito mais fluida, mas a partir de um certo ponto de processamento a quantidade de informação torna-se elevada e o programa acaba por colapsar.
O código utilizado neste teste está presente tanto no Apêndice Digital como no Anexo I deste documento.

3.2.1.4. Teste nº 4

Numa quarta abordagem, foi testada a linguagem de programação JavaScript, que foi executada em Max/MSP com o objeto "js".

O código utilizado neste teste está presente tanto no Apêndice Digital como no Anexo II deste documento.

3.2.1.5. Teste nº 5

Este foi o teste final, onde se investiu mais tempo, e cujas combinações resultaram num enorme espaço virtual ocupado. O processo e código utilizados para este teste foi escrito na linguagem de programação Python, e abrangeu apenas os dois primeiros sintetizadores em estudo (Elec7ro e NoiseMaker) ambos contendo 19 parâmetros cada um assumindo apenas 3 valores (0.0, 0.5 e 1.0). Ao todo foram gerados 5811 ficheiros de texto por sintetizador, cada um contendo 200.000 (duzentas mil) combinações de parâmetros, pesando em média 57Mb cada. No seu total, foram geradas 1.162.261.466 (um bilião, cento e sessenta e dois milhões, duzentos e sessenta e um mil, e quatrocentos e sessenta e seis) combinações por sintetizador.

Neste teste concentram-se os dois maiores problemas com que me deparei no capítulo da criação de combinações: tempo, pois este processo demorou cerca de um dia a ser realizado; e espaço, já que a totalidade das combinações apenas para dois sintetizadores ocupou cerca de 675Gb. Note-se que se aumentou o número de parâmetros para 19 reduzindo para 3 o número de valores que cada um assume.

O código utilizado neste teste está presente tanto no Apêndice Digital, onde se encontra também o primeiro e ultimo ficheiro de cada geração), como no Anexo III deste documento.

3.3. Resumo, Solução e Conclusões

Os exemplos que foram realizados e explanados nesta secção tiveram como principal objetivo a criação de um programa destinado à geração de um 'mapa' com todas as combinações possíveis de um determinado número de variáveis com valores entre 0.0 e 1.0. O mesmo mapa teria como destino a sua integração com quatro sintetizadores distintos cujos parâmetros teriam uma relação direta com os valores das variáveis.
Este sistema, constituído por conjuntos de contadores numéricos a operar em série foi testado em várias linguagens de programação, e teve um papel importante para a continuação do trabalho, na medida em que a ausência de um conjunto finito de combinações de parâmetros impossibilitaria a fase de treino do algoritmo 'neuronal', e desta forma a sua resolução determina o curso tanto da fase seguinte do projeto como de todo o trabalho.
Para a criação de um mapa com dimensões tão grandes é necessário tanto espaço virtual (em disco) como tempo para a realização da operação. Como foi explicado, o teste mais demorado foi naturalmente aquele que requisitou mais espaço para ser guardado. Mesmo que os dois problemas anteriores fossem resolvidos, surgiria um terceiro contratempo pois a capacidade de processamento necessária para um sistema computacional ler e interpretar a base de dados criada teria que ser igual ou superior à usada para solucionar os dois primeiros. Numa posterior fase do projeto apareceria outro problema: cada uma das combinações geradas teria que ser transformada em áudio analisada por um conjunto de descritores de áudio. O tempo e espaço reservados para esta fase seriam ainda maiores, visto não ter em minha posse capacidade computacional para levara cabo esta tarefa. A título de exemplo, se analisasse 4 amostras por segundo este seria o calculo para perceber quanto tempo demoraria o processo:

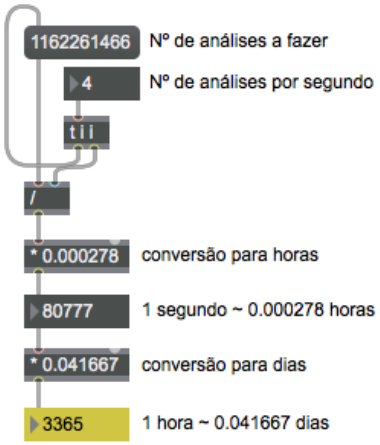


Figura 4 - Tempos de análise com 4 amostras por segundo

Analisando 4 amostras por segundo, que elevaria o consumo de CPU para 30% resultaria em 3365 dias para finalizar a análise. Se tivesse capacidade para analisar 10 amostras por segundo, precisaria de 1346 dias. Dado o caso remoto de conseguir analisar 100 amostras por segundo, em 134 dias o processo estaria concluído.

Na presença dos problemas descritos, chegou-se à conclusão que esta seria uma abordagem quase impossível dado o tempo disponível para a a realização do trabalho. Desta forma a solução mais viável seria a criação de um conjunto de combinações muito inferior aos últimos números abordados. As combinações resultariam de uma geração aleatória, e na sua totalidade seriam 1000, número modesto dadas as análises anteriores. Desta forma, o trabalho geral não teria que sofrer alterações, seguindo o mesmo rumo, mas os dados a serem estudados sofreriam uma redução abrupta no tamanho da sua amostra. Concluindo, e dados os últimos avanços, o foco deste trabalho sofre uma pequena alteração na medida em que todos os estudos e análises feitas aos sintetizadores de hora em diante terão como ponto de convergência uma pequena amostra da capacidade total de cada um. Assim, a generalização de conclusões relativas às capacidades de cada algoritmo de síntese, tornar-se-á errónea.

A próxima etapa do trabalho será selecionar quais os parâmetros e descritores a usar, para que sejam aplicadas as combinações aqui geradas. Para além desta seleção, no Capítulo 3 haverá uma secção de visualização de dados, que pretenderá perceber quais os parâmetros de síntese e descritores com melhor capacidade para satisfazer o processo neuronal.

No Anexo Digital encontra-se na Pasta ‘1000CombFixed’ os ficheiros com as 1000 combinações fixas para cada sintetizador, assim como o patch de MAX que as gerou. Mais tarde será concluído que Elec7ro e NoiseMaker têm o

mesmo número de parâmetros, e por isso usarão o mesmo ficheiro de combinações.

Tabela 4 – Fases do processo concretizadas

Na Tabela 4, surge de novo o processo base do trabalho. O primeiro problema está ultrapassado com a criação das combinações de âmbito fixo, e a seguinte etapa tem que ver com critérios de seleção de parâmetros de síntese e descritores. A implementação prática do trabalho sucede o seguinte capítulo.

4. Descritores e Parâmetros de Síntese

No capítulo anterior foram apresentados vários testes realizados de modo a decidir qual o melhor caminho a seguir dada a dimensão da base de dados necessária para cobrir todas as combinações de parâmetros de um sintetizador virtual. Concluiu-se que a melhor forma de solucionar o problema seria gerar um número de combinações fixas e muito inferiores aos valores anteriormente estudados, que servissem de treino para as redes neuronais.

No presente capítulo está presente a segunda parte do trabalho prático, que tem como objetivo apresentar os vários descritores e parâmetros de síntese usados, e o porquê dessas escolhas. Assim como no anterior, o presente capítulo apresenta um estudo fundamental para a próxima e ultima etapa do trabalho prático, assim como os resultados do mesmo. Serão também apresentadas análises gráficas relativas ao resultado das combinações de parâmetros atribuídas a cada sintetizador.

4.1. Sintetizadores Virtuais e Parâmetros

Para cada sintetizador foram criadas 1000 combinações aleatórias e fixas, ou seja, a cada sintetizador foram atribuídas as mesmas combinações de parâmetros em todos os testes realizados desde então. Como referido anteriormente, todos os parâmetros responsáveis pela evolução do som no domínio do tempo foram suprimidos. Para alem de não terem importância perante a ideia principal do projeto - a exploração e criação tímbrica, omitindo o facto da evolução sonora também ser responsável pela forma como percebemos um som -, aumentaria ainda mais o tempo tanto do treino do sistema neuronal, como da criação de um conjunto de soluções/combinações a serem treinados.

Na Tabela 5 estão representados todos os parâmetros selecionados segundo o critério de seleção, e os respetivos sintetizadores.

Elec7ro	NoiseMaker	Twist	U-No
VCOMODULATION	FILTERTYPE	CH1INTENSITY	SAW
VCFMODULATION	CUTOFF	FILTERTYPE	SUBOSC
VCORANGE	RESONANCE	FILTERCUTOFF	SUBOSCVOLUME
NOISEVOLUME	FILTERCONTOUR	FILTERQ	NOISEVOLUME
OSC1VOLUME	OSC1VOLUME	FILTER ENV DEPTH	SQUAREPWMODE
OSC2VOLUME	OSC2VOLUME	CH1ON-OFF	SQUAREPWINTENSITY
OSC3VOLUME	OSC3VOLUME	CH2TUNE	HPCUTOFF
CUTOFF	OSC1WAVEFORM	CH2PITCH	CUTOFF
RESONANCE	OSC2WAVEFORM	CH2INTENSITY	RESONANCE
OSC1WAVEFORM	OSC2FM	CH2ON-OFF	FILTERCONTOURBIAS
OSC1PW	OSC2PHASE		FILTERCONTOUR
OSC2WAVEFORM	OSC1PW		CUTOFFSMOOTH
OSC2FM	OSC1PHASE		MODDCO
OSC3WAVEFORM	RINGMODULATION		MODVCF
VCFBENDER	OSCBITCRUSHER		
FILTERTYPE	FILTERDRIVE		

Tabela 5 - Parâmetros a usar por sintetizador

4.2. Descritores

Nesta secção, será apresentada principalmente a análise aos descritores de áudio das duas bibliotecas a serem usadas.

4.2.1. Visualização de Dados

No inicio deste capítulo foi apresentada uma solução para o problema da criação da totalidade de combinações de parâmetros para um dado algoritmo de síntese, com a criação aleatória de um conjunto muito menor de combinações (1000), e que, por uma questão de coerência seriam fixas a cada sintetizador até ao fim do projeto.

Como as combinações foram geradas aleatoriamente, é importante percebermos a abrangência destes dados no espectro total de combinações, isto é, analisar se os sons daí resultantes satisfazem uma grande parte das capacidades de cada sintetizador, e se se encontram bem distribuídos. Para tal foram analisados os sons resultantes pelas duas bibliotecas de descritores 'Descriptors~' e 'Timbre Toolbox' e os dados foram inseridos em gráficos 2D com o auxilio da página web "Raw" para melhor visualização e análise.

Podemos partir do princípio que quanto mais abrangentes forem as combinações, maior conhecimento terá o sistema neuronal das capacidades do algoritmo de síntese, o que resulta num acréscimo de eficácia no mapeamento entre os parâmetros e os descritores. Para o produto final resultante desta dissertação tem que existir um mapeamento fiel entre os valores resultantes da análise e os parâmetros que a 'geraram'. Se um conjunto alargado de análises se centrar em torno de um valor, a resposta do sistema dentro desse âmbito será concisa, porque lhe foram oferecidos bastantes exemplos parecidos, mas possivelmente terá uma resposta mais vaga fora dessa área. Se por outro lado os descritores de áudio resultarem numa maior equidade numérica, a eficácia do sistema é compensada e distribuída de forma justa.

Dos seguintes exemplos ilustrados graficamente poderemos no final deste capítulo tirar as seguintes conclusões:

1) Qual(ais) sintetizador(es) oferece(m) as melhores capacidades[2] para o uso deste sistema;
2) Da análise resultante, quais os descritores com maior abrangência, passíveis de uma melhor resposta quando conciliados com o sistema neuronal.

Antes da visualização dos dados, é importante salientar novamente que todos os testes e análises que se seguem não representam nem tentam concluir a total capacidade de cada sintetizador, mas sim de um número reduzido de amostras (1000), que se estipularam como principal foco de estudo para este trabalho.

As seguintes imagens apresentam-nos, graficamente dispostas, as análises feitas em pares de descritores para todos os 'indivíduos' (combinações de parâmetros). Na Figura 5 podemos depreender um balanço entre os valores resultantes dos descritores *Energy* (eixo horizontal) e *Harmonic Ratio* (eixo vertical), ambos relativos à biblioteca 'Descriptors~'. Os sintetizadores Elec7ro, NoiseMaker e U-No apresentam formas muito semelhantes e de maior abrangência do que Twist, que se distancia ligeiramente dos restantes. Numa escala de 0 a 10, existe a nível geral uma grande aglomeração de indivíduos com valores de *Energy* superiores a 6, ou seja, mais de metade do âmbito total. Podemos concluir que o sintetizador U-No é aquele que representa os dois descritores em questão de forma mais equitativa. Por outro lado, as amostras de Twist concentram os seus descritores em torno da mesma área.

Na Figura 5.1 é-nos apresentada a visualização dos valores de todos os 'indivíduos' relativos aos descritores *Inharmonicity* (eixo horizontal) e *Harmonic Ratio* (eixo vertical). Daqui se conclui que a análise feita por 'Descriptors~' revela uma maior tendência para a geração de sons de cariz harmónico já que os valores de *Inharmonicity* (que se aglomeram na sua maioria abaixo do valor 5) são em média inferiores aos de *Harmonic* Ratio.

[2] Por 'capacidade', e apenas para o efeito desta dissertação, entende-se a aptidão espectral que um sintetizador oferece, ou seja, a capacidade de poder gerar o mais variado tipo de sons e não se focar apenas numa técnica de síntese.

Ambos os descritores operam de forma inversa. Assistimos a esta tendência de forma geral entre os quatro sintetizadores.

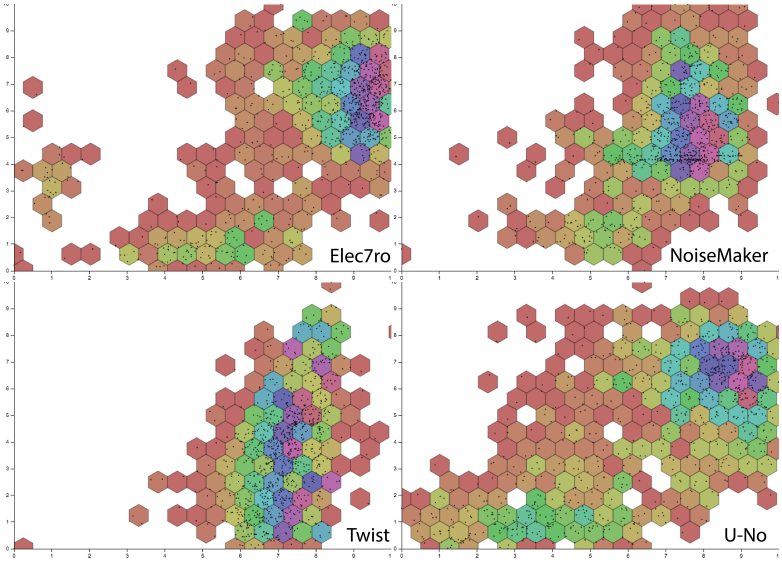


Figura 5 Visualização entre Energy e Harmonic Ratio

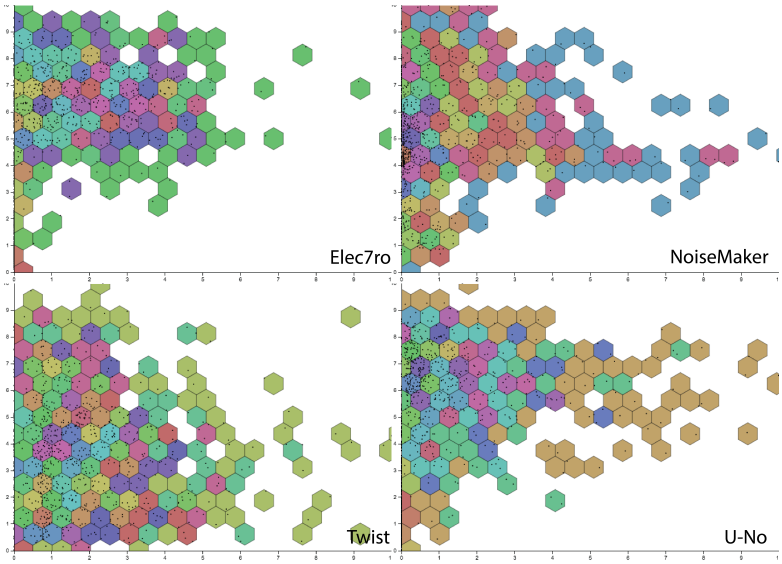


Figura 5.1 Visualização entre Inharmonicity e Harmonic Ratio

Contrariamente aos casos anteriores, e ainda recolhendo análises da mesma biblioteca de descritores, as Figuras 5.3 e 5.4 apresentam-nos valores não tão bem distribuídos. Na Figura 5.1 está presente o estudo entre *Inharmonicity* (eixo horizontal) e *Roughness* (eixo vertical). O primeiro foi já estudado, mas o segundo apresenta uma grande aglomeração em valores

abaixo de 2, o que indica que, de modo geral, os sintetizadores em questão tendem a gerar sons com pouca 'rugosidade'.

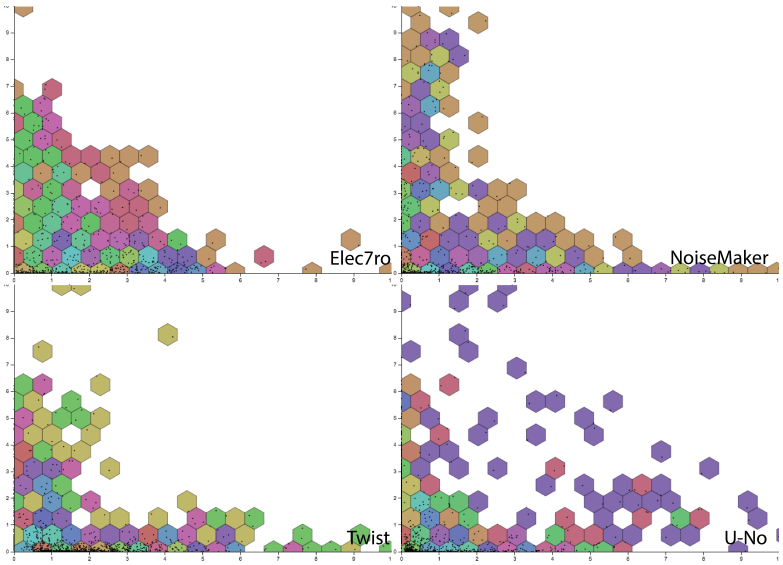


Figura 5.2 Visualização entre Inharmonicity e Roughness

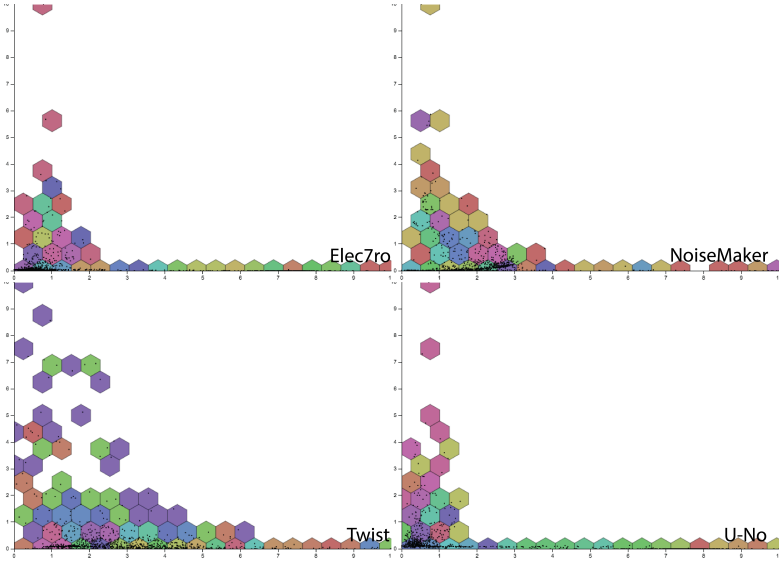


Figura 5.3 Visualização entre Spread e Brightness

A Figura 5.3 tem como objeto de estudo os descritores *Spectral Spread* (eixo horizontal) e *Brightness* (eixo vertical), e a Figura 5.4 apenas nos oferece de novo o descritor *Spectral Skewness* no eixo vertical. No primeiro caso assistimos a um aglomerado quase total de valores de *Brightness* muito baixos assim como os valores de *Spread*. É possível que numa fase avançada

do trabalho, o sistema neuronal tenha pouca capacidade de controlar os algoritmos de síntese de forma intuitiva nos casos dos descritores em questão, pois como foi referido anteriormente, carecendo de informação que abranja equitativamente o espectro de sons passíveis de ser gerados, podemos comprometer o comportamento do sistema.

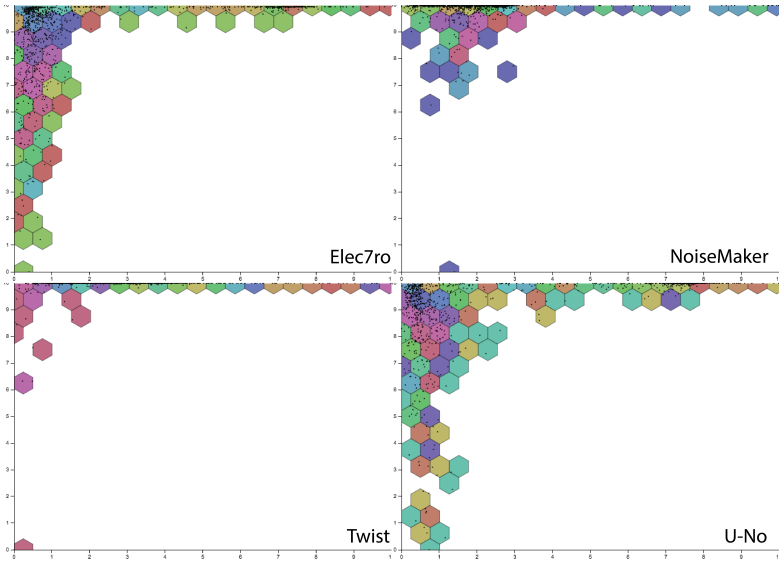


Figura 5.4 Visualização entre Spread e Skewness

Contrariamente ao *Brightness* e ao *Spread*, com tendência a se aglomerarem em torno dos valores mais baixos da escala, o *Spectral Skewness* concentra a grande maioria dos seus valores acima de 8, sendo que o Twist tem quase todos os seus indivíduos no máximo da escala, como nos mostra a Figura 5.4 que dispõe *Spectral Spread* na horizontal e *Spectral Skewness* na vertical. Mais uma vez é-nos apresentada uma situação de pouco conhecimento do espectro total de cada sintetizador.

De seguida apresentamos os gráficos resultantes da análise da biblioteca *Timbre Toolbox* realizados em MatLab para os mesmos sintetizadores, e para a mesma população (conjuntos de parâmetros).
Como seria de esperar, ambas as bibliotecas reagem de forma semelhante, principalmente no descritor *Spectral Skewness*, que já analisamos, e cuja maioria dos valores se encontra muito perto do seu máximo. A Figura 6 apresenta-nos não só o *Skewness,* mas também o descritor *Energy* que se encontra disposto no eixo horizontal, semelhante ao descritor *Harmonic Ratio* da biblioteca *Descriptors~*. Depreendemos que este descritor, cujos valores são calculados pela "soma de dos parciais harmónicos detetados num

especifico intervalo de tempo" Peeters et. al. (2011), se encontra na sua grande maioria em redor do valor mais baixo da escala. Em média existe mais informação entre os valores 0 e 3, do que no restante âmbito.

Na Figura 6.1 assistimos a uma melhor distribuição dos resultados dos descritores *Energy* e *Spectral Spread*, na horizontal e vertical respetivamente. Mais uma vez a analise do sintetizador Twist distancia-se dos restantes, mas ainda assim, no caso do segundo descritor existe uma distribuição semelhante.

Por ultimo são apresentados na Figura 6.2 *Noisiness* e *Inharmonicity*, dispostos no eixo horizontal e vertical, respetivamente. Assim como a anterior biblioteca de descritores analisou, existe uma maior tendência por parte dos quatro sintetizadores para a criação de sons harmónicos, já que os valores de *Inharmonicity* se encontram maioritariamente baixos, e muito juntos ao valor mínimo da escala.

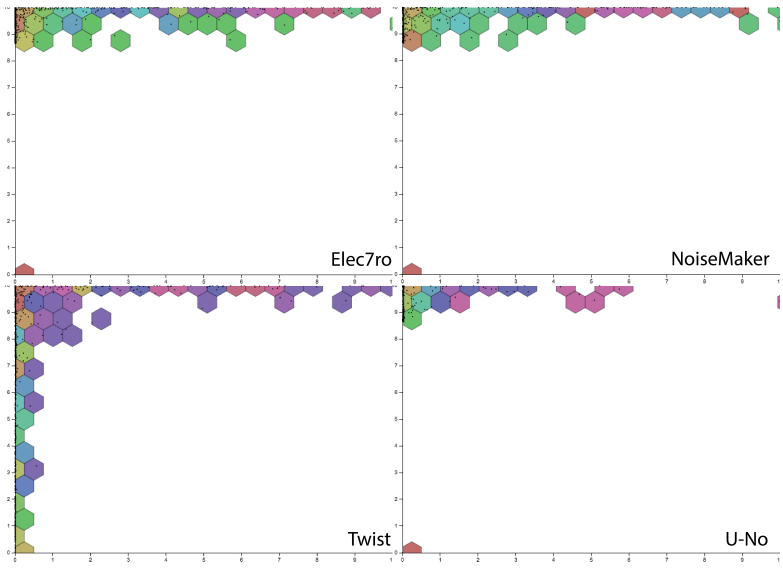


Figura 6 Visualização entre Energy e Skewness

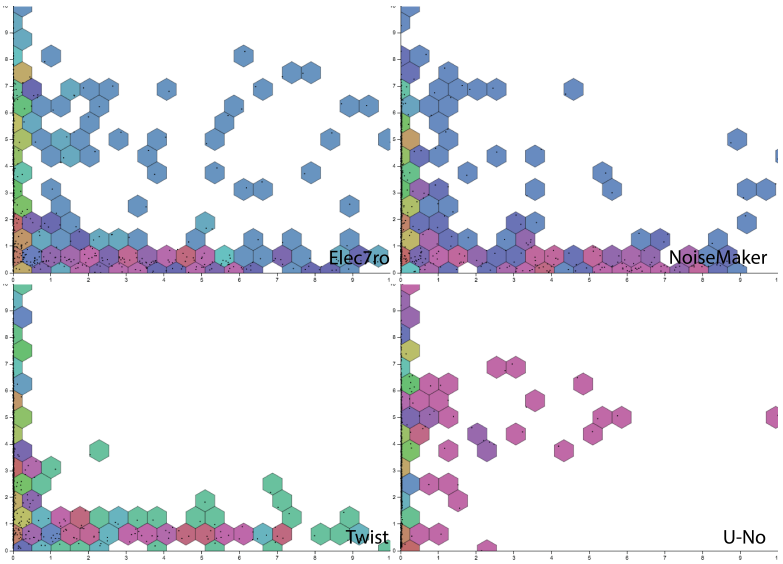


Figura 6.1 Visualização entre Energy e Spread

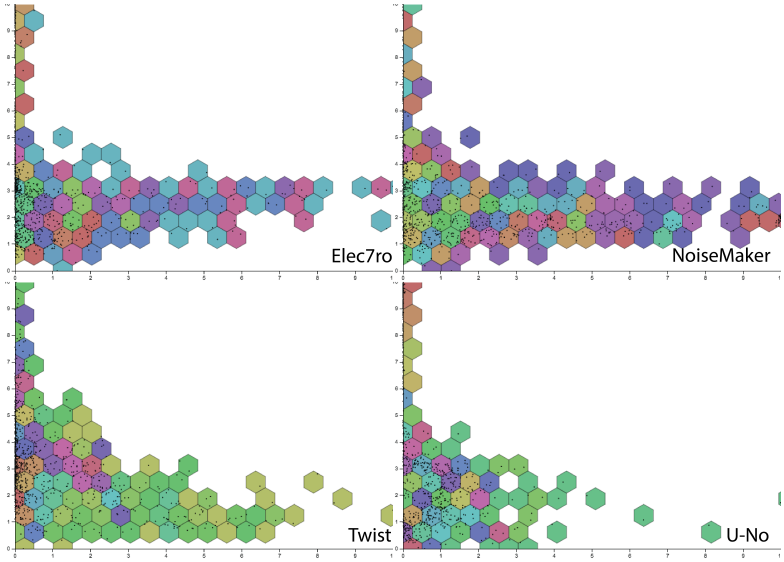


Figura 6.2 Visualização entre Noisiness e Inharmonicity

4.3. Conclusões

Foram analisados os dados recolhidos pelos descritores de áudio de duas bibliotecas diferentes: *Descriptors~*, objeto programado para funcionar em Max/MSP, do qual foram retirados valores de sete descritores distintos, e *Timbre Toolbox*, totalmente programado no ambiente de MatLab, que contribuiu com cinco descritores o mais próximo possível dos homónimos da primeira biblioteca. *Timbre Toolbox* foi utilizado como recurso e reserva a *Descriptors~*. Ambas mostraram igual potencial para contribuir na continuidade deste trabalho, visto que os resultados analisados por ambas foram coerentes, ainda que nesta altura do estudo teórico, dizer que os resultados apresentados são 'coerentes' não é mais do que teoria. Só quando os valores acima apresentados forem transportados para o domínio prático, poderemos perceber se o sistema tem ou não a capacidade se tornar intuitivo, e controlar dessa mesma forma os algoritmos de síntese.

Na sua generalidade, os sintetizadores em estudo mostraram aptidão para a continuação do trabalho, contudo, julgo que surgem dois obstáculos:

1) O facto de Elec7ro, NoiseMaker e U-No nos apresentarem capacidades muito semelhantes, e apenas Twist romper essa tendência. As análises feitas aos três primeiros mostram-nos uma 'população' distribuída de forma muito parecida no espectro de cada descritor. O caso mais gráfico é possivelmente o da Figura 5 entre os descritores *Energy* e *Harmonic Ratio*. Apresentam formas muito parecidas tanto no grande aglomerado como na 'cauda' que resulta dessa reunião de informação. A principal causa para este fenómeno, ainda que vaga e remota, é possivelmente o facto de os três sintetizadores terem sido desenvolvidos pela mesma marca (TAL). Para concluir esta ideia precisaríamos de muitos mais dados e testes do que aqueles que temos, mas ainda assim parece a mais plausível.

2) Consequência da alínea anterior, e já aqui exposto, o facto de existirem vários espaços abertos (sem informação) em grande parte do espectro de descritores como *Skewness* (em ambas as bibliotecas) ou *Energy* (*Timbre Toolbox*). Como vimos no Capítulo 3, tornou-se difícil e demorada a criação de todas as combinações possíveis de parâmetros para todos os sintetizadores. Mais à frente percebeu-se que quanto mais informação o sistema neuronal tiver ao seu dispor, melhor será a sua resposta, mas por outro lado demorará mais tempo a 'treinar' e a aprender com todos os exemplos.

Na minha opinião, e percebendo a quantidade de informação passível de ser gerada por um só sintetizador, a falta de informação sonora pode estar na base destes dois problemas. Possivelmente se acrescentássemos outros 1000 exemplos, a diferença não seria muita (dada a quantidade de combinações possíveis), mas se tivéssemos poder computacional suficiente para treinar uma rede neuronal com as capacidades totais de um sintetizador, teríamos com certeza uma maior área espectral abrangida. Por outro lado, deve ser ponderado o facto de um determinado algoritmo de síntese ser focado num tipo de som distinto, e desta forma não ter de todo capacidades para se propagar no espectro de cada descritor. Por exemplo, ao usarmos um sintetizador de som por modelos físicos, não podemos esperar que este tenha um som parecido com um sintetizador analógico e vice-versa. Do ponto de vista de cada um, o outro tem características diferentes, ou se quisermos, limitações diferentes. Estas diferenças têm repercussões na análise do áudio resultante.

O ultimo exemplo pode ajudar-nos a perceber o porquê do instrumento virtual Twist se diferenciar dos restantes. Enquanto que os algoritmos de TAL foram criados para emular o som de síntese analógica, e por isso são considerados sintetizadores virtuais analógicos, Twist é um sintetizador que usa *Spectral Morphing*, técnica de síntese sonora diferente da usada nos sintetizadores de TAL, e que se aproxima à síntese por modelos físicos.

A Tabela 7 tem representado o percurso até agora percorrido dentro do processo alvo do trabalho. As etapas de seleção dos parâmetros de síntese e descritores a usar, foram já abrangidas, e as condições foram criadas para a implementação do trabalho que será o principal foco no próximo capítulo. Serão explanados todos os passos percorridos para converter a informação aqui analisada, num sistema de otimização de parâmetros pensado para ter uma resposta intuitiva.

Tabela 6 – Processo base. Tanto os parâmetros de síntese como os descritores a usar, foram já abordados.

5. Implementação

Neste capitulo será apresentada a implementação do trabalho prático, usando toda a teoria e aprendizagem estudada nos capítulos anteriores. Esse estudo prévio, tornou mais simples a parte prática do trabalho, já que os principais problemas estavam resolvidos ou contornados.

5.1. Wekinator e treino das Redes Neuronais

Como já apresentado, Wekinator é um programa grátis e multiplataforma, destinado à criação de sistemas de Machine Learning. Usa algoritmos de gestão de dados com capacidade de mapear os exemplos que lhe são oferecidos às saídas desejadas, que é no fundo o tipo de sistema que este trabalho necessita. Desta forma, o Wekinator será responsável por criar uma ligação direta entre um conjunto de parâmetros e os valores dos descritores que daí resultam. Contudo, no fim deste processo serão os parâmetros de síntese a serem controlados pelos descritores. Desta forma:

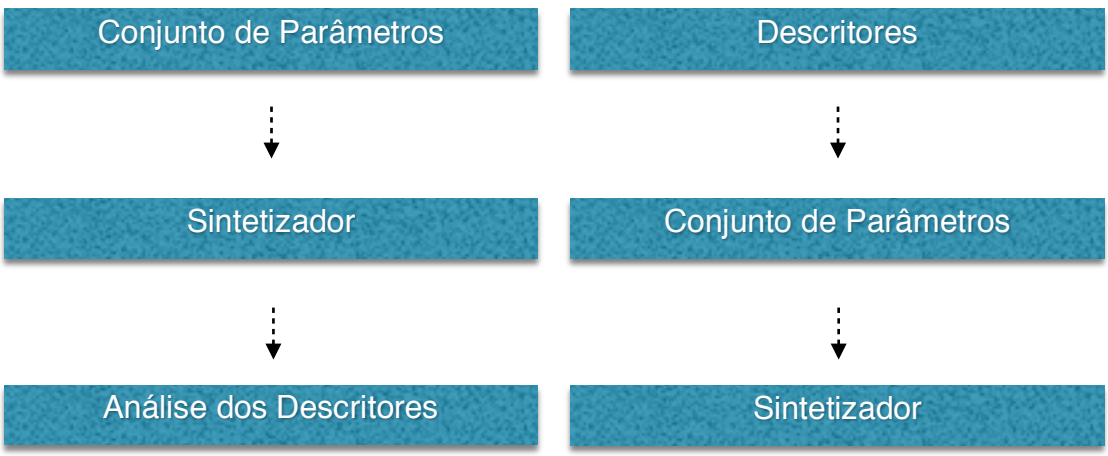


Tabela 7 – Processo base do trabalho prático

A Figura 7 contém do lado esquerdo o processo base do projeto, e estes foram os passos até agora:

1) Foi gerado um conjunto de parâmetros que controlam o algoritmo de síntese;
2) Os parâmetros foram interpretados pelo algoritmo de síntese;
3) Os sons resultantes dos primeiros parâmetros foram analisados pelos descritores de áudio.

Do lado direito encontramos o processo que pretendemos alcançar, onde os descritores foram transportados para o topo da 'cadeia', fornecendo desta vez um controlo de alto nível ao sintetizador em questão, ao invés de analisarem o resultado sonoro do sintetizador. Se no fim do processo pretendemos ter controlo manual sobre os descritores, estes tomarão a forma de *input* no Wekinator, enquanto que os parâmetros de síntese serão os *outputs*. No fundo, e a título de exemplo, queremos fazer corresponder à alínea de descritores:

Energy	Brightness	H. Ratio	Roughness	Inharm	Spread	Skewness
-13.80	0.81	0.79	0.44	0.04	0.72	-14

a respetiva alínea de parâmetros:

FilterType	Cutoff	Resonance	FilterContour	OSC1	OSC2	OSC3	…
0.68	0.3	0.04	0.88	0.4	0.42	0.32	…

Ao criar um novo projeto no Wekinator, somos convidados a especificar a quantidade de entradas e saídas a serem usados assim como as portas OSC que permitirão a comunicação com o MAX. A partir das mensagens OSC '/wek/inputs' e '/wek/outputs' temos acesso a uma porta de entrada no programa, por onde serão enviadas a cada iteração as mensagens relativas aos descritores (entradas) e os respetivos parâmetros (saídas).

A Figura 8 apresenta-nos a janela principal do projeto que acabámos de criar. Nela encontramos vários controlos tanto para a edição de cada saída (Figura 8.1) como para gravar, treinar e correr os exemplos dados (Figura 8.2).

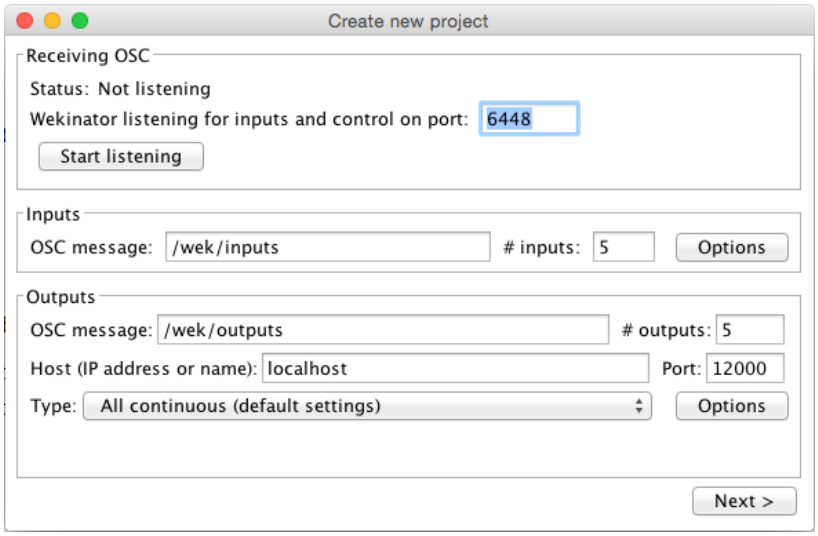


Figura 7 - Novo projeto de Wekinator

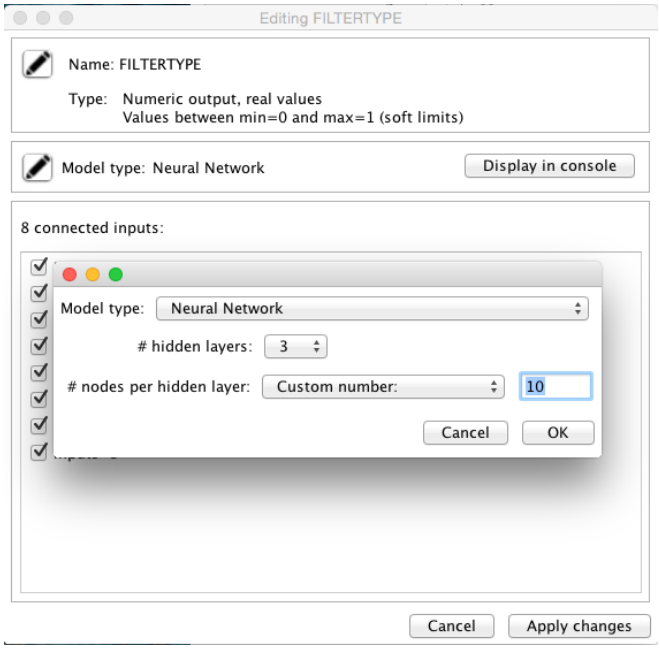


Figura 7.1 - Controlos e edição de parâmetros de saída em Wekinator

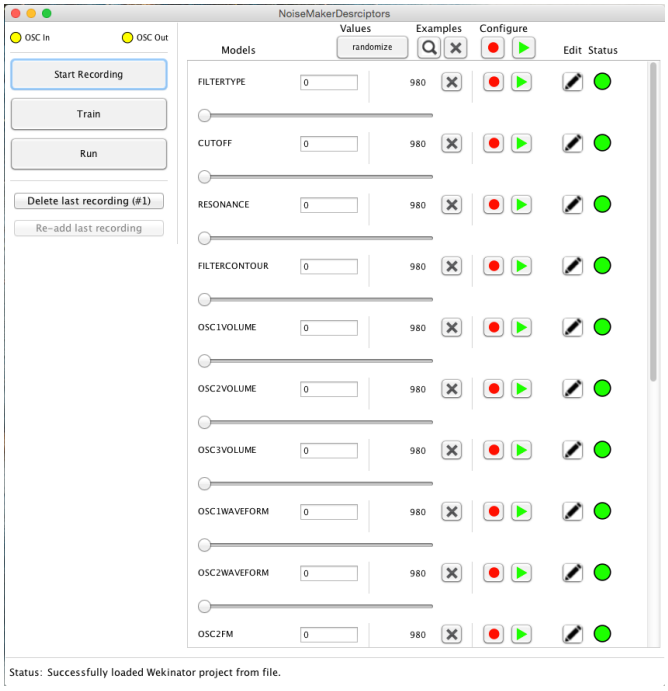


Figura 7.2 - Janela principal de Wekinator

Na edição de cada saída é possível escolher qual o algoritmo que este deve seguir (entre as opções *Neural Network*, *Linear Regression* e *Polynomial Regression*), assim como alguns parâmetros do algoritmo. No Capítulo 2 foi explanada a diferença entre *Linear Regression* e *Polynomial Regression*, e vimos que no modo Polinomial teríamos que especificar qual a ordem a ser usada. Mas numa base de dados da dimensão da nossa que foi criada aleatoriamente, não sabemos qual a ordem a usar, e possivelmente seria bastante alta, o que poria em causa o tempo de treino do algoritmo. A grande vantagem das Redes Neuronais comparativamente ao algoritmo Polinomial, é que resolvem facilmente o problema sem que seja especificado qualquer argumento antes de proceder ao treino, abrangendo todos os exemplos dados.

Assim que todas as saídas forem calibradas, podemos começar a gravar os exemplos. Um patch em MAX lê ambos os ficheiros com os descritores e os parâmetros, e envia-os via OSC, distribuindo-os entre entradas e saídas, respetivamente. Assim que o carregamento é feito, a base de dados criada no Wekinator é 'treinada' através no botão *Train*. É nesta fase que todos os mapeamentos são feitos, e as Redes Neuronais 'preenchidas' com os dados gravados. A Figura 9 apresenta-nos uma parte do patch criado para os vários propósitos do trabalho. Os dois objetos a vermelho contêm as informações

relativas aos descritores e parâmetros. Os objetos que aparecem a amarelo acrescentam o indicativo a cada mensagem fazendo com que sigam caminhos diferentes, e a verde encontra-se a secção responsável por receber a informação vinda do Wekinator, ambas via OSC.

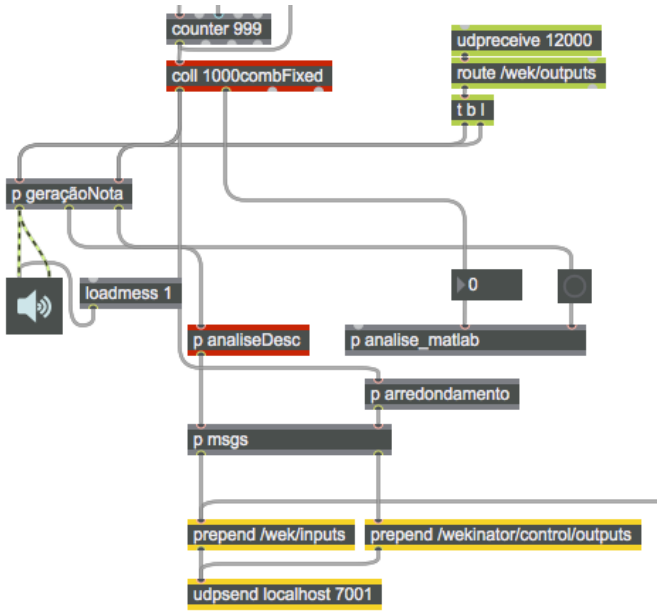


Figura 8 - Parte do patch de MAX contendo a comunicação OSC

Quando o treino do Wekinator é concluído, todo o sistema está finalmente apto a ser testado. É no MAX que serão controlados os valores de cada descritor, assim como a geração de som. O Wekinator faz apenas o mapeamento entre os valores que recebe e os parâmetros de cada sintetizador.
Na seguinte secção será estudada a resposta das Redes Neuronais a todo sistema.

5.2. Testes Práticos do Sistema

O *patch* de Max descrito anteriormente, servirá também para controlar o Wekinator e as suas saídas. Essa informação será enviada de novo para o MAX e para o sintetizador em questão. A Figura 10 apresenta-nos o *patch* completo onde está disponível o controlo de alto nível para todo o sistema.
A secção verde e laranja é referente à biblioteca *Descriptors~*, enquanto que a azul controla os descritores de *Timbre Toolbox*. O primeiro, foi treinado de formas diferentes:

1) Com todos os descritores para todos os parâmetros;
2) Com apenas quatro descritores para todos os parâmetros.
3) Teste de performance. É gerada uma combinação aleatória de parâmetros, analisada pelos descritores, e de seguida enviada para o Wekinator, que devolve a informação em forma de parâmetros de novo para o sintetizador.

Pode não parecer, mas as duas primeiras abordagens têm no final resultados diferentes. Senão vejamos, no primeiro caso dispomos de 1000 combinações para uma rede de mapeamento entre 7 descritores e e 15 parâmetros. Na segunda, temos o mesmo número de combinações, mas para uma rede mais leve de 4 descritores para os mesmos 15 parâmetros. Na Figura 11 vemos ilustrado esta mesma situação. Dentro das duas redes de mapeamento existe o mesmo número de combinações. Porém, o sistema irá tornar-se mais conciso e dessa forma mais coerente, se dispuser de menos elementos na rede. Desta forma:

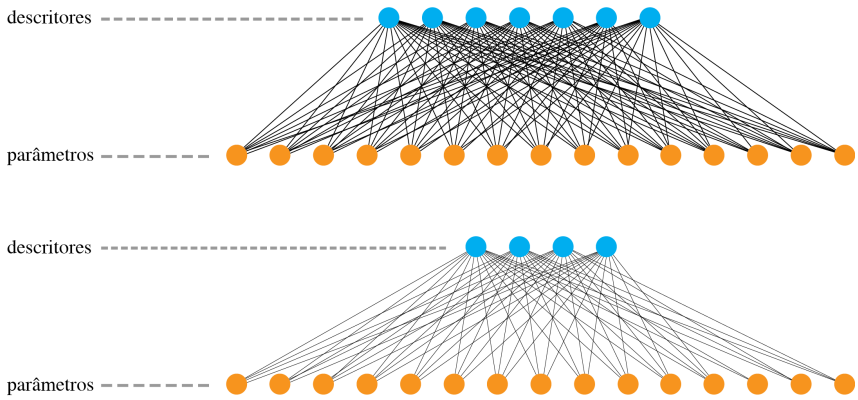


Figura 9 - Comparação entre treino com 7 ou 4 descritores

As próximas secções terão como objetivo o estudo e análise do comportamento tanto dos parâmetros de síntese como dos descritores, com vista na programação do patch final que servirá como interface deste trabalho. Contudo, na análise de descritores, não existem cálculos, formulas ou gráficos que comprovem as afirmações relativas a estes testes. São estudos auditivos, onde se põe à prova a componente intuitiva do sistema. É, portanto, um teste de cariz sensorial, inapto a ser medido.

5.2.1. Testes para sete descritores

O primeiro sintetizador a ser testado é o Elec7ro a ser controlado por 7 descritores. Os valores a ser enviados do MAX para Wekinator, em teoria deveriam estar dentro de um âmbito de 0 a 10, porque foi dentro desses valores que os descritores foram treinados, como vimos no Capítulo 4. Mas, assim como os algoritmos de regressão, também as Redes Neuronais oferecem uma resposta fora do âmbito para que foram treinadas, ou seja, se for enviado para o Wekinator um valor inferior a 0 ou superior a 10, este continuará a dar uma resposta que corresponde a nível vetorial àquela que foi dada dentro do âmbito. Desta forma, também os valores que não pertencem a essa escala serão testados.

É importante notar que apenas serão sujeitos a análise os descritores que actuarem de forma diferente do esperado.

5.2.1.1. Análise de Parâmetros

Começando pelo **Elec7ro**, ao fazer a comunicação pela primeira vez entre os dois programas, é detetado facilmente o primeiro problema: de todos os parâmetros associados a este sintetizador, nem todos oferecem uma resposta linear, mas sim classificativa, ou seja, parâmetros que identifiquem, por exemplo, o tipo de onda, não fazem a passagem gradual entre uma onda e a seguinte. O que acontece é: se o sintetizador oferecer 3 tipos de onda por oscilador, o âmbito do valor do parâmetro (0.0 a 1.0) é dividido em 3. Desta forma, e no caso do Elec7ro, quando o valor de 'OSC1WAVEFORM' passa de 0.33 para 0.34 resulta numa mudança de som abrupta que se percebe de forma óbvia. O mesmo fenómeno acontece entre os valores 0.66 e 0.67, assim como no parâmetro relativo ao tipo de filtro ('FILTERTYPE').

No caso do **NoiseMaker**, os parâmetros que constituem este problema têm que ver também com o tipo de filtro, e tipos de onda.

5.2.1.2. Análise de Descritores

Este é ser um campo que pode gerar alguma discussão já que alguns descritores parecem não atuar na devida forma.

O primeiro sintetizador a estudar é o **Elec7ro**. Começando pelo primeiro descritor *Energy*, que antes dos seus valores serem convertidos para uma escala de 0 a 10, tinha como objetivo analisar uma amostra e devolver o seu valor em decibéis, este parece não atuar da melhor forma. A causa pode invocar o facto de ser um descritor preparado para analisar a energia contida dentro de um determinado intervalo de frequência. Desta forma, se o valor de saída de *Energy* é dado em dB's, então não parece fazer sentido associa-lo a uma rede neuronal cuja resposta se pretende sobretudo harmónica. A nível prático, este problema é percetível ao variar os valores do descritor. Percebe-se que não é apenas a energia total do som que se altera, mas também conteúdo harmónico. Para solucionar este problema, o Wekinator disponibiliza a opção de desconectar uma determinada entrada a uma ou mais saídas, como mostra a Figura 12. O *input-1* (*Energy*) foi desconectado dos parâmetros que mais contribuíam para o problema, as saídas 8 e 13,

'CUTOFF' e 'OSC2FM' respetivamente. Este será um procedimento regular para os seguintes testes.

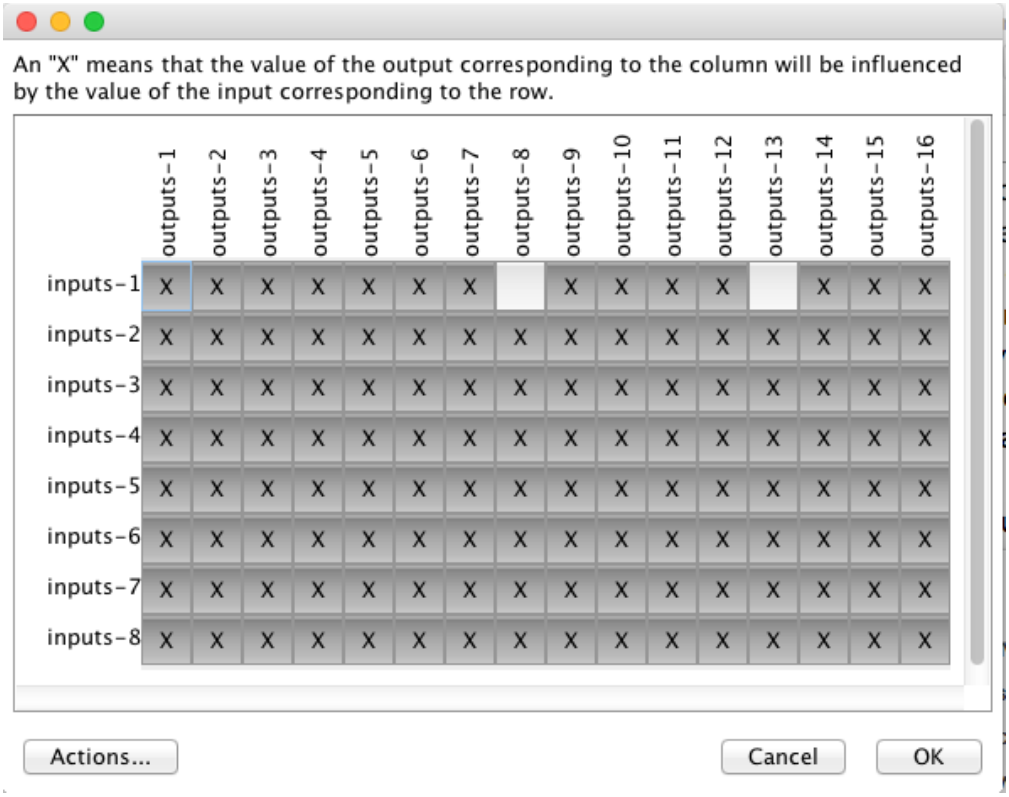


Figura 10 – Relação entre entradas e saídas no Wekinator

De seguida, o descritor que merece analise é *Harmonic Ratio*, já que dentro da escala em que se insere, responde de forma diferente em determinadas áreas. Após algumas experiências, conclui-se que *Harmonic Ratio* apenas responde da forma que é esperada, sensivelmente a partir de meio da escala para cima, ou seja de 5 para 10. Este fenómeno pode ser explicado com o estudo que foi feito no Capítulo 4 sobre este descritor, enfatizando mais uma vez a importância que a visualização de dados teve para o decorrer do trabalho. Como vimos na Figura 5 (pág. 39) que expõe em dois eixos os descritores *Energy* e *Harmonic Ratio*, este último tem a maior aglomeração de informação entre o valor 5 e 9. Como também foi destacado no último capítulo, a falta de exemplos de treino que abranjam da melhor forma o espectro de cada descritor é importante para performance dos mesmos, e este é um caso fiel a esse princípio.

Para continuar a análise aos restantes descritores, *Harmonic Ratio* será ajustado a 7, valor confortável dentro do âmbito onde tem mais exemplos treinados.

Roughness é o descritor que representa uma maior incógnita. Isto porque perceptualmente tem um comportamento mais plausível fora do âmbito em que foi analisado. Pessoalmente, considero que acrescenta 'rugosidade' ao som a partir do momento em que o seu valor ultrapassa 20, sendo que os seus valores foram 'escalados' para um âmbito de 0 a 10.
Referindo-se ao descritor *Roughness*, Bernardes (2011) afirma "O descritor de dissonância sensorial expressa aspetos inatos da perceção humana que regulam o encanto de uma sonoridade"[1].
Assim como o *Roughness*, também *Spectral Skewness* sofre o mesmo problema, mas desta vez com valores mais elevados. Este descritor apenas oferece alguma mudança significativa ao som a partir do valor 30, sensivelmente. Infelizmente não houve até à data uma resolução, nem tão pouco uma causa para este problema.
O próximo e último descritor que necessita de análise é *Spectral Spread*. Depois de estudada a sua 'população' na Figura 5.4 (pág. 41) do capítulo anterior, percebe-se que este descritor terá um comportamento parecido a *Harmonic Ratio*, já que o grande número de indivíduos com que contribuiu para o treino se encontra sensivelmente entre os três primeiros valores da escala. Desta forma, *Spread* mostra grande capacidade de resposta entre 0 e 3, sendo que daí em diante até ao fim da escala, tem poucos pontos de interesse. Importa notar que, assim como *Roughness* e *Skewness*, também este descritor tem uma resposta interessante fora da sua escala, quando se aproxima do valor 100.
Para o sintetizador **NoiseMaker** e **U-No**, o descritor *Energy* foi também desconectado de todos os parâmetros relativos a conteúdo harmónico, sendo que ficou associado maioritariamente aos volumes dos vários osciladores, assim como 'FILTERDRIVE', e 'ORBITCRUSHER', ambos relativos a distorção sonora.

Os restantes descritores têm uma resposta parecida com caso do Elec7ro, mas interessa salientar que *Spread* e *Skewness* têm agora respostas mais coerentes dentro do seu âmbito de treino.

Já o **Twist** tem vários pontos a favor neste tipo de teste. Os parâmetros *Spread*, *Roughness* e Skewness, que são postos de parte no seguinte teste, funcionam bastante bem com este sintetizador. Também aqui existem as quebras abruptas de som devido aos parâmetros classificativos, mas ainda

[1] De Inglês: "The descriptor sensory dissonance expresses innate aspects of human perception that regulate the "pleasantness" of a sonority." (Bernardes, 2011) página 62.

assim cada descritor oferece uma boa perceção da sua identidade. Já o *Energy* está confinado a uma determinada gama de valores onde existe conteúdo sónico, por isso irá ser regulado para 10, para que se prossiga o teste. Tanto *Brightness* como *Harmonic Ratio* funcionam como esperado, já o *Roughness* e *Inharmonicity* têm uma resposta mais intuitiva dentro de uma determinada gama de valores.

5.2.1.3. Testes para quatro descritores

Conforme foi descrito na segunda secção deste capítulo, existem vantagens em utilizar menos entradas para o mesmo valor de saídas. Desta forma, com a mesma quantidade de treinos existe uma maior coesão no mapeamento que é feito pelo sistema neuronal. O teste que se segue permite corroborar esta afirmação, na medida em que a redução para quatro dos sete descritores usados anteriormente, fortaleceu as capacidades dos que permaneceram.

Foram filtrados do primeiro conjunto de descritores *Energy*, *Brightness*, *Harmonic Ratio*, e *Inhamonicity*. Após algumas experiências com o sintetizador **Elec7ro**, conclui-se rapidamente que o sistema oferece melhores resultados do que o primeiro. A diferença está principalmente num fluxo mais suave de informação, em que ao incrementar o valor de um determinado descritor se percebe claramente a sua identidade. Também com menos entradas se consegue perceber melhor o efeito de cada descritor nos parâmetros de síntese. Por exemplo, valores altos de *Brightness* e/ou *Harmonic Ratio* resultam no incremento dos parâmetros 'NOISEVOLUME' e 'CUTOFF'. Também o descritor *Inharmonicity* está bastante relacionado com a Modulação por Frequência do segundo oscilador ('OSC2FM').

No caso do **NoiseMaker** e do **U-No** também esta adaptação teve sucesso, criando um sistema mais intuitivo e fácil de usar do que o anterior.

Depois de alguma calibração, os descritores do **Twist** acabam por retirar o melhor do sintetizador. A calibração foi em termos de desconexão de entradas para saídas como já realizado nos anteriores sintetizadores. Os parâmetros 'CH1ON-OFF' e 'CH2ON-OFF' contribuíam para que na maioria dos casos não houvesse conteúdo sónico.

5.3. Teste de Performance

Para o teste de performance foi gerada uma combinação aleatória de parâmetros e carregada num sintetizador. O resultado sónico foi depois analisado pelos descritores, e de seguida enviado para o Wekinator, que devolve a informação em forma de parâmetros de novo para o sintetizador. Em termos performativos, esta característica do sistema poderia ser interessante, na medida em que, em teoria, o sistema devolveria um som semelhante ao que foi gerado pela primeira vez, pois ambos teriam valores semelhantes nos vários descritores.

O sistema mostrou algumas debilidades na execução deste processo. Os resultados demonstram uma disparidade entre os dois sons, e as Figuras 12.1 e 12.2 representam isso mesmo, embora auditivamente, ambos tenham semelhanças.

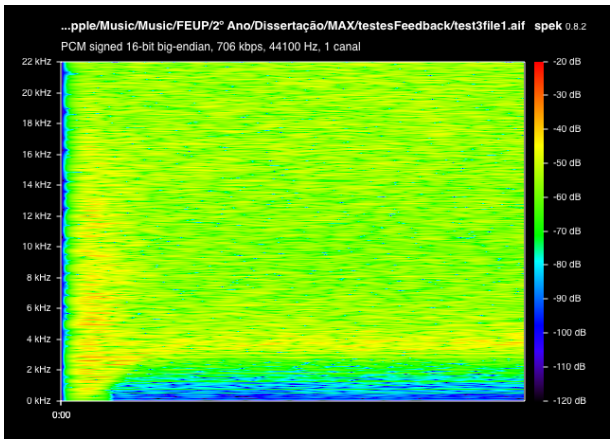


Figura 11– Som original de parâmetros aleatórios gerado com Elec7ro

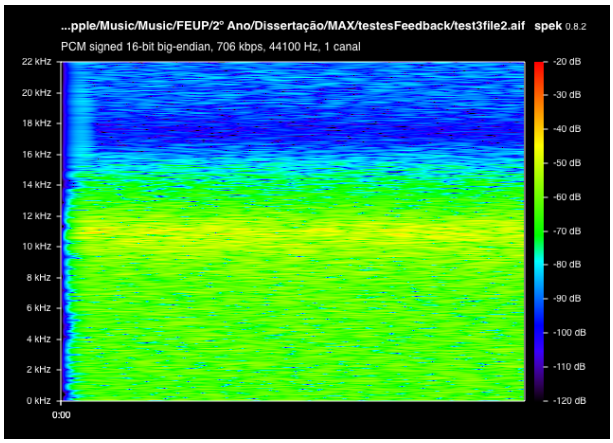


Figura 11.1 – Som devolvido pelo sistema

5.4. Conclusões

Foi apresentado e testado o sistema que representa o culminar de todo o estudo e informação presente nesta dissertação. Foram usadas análises a amostras de áudio geradas por quatro sintetizadores diferentes, para apresentar um modelo de controlo flexível de parâmetros de síntese sonora. Foram descritos todos os passos desde a programação feita em Max, aos diferentes mecanismos de operação do Wekinator, assim como os testes e treinos das Redes Neuronais. Cada descritor teve o seu foco e avaliação assim como os vários parâmetros.

Ao contrário de outros modelos de otimização automática de parâmetros que usaram Algoritmos Genéticos, e que estudámos anteriormente, este sistema é capaz de responder de forma ágil e intuitiva aos estímulos do utilizador, com recurso ao algoritmo de Redes Neuronais que se encontra à disposição no Wekinator, passando a ser possível uma otimização de parâmetros em tempo real.

Foi também realizado um teste de performance, em que o sistema reage a um som gerado por um dado sintetizador. Esse som é analisado segundo os mesmos descritores do sistema, e essa informação é enviada para o Wekinator que tem como função transformar esses dados em parâmetros prontos a serem carregados de novo no sintetizador. Embora seja uma característica de bastante interesse, ainda necessita de alguma calibração, já que os resultados não foram satisfatórios.

6. Conclusões e Trabalho Futuro

6.1. Satisfação dos Objetivos

Nesta dissertação, foi apresentado um sistema de otimização automática e intuitiva de parâmetros de síntese sonora, como proposto. O principal problema que surgiu teve que ver com a criação de um mapa de combinações de parâmetros, fundamental para o treino do sistema neuronal. Dada a dimensão das possibilidades, decidimos gerar aleatoriamente um número mais baixo de combinações, fixos para cada sintetizador, e que serviriam de amostra para todos os testes a efetuar. Os sons resultantes foram analisados segundo um conjunto de descritores de duas bibliotecas distintas, e os resultados foram expostos com o auxílio da visualização dos dados. De seguida, as Redes Neuronais foram treinadas com base nas combinações de parâmetros e consequente análise descritiva. O comportamento do sistema foi analisado do ponto de vista performativo, com especial atenção ao cariz intuitivo do mesmo. Por tudo aqui descrito, concluímos que os objetivos inicialmente propostos foram cumpridos.

6.2. Principais Conclusões

O sistema de otimização automática e intuitiva de parâmetros de síntese sonora proposto nesta dissertação, revelou-se uma ferramenta viável tanto para a composição musical, como para a musica performativa. A última pode lucrar particularmente com um sistema como o aqui proposto, não só por se mostrar uma ferramenta de uso intuitivo, mas também pelo facto da otimização automática de parâmetros ter a capacidade de calibrar qualquer número de parâmetros ao mesmo tempo., em tempo real. Em contexto de

execução ao vivo, esta tarefa não é possível para o performer, a menos que essa intensão esteja previamente automatizada.

Depois de realizados os testes sensoriais, com base na avaliação do comportamento de cada descritor, podemos responder a algumas perguntas colocadas no inicio deste documento:

1) As Redes Neuronais mostraram representar um algoritmo interessante para este género de abordagem. Para além de bastante ágil e responsivo, mostrou-se um algoritmo com curtos tempos de treino para a base de dados que dispúnhamos (treino com cerca de 1 minuto de duração para 1000 exemplos). Mas mesmo que a base de dados fosse bastante maior, existe a vantagem de o treino apenas ter que ser realizado uma vez, e o sistema responderá sempre com a mesma eficácia.
2) Para além de dispôs de tempos de treino relativamente reduzidos (mas sempre dependentes na dimensão da base de dados), este algoritmo tem a capacidade de responder a gestos rápidos com a mesma eficácia.
3) Mostrou-se bastante intuitivo na medida em que na maioria dos casos percebemos exatamente o efeito que cada descritor estava a ter no algoritmo de síntese. Esta característica, se combinada com a anterior, resulta numa boa ferramenta, não só de composição musical, mas também performativa.
4) É um sistema de otimização de parâmetros em tempo real, que lhe confere as capacidades que um performer sem recurso à computação não tem.

Para além dos últimos pontos, podemos acrescentar que a capacidade de computação é fulcral para um sistema deste tipo, principalmente na criação de parâmetros e consequente análise dos descritores. Vimos que quantos mais exemplos a Rede Neuronal tiver disponíveis para treino, mais fiável será a sua resposta. E dentro de uma dimensão tão grande de possibilidades e combinações, quanto mais preciso for o algoritmo neuronal, mais intuitivo o entenderemos.

6.3. Trabalho Futuro

Num trabalho futuro, este sistema pode ver melhorias ao nível da calibragem dos algoritmos neuronais de forma a torna-los mais precisos no seu

mapeamento. Mas sem duvida que vemos como promissora a ideia de transportar este sistema para o contexto da otimização automática e intuitiva de parâmetros de síntese sonora para um som dado.

Um dos problemas aqui expostos teve que ver com o número de combinações presentes no treino neuronal. Um treino de dimensões maiores tem a grande vantagem de tornar o sistema mais intuitivo. Mas apesar de tudo conferirá ao sistema uma maior capacidade tímbrica. Considero que numa escala tão alargada de combinações de parâmetros, há a possibilidade de existir pelo menos dois sons distintos, que depois de analisado pelos descritores ofereçam resultados parecidos ou até iguais. No entanto para cada combinação de descritores o sistema devolve apenas um resultado sónico. A possibilidade de explorar os restantes sons provenientes da mesma combinação de descritores seria um grande avanço para este trabalho. Dessa forma seria alargada a capacidade de explorar o espaço tímbrico de cada sintetizador.

Cada sintetizador ofereceu alguma resistência ao sistema, fazendo com que por vezes a resposta não fosse a esperada. O facto de se adequar a qualquer sintetizador externo pode ter prós e contras. É inegável a importância de um sistema pensado para se adequar a qualquer ambiente de síntese. Mas nenhum desses sintetizadores foi pensado para interagir com um programa deste tipo. Dessa forma, considero que o desenvolvimento de um algoritmo de síntese próprio para a utilização com este modelo pode ser bastante proveitoso.

Outro caminho será a implementação deste modelo para parâmetros e descritores que atuam no domínio do tempo. Para solucionar este problema teremos que retroceder novamente ao tamanho do treino e das combinações de parâmetros, e os problemas adjacentes, já que a quantidade de parâmetros sofreria um aumento.

Se todas as implementações anteriores se concretizarem, vejo de forma clara a possibilidade de tornar este modelo num instrumento musical (de natureza física ou virtual) que possa incorporar um contexto de performance e composição musical, que foi no fundo a grande ambição deste projeto.

7. Referências

1. Abdullah, M. H., & Blackburn, A. (2015) Spectromorphological notation: exploring the uses of timbral visualization in ethnomusicologi- cal works.
2. Blackburn, A., & Penny, J. (n.d.). Timbral notation from spectrograms: notating the un-notatable?
3. Bernardes, G. (2014). Content-based algorithmic-assisted audio composition.
4. Collins, N. (n.d.). Reinforcement learning for live musical agents.
5. Fiebrink, R. & Cook, P. R. (2010). The wekinator : a system for real-time , interactive machine learning in music rebecca fiebrink princeton university princeton university, *4*(3), 2005.
6. Fels, S. S., Hinton, G. (1998) glovetalkii - A Neural-Network Interface which Maps Gestures to Parallel Formant Speech Synthesizer Controls
7. Garcia, R. A., & St, A. (2001). Automating the desi g n of sound synthesis techniques using evolutionary methods machine listening group, 1–6.
8. Goudarzi, V. (n.d.). Neuroklang : Real-time Timbre Control using Neural Networks.
9. Groux, S. L. E. (2002). A neural network principal component synthesizer for expressive control of musical sounds.
10. Kotsiantis, S. (2007). Supervised Machine Learning: A Review of Classification Techniques, In *Frontiers in Artificial Intelligence and Applications* (VOL-160. Pags – 3-24)
11. Lee, M., & Wessel, D. (1992) "Connectionists Models for Real-Time Controlo f Synthesis and Compositional Algorithms", *ICMC Proceedings*.
12. Macret, M., & Pasquier, P. (2014). Automatic Design of Sound Synthesizers as Pure Data Patches using Coevolutionary Mixed-typed Cartesian Genetic Programming.

Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, 309–316. Http://doi.org/doi:10.1145/2576768.2598303

13. Macret, M., Pasquier, P., & Smyth, T. (n.d.). Automatic Calibration of Modified FM Synthesis to Harmonic Sounds using Genetic Algorithms, 0–7.
14. Macret, M. (2013). Automatic tuning of the op-1 synthesizer using a multi-objective genetic algorithm.
15. Malt, M., & Jourdan, E. (n.d.). Zsa . Descriptors: a library for real-time descriptors analysis.
16. Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., Paris, F., & Mcadams, S. (2011). The Timbre Toolbox : Extracting audio descriptors from musical signals, 130(5). Http://doi.org/10.1121/1.3642604
17. Smalley, D. (1997). Spectromorphology : explaining sound-shapes, 2(2), 107–126.
18. Uncini, A. (2003). Audio signal processing by neural networks, 55, 593–625. Http://doi.org/10.1016/S0925-2312(03)00395-3
19. Karaali, O., Corrigan, G., Gerson, I., & Road, E. A. (1996). Speech Synthesis with Neural Networks, (September), 45–50.
20. Yee-king, M., & Roth, M. (n.d.). An unsupervised software synthesizer programmer.
21. Yee-king, M. (2011) Automatic Sound Synthesizer Programming : Techniques and Applications.
22. Wessel, D., Drame C. & Wright, M. (1998), "Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis", *ICMC* (Univ. Of Michigan, Ann Arbor, USA, 1998)

Websites:

23. TAL Software - https://tal-software.com/home
24. Raw - http://raw.densitydesign.org
25. Nature of Code, Daniel Shiffman - http://natureofcode.com
26. Neural Networks and Deep Learning, Michael Nielsen - http://neuralnetworksanddeeplearning.com
27. EARS Glossary of Terms www.ears.dmu.ac.uk/spip.php?rubrique28
28. Weal, R. ElectroAcoustic Resource Site (http://ears.pierrecouprie.fr)

Anexos

Anexos

Anexo I - Teste nº 3

```
————————————————————
```

```
Option Base 1
Sub macro1()
Optimise (False)

MyDir = MyPath = ActiveWorkbook.Path

FileCount = 1

Columns("A:K").Clear
MyArray = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Row = 0

Loop1:
Column = 11
Row = Row + 1
Display = Display + 1

If Row > Rows.Count - 100 Then

    ActiveWorkbook.SaveAs Filename:= _
        "/apple/Desktop/Test/Test" & FileCount & ".xlsm", FileFormat:= _
        xlOpenXMLWorkbookMacroEnabled, CreateBackup:=False
    Columns("A:L").Delete Shift:=xlToLeft

t = MsgBox("Click on Ok to Continue", vbYesNo)
If t <> 6 Then Exit Sub
Row = 1

End If

Range(Cells(Row, 1), Cells(Row, 11)).Value = MyArray
Range(Cells(Row, 1), Cells(Row, 11)).Select
Loop2:
If Column = 0 Then GoTo Quit
If Int(MyArray(Column) + 0.0001) = 1 Then MyArray(Column) = 0: Column = Column
- 1: GoTo Loop2
MyArray(Column) = MyArray(Column) + 0.1
GoTo Loop1

Quit:
Optimise (True)
End Sub
Sub Optimise(Flag As Boolean)
On Error Resume Next
Application.ScreenUpdating = True
Application.DisplayAlerts = Flag
Application.EnableEvents = Flag
Application.DisplayStatusBar = Flag
ActiveSheet.DisplayPageBreaks = Flag
If Flag = False Then
```

```
Application.Calculation = xlCalculationAutomatic
Else
Application.Calculation = xlCalculationManual
End If
On Error GoTo 0
End Sub
```

———————————————————

Anexo II – Teste nº 4

———————————————————

```
var increment = 0.1;
var maximum = 1.0;
var list = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0];

function bang()
{
     list[9] = list[9] + increment;
     if(list[9] > maximum) {
          list[9] = 0.0;
          list[8] = list[8] + increment;
          if(list[8] > maximum) {
               list[8] = 0.0;
               list[7] = list[7] + increment;
               if(list[7] > maximum) {
                    list[7] = 0.0;
                    list[6] = list[6] + increment;
                    if(list[6] > maximum) {
                         list[6] = 0.0;
                         list[5] = list[5] + increment;
                         if(list[5] > maximum) {
                              list[5] = 0.0;
                              list[4] = list[4] + increment;
                              if(list[4] > maximum) {
                                   list[4] = 0.0;
                                   list[3] = list[3] + increment;
                                   if(list[3] > maximum) {
                                        list[3] = 0.0;
                                        list[2] = list[2] + increment;
                                        if(list[2] > maximum) {
                                             list[2] = 0.0;
                                             list[1] = list[1] + increment;
                                             if(list[1] > maximum) {
                                                  list[1] = 0.0;
                                                  list[0] = list[0] +
increment;

                                                  if(list[0] > maximum) {
                                                  reset();
                                                  }
                                             }
                                        }
                                   }
                              }
                         }
                    }
               }
          }
     }
```

```
     outlet(0, list);
}

function reset() {
     list = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.1];
}
```

———————————————————

Anexo III - Teste nº 5

———————————————————

```
parameters = ['VCOMODULATION', 'VCFMODULATION', 'VCORANGE', 'NOISEVOLUME',
'OSC1VOLUME', 'OSC2VOLUME', 'OSC3VOLUME', 'CUTOFF', 'RESONANCE',
'OSC1WAVEFORM', 'OSC1PW', 'OSC1SYNC', 'OSC2WAVEFORM', 'OSC2FM',
'OSC3WAVEFORM', 'VCFBENDER', 'MODBENDER', 'FILTERTYPE', 'HIGHCUT']

def frange(start, stop, step):
     i = start
     while i <= stop:
          yield i
          i += step

fileIter = 0
iter = 0
for p0 in frange(0.0, 1.0, 0.5):
     for p1 in frange(0.0, 1.0, 0.5):
          for p2 in frange(0.0, 1.0, 0.5):
               for p3 in frange(0.0, 1.0, 0.5):
                    for p4 in frange(0.0, 1.0, 0.5):
                         for p5 in frange(0.0, 1.0, 0.5):
                              for p6 in frange(0.0, 1.0, 0.5):
                                   for p7 in frange(0.0, 1.0, 0.5):
                                        for p8 in frange(0.0, 1.0, 0.5):
                                             for p9 in frange(0.0, 1.0,
0.5):
                                                  for p10 in frange(0.0,
1.0, 0.5):
                                                       for p11 in
frange(0.0, 1.0, 0.5):
                                                            for p12 in
frange(0.0, 1.0, 0.5):
                                                                 for p13 in
frange(0.0, 1.0, 0.5):
                                                                      for
p14 in frange(0.0, 1.0, 0.5):
     for p15 in frange(0.0, 1.0, 0.5):
          for p16 in frange(0.0, 1.0, 0.5):
               for p17 in frange(0.0, 1.0, 0.5):
                    for p18 in frange(0.0, 1.0, 0.5):
                              if iter % 200000 == 0:
```

```
                    if fileIter != 0:
                        file.close()
                    file = open('30combElec7ro%d' % fileIter,
'w')
                    fileIter = fileIter + 1
                file.write('%d, ' % iter)
                file.write(parameters[0])
                file.write(' %.1f ' % p0)
                file.write(parameters[1])
                file.write(' %.1f ' % p1)
                file.write(parameters[2])
                file.write(' %.1f ' % p2)
                file.write(parameters[3])
                file.write(' %.1f ' % p3)
                file.write(parameters[4])
                file.write(' %.1f ' % p4)
                file.write(parameters[5])
                file.write(' %.1f ' % p5)
                file.write(parameters[6])
                file.write(' %.1f ' % p6)
                file.write(parameters[7])
                file.write(' %.1f ' % p7)
                file.write(parameters[8])
                file.write(' %.1f ' % p8)
                file.write(parameters[9])
                file.write(' %.1f ' % p9)
                file.write(parameters[10])
                file.write(' %.1f ' % p10)
                file.write(parameters[11])
                file.write(' %.1f ' % p11)
                file.write(parameters[12])
                file.write(' %.1f ' % p12)
```

```
                                file.write(parameters[13])

                                file.write(' %.1f ' % p13)

                                file.write(parameters[14])

                                file.write(' %.1f ' % p14)

                                file.write(parameters[15])

                                file.write(' %.1f ' % p15)

                                file.write(parameters[16])

                                file.write(' %.1f ' % p16)

                                file.write(parameters[17])

                                file.write(' %.1f ' % p17)

                                file.write(parameters[18])

                                file.write(' %.1f;\n' % p18)

                                iter = iter + 1
file.close()

————————————————————————

parameters = ['filtertype', 'cutoff', 'resonance', 'filtercontour',
'osc1volume', 'osc2volume', 'osc3volume', 'osc1waveform', 'osc2waveform',
'oscsync', 'osc2fm', 'osc2phase', 'osc1pw', 'osc1phase', 'voices',
'ringmodulation', 'oscbitcrusher', 'vintagenoise', 'filterdrive']

def frange(start, stop, step):
     i = start
     while i <= stop:
          yield i
          i += step

fileIter = 0
iter = 0
for p0 in frange(0.0, 1.0, 0.5):
     for p1 in frange(0.0, 1.0, 0.5):
          for p2 in frange(0.0, 1.0, 0.5):
               for p3 in frange(0.0, 1.0, 0.5):
                    for p4 in frange(0.0, 1.0, 0.5):
                         for p5 in frange(0.0, 1.0, 0.5):
                              for p6 in frange(0.0, 1.0, 0.5):
                                   for p7 in frange(0.0, 1.0, 0.5):
                                        for p8 in frange(0.0, 1.0, 0.5):
                                             for p9 in frange(0.0, 1.0,
0.5):
                                                  for p10 in frange(0.0,
1.0, 0.5):
                                                       for p11 in
frange(0.0, 1.0, 0.5):
                                                            for p12 in
frange(0.0, 1.0, 0.5):
                                                                 for p13 in
frange(0.0, 1.0, 0.5):
```

```
                                                                    for
p14 in frange(0.0, 1.0, 0.5):
     for p15 in frange(0.0, 1.0, 0.5):
          for p16 in frange(0.0, 1.0, 0.5):
               for p17 in frange(0.0, 1.0, 0.5):
                    for p18 in frange(0.0, 1.0, 0.5):
                              if iter % 200000 == 0:
                                   if fileIter != 0:
                                        file.close()
                                   file = open('30combNoiseMaker%d' %
fileIter, 'w')
                                   fileIter = fileIter + 1
                              file.write('%d, ' % iter)
                              file.write(parameters[0])
                              file.write(' %.1f ' % p0)
                              file.write(parameters[1])
                              file.write(' %.1f ' % p1)
                              file.write(parameters[2])
                              file.write(' %.1f ' % p2)
                              file.write(parameters[3])
                              file.write(' %.1f ' % p3)
                              file.write(parameters[4])
                              file.write(' %.1f ' % p4)
                              file.write(parameters[5])
                              file.write(' %.1f ' % p5)
                              file.write(parameters[6])
                              file.write(' %.1f ' % p6)
                              file.write(parameters[7])
                              file.write(' %.1f ' % p7)
                              file.write(parameters[8])
                              file.write(' %.1f ' % p8)
                              file.write(parameters[9])
                              file.write(' %.1f ' % p9)
```

```
                                file.write(parameters[10])
                                file.write(' %.1f ' % p10)
                                file.write(parameters[11])
                                file.write(' %.1f ' % p11)
                                file.write(parameters[12])
                                file.write(' %.1f ' % p12)
                                file.write(parameters[13])
                                file.write(' %.1f ' % p13)
                                file.write(parameters[14])
                                file.write(' %.1f ' % p14)
                                file.write(parameters[15])
                                file.write(' %.1f ' % p15)
                                file.write(parameters[16])
                                file.write(' %.1f ' % p16)
                                file.write(parameters[17])
                                file.write(' %.1f ' % p17)
                                file.write(parameters[18])
                                file.write(' %.1f;\n' % p18)
                                iter = iter + 1
file.close()
```

————————————————————