RENATO A. A. BATTISTIN

IANITA : UM ESTUDO DE INTELIGÊNCIA ARTIFICIAL PARA GERAR MÚSICAS DO GÊNERO MAIS INFLUENTE DA CULTURA BRASILEIRA ATUAL - O FUNK

Orientador:

Prof. Ricardo Luis de Azevedo da Rocha

São Paulo
2022

Prof. Ricardo Luis de Azevedo da Rocha

AGRADECIMENTOS

Poderia agradecer talvez a todas as pessoas com quem tive interações na vida, pois sem uma delas é possível que tudo fosse outra coisa. Vejo em minha trajetória um cego que andava por uma floresta, ele enlouqueceu, colocou a mão no rosto em desespero, e finalmente retirou a venda que tapava seus olhos.

Agradeço a meu pai que conversou comigo quando eu tinha apenas 17 anos sobre minha escolha de carreira. Agradeço por meu caminho me levar à França, onde conheci pessoas incríveis e me redescobri. Agradeço a minhas supervisoras de pesquisa em bioinformática - Nelle Varoquaux e Sophie Abby - que me mostraram o quão interessante era o campo da inteligência artificial e me forneceram as bases que usei para desenvolver esse trabalho.

Em especial, agradeço a meu amigo Vinícius Lucena, que me deu a ideia para o tema deste trabalho em uma conversa no sofá do passado apartamento 66A. Agradeço a meus amigos e família que me incentivaram e me deram forças para completar esse projeto. Agradeço ao meu orientador por gostar e valorizar o contexto diferente que escolhi para esse estudo. E agradeço a minha namorada e amiga, Maria Clara, por ter realizado a revisão de todo esse documento sem pedir nada em troca.

"A arte existe porque a vida não basta"

-- Ferreira Gullar

RESUMO

O campo da aprendizagem profunda tem mostrado resultados muito promissores para a geração de música a partir de áudio cru. Esses formatos (.mpr, .wav, etc.) contém muita informação condensada e são acessíveis a praticamente qualquer usuário facilmente. Com isso em mente, este estudo se propõe a explorar e aplicar técnicas dessa área aplicada no contexto do funk brasileiro - gênero musical muito influente da cultura atual. Como gênero proveniente da classe mais vulnerável da sociedade, o funk nunca foi utilizado para um projeto de aprendizado de máquina, apesar de ter características interessantes para tal, como um ritmo bem marcado. Aplicando-se uma metodologia iterativa de desenvolvimento, percebeu-se as fortes limitações computacionais envolvidas nesse projeto, que não produziu resultados bons em redes treinadas a partir do zero, mas mostrou que é possível utilizar redes pré-treinadas com muito mais recursos para a finalidade proposta.

Palavras-Chave – Aprendizagem profunda, aprendizagem de máquina, áudio cru, funk brasileiro.

ABSTRACT

The field of deep learning has made impressive results in regards to music generation from raw audio. These formats (.mp3, .wav) contain a lot of condensed information and they are available to a wide variety of internet users. With that in mind, this study explores and applies machine learning techniques in the context of funk - recently the most influential brazilian music genre. It comes from the most vulnerable social class of our society, and hence it has not been used in an artificial inteligence project, even if it contains interesting characteristics to such: the well marked rhythm, for example. With an iterative development methodology, big computational limitations were discovered and compromised the success of the project in regards to training a neural network from scratch. However it was shown that it is possible to perform the task proposed with networks that had a lot more computational power involved in its training.

Keywords – Deep learning, machine learning, raw audio, brazilian funk.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Objetivos

O objetivo principal deste projeto é explorar possibilidades de inteligência artificial que sejam capazes de gerar músicas do gênero funk brasileiro a partir de uma entrada inicial, como um trecho de uma música, caracterizando uma geração condicionada. Dentro desse contexto, pode-se analisar e comparar os resultados obtidos com outros que representem o estado da arte na área de estudo de Machine e Deep Learning aplicados à música. E com essas informações gerar reflexões e considerações sobre esses campos.

Como projeto focado em desenvolver uma tecnologia nova e de resultados avaliados primeiramente de forma qualitativa, prevê-se uma metodologia experimental que vai testar diversas técnicas de aprendizado em conjunto com outras análises exploratórias dos dados. Com tal processo iterativo, espera-se entender e melhorar a qualidade das músicas produzidas com o tempo. Espera-se também, como objetivo secundário, gerar ideias e análises no âmbito cultural e musical sobre a complexidade e a produção atual do funk brasileiro, mostrando que a ideia de que o funk é um tipo de música fácil e simples não passa de um mito.

1.2 Motivação e Justificativa

O meio da engenharia se vê distante de expressões artísticas e culturais. Não há espaço na formação do engenheiro reservado para o aprendizado à apreciação ou criação de conteúdos mais artísticos e/ou criativos. E, infelizmente, isso dificulta a tão importante missão da área de formar profissionais com influência sobre o futuro do país, uma vez que a construção de uma boa sociedade não se faz apenas com números e métodos objetivos. A presença da arte e da cultura é essencial para o bem-estar do ser humano, fato já comprovado em vários estudos diferentes [7], e logo deve ser compreendida e levada em consideração na atuação da profissão.

O funk é expressão social e cultural da classe mais vulnerável da nossa sociedade, oriunda de processos históricos injustos que marcaram nosso país, e é também arte. Porém, devido às circunstâncias do nosso modelo de educação superior, que ainda é elitista, o funk não tem representação suficiente no meio acadêmico, principalmente dentro da área de exatas quando busca-se alguma interdisciplinaridade. Logo, este projeto é também para criar algo que diminua essa distância, uma ideia disruptiva para mostrar que é possível realizar um trabalho sério de engenharia com um gênero de música popular, e possivelmente incentivar a diversificação das áreas em que futuros projetos de engenharia prestarão atenção.

Além disso, o funk é conhecido pelo seu ritmo fortemente marcado e sua harmonia e composição simples, o que contrasta diretamente com estudos anteriores [8] que focam mais em gêneros como a música clássica, em que outros conceitos musicais são mais valorizados. Contudo, a diversidade e amplitude que o gênero ganhou criou vertentes muito distantes entre si, o que dificulta um processo de aprendizado de máquina. Isso permite teorizar as diferenças que aparecerão e motiva o trabalho a ser realizado como desafio de engenharia. Este estudo pode ser considerado um movimento matemático-antropofágico.

1.3 Organização do trabalho

Este relatório está dividido em seções que buscam explicar de forma estruturada os passos e conceitos necessários para a realização do experimento.

A seção "Aspectos Conceituais"conta com conceitos importantes e certa intuição matemática das transformações utilizadas pelas arquiteturas de Deep Learning para o entendimento do projeto como um todo, com isso ficará claro quais são os maiores desafios deste experimento. A seção de "Tecnologias Utilizadas"apresenta especificações técnicas que foram tomadas para a implementação do projeto, assim como descreve e justifica as ferramentas escolhidas para o projeto. A "Metodologia de Trabalho"explica o planejamento e fases previstas para a execução das ideias aqui apresentadas, detalhando mais a fundo como funciona o processo de composição da base de dados e treinamento dos modelos. Em seguida, temos a "Análise dos resultados", que busca explicitar o processo iterativo e investigativo que foi realizar o projeto. A seção de "Conclusões"fecha o trabalho com as principais reflexões, ideias e considerações aqui produzidas.

2 ASPECTOS CONCEITUAIS

Essa seção explora conceitos culturais sobre o contexto do gênero musical escolhido para o estudo, e técnicas matemáticas de forma a explicitar mais da engenhosidade e conhecimento acumulado que se encontra por trás das arquiteturas de Deep Learning atuais.

2.1 O funk na cultura brasileira

O primeiro tipo de funk a surgir no mundo teve origem nos Estados Unidos entre as décadas de 1960 e 1980 junto à ascensão do movimento negro. O gênero chegou às periferias do Brasil no final da década de 1970, com batida eletrônica agitada e dançante. Com o tempo, foi recebendo influências de elementos culturais afro-brasileiros. Nos anos 2000 houve uma grande popularização para outras partes da sociedade com grupos como Bonde do Tigrão, que obteve em 2001 um disco de platina pela Pró-Música Brasil.

Desde então houve uma explosão do estilo musical, com jovens encontrando no funk uma oportunidade de ascensão social, o que originou vários subgêneros. O "funk ostentação"marcado pela fetichização de mercadorias e demonstração de poder financeiro. O inovador "funk 150bpm", que surge no Rio de Janeiro e é caracterizado por uma batida mais acelerada. O "funk rave"ou "eletrofunk", popularizado pelo DJ GBR em 2016 e dando origem a vários outros subgêneros marcados por batidas mais eletrônicas e repetitivas como o "Mandelão", proveniente do Baile do Mandela do município Praia Grande de SP. O "funk pop", protagonizado atualmente por artistas como Anitta e Ludmilla, reconhecidas internacionalmente. O impacto cultural do gênero hoje deixa de atingir apenas o Brasil, com milhões de ouvintes em países como Estados Unidos, Portugal e Argentina [9].

Raros são os jovens que nunca tiveram contato com o gênero. Além de sua batida marcante, ritmada, pensada para dançar, o funk também possui letras bem características, muitas vezes falando sobre experiências juvenis da população das comunidades periféricas.

Temas sobre drogas, crime, violência, sexo são os mais abordados, tanto como exaltação quanto como denúncia do que se passa nas favelas. É reflexo da vivência na marginalização social, e pode se pensar que se originou primeiramente por uma falta de acesso dessas pessoas à sociedade que antes era representada na arte nacional em movimentos como a Música Popular Brasileira - MPB.

A cultura é dinâmica e se reconstruiu a partir da vivência dessas pessoas. A boemia, o romance e as questões da Música Popular Brasileira do século XX deram lugar ao carnal, ao terreno, e às relações líquidas, nunca antes representadas de forma tão ampla e crua num movimento artístico. Tal é a peculiaridade do funk brasileiro. Ainda assim, muitas pessoas criticam essa manifestação [10], e não compreendem seu valor como retrato social e como música pensada e com seu lugar de direito.

2.2 Formatos de áudio

Fisicamente, som é resultado de compressão e descompressão de moléculas (de ar geralmente) percebido pela nossa audição. Para tal fenômeno ser compreendido por um computador é necessário a conversão do sinal analógico (físico e contínuo) para o digital, e isso implica uma discretização do mesmo. Assim, áudio em seu formato cru ("raw audio") é uma série temporal de valores discretos de amplitude capturados com uma frequência de amostragem específica.

Atualmente, a representação padrão se chama "Pulse-code modulation"(PCM), e utiliza uma frequência de amostragem com intervalos de tempo regulares, geralmente 44,1 kHz, e aproxima os valores do sinal analógico para o nível discreto permitido mais próximo. A discretização em níveis de amplitude pode ser feita com 8, 16 ou 24 bits totalizando respectivamente 256, 65536 ou 16777216 valores possíveis. Como em todo processo de digitalização, há perda de informação nesse processo, porém tal perda torna-se facilmente imperceptível ao ser humano com uma boa escolha da frequência e níveis de amplitude.

Este formato de áudio cru é o formato que contém mais riqueza de informação, pois toda a onda sonora está codificada com suas nuances e detalhes. Isso não necessariamente facilita o trabalho com o formato no contexto de aprendizado de máquina, uma vez que tamanha riqueza implica um elevado custo computacional. Assim, existem outros tipos de arquivos de áudio relevantes para a área (e para produção musical), como é o caso do MIDI, uma representação de elementos musicais de alto nível. Neste projeto, o foco se encontra no formato de áudio cru, pois é o formato mais acessível para qualquer usuário

(pode-se encontrar facilmente áudios em mp3, m4a ou wav em qualquer lugar da internet). A seção a seguir disserta mais a respeito do que isso implica para o projeto.

2.3 Deep Learning e o desafio de trabalhar com áudio cru

Cada formato de áudio ou representação escolhida tem vantagens e desvantagens no processo de aprendizado e geração de música por modelos de deep learning [11]. O formato cru foi escolhido para este projeto, pois a melhor forma de compor uma base de dados com músicas de funk é utilizar arquivos disponíveis na internet que possuem justamente esse formato. Além disso, estudar as possibilidades e dificuldades de se extrair informação desse tipo de dado em tempo computacionalmente aceitável é escopo deste projeto.

A riqueza de informação presente em tal formato implica que um aprendizado bem realizado absorveria conceitos e detalhes da música sem perder grande parte dela. Contudo, não necessariamente a quantidade de informação em uma amostra se traduz em bom aprendizado, uma vez que muito detalhe e ruído podem atrapalhar os algoritmos de aprendizado, que falham em entender[1] a estrutura geral e podem se apegar a minúcias que nada tem a ver com o que é o gênero que se estava aprendendo.

Computacionalmente, utilizar áudio cru é extremamente custoso. Por exemplo, se utilizamos 5 segundos de música amostrado à 44,1kHz e codificado como 16-bit, teremos 220500 pontos que podem assumir 1 dentre 65536 valores. Tal estrutura é custosa e indicaria que para gerar 1 segundo de música precisaríamos executar o processo de geração 44100 vezes, o que pode ser extraordinariamente lento dependendo do hardware disponível para tal processamento.

Assim, trabalhar com áudio cru requer técnicas para reduzir o tamanho dos dados de entrada. Reduzir a frequência de amostragem para diminuir a quantidade de pontos por segundo é uma boa técnica, mas deve ser feita com cautela. O teorema de amostragem de Nyquist-Shannon diz que a frequência de amostragem deve ser no mínimo o dobro da maior frequência sonora da amostra para não haver perda significativa de informação ou a criação de artefatos indesejados (44,1kHz geralmente é escolhido justamente pelo limiar da audição humana estar próximo de 20kHz), isso implica que reduzir a frequência de amostragem pode levar à perda de informação de sons que contém componentes mais

[1] "Entender"é utilizado para falar de forma sucinta e de fácil compreensão sobre o processo implícito de extração de descritores, atribuição de pesos, e decisão de classes realizado por arquiteturas de machine learning.

agudos e criar ruído através do trecho alterado.

Uma técnica usada frequentemente, empregada inclusive pelo time de aprendizado de máquina da Google DeepMind [3], chama-se codificação μ-law e é uma transformação que visa diminuir a quantidade de níveis sonoros de uma amostra baseado na percepção auditiva humana. Assim, um áudio de 16 bits, por exemplo, pode ganhar uma representação de 8 bits com um mínimo de perda de informação. E se pensarmos que esses níveis são categorias independentes, pois não é um espectro contínuo, com tal transformação reduzimos um problema de classificação de 65536 classes distintas para apenas 256.

Após o bom pré-processamento dos dados ainda se faz necessário técnicas engenhosas para a alimentação dos modelos - como o uso de geradores a serem executados a cada etapa do treinamento ao invés de armazenar todos os dados de todas as etapas na memória RAM - e também é de crucial importância a boa escolha da arquitetura, pois a música tem componentes locais, como timbre, e componentes globais, como ritmo, e escolher camadas para os modelos que capturem ambos os contextos é algo que tem sido estudado e investigado. Nas seções a seguir, são mostradas técnicas e operações matemáticas utilizadas para capturar esses contextos, além de um pouco da evolução das redes neurais artificias para geração e processamento de áudio.

2.3.1 O que é uma rede neural?

A área da inteligência artificial é muito vasta. Tendo grande relevância em 1950 com Alan Turing e o teste de Turing [12], atualmente compreende diversas técnicas, métodos e aplicações. Dentre eles, a subárea do Machine Learning contém algoritmos bem fundamentados matematicamente que parecem simular certa inteligência quando são otimizados para dada tarefa - por exemplo, a classificação de amostras em um ou mais grupos. Contudo, essa área também contém uma das ideias mais não matematicamente fundamentadas: a rede neural [12].

Uma rede neural é formada por um conjunto de unidades chamadas neurônios. Estes são compostos basicamente por um conjunto de entrada, uma saída, e uma função de ativação - responsável por "interpretar" o valor obtido pelo processamento das entradas e produzir uma saída. Um grande conjunto desses em camadas pode ser otimizado através de várias iterações que buscam minimizar o erro encontrado ao se comparar os resultados obtidos pela rede com a realidade. Percebe-se rapidamente que não há explicitamente uma garantia matemática de que algo será aprendido [12].

Também percebeu-se que, aproximadamente, quanto mais neurônios e mais dados (existem patamares para ambos os casos) eram adicionados, melhor o modelo aprendia a realizar sua tarefa. Redes com muitas camadas, que puderam existir graças aos avanços computacionais vistos a partir dos anos 2000, logo criaram uma nova subárea - o Deep Learning. E são essas redes atualmente que parecem conseguir captar uma faceta da cognição e da experiência humana e produzir resultados que se misturam com a própria subjetividade humana da área de aplicação.

2.3.2 RNNs e CNNs

Dois tipos de redes neurais cujos conhecimentos são essenciais para o processamento e geração de música são a rede neural convolucional (CNN) e a rede neural auto-regressiva (RNN).

Redes convolucionais são conhecidas pelas suas aplicações em imagens e outros conteúdos que contém grande quantidade de informação. Não seria fácil para uma rede de neurônios interpretar uma entrada com 1 milhão de pixeis, a camada que contém a convolução é justamente utilizada para reduzir a quantidade de informação bruta e adquirir novos descritores da entrada com intuito de melhor compreender os arredores de um ponto ao invés de apenas a informação contida nele próprio. Nesse caso bidimensional da imagem, um exemplo de convolução pode ser imaginado como um filtro móvel de tamanho $n \times n$ (tal área de ação é chamada de campo receptivo da convolução) que contém um modo de calcular um novo valor baseado nos $n \times n$ pixeis abaixo do filtro. Assim, a convolução é uma operação essencial para se extrair e fornecer noções a respeito dos arredores de um dado para uma rede neural. No caso da música, ainda que seja unidimensional, podemos facilmente representá-la de forma bidimensional, o que permite a aplicação de uma camada de convolução. Isso é importante, pois um único valor de amplitude isolado não é informação suficiente para que um modelo possa compreender os conceitos que estão presentes nesse tipo de arte. Mais sobre essa transformação 1D para 2D será descrito na seção sobre a rede Wavenet.

A outra rede neural indispensável para se criar arquiteturas para geração de música é a auto-regressiva. Esse tipo de rede é caracterizado pela ciclagem de inputs, isso quer dizer que após uma primeira passagem da entrada pelos neurônios da rede, ele será realimentado para ser novamente considerado na regulação do modelo. Com isso, temos informação da entrada anterior assim como a presente. Para dados sequenciais (ou temporais), em que a ordem de apresentação dos dados importa (como na música), esse tipo de retroalimentação

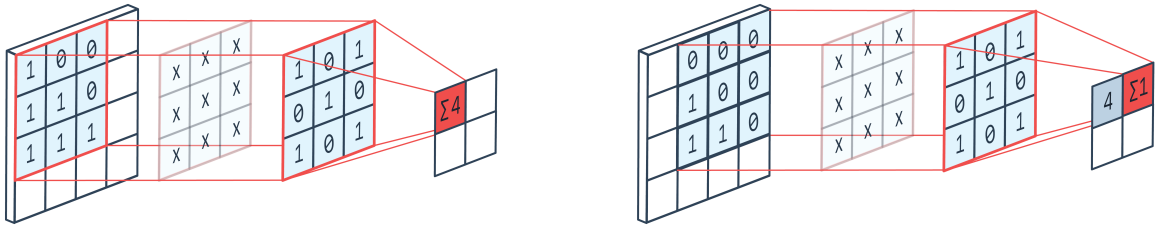

Figura 1: Ilustração de uma convolução 2D [1]: uma amostra passa por um filtro e resulta em um novo conjunto de dados reduzido e mais acessível a um eventual modelo.

é essencial. É necessário considerar o que veio antes para classificar o seguinte. O processo de geração também se baseia nessa ideia.

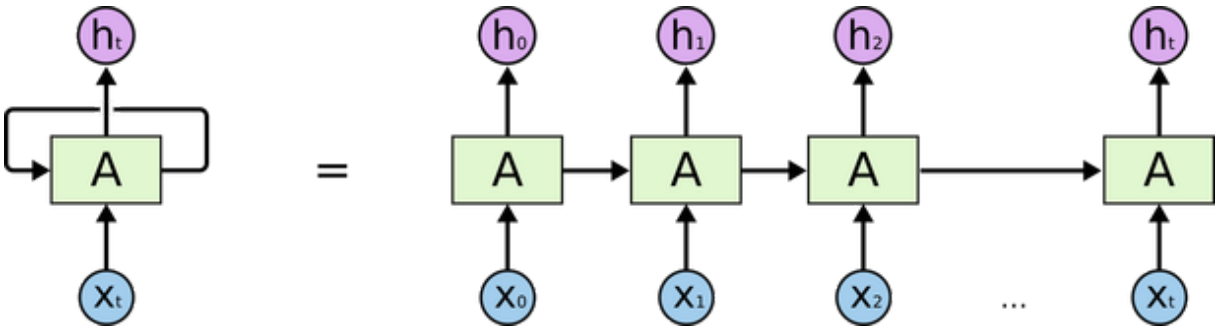

Figura 2: Ilustração de uma arquitetura RNN [2]: as saídas obtidas no passo anterior servem como entrada para a próxima etapa em conjunto com uma nova.

Antigamente era muito comum RNNs encontrarem momentos onde o gradiente de descida se tornava nulo ou extremamente grande para sequências longas devido à problemas na propagação do erro das saídas através dos modelos [13]. Uma melhoria para lidar com esse problema ocorreu com o surgimento da engenhosa camada LSTM (Long-short term memory) que pode ser vista como uma subunidade utilizada dentro de redes auto-regressivas mais atuais e responsável por controlar a entrada de novos inputs e o esquecimento de velhos mais sofisticadamente de forma a maximizar o aprendizado e aumentar a capacidade de memória do modelo. A ideia é justamente treinar uma parte do modelo para ser responsável por selecionar o que pode ser esquecido e o que deve ser propagado de forma a controlar o erro propagado sem prejudicar o aprendizado. Anteriormente, o erro era propagado sem processamento, e isso causava o problema mencionado no início do parágrafo.

Tais melhorias realizadas em modelos anteriores são base para evolução na área de deep learning, caracterizando um processo de desenvolvimento iterativo e colaborativo dentro da comunidade envolvida. É muito comum visualizar os blocos ou redes como caixas-pretas por causa da grande complexidade envolvida na soma de todos os conceitos

que ali interagem. O resto desta seção é um bom exemplo do trabalho conjunto que leva à evolução dessa área assim como certa investigação dos mecanismos que possibilitam os resultados gerados no estado da arte da geração de áudio por inteligência artificial.

2.3.3 WaveNet

Sem dúvidas o artigo mais impactante dos últimos 10 anos no contexto de geração de música (e outros tipos de áudio como fala a partir de texto, que foram até mais destacados no próprio artigo) a partir de áudio cru é [3] da equipe DeepMind da Google. A arquitetura proposta Wavenet hoje serve de base para os melhores modelos criados e juntou diversas técnicas de CNNs e RNNs para a facilitação da difícil tarefa que é trabalhar com áudio cru. O principal modelo que inspirou a arquitetura da Wavenet foi o pixelCNN [14], um modelo causal focado para a geração de imagens, sendo a principal e crucial contribuição do artigo [3] a utilização da convolução dilatada.

Antes de entrar nesse tipo diferente de convolução, Wavenet recebe como input o áudio cru, 16kHz de frequência de amostragem, e produz como output um outro vetor de mesmo tamanho. Os valores x_t de tal vetor são baseados na probabilidade $p(x_t) = p(x_t|X_{i<t})$, sendo X_i os valores obtidos no passo anterior ou dados diretamente pelo input, isso significa que todo valor x_t é estimado com base em todos os valores que são anteriores ao instante t. Tal restrição modela a causalidade necessária para a geração sonora, ou seja, a rede neural não possui informação sobre o tempo futuro para predizer qual o valor de x no tempo t pressente. Outro fator de extrema importância para a fase de treinamento é o fato de que todo x_t de uma camada do modelo pode ser calculado paralelamente, pois a previsão de x_t depende apenas de $X_{i<t}$ realizado no passo anterior, não depende de x_{t-1} como é o caso da maioria das redes auto-regressivas. Obviamente na fase de geração essa vantagem não existe, pois há dependência direta entre as previsões.

Sobre a convolução dilatada, ela é essencial para a extração de informações do contexto local e global da música. Novamente, a música possui características em níveis de resolução distintos, mas simplesmente utilizar uma convolução com campo receptivo grande o suficiente para adquirirmos informações globais seria computacionalmente inviável, teríamos muitos cálculos para cada passo. Assim, a convolução dilatada é uma convolução com espaçamentos (um vetor com vários zeros) dentro da sua área de ação, permitindo ao modelo possuir um grande campo receptivo sem aumentar exponencialmente seu custo de processamento. Juntando diferentes camadas idênticas, cada uma com várias convoluções dilatadas com tamanhos de espaçamento distintos, e que recebem

informação da camada interior, chega-se a um modelo preparado para analisar e conhecer áudio, e mais especificamente música, em várias escalas. Tal modelo é capaz de entender[1] as peculiaridades da sucessão direta entre dois valores na série temporal (que é o áudio) quando utiliza um campo receptivo pouco espaçado até noções de ritmo quando utiliza um campo muito espaçado.

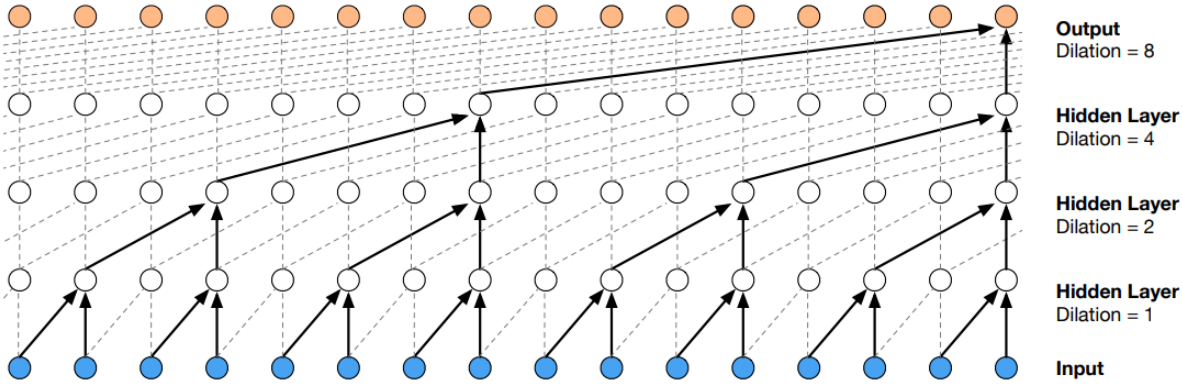

Figura 3: Ilustração de um agrupamento de convoluções dilatadas [3], elemento principal da rede Wavenet. Todas as camadas mostradas contribuem com dados, e se diferenciam por diferentes espaçamentos entre seus filtros, ainda que tenham o mesmo tamanho.

Além disso, para a confecção da saída de fato é usada uma técnica chamada de pulos de conexão (Skip-connections), que transportam saídas de camada intermediárias diretamente para a camada final do modelo ao invés de realizar todos os ciclos auto-regressivos. Isso acelera a convergência e provê mais informação para a síntese da previsão do modelo.

Vale também dizer que inicialmente os sinais de áudio eram modelizados como variáveis contínuas, pois fora do mundo digital eles são de fato contínuos. Entretanto, foi mostrado que a pressuposição de distribuições de probabilidade ao se trabalhar com uma variável contínua era danosa aos resultados, e uma modelização que nada pressupõe sobre a forma da distribuição dos dados, no caso a categórica através de uma distribuição softmax, era mais flexível e condizente para a tarefa de áudio.

Tem aqui uma reflexão. O fato do áudio ser implicitamente continuo não é utilizado em nenhum momento nem mesmo nas implementações derivadas da Wavenet. Não é estranho pensar que tal noção possa ser útil em algum contexto, por exemplo na criação de uma métrica para avaliação de um modelo em treinamento, uma vez que um elemento de uma música codificada em 64 níveis $X_k = 48$ sendo previsto pelo modelo como $x_k = 50$

[1] "Entender"é utilizado para falar de forma sucinta e de fácil compreensão sobre o processo implícito de extração de descritores, atribuição de pesos, e decisão de classes realizado por arquiteturas de machine learning.

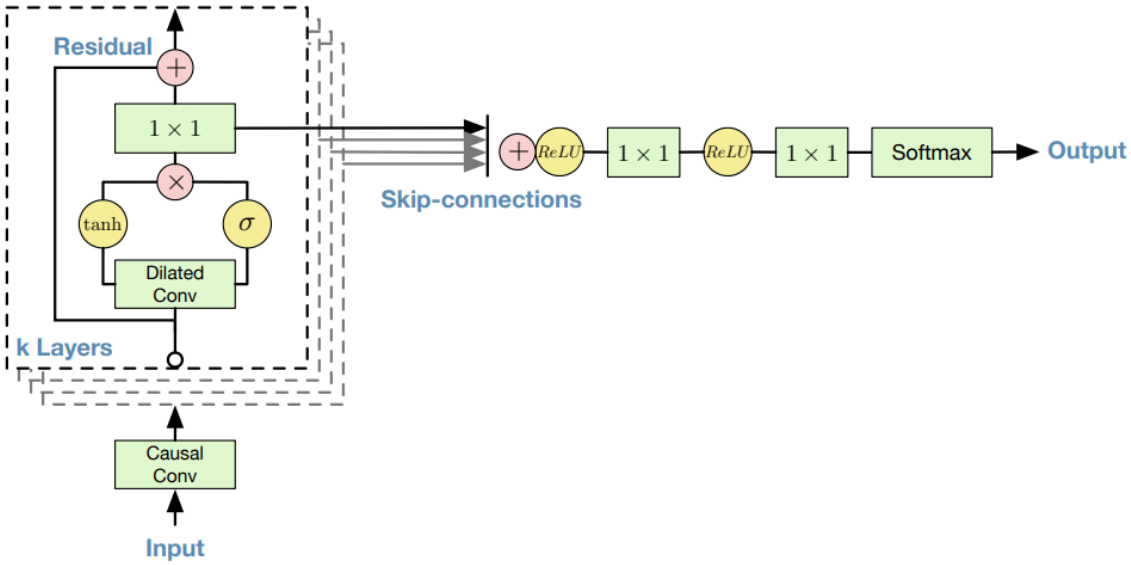

Figura 4: Ilustração da arquitetura Wavenet. É possível visualizar os pulos de conexão para a saída. Informações sobre os demais símbolos utilizados para a produção da imagem podem ser encontradas no artigo original. [3]

deveria ser menos punido do que uma previsão $x_k = 3$, e até onde avançou a pesquisa realizada neste projeto, não foi dada a devida e plausível atenção a esse fato.

2.3.4 SaShiMi

Um dos maiores problemas da Wavenet é seu grande número de parâmetros, que aumenta seu tempo de treinamento, e seu campo receptivo que mesmo largo falha para sequências mais longas que 4000 elementos. Felizmente, outros estudos realizados, inspirando-se nas ideias da rede Wavenet e em outras arquiteturas, conseguiram solucionar muitos de seus problemas aqui mencionados e avançaram consideravelmente o estado da arte no contexto de geração de música.

Um dos últimos trabalhos na área, publicado no início de 2022 por Karan Goel et Al. [4], propõe uma arquitetura chamada SaShiMi que utiliza as convoluções paralelas como base, mas engenhosamente melhora a velocidade de treinamento e a geração de amostras, além de expandir a capacidade de aprendizado para muito além de 4000 elementos - chega-se a 128000 com esse novo modelo. Tal arquitetura foi o principal modelo a ser utilizado neste projeto e, portanto, merece o detalhamento de suas ideias e da forma como o problema de aprender longas sequências foi matematicamente modelizado para a concretização desta rede neural.

2.3.4.1 Longas sequências

Modelizar longas sequências e obter resultados práticos a partir do estudo das mesmas sempre foi um desafio para as redes neurais. Recentemente foi criado um conjunto de testes de referência para avaliar a capacidade de um modelo de processar e solucionar diversos problemas envolvendo tipos volumosos de dados. Desde então Long Range Arena (LRA) [15] tem sido usado para testar novas ideias e modelizações para este tipo de problema.

Em 2021, Albert Gu, Karan Goel et Al. publicaram um estudo [16] que propõe uma modelização de longas sequências baseada no modelo fundamental do espaço de estados (SSM), uma técnica bem conhecida nas áreas de controle. Eles obtiveram resultados sem precedentes nos testes LRA. O melhor exemplo desse sucesso pôde ser verificado com o teste chamado Pathfinder-X, que se trata de um teste visual em que o modelo recebe uma imagem contendo dois pontos e caminhos tracejados e tem como objetivo dizer se os dois pontos são ou não conectados pelos caminhos mostrados. A versão simplificada (Pathfinder) contém imagens de 32×32 pixeis e foi solucionada por vários modelos, porém a versão X mencionada, que contém imagens de 128×128 - 16000 pixeis, não havia sido satisfatoriamente solucionada por nenhum modelo anteriormente publicado e ainda assim essa modelização obteve mais de 95% de acurácia no teste [16].

Futuramente, os mesmos pesquisadores utilizaram a parametrização do espaço de estados proposta, nomeada S4 - Structured State Space, como bloco fundamental da arquitetura SaShiMi. Nas seções seguintes é discutido um pouco mais a fundo a motivação e detalhamento dessa técnica.

2.3.4.2 Modelizando com SSM

Em teoria, as sequências poderiam ser modelizadas através da equação:

$$\begin{cases} x'(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

Em que $y(t)$ é a saída, $u(t)$ é a entrada, e $x(t)$ é um estado intermediário. É necessário, porém, realizar a discretização dessa modelização para podermos trabalhar com áudio digital (uma série temporal discreta). Para isso introduziremos um passo Δ, o que transforma a entrada contínua $u(t)$ na entrada discreta $u_k = u(k\Delta)$. Também, por mo-

tivos de simplicidade, assume-se $D = 0$, e por fim realiza-se aproximações das matrizes envolvidas através do método bilinear [17], A é aproximada por $\bar{A}$, e as outras matrizes são análogas. Logo, a modelização completa discreta é:

$$\begin{cases} x_k = \bar{A}x_{k-1} + \bar{B}u_k & \bar{A} = (I - \frac{\Delta}{2} \cdot A)^{-1}(I + \frac{\Delta}{2} \cdot A) \\ y_k = \bar{C}x_k & \bar{B} = (I - \frac{\Delta}{2} \cdot A)^{-1}\Delta B \quad \bar{C} = C \end{cases}$$

Desenvolvendo a equação de y_k para $x_{-1} = 0$, chega-se a:

$$y_k = \bar{C}\bar{A}^k\bar{B}u_0 + \bar{C}\bar{A}^{k-1}\bar{B}u_1 + ... + \bar{C}\bar{A}\bar{B}u_1 + \bar{C}\bar{B}u_k$$

Tal expressão pode ser escrita como uma convolução com kernel $\bar{K}$:

$$y = \bar{K} * u$$

Computar $\bar{K}$ não é trivial, e contribuições realizadas pela modelização em espaços estruturados são comentadas na próxima seção. Contudo, ainda aqui é preciso notar a existência do termo $\bar{A}^k$ nas equações acima. Pode-se cair facilmente em um problema de desaparecimento ou explosão de gradientes ao se multiplicar várias vezes essa matriz (bem similar ao problema enfrentado pelas RNN, que foi solucionado com a ideia LSTM). Assim, é necessário restringir a matriz A a uma classe de matrizes que respeite as restrições para ser considerada dentro da estrutura HiPPO (high-order polynomial projection operators) [18] de forma a evitar o problema causado pela potenciação de A. Tem-se que $A \in R^{n \times n}$ para uma matriz HiPPO é definida como:

$$A_{ij} = -\begin{cases} (2i+1)^{\frac{1}{2}}(2j+1)^{\frac{1}{2}} & i > j \\ i+1 & i = j \\ 0 & i < j \end{cases}$$

2.3.4.3 S4 - Structured State Space

Seria de grande utilidade se a matriz A fosse diagonal. Para se aplicar uma técnica de diagonalização, pode-se realizar a substituição $x = V\tilde{x}$ na modelização do SSM, escrita de forma simplificada:

$$\begin{cases} V\tilde{x}' = AV\tilde{x} + Bu \\ y = CV\tilde{x} \end{cases}$$

Isolando $\tilde{x}'$ na equação acima:

$$\begin{cases} \tilde{x}' = V^{-1}AV\tilde{x} + V^{-1}Bu \\ y = CV\tilde{x} \end{cases}$$

É possível que $V^{-1}AV$ seja diagonal, mas para tal usa-se o domínio dos complexos C e não mais dos reais R. Contudo, algumas matrizes de conjugação V para que a matriz HiPPO seja diagonalizável possuem entradas que crescem exponencialmente com o tamanho da matriz, o que as torna mal condicionadas (são matrizes com grande sensibilidade, em que uma pequena alteração na entrada causa grandes diferenças na saída, o que pode gerar problemas ao utilizar-se métodos que dependem dela, como o cálculo de sua inversa) e, portanto, difíceis de computar em vários contextos.

Idealmente, A é diagonalizável por uma matriz $V \in C^{N\times N}$ perfeitamente condicionada, o que implica $A = VDV^*$, sendo D uma matriz diagonal. Isso implica que idealmente A é uma matriz normal, porém a definição de matriz normal, que é respeitar $M^*M = MM^*$ não contém a definição HiPPO declarada na seção anterior. Ainda assim, a matriz HiPPO pode ser decomposta em uma soma de uma matriz normal mais uma matriz low-rank (que é uma matriz que contém muito menos informação independente do que suas dimensões permitiriam, podendo ter várias linhas nulas, por exemplo, o que a torna de fácil computação):

$$A = VDV^* - PQ^T$$

Sendo $P, Q \in R^{N\times 1}$. Essa transformação por si só ainda não é útil, pois fazer uma potência de uma soma ainda é muito custoso. Para tratar deste gargalo, foram propostas três novas técnicas:

- Ao invés de se computar $\bar{K}$ diretamente, é computado seu espectro avaliando sua função geradora em raízes específicas e depois aplicando uma transformada de Fourier inversa
- A técnica acima troca a potenciação por uma inversão de uma soma, que pode ser simplificada aplicando-se a identidade de Woodbury

- E por fim, pode-se mostrar que a representação atingida é equivalente à computação de um kernel de Cauchy

Detalhes das provas dessas técnicas e como elas interagem para criar o algoritmo que computa eficientemente $\bar{K}$ podem ser encontrados no artigo original [16]. O importante de se ressaltar é que a complexidade de se calcular $\bar{K}$ de uma forma direta, com dimensão intermediária N e comprimento de sequência L, requer $O(N^2L)$ operações e $O(NL)$ de memória para sua execução. Após todo o trabalho algébrico exposto aqui de forma simplista, tem-se $O(N + L)$ operações e $O(N + L)$ para memória, o que é uma melhora de eficiência no algoritmo gigante e explica um pouco como eles foram capazes de atingir resultados avançando o estado da arte.

2.3.4.4 De volta ao SaShiMi

O artigo da arquitetura SaShiMi [4] faz primeiramente uma contribuição teórica à modelização S4, ao invés da matriz A ser parametrizada como uma diagonal D mais uma fatoração low-rank PQ^*:

$$A = D + PQ^*$$

Ela é parametrizada como:

$$A = D - PP^*$$

Basicamente, restringe-se a fatoração. O problema que acontecia era que em alguns casos durante a geração autoregressiva do modelo a estabilidade das potências de A acabava se perdendo. Com essa nova fatoração, esse problema não acontece mais e nenhuma das propriedades necessárias mostradas na seção anterior foram afetadas.

Colocando a teoria matemática de lado, a arquitetura completa SaShiMi conta com blocos da modelização S4 com funções de ativação não lineares, etapas de normalização e conexões residuais. Ela utiliza também, diferentemente da Wavenet, camadas de Up e Down Pool que transformam as dimensões do input - diminuindo a frequência de amostragem antes dos blocos S4 e voltando ao tamanho original depois. Juntando os resultados retransformados para a dimensão correta de todas as camadas treinadas em inputs de dimensões diferentes tem-se a saída do SaShiMi - de mesma dimensão da entrada original. Na figura 5 pode-se ver o esquema da arquitetura.

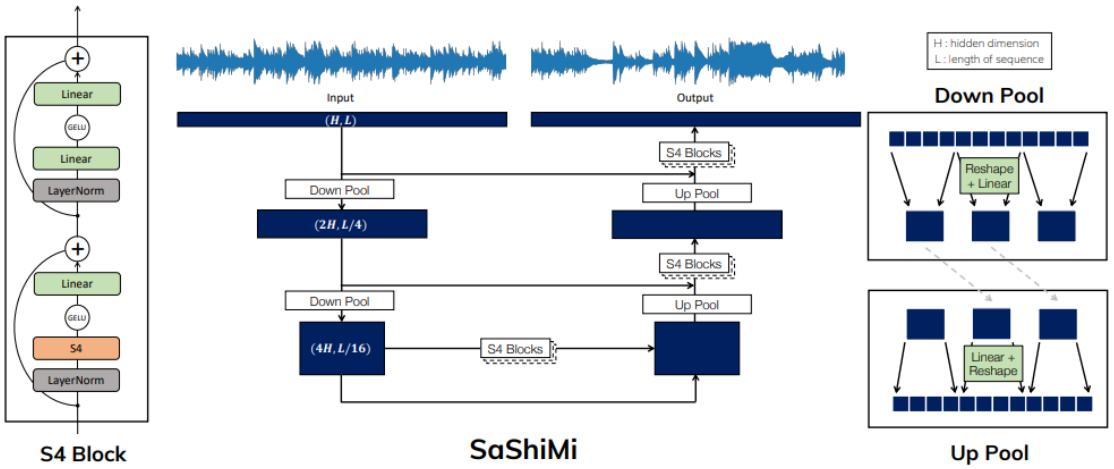

Figura 5: Ilustração da arquitetura SaShiMi [4]. À esquerda pode ser visto o detalhamento de um bloco S4, que possui camadas de normalização ("LayerNorm") e funções de ativação não lineares ("GELU") e lineares ("Linear"). No meio e à direita pode ser visto a propagação e transformação do sinal de entrada, de modo que o aprendizado da rede é realizado em entradas de diferentes dimensões, que são depois corrigidas para o tamanho original.

A integração de SaShiMi como componente para outras arquiteturas também foi testado. O principal exemplo realizado foi utilizando a rede DiffWave [19], que diferentemente das outras emprega uma técnica de geração que parte de um ruído como sinal de entrada e iterativamente reorganiza todo o áudio até um formato mais condizente com o aprendizado da rede, criando um modo muito interessante para geração não-condicional. Originalmente, o bloco convolucional incluso na arquitetura era uma versão da Wavenet, porém foi mostrado que substituindo-a pelo SaShiMi o modelo atingiu resultados ainda melhores [4].

É notável a quantidade de conhecimento que existe no desenvolvimento de uma rede neural que representa o estado da arte. Existem diversos níveis de resolução da arquitetura que podem ser compreendidos e que carregam um grande nível de complexidade. As seções de aspectos conceituais aqui mostradas servem para tentar promover uma clareza maior em relação às técnicas utilizadas na concepção de modelos, assim como certa apreciação pelo trabalho já realizado e muitas vezes negligenciado por projetos que utilizam o que foi produzido apenas como caixa-preta.

2.3.5 Jukebox

A rede chamada Jukebox [6], publicada em 2020 pelo grupo OpenAI, é uma arquitetura de rede neural profunda que foi treinada originalmente em uma enorme base de

dados com um grande poder de processamento, detalhes a respeito podem ser lidos na seção Análise dos Resultados. Ela foi utilizada nesse projeto como prova de conceito da capacidade geracional de áudio identificável como música, em especial funk, pelas redes atuais.

Esta seção explica um pouco dos conceitos teóricos envolvidos nessa rede.

2.3.5.1 Vector Quantisation - Variational Auto Encoder - VQ-VAEs

Um encoder é uma estrutura matemática capaz de realizar compressão e descompressão de um tipo de dado. Idealmente, as perdas nesse processo devem ser mínimas para um bom algoritmo de compressão. Um variational autoencoder (VAE) é uma técnica de machine learning que busca aprender uma dessas estruturas codificadoras para um tipo de dado específico.

Assim, uma entrada para esse tipo de rede é processada até atingir uma nova representação de dimensionalidade reduzida. E essa transformação é otimizada de forma a se minimizar o erro de reconstrução pela parte responsável pela decodificação dessa nova representação. Originalmente, essa representação reduzida é interpretada como parâmetros para modelização e construção de um espaço de probabilidades composto por diversas curvas (Gaussianas, por exemplo), e são a partir de amostras geradas com base nessas distribuições que o decodificador é treinado. Contudo, no caso da arquitetura Jukebox, foi optado por não utilizar a representação encontrada como parâmetros de distribuições probabilísticas, mas sim como vetores. Abaixo, uma imagem ilustrando as etapas envolvidas em um VAE:

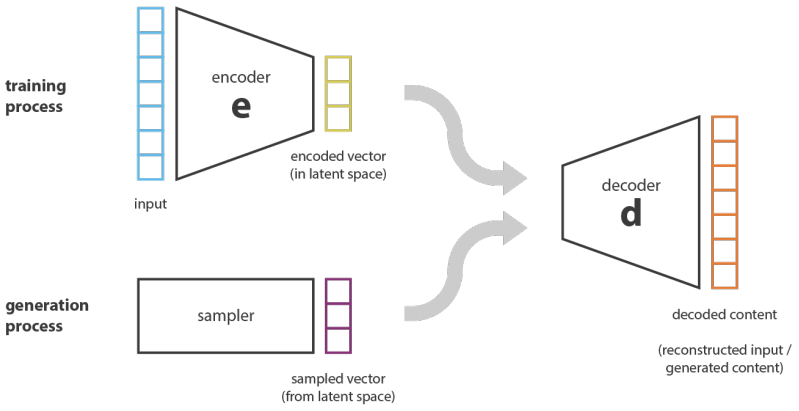

Figura 6: Ilustração de um variational autoencoder retirada do site [5]. pode-se ver as etapas de codificação (encoder), e decodificação (decoder). O sampler serve para gerar amostras a partir da representação encontrada para treinar o decoder.

Especificamente para a rede Jukebox foram treinados três autoencoders distintos para janelas de áudio de tamanhos também distinto. Tal técnica permite a compreensão de elementos da música perceptíveis em pequena, média e larga escala.

2.3.5.2 Transformers

Tendo a compressão para o espaço representativo encontrada e a descompressão bem treinados com um mínimo de perda, surge a necessidade de gerar uma amostra (uma sequência) a partir dos códigos comprimidos que resulte em música. É aqui que entram os Transformers, uma arquitetura proposta em 2017 pela equipe Google Brain [20].

Ela foi criada para processar sequências e se encontra no mesmo grupo que RNNs e LSTMs, porém, como novidade, possui mais elementos de paralelização, o que a torna mais vantajosa para tarefas com estruturas de dados grandes - como áudio cru, e possui em seu algoritmo a técnica conhecida como atenção, inspirada na atenção cognitiva dos seres humanos. Em machine learning, ela é a ideia de definir pesos variáveis e treináveis para os elementos da sua entrada, desse modo o modelo é capaz de aprender e atribuir mais peso para as partes que forem mais essenciais naquele contexto.

Na rede Jukebox, são treinados um Transformer para cada nível de codificação criado na etapa anterior. Os níveis com maior granularidade - mais próximos da frequência de amostragem do áudio original, consideram a geração proveniente dos níveis anteriores, que aprenderam conceitos de grande escala, por meio de redimensionamento de suas saídas através de “upsamplers”. Com isso, gera-se áudio de qualidade em que se pode verificar conceitos musicais de diversas escalas. Abaixo uma imagem que ilustra esse processo de geração:

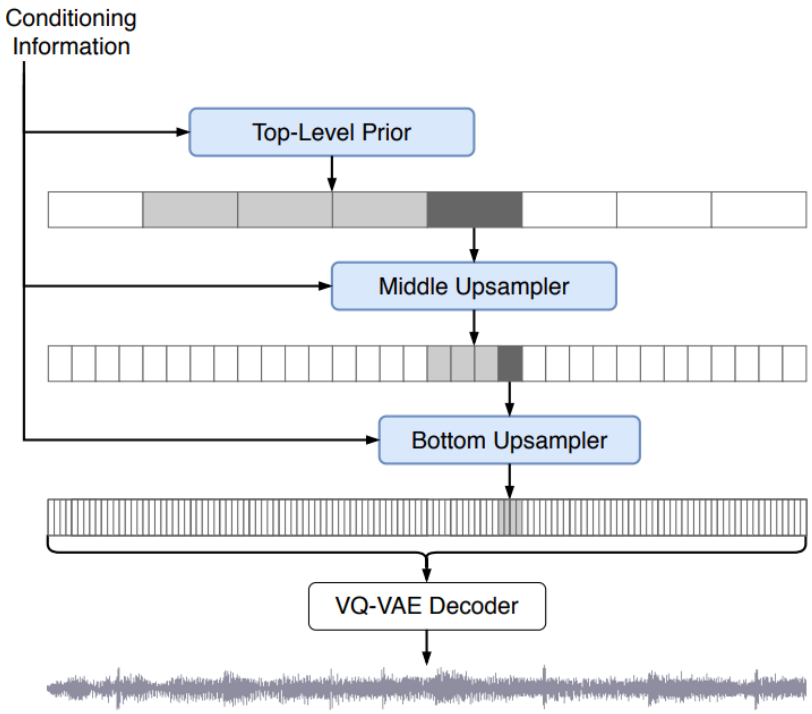

Figura 7: Ilustração retirada do artigo original [6]. Podemos ver as diversas escalas em que as gerações são realizadas. Apenas a última é de fato decodificada para o formato de áudio. Conditioning information nesse caso pode ser extratos iniciais de uma música para server de início para a geração, um gênero, um artista, ou até informações sobre a letra da música.

3 TECNOLOGIAS UTILIZADAS

O ambiente de desenvolvimento proporcionado pela Google: Google Colab foi escolhido para o projeto, pois realizar o treinamento de uma rede neural e posterior geração de áudio exige um alto desempenho computacional que não é facilmente contornável. Assim, a disponibilização remota de GPUs de alto desempenho promovida pela plataforma é essencial para todas as fases do projeto. Com o tempo de treinamento elevado da rede SaShiMi foi necessário contratar a versão paga desse serviço [1]. Com ela, foi possível realizar treinamentos mais longos e com GPUs mais rápidas.

A linguagem utilizada é Python, pois é a ferramenta com maior suporte e diversidade para trabalhar com Deep Learning. O formato dos áudios presentes na base de dados é '.mp3', e eles foram obtidos manualmente com o auxílio do programa "aTube Catcher"que permite realizar o download de vídeos do Youtube e salvar facilmente seu áudio.

Os códigos usados para definição das arquiteturas, treinamento das redes e geração de áudio foram adaptados a partir dos repositórios Github:

- https://github.com/HazyResearch/state-spaces - para a rede SaShiMi.
- https://github.com/openai/jukebox - para a rede Jukebox

Várias bibliotecas Python foram utilizadas, dentre elas destacam-se:

- *librosa*: permite a leitura e análise de amostras de áudio, extraindo diversos descritores e fornecendo ferramentas para a exploração dos sinais.
- *pyTorch*: é a biblioteca mais utilizada atualmente para Deep Learning: define as arquiteturas, cria o contexto de treinamento e o executa
- *scikit-learn*: é uma biblioteca muito utilizada no Machine Learning clássico. Ela possui diversos métodos e algoritmos para classificação, regressão, agrupamento,

[1] O custo de serviço do ambiente de desenvolvimento virtual Colab Pro+ foi de R$258,00 ao mês

etc. Aqui, ela foi usada para realizar uma análise não supervisionada dos dados de forma a ajudar na filtragem da base de dados

- *numpy*: é uma biblioteca que define e auxilia operações matemáticas mais complexas do que as tradicionais encontradas na linguagem
- *seaborn, plotly*: servem para produzir gráficos rapidamente e que ainda sejam esteticamente agradáveis

Além disso, para divulgação dos resultados, foi escolhida a plataforma "Streamlit", que contém um módulo de desenvolvimento para Python. Ela possui todas as funcionalidades necessárias para a disponibilização de texto, imagens e áudio do projeto, e apresenta relativa simplicidade dentro do contexto de criação de um website. O link para este projeto é: https://renatoaab-tccsite-app-5v0miw.streamlit.app/.

4 METODOLOGIA DE TRABALHO

Este estudo foi desenvolvido a partir de um processo iterativo. A pesquisa em inteligência artificial se inicia pela enunciação do problema, neste caso, treinar um modelo que seja capaz de aprender e gerar novas amostras de funk brasileiro. Com isso em mente, é preciso escolher um modelo que possa ser capaz de cumprir a tarefa, aqui o principal é a arquitetura SaShiMi, detalhada na seção anterior. E também deve-se compor uma base de dados que seja relevante e representativa do gênero de modo a fornecer boas amostras ao modelo. A iteração presente nesse estudo se encontra no fluxo de projeto: compor uma base de dados, treinar o modelo, avaliá-lo e gerar novos conhecimentos para compor uma base de dados melhor ou definir configurações mais adequadas para o modelo, e reiniciar o processo. Com o passar das mesmas, espera-se produzir resultados melhores ou, no mínimo, entender mais sobre o problema e as dificuldades de executá-lo.

4.1 Composição da base de dados

A primeira base de dados criada foi feita a partir do download das 100 músicas de funk brasileiro mais tocadas de 2021 disponibilizadas por um site focado nesse tipo de serviço.

4.1.1 Pré-processamento

Não adianta utilizar os arquivos brutos e esperar qualquer aprendizado de qualquer rede. É necessário realizar pré-processamentos nos dados para adequá-los ao modelo utilizado.

No artigo da rede SaShiMi [4], o teste realizado com a base de dados chamada YouTubeMix trabalhou com um áudio extraído de um vídeo de 4 horas do YouTube chamado "4 HOURS Peaceful Relaxing Instrumental Piano Music-Long Playlist". Esse áudio foi separado em partes de 8 segundos com 16kHz de frequência de amostragem, o que totali-

zou 1800 séries temporais com 131072 elementos. Além disso, também foi realizada uma transformação dos níveis sonoros de modo a diminuir sua complexidade.

4.1.1.1 Transformação de níveis sonoros

Áudios geralmente são codificados e salvos com uma configuração de 16 bits, o que implica 65536 níveis sonoros distintos. No contexto de machine learning, em que o modelo tenta aprender a prever qual seria o próximo nível sonoro baseado em amostras passadas, prever qual das 65536 classes seria a mais adequada é uma tarefa extremamente difícil e avaliá-la seria muito delicado. Assim, reduzir a quantidade de níveis das amostras é etapa essencial do pré-processamento dos dados a serem utilizados no treinamento do modelo.

Os dois métodos principais é uma transformação linear e uma orientada pela seguinte função (μ-law) em que μ é relacionado à quantidade de níveis de codificação:

$$f(x) = sign(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)}$$

Visualmente, a linear não troca a distribuição original, mas a μ-law agrupa vários níveis sonoros extremos para um único conjunto e foca em separar melhor os próximos de zero, deste modo argumenta-se que existe menos perda de informação para um ouvinte humano:

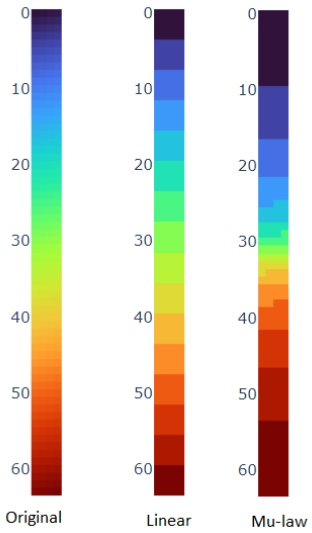

Figura 8: A ilustração contém um vetor "Original", à esquerda, composto pelos números de 0 a 255, representados de acordo com uma escala de cor, e agrupados em 4 colunas, de forma a criar uma matriz 64×4. No centro, há a transformação linear de 256 níveis para 16, em que não há mudança no agrupamento de valores. À direita, pode-se ver a transformação μ-law de 256 para 16, que altera as distribuições resultantes. Feito por plotly.

Para a base de dados YouTubeMix foi utilizada a transformação μ-law com 256 níveis sonoros. Contudo, uma análise qualitativa e comparativa informal do quanto poderiam ser reduzidos os níveis sonoros sem prejudicar a clareza e musicalidade do som mostrou que 64 níveis sonoros eram suficientes para os funks aqui mencionados. Amostras com diferentes quantidades de níveis sonoros podem ser escutadas no site de divulgação do projeto: https://renatoaab-tccsite-app-5v0miw.streamlit.app/.

4.1.1.2 Redução da frequência de amostragem

A primeira abordagem foi orientar a nova base de dados composta pelos 100 funks mais ouvidos de 2021 de acordo com as configurações da base de dados YouTubeMix. Contudo, ao realizar a divisão das músicas em trechos de 8 segundos (sem superposição de janelas) utilizando a frequência de amostragem original, verificou-se que ocorria um erro no ambiente de execução proporcionado pela Google: a GPU disponível para treinamento não conseguia suprir os requisitos pedidos pela biblioteca PyTorch para a execução do treinamento.

Foi identificado que o problema era o grande comprimento das amostras alimentadas ao modelo. Elas deveriam ser reduzidas. O problema é que a frequência de amostragem está diretamente ligada com a boa qualidade sonora do áudio. A maioria dos arquivos possui 44,1kHz de frequência amostral, pois, de acordo com a lei de Nyquist, esse valor seria suficiente para captar frequências de até 22,05kHz - praticamente o limite superior da audição humana. Isso quer dizer que quanto mais se reduz a frequência de amostragem, mais informação sobre frequências agudas é perdida, mas elas ainda são processadas, o que cria artefatos e ruídos que prejudicam a qualidade do som.

Foi realizado um teste qualitativo para identificar a menor frequência amostral para a qual o som ainda mantinha clareza e musicalidade, ao mesmo tempo que resolvia o problema de falta de memória na GPU. Para uma sequência de 8 segundos, a frequência que resolvia o problema de memória causava um dano considerável na qualidade do som. Assim, foram criados trechos de 4 segundos com frequência amostral de 7kHz, atendendo tanto aos requisitos de memória quanto a uma qualidade sonora razoável. Amostras exemplificando a redução da frequência de amostragem podem ser ouvidas em: https://renatoaab-tccsite-app-5v0miw.streamlit.app/.

4.1.1.3 Separação de acompanhamento e vocal

Logo no início, foi pensado que utilizar um áudio de funk que contivesse tanto vocal quanto acompanhamento poderia causar problemas para o aprendizado da rede, devido à grande diferença encontrada entre a parte cantada e a parte que caracteriza a batida do funk. Ainda que a forma de cantar funk tenha tendências, atualmente isso se diversificou bastante: o funk pop nada tem a ver no estilo de cantar com o funk ostentação por exemplo. O mais característico do funk em relação ao vocal são de fato as letras, e como a semântica e conteúdo das mesmas dificilmente seria aprendida por uma rede focada em áudio (esse trabalho seria mais para uma rede de NLP - Natural Language Processing), foi escolhido retirar a parte do vocal e focar apenas na geração de uma batida/acompanhamento de funk, que é parte também característica desse gênero musical e essa sim poderia ser compreendida pela arquitetura proposta.

Realizar a separação automática de acompanhamento e vocal não é uma tarefa trivial, o sinal de áudio contém informações fortemente misturadas sobre seus componentes. Isolar esses componentes é outro foco de estudo de deep learning aplicado a música. Para simplicidade, foi utilizada uma rede neural já treinada para realizar essa tarefa: Spleeter [21], disponibilizada pela empresa de streaming de áudio Deezer e acessível através de uma API para Python. Utilizando esta rede, foi possível realizar uma separação do áudio em acompanhamento e vocal.

Uma análise qualitativa dos resultados indica que alguns trechos foram bem separados, mas outros nem tanto. O método não é perfeito, pois é uma área de estudo em andamento, e para piorar: Spleeter foi treinado com vários gêneros de música presentes no servidor da Deezer [22] e provavelmente com grandes tendências ao inglês, e no funk brasileiro não é incomum identificar um elemento rítmico (como uma batida) sendo realizado pelo vocal, por exemplo o verso e título da música “Bum bum tam tam”, o que explicaria uma maior dificuldade de separação em alguns casos.

4.1.2 Filtragem e análise não supervisionada

Outro processo envolvido na criação de bases de dados é a identificação e remoção de amostras muito diferentes do resto da base. Para mais de 3000 trechos, que é próximo à quantidade utilizada por todas as bases criadas, é inviável realizar essa separação manualmente escutando todas as amostras. Para isso, é preciso utilizar técnicas de extração de descritores (features) dos áudios para automatizar pelo menos parte do processo.

A primeira propriedade calculada e analisada nesta fase foi uma estimativa do tempo da música - batidas por minuto. Essa é uma das características mais presentes em funks, é bem comum que eles sejam classificados com base nesse conceito. Por exemplo, funks com 150 bpm (batidas por minuto) ficaram mais famosos em uma fase mais recente do gênero, após 2015.

Para calcular essa medida foi usada a biblioteca "librosa". A função "onset_strength" calcula o envelope correspondente à intensidade sonora ao longo da duração do áudio. Abaixo uma ilustração:

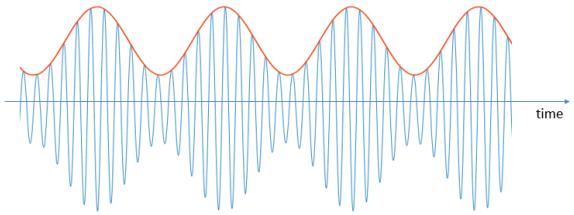

Figura 9: Em vermelho o envelope correspondente à onda mostrada em azul.

E se utilizando dessa informação como entrada, a função "librosa.beat.tempo" consegue estimar a quantidade de batidas por minuto de um trecho. Abaixo, um gráfico mostrando a distribuição de tempo encontrados na base de dados composta apenas pelo subgênero mandelão:

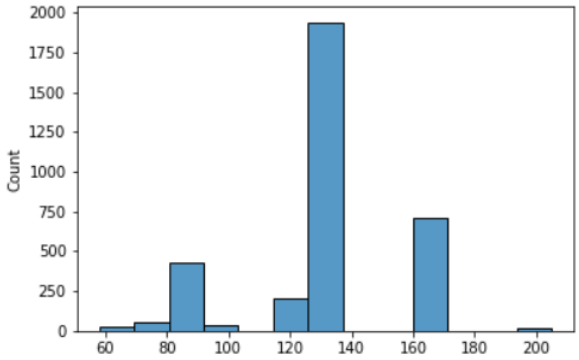

Figura 10: Distribuição do tempo calculado para cada trecho de 4 segundos presente na base de dados composta por amostras de funk mandelão. Feito utilizando a biblioteca seaborn.

A grande maioria das amostras se encontra entre 100 e 180 bpm, o que faz sentido. Ao se escutar alguns dos trechos abaixo de 100 bpm não foi possível distingui-los daqueles dentro da faixa principal que continha a maior parte das amostras. Por outro lado, os trechos com mais de 180 bpm continham amostras que soavam muito diferentes do resto,

com transições estranhas ou batidas de fato muito aceleradas. Estes foram removidos da base de dados.

Outro método utilizado para filtrar as amostras foi um simples aprendizado não supervisionado. A partir da biblioteca "sklearn" foi aplicado o algoritmo "k-means", um algoritmo de agrupamento que classifica vetores em subgrupos de modo a reduzir a variância encontrada entre cada grupo. O tipo de vetor de entrada escolhido foi a frequência amostral de cada nível sonoro de uma amostra codificada em 64 níveis pela μ-law. Calculando-se esse vetor para todo trecho da base de dados e testando diferentes números de grupos para o "k-means" obteve-se distribuições como a seguinte:

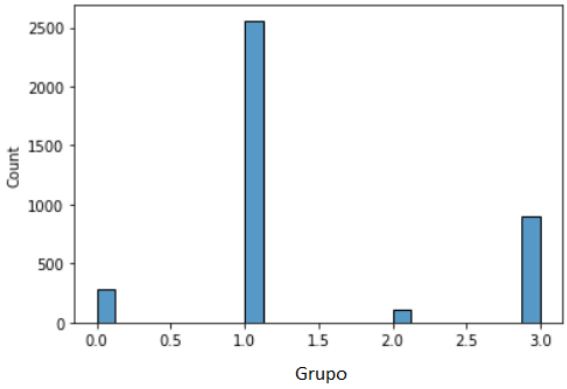

Figura 11: Distribuição em grupos encontrada pelo algoritmo "k-means" para $n = 4$, realizada com a base de dados composta por funks da produtora Kondzilla. Feito utilizando a biblioteca seaborn.

Observando amostras presentes em cada grupo (Figura 12), podemos identificar que os grupos 0 e 2 contém trechos com muitos valores nulos, o que indica momentos de completo silêncio ou ruído causado pelo pré-processamento, e por isso todos os trechos ali presentes foram desconsiderados. Já os grupos 1 e 3 possuem níveis sonoros bem distribuídos, e escutando algumas das amostras presentes em cada agrupamento pode-se compreender uma tendência de faixas de transição no grupo 3, que mudam de uma parte da música para outra de batida, ritmo ou mesmo melodia diferente, enquanto o grupo 1 é mais consistente em apenas uma batida. Como o funk é composto de transições tanto quanto consistência, e a base de dados ainda contém muito mais amostras classificadas como 1, as amostras em 3 não foram removidas.

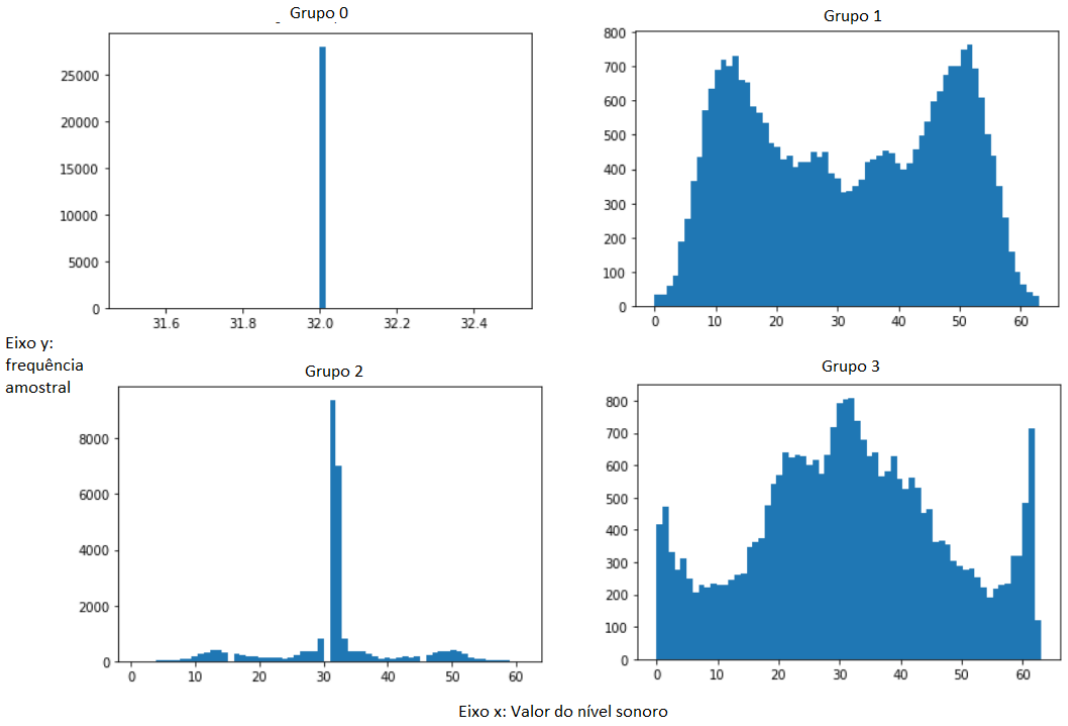

Figura 12: Exemplos de amostras encontradas em cada grupo. Vale lembrar que o valor 32 é correspondente ao nível sonoro nulo. Feito utilizando a biblioteca seaborn.

4.1.3 Bases de dados

Abaixo são descritas as bases de dados que foram compostas para a realização deste projeto, assim como as técnicas nela empregadas na etapa de pré-processamento dos dados.

4.1.3.1 100 funks mais ouvidos de 2021 (Base 1)

Finalmente, a primeira base de dados composta para o treinamento de uma rede SaShiMi a partir do zero contém 100 músicas que passaram pela rede Spleeter gerando uma faixa de vocal e uma de acompanhamento. Em seguida, as de acompanhamento foram divididas em trechos de 4 segundos sem superposição a uma frequência de amostragem de 7kHz e codificadas pela transformação μ-law de 6 bits, totalizando 4707 amostras de 28000 elementos com 64 classes (níveis sonoros) possíveis.

4.1.3.2 Funks da produtora Kondzilla (Base 2)

A segunda base foi composta utilizando o software “aTubeCatcher” e focou em obter músicas editadas por apenas uma produtora. Foi feito manualmente o download das faixas

de áudio dos 76 vídeos mais populares do canal Kondzilla em formato .mp3. Em seguida, essas faixas passaram pela rede Spleeter, que as separou em vocal e acompanhamento. Esta foi, então, dividida em janelas de 4 segundos sem superposição a uma frequência de amostragem de 7kHz e codificadas pela transformação μ-law de 6 bits. Além disso, foi realizado o processo de filtragem a partir da análise não-supervisionada com inuito de retirar amostras muito diferentes do resto da base de dados. Ao total, a base é composta por 3135 amostras de 28000 elementos com 64 classes (níveis sonoros) possíveis.

4.1.3.3 Acompanhamentos de funk do subgênero mandelão (Base 3)

A terceira base focou em utilizar apenas um subgênero do funk - o mandelão. Com o uso do "aTubeCatcher", foi feito o download de 97 faixas de áudio provenientes de vídeos do YouTube, sem canal específico, do tipo "acompanhamento de mandelão". Como as faixas não contém vocal (elas são feitas justamente para servirem de base para compositores de funk), não foi preciso utilizar a rede Spleeter. Os áudios foram divididos em trechos de 4 segundos de duração sem superposição a uma frequência de amostragem de 7kHz e codificadas pela transformação μ-law de 6 bits. Em seguida, elas passaram pelo processo de filtragem orientada pela análise não-supervisionada, totalizando 3418 amostras de 28000 elementos com 64 classes (níveis sonoros) possíveis.

4.2 Treinamento e validação

O treinamento da arquitetura SaShiMi foi realizado adaptando o código disponível em "https://github.com/HazyResearch/state-spaces" e utiliza primariamente a biblioteca PyTorch. Apesar da ordem aqui documentada, o processo de desenvolvimento desse projeto funcionou em sequências de produzir uma base de dados, treinar o modelo em cima dela, avaliar as amostras geradas e propor novas bases de dados ou parâmetros para o modelo utilizado. A avaliação das amostras foi feita de forma subjetiva, buscando reconhecer padrões de funk nos áudios gerados.

Para todas as bases de dados utilizadas, 88% das amostras foram usadas para a fase de treinamento da arquitetura - cálculo dos parâmetros das camadas através do método de otimização chamado "AdamW", e os outros 12% foram dedicados a fase de validação, em que o modelo aplica os pesos calculados na etapa anterior em amostras novas e avalia como está sendo o desempenho através de métricas pré-determinadas. Além disso, uma quantidade mínima de trechos foram reservados para teste, fase em que o modelo treinado

realiza previsões sobre dados que ele nunca teve contato anteriormente.

4.2.1 Métricas utilizadas

As métricas utilizadas e calculadas para treinamento, validação e teste dos modelos foram:

- **Perda**: a função de perda indica uma quantidade que representa a diferença quantitativa entre uma previsão gerada pelo modelo e o valor verdadeiro que deveria ser previsto. No caso de SaShiMi, a função específica utilizada se chama "cross entropy". E o valor que o algoritmo de otimização busca minimizar é justamente a "cross entropy" calculada para a partição de validação dos dados.

- **Acurácia**: é uma métrica definida pela razão entre previsões corretas sobre número total de previsões. Vale notar que para uma tarefa de classificação entre 64 classes, o valor obtido por um modelo chutando aleatoriamente a classe de uma previsão tende a 0.016 ao invés de 0.5 como no caso clássico de apenas 2 classes.

- **Acurácia para** k: é uma métrica parecida com a acurácia tradicional, porém, como SaShiMi produz probabilidades de uma previsão para cada classe, podemos selecionar os k maiores valores e considerar o vetor formado por essas k classes. Se a classe correta se encontra dentro deste vetor, a previsão é considerada k-correta. Os valores de k utilizados foram 3,5 e 10.

4.2.2 Hiperparâmetros de SaShiMi

Uma arquitetura de machine learning possui dois tipos de parâmetros: aqueles aprendidos durante o treinamento, e aqueles escolhidos previamente pela pessoa que executa o treinamento e que são conhecidos como hiperparâmetros. O modelo SaShiMi possui os seguintes hiperparâmetros relevantes, com nomes apresentados como no arquivo de configuração disponível no github "sashimi.yaml":

- **n_layers**: indica a quantidade de camadas do tipo "S4 block" (Figura 5) do modelo. Conjuntos de camadas adicionais são inseridos para cada redimensionamento de entrada presente na arquitetura completa;

- **pool**: esse hiperparâmetro é composto por dois valores que devem ser indicados da seguinte forma:

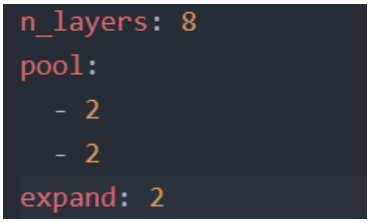

Figura 13: Trecho do arquivo "sashimi.yaml", que contém as configurações e hiperparâmetros da arquitetura. Os valores aqui mostrados criariam uma rede com 3 blocos (original, e mais um para cada redimensionamento de fator 2 realizado de um total de 2) de 8 camadas cada.

o primeiro valor indica o fator de compressão que vai redimensionar a entrada daquele ciclo. O segundo valor indica a quantidade de vezes que essa transformação será aplicada aos dados ao longo de sua arquitetura - correspondem aos blocos chamados "Up pool" e "Down pool" mostrados na Figura 5;

- **ff**: esse hiperparâmetro indica a quantidade de conexões chamadas "feed forward" presentes no modelo. Elas são as ligações que enviam dados anteriores a um processamento para serem considerados na composição da saída. Na Figura 5 são ilustradas como conexões horizontais que não passam por um conjunto "S4 block".

- **dropout**: esse hiperparâmetro é proveniente de uma técnica de mesmo nome [23] muito utilizada no treinamento de redes neurais para evitar o fenômeno chamado de "overfitting", em que a rede se apega a detalhes presentes na partição de treinamento e acaba perdendo seu poder de generalização. Isso é expresso nas métricas por uma grande diferença em favor da fase de treinamento ao se comparar com os mesmos índices calculados para a base de validação. O dropout lida com isso zerando alguns parâmetros da rede aleatoriamente e o hiperparâmetro aqui expresso indica a probabilidade de um parâmetro ser desconsiderado. Contudo, não foi relatado esse tipo de problema em nenhum experimento que utilizou a arquitetura SaShiMi [4], e portanto esse hiperparâmetro foi mantido como nulo por padrão.

4.2.3 Limitações de treinamento

Mesmo com um número limitado de camadas, operações de redimensionamento, e base de dados reduzida, não é incomum realizar um treinamento com mais de 1.4 milhão de parâmetros internos. Esse grande número de iterações é custoso computacionalmente.

O ambiente de execução da Google proposto na seção de "Tecnologias Utilizadas" na versão gratuita não possui as funcionalidades nem a capacidade necessária para a rea-

lização dos treinamentos em questão. Isso ocorre devido à longa duração dos mesmos que deveriam continuar em andamento mesmo com a desconexão do usuário com o ambiente remoto. Felizmente, a versão paga possui essa funcionalidade.

Entretanto, ainda existem dificuldades: o Colab possui duas linhas de GPU - a padrão e a premium, e suas disponibilidades são variadas e gastam créditos de uso da plataforma. Ao se utilizar uma GPU padrão é possível realizar uma época, um ciclo do treinamento, em aproximadamente 1 hora. Considerando agora que são necessárias perto de 100 épocas para se obter a confirmação de um patamar da métrica monitorada, que indica que o modelo provavelmente já não consegue melhorar consideravelmente, o tempo total de treinamento chega a 100 horas.

Para piorar, o ambiente de execução tem limite para o uso ininterrupto. Assim, não é possível realizar essas 100 horas de uma vez. No meio dessa execução é necessário recomeçar algumas vezes o processamento a partir de um checkpoint salvo da última etapa de treinamento antes da desconexão do ambiente. Como a desconexão ocorre durante momentos imprevisíveis, com intervalos de duração diferentes cada vez, perde-se tempo entre uma desconexão e um recomeço do processamento. Logo, é comum que os treinamentos durem períodos próximos a 1 semana.

Por fim, com a GPU premium cada época é realizada 6 vezes mais rapidamente, e conseguem ser concluídos em apenas 1 dia. Entretanto, tanto a GPU premium quanto a padrão gastam créditos de uso da plataforma, e a premium gasta proporcionalmente mais com sua velocidade. Assim, eventualmente os créditos comprados pelo valor mensal acabam e é necessário adquirir mais créditos[1], criando uma limitação financeira além de temporal para a progressão dos treinamentos e desenvolvimento da pesquisa.

Como observação: serviços de disponibilidade de processamento remoto a partir de GPUs estão sujeitos à procura do mercado de criptomoedas. Essa área utiliza muito das unidades de processamento do planeta e possui uma alta demanda energética, e isso certamente influencia na regulação dos preços desses serviços, que ficam mais caros pela alta procura. Infelizmente, essa concorrência atrapalha o desenvolvimento de pesquisas em inteligência artificial e outras áreas do conhecimento que exigem muito processamento.

[1] 100 créditos custam R$58,00

5 ANÁLISE DOS RESULTADOS

Esta seção se dedica a explicitar os resultados encontrados a partir dos treinamentos realizados, além de explicar as decisões tomadas para a progressão deste estudo.

5.1 Treinamento 1

O primeiro treinamento realizado utilizou a base de dados composta pelos 100 funks mais populares de 2021 (base 1). Foram utilizados os seguintes hiperparâmetros:

Hiperparâmetro	Valor
n_layers	4
pool(1)	2
pool(2)	2
ff	2
dropout	0.0

Tabela 1: Hiperparâmetros utilizados para treinamento da rede SaShiMi com a base de dados 1.

Esses hiperparâmetros são menores do que os utilizados no treinamento original da rede [4] devido à alocação de memória na GPU necessária para armazenar o modelo - as GPUs padrão do Google Colab possuem 16GB de memória. O treinamento demorou aproximadamente 1 semana, e as métricas calculadas ao final do resultado foram as seguintes:

Épocas: 100	Treinamento	Validação	Teste
Acurácia	0.326	0.300	0.291
Acurácia para 3	0.651	0.623	0.595
Acurácia para 5	0.788	0.765	0.738
Acurácia para 10	0.911	0.897	0.880

Tabela 2: Métricas obtidas ao final de 100 épocas do treinamento da arquitetura SaShiMi com a base de dados 1.

Podemos ver pelas métricas que existe uma acurácia melhor que a chance, porém isso não é suficiente para implicar na boa geração de uma música de funk. Os processos de

geração utilizando esse modelo, que duram em torno de 30 minutos, não produziram resultados subjetivamente agradáveis. As amostras apresentam muitos ruídos e anomalias que apenas machucam os ouvidos. Elas se encontram disponíveis no site: https://renatoaab-tccsite-app-5v0miw.streamlit.app/.

Um outro ponto importante de se notar é que parece existir um fenômeno leve de overfitting neste treinamento, explicitado pela diferença de desempenho entre as partições de treinamento e validação. Ao se investigar mais a fundo, foi encontrado um problema na parte do algoritmo que deveria misturar as amostras. Assim muitos dos trechos na parte de validação são de uma mesma música, uma que a fase de treinamento praticamente não viu.

Pensando em estudos gerais na área de inteligência artificial, isso não é necessariamente ruim, uma vez que para resultados consistentes é importante ter uma partição de validação independente da de treinamento. Contudo, nessa base de dados, as músicas escolhidas são muito distintas entre si. Existem diversos subgêneros de funk no dataset: funk pop, funk mandelão, funk rave, funk ostentação, funk carioca, etc. E esses estilos são consideravelmente diferentes entre si, supõe-se que isso dificulta a compreensão geral do estilo e das características que definem o gênero funk, debilitando a aprendizagem do modelo.

Com isso em mente, para os treinamentos seguintes foi tomada a decisão de se compor novas bases de dados e misturar aleatoriamente os trechos. A primeira das novas bases tenta reduzir a variância das amostras ao se escolher funks provenientes de apenas uma produtora (Kondzilla, no caso) uma vez que, nesse meio musical, a produtora tem muito controle sobre a edição da música na parte da batida - às vezes mais até que o próprio artista. A segunda base criada por esses insights tenta reduzir a variância a partir da seleção de um subgênero específico ao invés de abranger o funk como um todo. No caso, o escolhido foi o mandelão devido a sua batida bem marcada e consistência entre seus artistas. Nesta segunda, também foi testado retirar o pré-processamento que excluiria a voz do cantor, pois essa etapa cria artefatos nos trechos que poderiam também atrapalhar a aprendizagem.

5.2 Treinamento 2

O segundo treinamento realizado utilizou funks da produtora Kondzilla, os hiperparâmetros foram mantidos do primeiro treinamento:

Hiperparâmetro	Valor
n_layers	4
pool(1)	2
pool(2)	2
ff	2
dropout	0.0

Tabela 3: Hiperparâmetros utilizados para treinamento da rede SaShiMi com a base de dados 2.

Após novamente 1 semana, as métricas calculadas ao final do resultado foram as seguintes:

Épocas: 100	Treinamento	Validação	Teste
Acurácia	0.470	0.469	0.462
Acurácia para 3	0.856	0.855	0.851
Acurácia para 5	0.945	0.943	0.942
Acurácia para 10	0.989	0.988	0.988

Tabela 4: Métricas obtidas ao final de 100 épocas do treinamento da arquitetura SaShiMi com a base de dados 2.

Pode-se perceber uma melhora considerável dos resultados. Aparentemente o modelo conseguiu entender um pouco mais a partir dessa base de dados. Escutando as amostras geradas - este processo nunca é muito demorado -, elas não apresentam de fato uma musicalidade subjetivamente falando, porém já mostram certa consistência na faixa de frequências geradas. A rede optou por gerar áudios mais graves, o que promoveu uma melhoria nas métricas, mas apenas isso não foi suficiente. As amostras geradas estão disponíveis em:

Vale comentar aqui que outra proposta considerada foi a de utilizar uma base de dados composta por músicas de apenas um artista, e de preferência um que tivesse mantido seu estilo ao longo do tempo. Foi testada uma base de dados compostas por músicas apenas do DJ Mandrake, porém a dificuldade de se formar uma base com no mínimo 100 músicas debilitou a capacidade de aprendizagem e esse ideia acabou sendo descartada, mas tem valor de exploração em contextos específicos.

5.3 Treinamento 3

O treinamento com a terceira base de dados, composta apenas pelo subgênero mandelão, serviu também para exemplificar a diferença causada pela escolha dos hiperparâmetros. Estes devem ser otimizados através de diversos treinamentos, porém como um treinamento dura 1 semana em um serviço limitado e pago, essa prática foi deixada de lado e foi apenas testada aqui. Mesmo nos artigos em que as arquiteturas aqui utilizadas foram publicadas [3,4,6], pouco se vê a respeito do processo de afinamento dos hiperparâmetros, provavelmente também pela duração e custo energético dos treinamentos.

Os primeiros testados foram hiperparâmetros:

Hiperparâmetro	Valor
n_layers	2
pool(1)	2
pool(2)	2
ff	2
dropout	0.0

Tabela 5: Hiperparâmetros utilizados para treinamento da rede SaShiMi com a base de dados 3.

As métricas produzidas após 1 semana foram as seguintes:

Épocas: 95	Treinamento	Validação	Teste
Acurácia	0.368	0.361	0.369
Acurácia para 3	0.687	0.679	0.690
Acurácia para 5	0.810	0.803	0.813
Acurácia para 10	0.920	0.917	0.922

Tabela 6: Métricas obtidas ao final de 95 épocas do treinamento da arquitetura SaShiMi com a base de dados 3.

O segundo conjunto de hiperparâmetros foi:

Hiperparâmetro	Valor
n_layers	4
pool(1)	2
pool(2)	2
ff	1
dropout	0.0

Tabela 7: Hiperparâmetros utilizados para treinamento da rede SaShiMi com a base de dados 3.

As métricas calculadas ao final do treinamento foram as seguintes:

Épocas: 90	Treinamento	Validação	Teste
Acurácia	0.343	0.344	0.346
Acurácia para 3	0.659	0.659	0.666
Acurácia para 5	0.787	0.786	0.794
Acurácia para 10	0.907	0.908	0.912

Tabela 8: Métricas obtidas ao final de 90 épocas do treinamento da arquitetura SaShiMi com a base de dados 3.

Pode-se notar uma melhora em relação ao primeiro treinamento. As amostras geradas, porém, sofrem dos mesmos problemas do primeiro, o que indica que o modelo não aprendeu em um nível suficiente as características que compõem um funk mandelão. Elas podem ser ouvidas no site: https://renatoaab-tccsite-app-5v0miw.streamlit.app/.

Vale notar que a diminuição do número de camadas - foram utilizadas 2 no primeiro conjunto de hiperparametros do treinamento 3 contra 4 no segundo conjunto - pouco afetou as métricas encontradas nesse nível de aprendizado. A mudança das conexões "feedforwrd" (ff) foi mais significativa para a mudança dos resultados, retirar uma delas piorou o modelo. Isso é importante pois o número de camadas está diretamente ligado ao número de parâmetros treináveis da rede, que estão relacionados com o tempo de treinamento e a quantidade de memória necessária para alocação da rede no computador que a está treinando.

Este resultado também indica que, apesar da melhora em relação ao treinamento 1, ela possivelmente não teve a ver com a remoção da etapa utilizando a rede Spleeter. É mais provável que a melhora se deu pela maior consistência entre as músicas compondo a base de dados, e que os artefatos às vezes originados pela pré-processamento passaram despercebidos na escala do total de amostras.

5.4 Prova de conceito

Como prova de conceito foi pensado em utilizar uma rede já treinada, que fez uso de uma enorme base de dados, muito tempo e unidades de processamento. A rede escolhida e mencionada na seção de Aspectos Conceituais, Jukebox do grupo OpenAI, é na verdade uma junção de 3 redes que trabalham juntas para formar áudio nos mais diversos níveis de granularidade: a camada VAE, um upsampler e um prior.

O modelo foi treinado em trechos de 9 segundos, com codificação 32-bit (o que resulta em mais de 4 milhões de classes), em uma base de dados composta por 1,2 milhão de músicas, sendo 600 mil delas em inglês. Letras e outros dados também foram levados em

conta como forma de condicionamento da rede [6]. A primeira camada, com 2 milhões de parâmetros treináveis, foi treinada durante 3 dias utilizando 256 GPUs do tipo Tesla V100 - uma GPU de nível comercial ideal para infraestruturas especializadas em processamento de dados. Para se ter noção de quanto poder computacional existe em uma V100, cada placa custa em torno de U$ 15000,00. As camadas de upsamplers foram treinadas durante 2 semanas com 128 V100s e a última camada do modelo - o prior, com mais de 5 bilhões de parâmetros - foi treinada durante 4 semanas em 512 V100s [6].

Inicialmente foi considerado realizar o treinamento de uma versão reduzida da rede com uma base composta por funks, porém, comparativamente com os recursos do treinamento original, não havia recursos disponíveis suficientes para garantir que a rede atingisse seu potencial real. Logo, utilizou-se apenas a rede "1b_lyrics"já treinada disponibilizada pelo grupo OpenAI. Ela recebeu um trecho de 15 segundos de um dos funks da produtora Kondzilla e gerou mais 15 segundos de áudio baseado em sua entrada inicial.

Os resultados gerados, disponíves em https://renatoaab-tccsite-app-5v0miw.streamlit.app/, mostram que já se encontrou formas artificiais de se criar músicas de vários gêneros. Houve resultados que fugiram do estilo musical apresentado, mas foram produzidas amostras que seguem as características de um funk. A base de dados na qual a rede foi treinada dificilmente continha muitos funks - talvez apenas funks do subgênero funk pop, que acabaram tendo mais relevância internacional nos últimos tempos - e isso dificulta a compreensão da rede com relação ao gênero. Contudo, ainda assim, foi produzido um trecho condizente com o mostrado anteriormente, o que indica um forte poder de generalização. Com essa base pronta, infelizmente é impossível realizar uma geração não condicionada com um trecho de uma amostra, pois não há representatividade do gênero funk entre as músicas processadas pela rede.

Esta rede utilizada também foi treinada com as letras das músicas, logo compreende o papel do vocal e aprendeu certa fonética para criar sons parecidos com alguém cantando. A diferença é que os sons criados não representam palavras, falta certa restrição e modelização semântica. Além disso, é possível notar ainda o viés da fonética da língua inglesa, o que era de se esperar, mas ainda instiga a imaginação considerar uma base apenas de músicas em português, em que os resultados seriam mais condizentes.

6 CONCLUSÕES

A engenharia deve ser a área mais interdisciplinar que temos. É um pouco essa a ideia por trás da profissão. O engenheiro é aquele profissional que vai criar soluções e projetos e com eles causar mudanças na realidade. Tal potencialidade não deve ser ignorada e possuir conhecimento necessário para usufruir dela da melhor forma possível em prol da sociedade também é responsabilidade desse profissional.

É dentro desse contexto que o engenheiro deve conhecer, entre outras coisas várias, arte e cultura - pontos essenciais para o enriquecimento da experiência de vida moderna. Por isso é importante existir uma aproximação desses conceitos com a formação dessas pessoas. Este projeto teve o intuito de reduzir essa distância escolhendo um contexto artístico - a música de gênero funk - deixado de lado pelo meio acadêmico e realizando um estudo sobre técnicas avançadas de inteligência artificial aplicadas a esse contexto. O funk, muito marginalizado e alvo de preconceito por parte da sociedade, foi escolhido particularmente para tentar dar visibilidade a uma parte da arte nacional em um meio distante de suas manifestações tradicionais.

Recentemente têm surgido redes neurais profundas capazes de investigar e reproduzir facetas da subjetividade criativa humana. É curioso ver um modelo pautado em regras matemáticas se organizar a ponto de criar arte que seja tão boa qualitativamente quanto a criada por um humano. É um processo matemático-antropofágico, em que um conjunto de instruções - um algoritmo - é capaz de receber e processar as obras artísticas da humanidade e em seguida gerar algo novo. São reflexões como essas que indicam que complexidade entre unidades simples geram conceitos abstratos e que transcendem a própria essência daquilo que os criou.

A própria configuração ou organização dos neurônios de uma rede artificial contém a informação necessária para codificar e "entender" um conceito subjetivo humano. Talvez o próprio ser humano seja um conjunto de várias configurações. Não se pode subestimar a capacidade cerebral do ser humano: apenas para compreender uma parte dos conceitos subjetivos de nossa psique foram necessários anos de desenvolvimento na área de inte-

ligência artificial e um investimento absurdo em infraestrutura para o treinamento das redes produzidas. A quantidade de energia e processamento utilizado em redes que representam o estado da arte na geração de amostras artísticas tornam impossível a exploração a fundo dessa área pelo usuário padrão. Apenas grandes grupos de pesquisa ou empresas com setores especializados nisso têm a capacidade de realizar os treinamentos com a quantidade de informação necessária a uma rede. Se um dia existir uma inteligência artificial genérica, similar a um humano, provavelmente existirão pouquíssimas porque haverá um limite de recursos disponíveis e que façam sentido serem empregados para isso.

Os treinamentos aqui realizados utilizaram uma das arquiteturas mais rápidas e novas da atualidade para geração de música a partir de áudio cru: SaShiMi [4]. Entretanto, ainda foram sentidas diversas demandas de recursos - em unidades de processamento (GPUs) - que acabaram limitando o potencial da rede e formato de áudio escolhidos. As amostras geradas não possuem musicalidade e não podem ser consideradas um funk. Porém, como prova de conceito, ao utilizar-se de a rede Jukebox, que usou uma base de dados maior com e foi treinada com muito mais processamento, foi possível obter uma geração condicionada relevante. Isso mostra que existe o potencial da utilização dessas redes para geração de funk, mas novamente o gênero sofre de falta de visibilidade em meios de pesquisa, e devido ao alto custo de se conduzir uma pesquisa de ponta em inteligência artificial no Brasil, o funk provavelmente não será compreendido por uma rede neural artificial no futuro próximo.

REFERÊNCIAS

[1] 2D Convolution. https://peltarion.com/knowledge-center/modeling-view/build-an-ai-model/blocks/2d-convolution. Accessed: 2022-10-20.

[2] COLEMAN, D. *BASIC ARCHITECTURE OF RNN AND LSTM.* https://pydeeplearning.weebly.com/blog/basic-architecture-of-rnn-and-lstm. Published: 2017-01-18.

[3] OORD, A. v. d. et al. Wavenet: A generative model for raw audio. arXiv, 2016.

[4] GOEL, K. et al. It's raw! audio generation with state-space models. arXiv, 2022.

[5] ROCCA, J. *Understanding Variational Autoencoders (VAEs).* 2019. Disponível em: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>.

[6] DHARIWAL, P. et al. *Jukebox: A Generative Model for Music.* arXiv, 2020. Disponível em: <https://arxiv.org/abs/2005.00341>.

[7] WHEATLEY, D.; BICKERTON, C. Subjective well-being and engagement in arts, culture and sport. *Journal of Cultural Economics*, March 2016.

[8] BRIOT, J.-P.; HADJERES, G.; PACHET, F.-D. *Deep Learning Techniques for Music Generation – A Survey.* arXiv, 2017. Disponível em: <https://arxiv.org/abs/1709.01620>.

[9] FREITAS, P. *Funk é cultura? Entenda a origem do estilo de música controverso.* 2021. Disponível em: <https://www.megacurioso.com.br/artes-cultura/118045-funk-e-cultura-entenda-a-origem-do-estilo-de-musica-controverso.htm>.

[10] PEREIRA, A. B. "a maior zoeira": experiências juvenis na periferia de são paulo. Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo, 2010.

[11] NATSIOU, A.; O'LEARY, S. Audio representations for deep learning in sound synthesis: A review. arXiv, 2022.

[12] SKANSI, S. *Introduction to Deep Learning.* [S.l.]: Springer, 2018.

[13] MIGUEL, T. *How the LSTM improves the RNN.* https://towardsdatascience.com/how-the-lstm-improves-the-rnn-1ef156b75121. Published: 2021-01-01.

[14] OORD, A. v. d.; KALCHBRENNER, N.; KAVUKCUOGLU, K. *Pixel Recurrent Neural Networks.* arXiv, 2016. Disponível em: <https://arxiv.org/abs/1601.06759>.

[15] TAY, Y. et al. *Long Range Arena: A Benchmark for Efficient Transformers*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2011.04006>.

[16] GU, A.; GOEL, K.; Ré, C. *Efficiently Modeling Long Sequences with Structured State Spaces*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2111.00396>.

[17] TUSTIN, A. A method of analysing the behaviour of linear systems in terms of time series. Journal of the Institution of Electrical Engineers-Part IIA: Automatic Regulators and Servo Mechanisms, 1947.

[18] GU, A. et al. *HiPPO: Recurrent Memory with Optimal Polynomial Projections*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2008.07669>.

[19] KONG, Z. et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2009.09761>.

[20] VASWANI, A. et al. *Attention Is All You Need*. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1706.03762>.

[21] MOUSSALAM, M. *Releasing Spleeter: Deezer Research source separation engine*. https://deezer.io/releasing-spleeter-deezer-r-d-source-separation-engine-2b88985e797e. Published: 2019-11-04.

[22] HENNEQUIN, R. et al. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, v. 5, p. 2154, 06 2020.

[23] SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan 2014. ISSN 1532-4435.