UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Andrey Lucas Herchonvicz

Neuroevolução de Redes LSTM: uma aplicação na detecção de engano via voz

Florianópolis
2023

Andrey Lucas Herchonvicz

Neuroevolução de Redes LSTM: uma aplicação na detecção de engano via voz

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Orientador: Prof. Rafael de Santiago, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Herchonvicz, Andrey Lucas
Neuroevolução de Redes LSTM: uma aplicação na detecção de engano via voz / Andrey Lucas Herchonvicz ; orientador, Rafael de Santiago, 2023.
74 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciência da Computação. 2. Aprendizado Profundo. 3. Algoritmo Genético. 4. Neuroevolução. 5. Detecção de engano através da fala. I. Santiago, Rafael de. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Andrey Lucas Herchonvicz

Neuroevolução de Redes LSTM: uma aplicação na detecção de engano via voz

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Andrei de Almeida Sampaio Braga, Dr.
Universidade Federal da Fronteira Sul

Prof. Mateus Grellert da Silva, Dr.
Universidade Federal de Santa Catarina

Prof. Mauro Roisenberg, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

GOV.BR
Documento assinado digitalmente
PATRICIA DELLA MEA PLENTZ
Data: 01/08/2023 16:14:18-0300
CPF: ***.243.700-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

ICP-Edu
Documento assinado digitalmente
Rafael de Santiago
Data: 31/07/2023 18:21:02-0300
CPF: ***.277.019-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Rafael de Santiago, Dr.
Orientador

Florianópolis, 2023.

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram para a realização deste trabalho. Primeiramente gostaria de agradecer meu orientador, Rafael de Santiago, cuja orientação e sabedoria foram de valor inestimável. Agradeço a sua paciência, dedicação e disposição em compartilhar seus conhecimentos que contribuíram para o desenvolvimento desta dissertação.

Gostaria também de expressar minha gratidão aos meus pais, Ana e Joni e minha irmã Angela. Seu apoio ao longo dos anos foram fundamentais para a minha jornada acadêmica. Suas palavras de incentivo e sacrifícios nunca serão esquecidos.

Além disso, gostaria de agradecer aos meus antigos colegas de trabalho da antiga equipe de pesquisa aplicada da Senior Sistemas, liderada pelo Márcio Jasinski. O apoio, colaboração e troca de conhecimentos foram essenciais para o desenvolvimento das ideias apresentadas neste trabalho. Sou grato por ter compartilhado essa experiência profissional com vocês e pela amizade que construímos ao longo do tempo.

Por fim, gostaria de dedicar um agradecimento especial a minha esposa Mariza, cujo apoio incansável e encorajamento foram fundamentais em cada etapa desse processo. Nos momentos mais dificies, foi ela quem me motivou a continuar, lembrando-me do propósito e da importância do meu trabalho. Seu amor e compreensão foram uma inspiração constante, e serei eternamente grato.

RESUMO

A tarefa de detectar mentiras tem uma longa história, desde o uso do polígrafo e, recentemente, a detecção de mentiras em conversas tem se mostrado um desafio complexo. A utilização dessa tecnologia pode ser aplicada em diversas áreas, como segurança, cibersegurança, recursos humanos, psicologia e também para interrogatório de suspeitos. Devido à dificuldade de detectar mentiras por meio da fala, muitas abordagens estão aplicando aprendizado de máquina combinando áudio da fala e características textuais da transcrição de áudio. Muitas técnicas foram desenvolvidas para detectar mentiras por meio da fala, e o objetivo deste trabalho é discutir mais detalhadamente essas abordagens que foram identificadas por meio de uma revisão sistemática da literatura. Assim, este trabalho apresenta um histórico que mostra aspectos da mentira, detecção de mentiras e técnicas usadas para esse propósito. Também são discutidas técnicas baseadas em aprendizado profundo para detectar mentiras, bem como outros aspectos, como conjuntos de dados e métricas disponíveis. Além disso, o objetivo é obter melhores resultados na detecção de mentiras por meio da proposta de uma abordagem de Neuroevolução, que combina redes neurais e algoritmo genético. Os resultados do método proposto alcançaram uma pontuação F1-score de 63,06% sem *overfitting*, utilizando o conjunto de dados *Bag-of-Lies*. Os resultados mostram que a abordagem proposta pode servir para melhorar as arquiteturas de redes neurais por meio de evolução guiada, e pode ser estendida para outras tarefas. Além disso, o desempenho da abordagem pode ser ainda mais otimizado investigando outras características das redes *Long short-term memory* (LSTM), como configurações ótimas de hiperparâmetros. Finalmente, conclui-se este trabalho discutindo as limitações e examinando trabalhos promissores e futuros.

Palavras-chave: Aprendizado Profundo. Algoritmo Genético. Neuroevolução. Detecção de engano através da fala.

ABSTRACT

The task of detecting deception has a long history since using the polygraph and, not long ago, spot deception in conversational speech has been proved to be a complex challenge. The use of this technology can be applied in many fields such as security, cybersecurity, human resources, psychology and also for suspect interrogation. Due to the difficulty of detecting lies through speech, many approaches are applying machine learning combining audio of speech and textual characteristics from audio transcription. Many techniques have been developed to spot deceit through speech, and the purpose of this work is to discuss in more detail these approaches that were identified by a systematic literature review. Thus, we present aspects of lying, deception detection and techniques used for this purpose. Also, we discuss deep learning-based techniques to detect deception and also other aspects such as available datasets and metrics. Moreover, we aim to obtain better results in deception detection by proposing a Neuroevolution approach, which combines Neural Networks and Genetic Algorithm. The results of our method achieved a f1-score of 63.06% with no overfitting, using the Bag-of-Lies dataset. Our results show that our approach can be useful for improving neural network architectures through guided evolution, and it can be extended to other tasks. Moreover, the performance of our approach can be further optimized by investigating other characteristics of Long short-term memory (LSTM) networks, such as optimal hyper-parameter configurations. Finally, we conclude this work by arguing the limitations and examining promising and future works.

Keywords: Deep Learning. Genetic Algorithm. Neuroevolution. Speech Deception Detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A invenção do polígrafo é considerado o início da detecção de enganos assistida. Esta tecnologia não está disponível para o público tendo sido aplicada por alguns governos para interrogar suspeitos de crimes (PLACE, 2010). Se essa tecnologia fosse acessível, muitos campos poderiam se beneficiar. Os Recursos Humanos (RH) podem detectar discursos enganosos de candidatos em uma entrevista de emprego (PANOVA; POSTOLOV, 2015). Instituições financeiras e seguradoras podem usar essa tecnologia como ferramenta auxiliar para capturar declarações enganosas antes de conceder um empréstimo ou registrar sinistros (LAMPERT, 1963). Finalmente, os aplicativos de computador podem usar essa tecnologia como método de autenticação (BAHMANI; NASAB, 2016).

Nos últimos anos, abordagens baseadas em aprendizado profundo têm sido usadas para classificar declarações enganosas ou verdadeiras. Tais métodos estão evoluindo e alcançando melhores resultados à medida que o poder computacional vem se tornando mais acessível (MENDELS et al., 2017). Outro aspecto importante são as variáveis utilizadas para detectar mentiras por meio da fala. Existem muitos tipos diferentes, como Coeficientes Cepstral de Frequência Mel (MFCC) (LEVITAN et al., 2015), Taxa de Cruzamento Zero (ZCR) (FU et al., 2019), Frequência Fundamental (F0), recursos acústicos e prosódicos (PÉREZ-ROSAS et al., 2015). Outra possibilidade é transcrever a fala e extrair características do texto (KOPEV et al., 2019).

Entre as técnicas de aprendizado profundo, as redes neurais recorrentes (RNNs) são especialmente úteis para processar sequências de dados, e uma das arquiteturas mais populares de RNNs é a *Long short-term memory* (LSTM). No problema de detecção de enganos, alguns trabalhos usaram essa abordagem. Marcolla (MARCOLLA; SANTIAGO; DAZZI, 2020) propõe uma abordagem para detectar mentiras usando estresse de voz com redes LSTM. Seu método utiliza apenas voz e o conjunto de variáveis é composto por características do MFCC extraídas de falas gravadas pelo autor na língua portuguesa do Brasil. Além disso, Hershkovitch (NEITERMAN; BITAN; AZARIA, 2020) também propôs uma rede LSTM usando falas em inglês e hebraico. O tipo de dado de entrada utilizado neste trabalho foi um espectrograma das sequências de áudio.

Variações de LSTM também são aplicadas à detecção de enganos. Chou (CHOU; LIU; LEE, 2019) propôs um conjunto de um LSTM Bidirecional (BiLSTM) e uma camada densa com mecanismo de *attention*. Neste trabalho, as variáveis foram extraídas usando discursos individuais e a interação entre as falas do entrevistado e entrevistador do conjunto de dados *Daily Deceptive Dialogues Corpus of Mandarin* (DDDM) (HUANG et al., 2019). Xie (XIE et al., 2018) usou uma combinação de rede neural recorrente com operações de convolução (Conv-BiLSTM), BiLSTM e camadas totalmente conectadas utilizando um conjunto de dados próprio e o corpus *Columbia-SRI-Colorado* (CSC) (HIRSCHBERG et al., 2005). Finalmente, Levitan (LEVITAN et al., 2015) propôs uma combinação de um BiLSTM com redes perceptrons de multicamada (MLP) onde o autor utilizou variáveis acústicas e lexicais extraídas do *Columbia*

X-Cultural Deception (CXD) Corpus (LEVITAN et al., 2015) bem como as variáveis acústicas.

Um dos principais problemas para detectar enganos é a falta de dados. Existem alguns conjuntos de dados disponíveis na internet, mas cada corpus tem características diferentes. Além disso, as redes neurais profundas alcançam os melhores resultados com uma enorme quantidade de dados. Por outro lado, não há uma grande disponibilidade de dados rotulados disponíveis. Para resolver este problema, (FU et al., 2019) propôs uma abordagem usando uma técnica semi-supervisionada composta por um codificador, um decodificador e um classificador.

Esse trabalho propõe uma forma de resolver o problema para detectar enganos usando a fala e como técnicas de redes neurais profundas podem ser aplicadas a essa tarefa. Além disso, esse trabalho busca utilizar os últimos avanços da literatura para detectar enganos usando abordagens de aprendizado profundo. Por fim, uma nova abordagem para este problema é proposta usando a técnica de Neuroevolução.

Para exemplificar o potencial das técnicas de Neuroevolução, (YERIGERI; RAGHA, 2019) propôs a utilização da técnica para evoluir uma rede neural que reconhece emoções. Esta estratégia de Neuroevolução também pode ser aplicada em redes neurais para encontrar os hiperparâmetros ideais e/ou topologia de rede. De acordo com (ABBASI et al., 2015; MAJDI; BEIKI, 2010), essa abordagem tem potencial para obter melhores resultados quando comparada à tentativa e erro e consome menos tempo.

1.1 PROBLEMA DE PESQUISA

Até onde vai o conhecimento do autor desse trabalho, não há nenhum outro estudo usando a Neuroevolução para detecção de enganos através da fala. Então, o trabalho busca entender como a Neuroevolução pode ser utilizada e quão bem desempenha no contexto desse tipo de detecção.

1.1.1 Pergunta de Pesquisa

Esse estudo apresenta a seguinte pergunta de pesquisa:

- Poderia uma abordagem de Neuroevolução alcançar melhores resultados em comparação com os modelos de aprendizado profundo do estado da arte, na detecção de declarações enganosas por meio da fala?

1.1.2 Hipótese

A pesquisa considera as seguintes hipóteses:

- Nula: Uma abordagem baseada em Neuroevolução não alcança resultados superiores quando comparada a outros métodos de aprendizado profundo para detectar enganos por meio da fala.

- Alternativa: Uma abordagem baseada em Neuroevolução alcança resultados superiores quando comparada a outros métodos de aprendizado profundo para detectar enganos por meio da fala.

1.2 OBJETIVOS

1.2.1 Objetivo Principal

Propor uma abordagem Neuroevolutiva para a detecção de enganos através da fala.

1.2.2 Objetivos Específicos

Para atingir o objetivo deste trabalho, os seguintes objetivos específicos devem ser alcançados:

- Identificar os trabalhos do estado da arte na detecção de enganos através da fala usando redes neurais profundas com ou sem otimização.
- Estabelecer os conjuntos de dados mais usados e comparáveis.
- Desenvolver um método baseado em Neuroevolução para detectar enganos na fala.

1.3 METODOLOGIA DE PESQUISA

Com a intenção de responder à pergunta de pesquisa, os passos necessários são apresentados a seguir:

1. Desenvolver uma pesquisa com arquitetura de redes neurais profundas do estado da arte para detecção de enganos através da fala;
2. Comparar os métodos mais avançados;
3. Identificar os conjuntos de dados mais usados e disponíveis para comparação;
4. Desenvolver uma abordagem Neuroevolutiva;
5. Realizar análises comparativas para determinar o desempenho do método proposto em relação ao estado da arte;
6. Analisar, reportar e divulgar os resultados obtidos;
7. Disponibilizar o código-fonte do trabalho em um repositório público sob a licença MIT (*Massachusetts Institute of Technology License*) (MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 1988)

1.4 CONTRIBUIÇÕES

Os resultados da revisão da literatura desse trabalho foram publicados como um *survey* em (HERCHONVICZ; SANTIAGO, 2021). Todo o código utilizado pelo trabalho pode ser encontrado em um repositório no GitHub (HERCHONVICZ, 2023).

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante deste trabalho está dividido da seguinte forma. O capítulo 2 apresenta uma revisão teórica sobre detecção de engano, além de explorar técnicas de aprendizado de máquina profundo aplicado à tarefa de detecção de engano na fala. No capítulo 3 é descrita a estratégia utilizada para realizar uma revisão da literatura e uma visão geral dos trabalhos relacionados. O capítulo 4 apresenta o método de Neuroevolução para redes LSTM, incluindo a preparação e persistência dos dados, configurações preliminares das redes neurais, algoritmo genético e considerações sobre o método proposto. Os experimentos e análises realizados, incluindo as configurações do algoritmo genético e das redes LSTM, a análise dos resultados e a identificação do melhor indivíduo são apresentados no capítulo 5. Finalmente, no capítulo 6, são apresentadas as conclusões do estudo e apontados possíveis trabalhos futuros.

2 FUNDAMENTOS

Este capítulo destina-se a descrever os fundamentos teóricos relacionados à detecção de enganos e também os métodos de detecção automática de enganos usados nesse trabalho. Para tanto, serão abordados tópicos como mentira e engano, redes neurais profundas, LSTM, aprendizado de máquina automatizado, algoritmo genético, Neuroevolução e MFCCs. Com uma visão geral desses conceitos, será possível entender melhor o processo de detecção de enganos através da fala e a contribuição deste trabalho para a área.

2.1 MENTIRA E ENGANO

A mentira e o engano estão intrinsecamente relacionados. A mentira é uma forma específica de engano, onde se comunica um falso testemunho para enganar outra pessoa. Não é considerado mentira quando a pessoa que faz a declaração acredita que está dizendo a verdade. Além disso, formas de comunicação em que a pessoa sabe que a frase não é verdadeira não podem ser caracterizadas como mentira, um exemplo disso é a ficção (CHISHOLM; FEEHAN, 1977).

O estresse é um aspecto importante que pode ser relacionado com a mentira. Alguns estudos como (DEPAULO et al., 1996), sugerem que sob estresse as pessoas tendem a tomar decisões menos precisas, aumentando assim a probabilidade de mentir. Além disso, o estudo (JR; DEPAULO, 2006) explorou a relação entre mentira e estresse. Os resultados indicaram que, em situações de alto estresse, os indivíduos que mentem podem apresentar mais dificuldade em manter a coerência e a precisão em suas respostas, o que pode levar a revelações inconsistentes e reveladoras.

Estima-se que a mentira e o engano surgiram na pré-história junto com a criação da escrita e da moeda. Ao registrar o controle da comercialização de bens e produtos, a mentira foi utilizada como estratégia de manipulação para a sobrevivência dos grupos (FERNÁNDEZ, 2006). Na Europa da Idade Média, a verdade e a mentira eram associados a entidades e práticas religiosas. Esses conceitos eram utilizados como ferramenta para atrair pessoas, alegando uma disputa entre o bem e o mal (ASIMOV, 2009).

2.1.1 Principais tipos de mentira e engano

Existem alguns fatores que levam as pessoas a mentir e enganar. A natureza humana tem uma tendência a projetar uns aos outros, especialmente as pessoas que desempenham um papel importante na vida de um indivíduo. Um indivíduo pode enganar as pessoas importantes para ele para manter relacionamentos e evitar machucá-los (BRÅTEN, 2009; COLE, 2001).

Outro tipo comum de mentira é evitar a punição. Esse comportamento é observado desde a infância, onde a mentira é usada para evitar punições ou constrangimentos. Por exemplo, quando uma criança ou adolescente não faz a lição de casa e usa uma mentira para justificar

e evitar notas baixas (LEWIS, 1993). Em adultos, esse comportamento geralmente tem um contexto mais sério, onde o engano é usado para evitar a perda de dinheiro, liberdade, reputação e até mesmo a vida (FORNACIARI; POESIO, 2013; SUNG; PENTLAND, 2005).

A mentira também é usada como ferramenta inteligente de manipulação e gestão pessoal para benefício próprio. Nesse cenário, as mentiras mais comuns são acumular riqueza, segurança, status e atrair um companheiro(a) (WILLIAMS et al., 2009; LEVINE; KIM; HAMEL, 2010).

Além de enganar outras pessoas, pode-se enganar a si mesmo. O autoengano acontece quando uma pessoa aceita como verdadeira uma realidade que é falsa sem estar ciente disso. Por exemplo, quando uma pessoa tenta se convencer de que algo é verdade quando não é. Outra situação muito comum no autoengano é a interpretação tendenciosa de um fato, levando a uma narrativa enganosa do evento real (MELE, 2000; HIPPEL; TRIVERS, 2011).

Há pessoas com uma tendência duradoura e incontrolável de mentir. Esse comportamento é conhecido como mentira patológica (DIKE; BARANOSKI; GRIFFITH, 2005). Apesar de ser um conceito pouco compreendido pela ciência, muitos pesquisadores acreditam que se trata de uma doença psicológica, onde o indivíduo não consegue parar de mentir, independente da motivação para a mentira (GRUBIN, 2005; MUZINIC; KOZARIC-KOVACIC; MARINIC, 2016).

Desde o século XX, estudos têm sido realizados com o objetivo de determinar se os padrões genéticos estão relacionados a mentira e engano. Um dos primeiros experimentos conhecidos foi realizado em 1928 pelos pesquisadores Hartshorne e May, onde um padrão semelhante de falso testemunho entre irmãos foi relatado (HARTSHORNE; MAY, 1928).

Apesar de não ser bem recebido pela comunidade científica, outros estudos foram publicados posteriormente buscando correlacionar o engano com uma base genética de um indivíduo (CLONINGER; REICH; GUZE, 1978; CADORET; CAIN; CROWE, 1983; MOFFITT, 2017).

2.1.2 Métodos para detecção de engano

Os métodos para detecção de mentiras e enganos são um assunto de grande interesse para os humanos há séculos. Desde os primórdios da civilização humana, as pessoas procuram entender as motivações e os comportamentos dos outros. Nos últimos anos, a tecnologia desempenhou um papel significativo na evolução das técnicas de detecção de mentiras.

Uma das primeiras formas de detecção de mentiras foi o uso de tortura. Nos tempos antigos, as pessoas acreditavam que a dor física causada pela tortura levaria um mentiroso a confessar. Este método foi usado em várias sociedades, incluindo a Roma e a Grécia antigas (INNES, 2012). No entanto, percebeu-se que a tortura não era um método confiável para detectar mentiras, pois as pessoas confessavam qualquer coisa para parar a dor (O'MARA, 2015).

Ao mentir, os seres humanos fazem esforços cognitivos de forma involuntária. Este tipo de pista é explorado em ambientes forenses onde o polígrafo é usado para determinar a fala

enganosa. Os polígrafos foram desenvolvidos como um método mais científico para detectar mentiras no século XX. O aparelho detecta mentiras através da análise do teste do polígrafo onde são combinadas várias medidas como pressão arterial, pulso, respiração e transpiração, o que pode indicar quando uma pessoa está mentindo. No entanto, esta tecnologia não é acessível e só pode ser usada em instituições governamentais (PLACE, 2010). Além disso, a precisão dos testes do polígrafo tem sido objeto de controvérsia (IACONO, 2008). Muitos estudos mostraram que os testes não são totalmente confiáveis e podem ser influenciados por fatores como ansiedade ou medicamentos (REINACH; LOUW, 2002; COOK; MITSCHOW, 2019).

Nos últimos anos, novas tecnologias foram desenvolvidas para melhorar a precisão da detecção de mentiras. Uma dessas tecnologias é a máquina fMRI (ressonância magnética funcional) (KOZEL et al., 2005). A máquina fMRI mede as mudanças no fluxo sanguíneo para diferentes partes do cérebro, o que pode indicar quando uma pessoa está mentindo. Este método ainda está em fase experimental, mas os resultados iniciais são promissores. Outra tecnologia desenvolvida é a máquina de rastreamento ocular (DARWISH; BATAINEH, 2012). Esta máquina rastreia os movimentos dos olhos de uma pessoa, o que pode indicar quando uma pessoa está mentindo. Este método é baseado na teoria de que quando uma pessoa está mentindo, ela tende a evitar o contato visual.

Com o aumento do poder computacional, os métodos para detectar enganos vêm evoluindo no campo da inteligência artificial. De fato, é possível extrair um grande número de pistas usando apenas aspectos da fala como tom, energia, duração, espectro e cepstral (GRACIARENA et al., 2006). Além disso, os dados de áudio podem ser transcritos e combinados com variáveis de áudio e características do texto (MENDELS et al., 2017). Por isso, detectar enganos por meio da fala é um campo promissor e tem muito a ser explorado.

Além disso, há outras oportunidades a serem exploradas. Uma delas é a necessidade de mais dados, como foi apontado por (MARCOLLA; SANTIAGO; DAZZI, 2020). Outro tópico que pode ser explorado é o conjunto de variáveis mais relevantes para se detectar enganos através da fala. Existe a possibilidade de incluir características comportamentais e da personalidade da pessoa, conforme indicado por (CHOU; LIU; LEE, 2019). Para validar a robustez dos sistemas de detecção de engano, (MENDELS et al., 2017) identificou melhorias testando sua abordagem em diferentes conjuntos de dados. Por fim, (FU et al., 2019) sugeriu a melhoria do modelo por meio da combinação de outras abordagens de redes neurais profundas.

2.2 APRENDIZADO PROFUNDO

Em sua essência, o aprendizado profundo é construído sobre o conceito de redes neurais artificiais, inspiradas na estrutura e função do cérebro humano. Essas redes consistem em nós interconectados, ou “neurônios”, que se comunicam entre si por meio de uma série de operações matemáticas. Ao ajustar os pesos e vieses dessas conexões, os algoritmos de aprendizado profundo podem aprender a reconhecer padrões e fazer previsões com base nos dados de entrada (GOODFELLOW; BENGIO; COURVILLE, 2016).

As Redes Neurais Artificiais são compostas de neurônios, camadas e conexões entre os neurônios. Um neurônio recebe a soma de todas as saídas dos neurônios anteriores como entrada, realiza cálculos nesses dados e passa os resultados para outros neurônios por meio das conexões. As ligações estão associadas ao peso onde a entrada é multiplicada pelo peso correspondente. Cada um desses neurônios está associado a um viés adicionado à entrada. Por fim, esse valor é passado para uma função de ativação, que determina se o neurônio correspondente será ativado ou não (BISHOP et al., 1995; GOODFELLOW; BENGIO; COURVILLE, 2016).

A estrutura de camadas de uma rede neural é mostrada na Figura 1. Uma camada é composta por um conjunto de neurônios. A primeira camada de uma rede é a camada de entrada e a última é a camada de saída. Cada camada entre essas duas é chamada de camadas ocultas, onde a saída da camada anterior fornece entrada para a próxima camada. Esse processo de passar informações de uma camada para outra é chamado de propagação direta. Uma vez que a previsão de saída é comparada com a saída real, a rede neural calcula a perda e ajusta os pesos da rede usando a técnica de retropropagação (BISHOP et al., 1995).

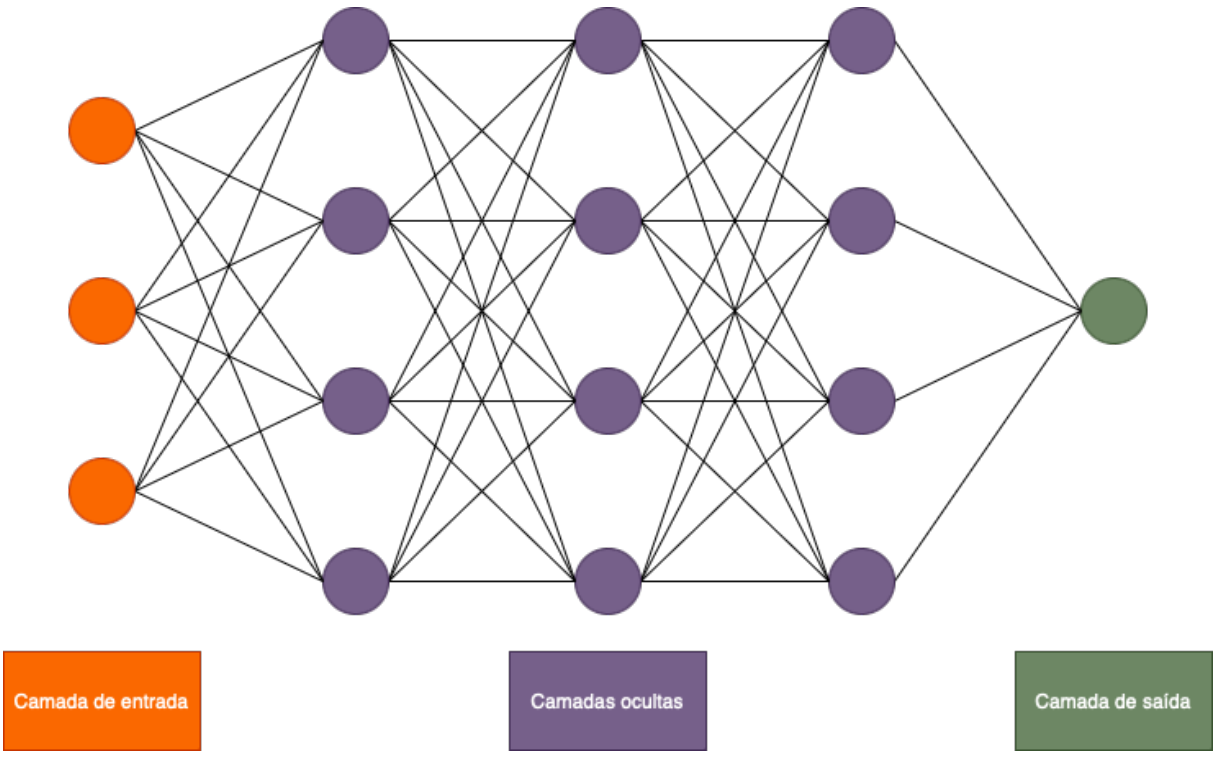

Figura 1 – Estrutura de uma rede neural

A propagação direta e a retropropagação são executadas muitas vezes em várias entradas. Este processo é repetido até que a rede consiga prever os resultados corretamente na maioria dos casos (BISHOP et al., 1995; GOODFELLOW; BENGIO; COURVILLE, 2016). As redes neurais aumentam o desempenho à medida que mais dados são coletados. Assim, atualmente há mais dados e poder computacional disponíveis e é por isso que este campo tem recebido muita atenção, não só da comunidade científica, mas também da indústria (JORDAN;

MITCHELL, 2015; CHEN; LIN, 2014).

A ideia de modelar matematicamente um neurônio surgiu em 1943, quando Warren McCulloch e Walter Pitts propuseram a construção de circuitos lógicos booleanos (MCCULLOCH; PITTS, 1943). Em 1949, Donald Hebb propôs a aplicação de pesos nas conexões entre os neurônios para ajustar a intensidade entre as conexões (HEBB, 2005). A primeira rede neural foi construída por Frank Rosenblatt em 1957, onde ele usou o modelo de Hebb para construir um computador capaz de aprender o reconhecimento de padrões usando sensores (ROSENBLATT, 1958).

A partir da década de 1960, após a publicação do livro "Perceptrons" de Marvin Minsky e Seymour Papert (MINSKY; PAPERT, 2017), a pesquisa usando redes neurais foi abandonada. Finalmente, em 1986, David Rumelhart publicou um artigo sobre retropropagação na revista Nature e desde então avanços significativos surgiram (RUMELHART; HINTON; WILLIAMS, 1986).

O aprendizado profundo é um subconjunto do aprendizado de máquina, que é um ramo da inteligência artificial que permite que as máquinas aprendam com os dados e façam previsões ou decisões. O aprendizado profundo, em particular, refere-se ao uso de redes neurais artificiais compostas por várias camadas para modelar e resolver problemas complexos. O aprendizado profundo tornou-se cada vez mais popular nos últimos anos, graças à sua capacidade de produzir resultados de ponta em uma ampla gama de aplicações, incluindo reconhecimento de imagem e fala, processamento de linguagem natural e direção autônoma (LECUN; BENGIO; HINTON, 2015).

Uma das principais vantagens do aprendizado profundo é sua capacidade de lidar com dados complexos, como imagens e vídeos. As técnicas tradicionais de aprendizado de máquina geralmente tem dificuldade com esses dados, pois o número de variáveis e as possíveis combinações dessas variáveis podem rapidamente se tornar inviável. Os algoritmos de aprendizado profundo, por outro lado, podem aprender a extrair essas variáveis de dados brutos, sem a necessidade de engenharia de variáveis explícita (GOODFELLOW; BENGIO; COURVILLE, 2016; LECUN; BENGIO; HINTON, 2015).

Houve vários avanços significativos no aprendizado profundo na última década, incluindo o desenvolvimento de redes neurais convolucionais (CNNs) para reconhecimento de imagem (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), RNNs para processamento de dados sequenciais (GRAVES; SCHMIDHUBER, 2005) e redes adversárias generativas (GANs) para síntese de imagens e vídeos (GOODFELLOW et al., 2020). Esses modelos alcançaram um alto desempenho em uma ampla gama de *benchmarks* tendo sido implantados em produtos e serviços comerciais.

Para medir o desempenho de um modelo de aprendizado de máquina, incluindo redes neurais, métricas de avaliação são utilizadas. Essas métricas fornecem medidas quantitativas para avaliar a qualidade das previsões feitas pelo modelo em um determinado conjunto de dados. As métricas podem variar de acordo com o tipo de problema a ser resolvido, classificação ou regressão. Para problemas de classificação, as métricas de avaliação amplamente utilizadas são:

precisão, recall, acurácia e F1-score.

- Acurácia: A acurácia mede a taxa de acerto geral das previsões do modelo, calculando a proporção de amostras corretamente previstas, independentemente da classe. Dado que, TP significa verdadeiro positivo, TN verdadeiro negativo, FP falso positivo e FN falso negativo, a acurácia é calculada pela fórmula: $(TP+TN)/(TP+TN+FP+FN)$.
- Precisão: A precisão mede a proporção de amostras positivas corretamente previstas entre todas as amostras previstas como positivas. Ela quantifica a capacidade do modelo de evitar erros de falso positivo. O valor de precisão é calculado pela fórmula: $TP/(TP+FP)$.
- *Recall*: O *recall* calcula a proporção de instâncias positivas corretamente previstas em relação a todas as instâncias positivas reais. Ele reflete a capacidade do modelo de evitar falsos negativos. O valor de *recall* é calculado pela fórmula: $TP/(TP+FN)$.
- F1-score: O F1-score é uma média harmônica de precisão e *recall* e fornece uma medida balanceada entre eles. O F1-score é calculado pela fórmula: $2 \times (Precisão \times Recall)/(Precisão + Recall)$.

Apesar de seus sucessos, o aprendizado profundo ainda enfrenta vários desafios, incluindo a necessidade de grandes quantidades de dados rotulados, o potencial *overfitting* e a dificuldade de interpretar e explicar as representações aprendidas. No entanto, pesquisas em andamento em áreas como aprendizado por transferência, aprendizado não supervisionado e explicabilidade estão abordando esses desafios e abrindo novos caminhos para a inovação em aprendizado profundo (SAMEK; WIEGAND; MÜLLER, 2017).

2.3 LSTM

RNNs são redes neurais usadas para modelar dados de sequência, como áudio, vídeo, texto e séries temporais. A diferença entre MLP e RNN é que a RNN tem um mecanismo de looping onde cada neurônio tem duas entradas. A primeira é recebida dos neurônios da camada anterior e a segunda entrada, chamada de estado oculto, é recebida de eventos anteriores e o estado oculto inicial é inicializado com zero (MEDSKER; JAIN, 2001). O problema com RNNs é que a memória é curta, fazendo com que os neurônios quase se esqueçam dos estados anteriores. Além disso, o gradiente que ajusta os pesos pode ser muito pequeno e os pesos mal podem ser ajustados. Este problema é conhecido como problema de gradiente de fuga (PASCANU; MIKOLOV; BENGIO, 2013).

Para resolver esses problemas, a rede LSTM foi criada. LSTM é um tipo de RNN que aprende a manter apenas informações relevantes para fazer previsões. Essencialmente, ele tem dois estados ocultos, um é a memória de curto prazo (ou estado oculto) que é o mesmo utilizado nas redes RNNs e a memória de longo prazo (ou estado da célula). A célula LSTM tem o que é

chamado de portas para regular o fluxo de informações. As portas podem aprender quais dados da sequência devem ser mantidos e quais não devem e podem ser descartados (HOCHREITER; SCHMIDHUBER, 1997).

A arquitetura da célula LSTM é mostrada na Figura 2. A LSTM possui três operações principais chamadas de portas, onde cada uma é responsável por uma tarefa. A primeira é chamada de "porta do esquecimento" (f_t) usado para descartar informações irrelevantes. O valor de entrada (x_t) e a memória curta (h_{t-1}) são multiplicados por pesos (W_f e U_f) e adicionados algum viés (B_f). Finalmente, uma função sigmoide σ é aplicada para gerar um número entre 0 e 1.

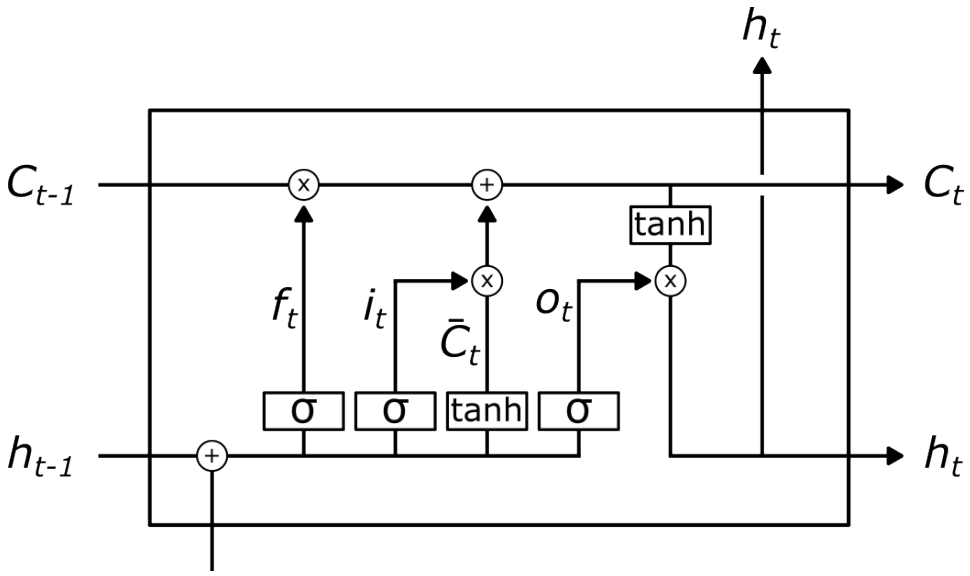

Figura 2 – Arquitetura de uma célula LSTM

A função da porta de esquecimento é definida da seguinte forma:

$$f_t = \sigma(x_t \times U_f + h_{t-1} \times W_f + B_f)$$

A "porta de entrada" i_t decide quais informações devem ser atualizadas no estado da célula. Esta operação multiplica a entrada x_t e o estado oculto h_t pelos respectivos pesos U_i e W_i, adiciona um viés B_i e usa uma função sigmoide para gerar um valor entre 0 e 1. O estado oculto e a entrada também são passados para uma função *tanh* para gerar candidatos $\bar{C}_t$ para o estado da célula. Em seguida, i_t e $\bar{C}_t$ são multiplicados para decidir qual informação do candidato é importante enviar para o estado da célula. Por sua vez, i_t e $\bar{C}_t$ são definidos da seguinte forma:

$$i_t = \sigma(x_t \times U_i + h_{t-1} \times W_i + B_i)$$

$$\bar{C}_t = tanh(x_t \times U_c + h_{t-1} \times W_c + B_c)$$

Nesse momento, a célula LSTM está pronta para atualizar o estado da célula C_t multiplicando o estado anterior da célula C_{t-1} com a saída da porta de esquecimento f_t e então

somando com a multiplicação dos resultados da porta de entrada i_t e os candidatos $\bar{C}_t$. Isso criará um novo estado de célula e enviará as informações para a próxima célula LSTM. Este cálculo é definido da seguinte forma:

$$C_t = C_{t-1} \times f_t + i_t \times \bar{C}_t$$

A última porta é a porta de saída o_t e então é possível calcular o novo estado oculto h_t. O primeiro passo é passar a entrada x_t e o estado oculto anterior h_{t-1} para uma função sigmoide para obter o valor da porta de saída o_t. Em seguida, o estado da célula é passado para uma função *tanh*. Finalmente, esses valores são multiplicados e a saída é o estado oculto. O cálculo da porta de saída e o valor do estado oculto são definidos da seguinte forma:

$$o_t = \sigma(x_t \times U_o + h_{t-1} \times W_o + B_o)$$

$$h_t = o_t \times tanh(C_t)$$

2.4 APRENDIZADO DE MÁQUINA AUTOMATIZADO

Aprendizado de máquina automatizado (AutoML) é o processo de automatizar o design e a aplicação de modelos de aprendizado de máquina. Técnicas de AutoML são usadas para desenvolver e otimizar modelos de aprendizado de máquina, reduzindo a necessidade de intervenção humana. O AutoML tornou-se um importante campo de pesquisa, pois pode melhorar a eficiência e a eficácia dos processos de aprendizado de máquina (WARING; LINDVALL; UMETON, 2020).

A Figura 3 mostra a comparação entre o fluxo de desenvolvimento tradicional de uma aplicação de Machine Learning (a) e o fluxo de desenvolvimento com AutoML (b). O AutoML envolve a automação do processo de aprendizado de máquina, que inclui pré-processamento de dados, engenharia de variáveis, seleção de modelo, ajuste de hiperparâmetros e implantação de modelo. Ele permite que os usuários criem modelos de aprendizado de máquina sem conhecimento profundo dos algoritmos e da matemática relacionada. As plataformas AutoML geralmente incluem vários componentes, como ferramentas de preparação de dados, seleção de modelo e algoritmos de otimização de hiperparâmetros. As plataformas AutoML são projetadas para automatizar todo o *pipeline* de aprendizado de máquina, desde o pré-processamento de dados até a implantação do modelo (HE; ZHAO; CHU, 2021).

Os avanços mais recentes no AutoML se concentraram em melhorar o desempenho dos modelos de aprendizado de máquina e reduzir o tempo e os recursos necessários para treinar modelos. Modelos baseados em redes neurais profundas podem ser otimizados por soluções AutoML. Redes neurais profundas podem aprender padrões complexos e relacionamentos em dados, o que pode levar a um melhor desempenho. AutoML também pode ser utilizado com técnicas de aprendizado por reforço na otimização de hiperparâmetros (DONG et al., 2019).

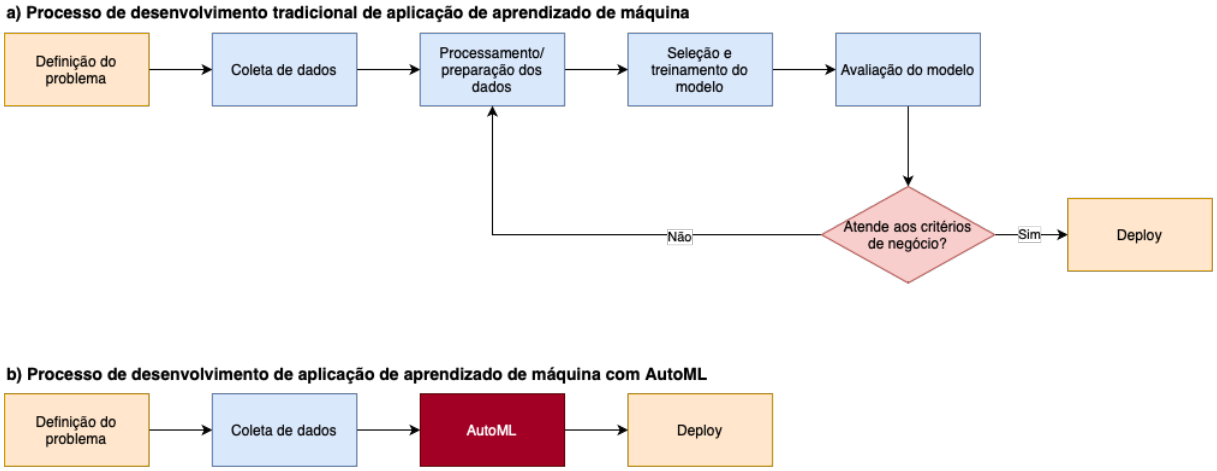

Figura 3 – Comparação entre os fluxos de desenvolvimento tradicional de uma aplicação de Machine Learning (a) e o desenvolvimento com AutoML (b).

O aprendizado por reforço pode aprender com experiências anteriores e melhorar a seleção de hiperparâmetros para modelos de aprendizado de máquina.

Existem vários métodos usados no AutoML, incluindo algoritmos genéticos, otimização bayesiana e otimização baseada em gradiente (LEDELL; POIRIER, 2020). Os algoritmos genéticos envolvem a evolução de populações de soluções candidatas e a seleção da melhor solução com base em pontuações de *fitness*. A otimização bayesiana envolve a construção de um modelo probabilístico da função objetivo e seu uso para guiar a seleção de hiperparâmetros. A otimização baseada em gradiente envolve o uso de gradientes para otimizar hiperparâmetros. Esses métodos são projetados para automatizar o processo de ajuste de hiperparâmetros, que é um componente crítico do aprendizado de máquina.

2.5 ALGORITMO GENÉTICO

No início da década de 1950, pesquisas começaram a desenvolver abordagens de evolução natural inspiradas na ideia darwiniana de sobrevivência do mais apto para problemas de otimização. Notavelmente, Algoritmos Evolutivos (EA) são usados para encontrar a melhor aproximação da solução ótima (YU; GEN, 2010). O algoritmo evolutivo mais comum é o Algoritmo Genético (GONG; YAN; ZUO, 2010).

A Figura 4 mostra as etapas envolvidas no processo do Algoritmo Genético (GA). O GA é uma abordagem meta-heurística inspirada na seleção natural usada para encontrar soluções aproximadas para problemas de otimização combinatória. O GA foi desenvolvido por John Holland na década de 1960 e tornou-se popular em 1975, quando Holland publicou seu livro *Adaptation in Natural and Artificial Systems* (HOLLAND et al., 1992). O GA possui cinco fases: (i) inicialização da população; (ii) cálculo do *fitness*; (iii) seleção; (iv) cruzamento; e (v) mutação.

Primeiramente, na fase inicial da população, o GA gera um conjunto de possíveis

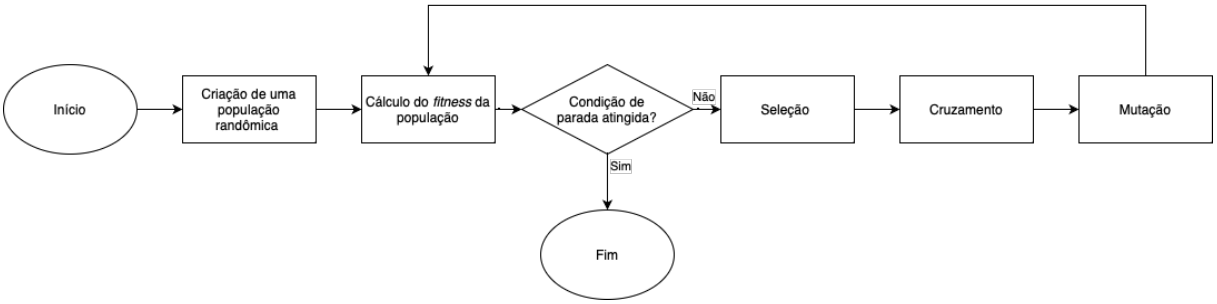

Figura 4 – Etapas do Algoritmo Genético.

soluções aleatórias, chamadas de cromossomos ou indivíduos. Um indivíduo é formado por um conjunto de genes onde cada gene é uma propriedade da solução. Em segundo lugar, o GA avalia um escore para cada indivíduo. Esse escore é chamado de *fitness* sendo calculado através da função *fitness*. Isso determinará o que torna uma solução melhor que a outra. A função *fitness* pode ser usada para maximizar ou minimizar algo de acordo com o problema em questão.

A próxima fase é a seleção onde os indivíduos são escolhidos de uma população. Existem várias abordagens sobre como selecionar indivíduos (JEBARI; MADIAFI, 2013). A abordagem de seleção original é conhecida como Seleção de Roleta (*Roulette Wheel Selection* (RWS) do inglês). No RWS, a chance de um indivíduo ser escolhido para reprodução é proporcional à sua pontuação de *fitness*. Essa abordagem pode ser tendenciosa e, para resolver esse problema, a Seleção Estocástica foi proposta. A Seleção Estocástica obtém uma amostragem das soluções com um único valor aleatório, escolhendo-as em intervalos uniformemente espaçados. Seleção de Torneio seleciona indivíduos aleatórios e compara sua aptidão em um "torneio". O vencedor do torneio é selecionado para reprodução. Por fim, a mais comum é a Seleção de Elitismo, onde são escolhidos os melhores indivíduos.

Uma vez que os pais são escolhidos, eles podem ser combinados. Este processo é conhecido como Cruzamento. Como na fase de seleção, existem muitas estratégias de como cruzar os genitores selecionados. A estratégia original proposta por Holland é chamada *One-point crossover* (ALGORITHMS, 1992) onde uma parte do primeiro genitor é selecionada da posição inicial até um limiar aleatório e os genes restantes são selecionados do segundo genitor, e vice-versa. O cruzamento de dois pontos, utiliza dois limiares aleatórios e os genes entre esses limiares são trocados entre os pais. Finalmente, no cruzamento uniforme, cada gene é escolhido aleatoriamente dos pais.

Após a geração de todos os novos indivíduos, o GA inicia a fase de Mutação. A mutação altera aleatoriamente os genes dos novos indivíduos. A mutação da cadeia de bits é aplicada a genes binários e inverte os valores em posições aleatórias. A mutação gaussiana substitui genes aleatórios de acordo com uma distribuição gaussiana do gene original. A mutação uniforme substitui um gene por um valor aleatório uniforme entre um intervalo.

Finalmente, após a fase de mutação, o GA avalia a nova população e o restante do

processo é repetido até que um critério de parada seja satisfeito. Cada iteração desse *looping* é chamada de geração. Um critério de parada pode ser condicionado ao valor de *fitness* dos indivíduos ou até que o GA atinja o número máximo de gerações. O indivíduo mais apto é a melhor solução.

2.6 NEUROEVOLUÇÃO

A Neuroevolução é um subcampo da inteligência artificial que combina princípios de redes neurais e EAs para otimizar redes neurais artificiais. A ideia por trás da Neuroevolução é usar EAs para evoluir a estrutura e os pesos das redes neurais, que podem ser treinadas para realizar tarefas específicas. Em essência, a Neuroevolução visa automatizar projetos de redes neurais buscando otimizar possíveis arquiteturas e parâmetros (FLOREANO; DÜRR; MATTIUSSI, 2008).

No início da década de 1990, pesquisadores começaram a aplicar EAs para evoluir os pesos de redes neurais como alternativa à retropropagação (DASGUPTA; MCGREGOR, 1992). Para evoluir, não apenas os pesos, mas também a topologia de rede, foi proposta uma das mais conhecidas técnicas de Neuroevolução chamada *NeuroEvolution of Augmenting Topologies* (NEAT) (STANLEY; MIIKKULAINEN, 2002). O NEAT é um algoritmo genético que desenvolve redes neurais adicionando e removendo nós e conexões ao longo do tempo. Mostrou-se eficaz em uma ampla gama de tarefas, incluindo jogos, controle e classificação (PAPAVASILEIOU; CORNELIS; JANSEN, 2021). O algoritmo NEAT possui diversas extensões desenvolvidas para problemas específicos no campo da Neuroevolução como rtNEAT (STANLEY; BRYANT; MIIKKULAINEN, 2005), HyperNEAT (STANLEY; D'AMBROSIO; GAUCI, 2009), cgNEAT (HASTINGS; GUHA; STANLEY, 2009) e odNEAT (SILVA et al., 2015).

Mais recentemente, EAs como Algoritmos Genéticos e a Estratégia Evolutiva Natural (NES) foram aplicados no aprendizado por reforço para evoluir arquiteturas neurais profundas. Os resultados mostram que essas duas estratégias podem superar as redes neurais profundas modernas, como redes Q profundas (DQN) e A3C (SALIMANS et al., 2017; SUCH et al., 2017).

Outro avanço importante na Neuroevolução é o uso de funções de *fitness* projetadas especificamente para redes neurais. As funções de *fitness* podem ser usadas para orientar a evolução das redes neurais, medindo seu desempenho em uma tarefa específica. Por exemplo, uma função de *fitness* pode recompensar redes neurais capazes de classificar imagens corretamente ou capazes de controlar um robô de uma determinada maneira. Ao selecionar as redes neurais mais adequadas, o algoritmo pode desenvolver redes mais complexas e eficazes ao longo do tempo (VERBANCSICS; HARGUESS, 2015).

Existem vários métodos usados em Neuroevolução que envolvem o uso de EAs para treinar e evoluir arquiteturas de redes neurais. Os principais EAs empregados em DNNs, RNNs e CNNs são GAs, Programação Genética (GP) e Estratégias Evolutivas (ES). Outros incluem Evolução Diferencial (DE), Evolução Gramatical (GE) e NEAT. GAs buscam soluções quase

ótimas, enquanto o GP faz programação automatizada. O ES usa mutação como operador principal e o cruzamento como opcional. O DE é eficiente em espaços contínuos e converge mais rápido. GE usa sequência binária para selecionar regras de produção em uma gramática em Backus-Naur. NEAT evolui redes neurais com marcação histórica, especiação e estrutura mínima para encontrar soluções melhores incrementalmente (GALVÁN; MOONEY, 2021).

2.7 MFCCS

MFCC é uma técnica amplamente utilizada para processamento de sinais de voz e áudio. Os MFCCs são uma representação compacta do envelope espectral de um sinal de som, usados como variáveis em muitas tarefas de reconhecimento de fala e classificação de áudio (ZHENG; ZHANG; SONG, 2001).

A técnica MFCC envolve várias etapas, que podem ser resumidas da seguinte forma (SHAFIQUE et al., 2021; HAN et al., 2006):

1. Pré-ênfase: amplifica os componentes de alta frequência do sinal para equilibrar o espectro de frequência.
2. Enquadramento: segmenta o sinal em pequenos quadros para analisar o conteúdo espectral ao longo do tempo.
3. Janelamento: aplica uma função de janela para reduzir o vazamento espectral causado pelo enquadramento.
4. Transformação de Fourier Discreta (DFT): calcula o espectro de potência de cada quadro usando a Transformação de Fourier Rápida (FFT).
5. Banco de filtros da escala Mel: aplica um banco de filtros triangulares ao espectro de potência, que se aproximam da resposta de frequência do sistema auditivo humano.
6. Escala logarítmica: usa o logaritmo das energias do banco de filtros para comprimir a faixa dinâmica.
7. Transformação Discreta do Cosseno (DCT): aplica uma DCT às energias logarítmicas do banco de filtros, que descorrelaciona os coeficientes e enfatiza o envelope espectral.

As etapas necessárias para calcular os MFCCs para obter a representação de vetores utilizados como entrada para os algoritmos de machine learning podem ser vistas na Figura 5. Depois que os MFCCs são calculados, eles podem ser usados como dados de entrada para vários algoritmos de aprendizado de máquina, incluindo aprendizado profundo. Em algoritmos de aprendizado profundo, os MFCCs são frequentemente usados em combinação com outros tipos de variáveis, como espectrogramas (MEGHANANI; ANOOP; RAMAKRISHNAN, 2021) ou formas de onda (JUVELA et al., 2018), para melhorar o desempenho dos modelos que utilizam áudio.

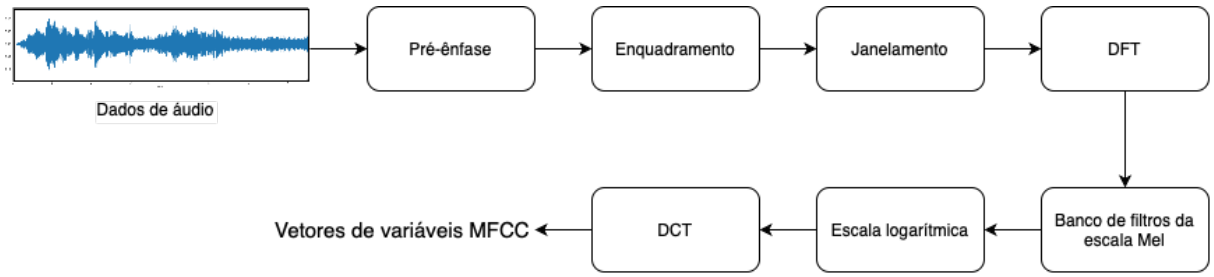

Figura 5 – Etapas para calcular os MFCCs.

Em tarefas de reconhecimento de fala, modelos de aprendizado profundo, como CNNs (CHOWDHURY; ROSS, 2019) e RNNs, demonstraram se beneficiar do uso de MFCCs como variáveis de entrada. Os MFCCs também podem ser combinados com outras variáveis, como valores baseados em fonemas, para melhorar ainda mais o desempenho dos modelos de reconhecimento de fala.

Em tarefas de classificação de áudio, modelos de aprendizado profundo, como redes CNNs e LSTMs, têm sido usados com MFCCs como variáveis de entrada (DENG et al., 2020; SWEDIA et al., 2018; BAE; CHOI; KIM, 2016). Os MFCCs podem capturar o conteúdo espectral do sinal de áudio e podem ser usados para distinguir diferentes tipos de sons, como fala, música e sons ambientais.

Os MFCCs são uma técnica poderosa para processamento de sinais de voz e áudio e podem ser usados como variáveis de entrada em vários algoritmos de aprendizado profundo para reconhecimento de fala e tarefas de classificação de áudio. O uso de MFCCs em combinação com outros tipos de variáveis e modelos de aprendizado profundo pode levar a melhorias significativas no desempenho.

3 REVISÃO DA LITERATURA

Este capítulo apresenta o processo de identificar e selecionar os trabalhos mais adequados relacionados à detecção de enganos de fala usando aprendizado profundo.

3.1 ESTRATÉGIA E CRITÉRIOS DE PESQUISA

A estratégia de busca considerou artigos de congressos publicados e artigos que contenham as seguintes consultas no título, resumo ou palavras-chave:

- “neural network” OU “deep learning” OU “neuroevolution” E;
- (“voice” OU “speech” OU “lexical”) E (“deception” OU “deceit” OU “lie” OU “stress”) E “detection”.

O processo de busca foi realizado no *ACM Digital Library*, *IEEE Xplore*, *Scopus* e *Web of Science corpus*, que são alguns dos maiores indexadores de artigos disponíveis. Além disso, foi utilizada a ferramenta de busca do Google voltada para pesquisa de trabalhos acadêmicos, o Google Acadêmico. Por fim, também foi utilizado o repositório de artigos da *Cornell University*, *arXiv.org*.

A pesquisa de detecção de enganos é um assunto presente em muitos campos, como ciência da computação, psicologia e medicina. A estratégia de busca focou em documentos publicados apenas em ciência da computação e descartou pesquisas de outras áreas. Além disso, alguns critérios foram utilizados, e o processo de seleção dos trabalhos foi o seguinte:

1. Busca de artigos em indexadores;
2. Remoção de duplicados;
3. Remoção de artigos inválidos;
4. Descarte de artigos publicados antes de 2010;
5. Remoção de artigos com as palavras “vídeo”, “image”, “twitter” ou “tweet” no título;
6. Descarte de trabalhos não relacionados à computação;
7. Pesquisas para detectar mentiras usando outras técnicas, ao invés de apenas fala e fala e texto foram rejeitadas;
8. Foram considerados apenas artigos que usam métodos de aprendizado profundo ou rede neural;
9. Problemas de classificação multiclasse foram descartados;

10. Remoção de artigos que não estejam focados apenas em aprendizado profundo e Neuroevolução;

11. Remoção de artigos não relacionados a esta pesquisa.

Na etapa 1, foi feita uma busca nos indexadores utilizando os critérios de busca traduzidos para a linguagem de consulta disponível em cada corpus. Os resultados foram armazenados em arquivos BibTex. A linguagem de programação *Python* foi utilizada em conjunto com o *notebook jupyter*, que é um ambiente de desenvolvimento interativo, para ler o arquivo *BibTex* e realizar o passo 2 até o passo 8.

Artigos com mesmo título e DOI, e mesmo título e ano foram descartados. Após descartar os duplicados, restaram 5 artigos com o mesmo título. Após a abertura dos trabalhos, foi constatado que em todos os casos os artigos estavam duplicados. Por se tratar de trabalhos duplicados, esses estudos foram descartados.

Na etapa 3, foi removido os documentos inválidos, como capas, sumários e livros de conferências. O próximo passo foi descartar artigos com mais de 50 páginas, com o objetivo de remover livros. Em seguida, na etapa 5 foram impressos todos os artigos que continham as palavras “video”, “image”, “twitter” ou “tweet” no título e foi percebido que era possível descartar esses artigos com segurança.

As etapas 6 e 7 foram realizadas simultaneamente. Todos os títulos foram impressos dos trabalhos que restaram no conjunto de dados, os documentos não relacionados à computação foram descartados e também todas as pesquisas que utilizaram outras características para detecção de engano que não a fala e fala e texto. As abordagens que utilizam vídeos como fonte de informação foram excluídos, pois o foco deste estudo está na análise de dados de áudio e texto. Além disso, há uma quantidade limitada de conjunto de dados disponíveis para detecção de engano. Enquanto é possível trabalhar com áudio extraídos de conjuntos de dados em vídeos, não é possível extrair o vídeo de conjuntos de dados que se baseiam apenas em áudio, restringindo assim a comparação entre os trabalhos.

Na etapa 8, todos os artigos que usam métodos de aprendizado profundo ou rede neural para detectar mentiras foram marcados. Esse processo foi realizado por meio da leitura de títulos, resumos e também de uma análise superficial no conteúdo dos documentos.

Da etapa 9 até o final do processo foi realizada pela leitura de todas as obras que sobraram. Assim, os passos restantes foram realizados ao mesmo tempo. Na etapa 9, foi percebido que alguns trabalhos estavam realizando classificação usando mais de duas classes. Essa questão estava relacionada a pesquisas que visavam identificar estresse utilizando conjuntos de dados de emoções e agrupando diferentes emoções como estresse ou não estresse. Com base nesse fato, esses estudos foram desconsiderados, com exceção de um trabalho cujo objetivo é detectar engano utilizando três classes: verdade, meia-verdade e mentira.

Houve alguns trabalhos que não estavam focando em técnicas usando apenas aprendizado profundo. Esses trabalhos se concentraram em testar conjuntos de variáveis e comparar

usando diferentes tipos de métodos, como SVM, Árvores de Decisão, Regressão Logística, etc. Portanto, esses trabalhos foram descartados na etapa 10.

Por fim, na etapa 11 foram identificados que alguns artigos não estavam relacionados esta pesquisa. Nesse sentido, foram identificadas pesquisas que extraem recursos da eletroencefalografia para detecção de emoções, trabalhos que utilizaram apenas a transcrição de áudio como variável e artigos com foco no acento da palavra (pronúncia) e não no estresse emocional.

Os resultados da estratégia e critérios de busca podem ser vistos na Figura 6. Esses números representam os valores consolidados, uma vez que o processo foi executado em 2021 onde foram encontrados 9 trabalhos e novamente em 2023 onde foram incluídos dois trabalhos. Após passar por todas as etapas, foram selecionados os 11 artigos mais relevantes para este trabalho. Os artigos selecionados são mostrados na Tabela 3.1.

Os identificadores com seus respectivos estudos são: S1(NEITERMAN; BITAN; AZARIA, 2020); S2(MARCOLLA; SANTIAGO; DAZZI, 2020); S3(CHOU; LIU; LEE, 2019); S4(FU et al., 2019); S5(KOPEV et al., 2019); S6(YERIGERI; RAGHA, 2019); S7(XIE et al., 2018); S8(HAN; BYUN; KANG, 2018); S9(MENDELS et al., 2017); S10(HILMY et al., 2021); S11(MANSBACH; AZARIA, 2023).

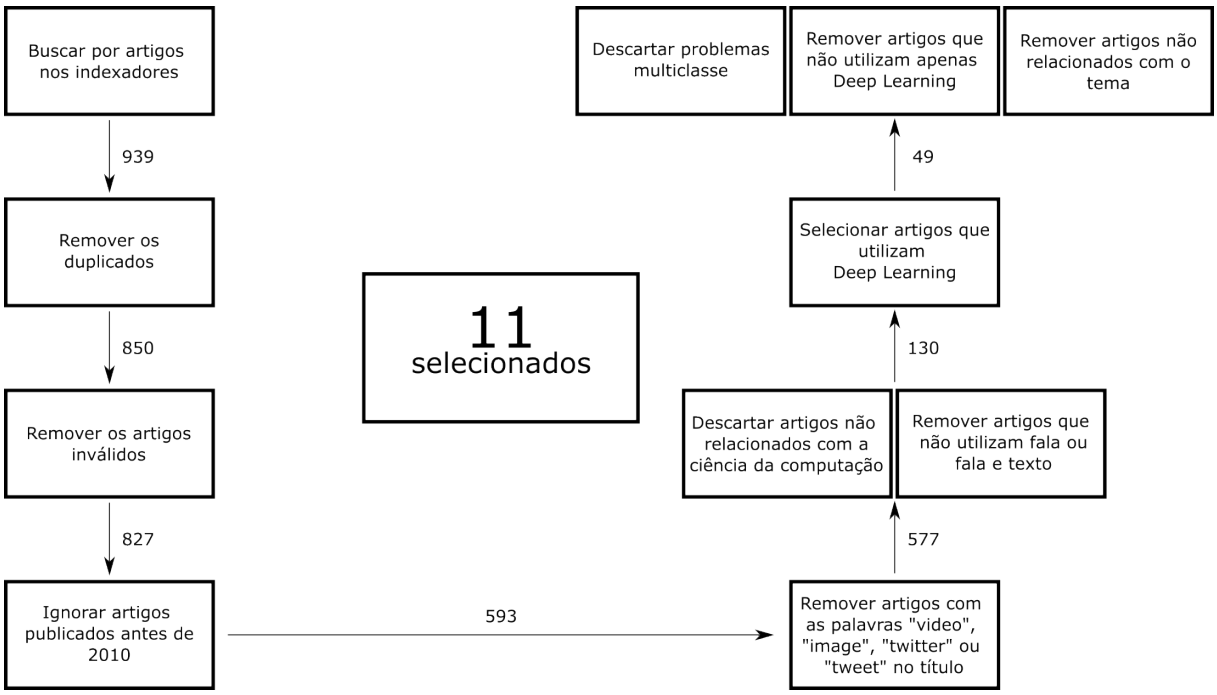

Figura 6 – Etapas de execução da Revisão Sistemática

ID	Ano	Problema	Método	Conjunto de dados	Métrica
S1	2020	Detecção de engano	LSTM	Próprio	59,97% (acc)
S2	2020	Detecção de estresse	LSTM	Próprio	72,5% (acc)
S3	2019	Detecção de engano	BLSTM-DNN	DDDM	74,39% (F1)
S4	2019	Detecção de engano	SS-ANE	CSC Próprio	62,78% (acc) 63,89% (acc)
S5	2019	Detecção de engano	MLP	Próprio	51,04% (acc)
S6	2019	Detecção de estresse	Neuroevolution	SUSAS CMU Mahathi	85,53% (acc) 84,2% (acc)
S7	2018	Detecção de engano	CovBLSTM w/ Skip	CSC Próprio	68,4% (acc) 80,85% (acc)
S8	2018	Detecção de estresse	LSTM	Próprio	65,2% (acc)
S9	2017	Detecção de engano	LSTM + MLP	CXD	63,90% (F1)
S10	2021	Detecção de estresse	CNN	Próprio	61% (acc)
S11	2023	Detecção de engano	Meta-learning	Próprio	38,57% (F1)

Tabela 1 – Trabalhos Selecionados

A maioria dos trabalhos selecionados está usando rede neural recorrente e suas variações. A abordagem baseada em redes neurais LSTM puras foi usada em S1, S2 e S8. As obras S3, S7 e S9 estão usando LSTM em combinação com outras redes ou variações de LSTM. Ainda, S4 propôs um método chamado SS-ANE composto por um codificador e um decodificador. Dois artigos foram incluídos como exceção durante o processo de revisão sistemática que são S5 e S6. O trabalho de S5 propôs uma detecção de engano com três classes: verdade, meia-verdadeiro e mentira. Este artigo foi incluído por estar usando dados reais de debates políticos. Finalmente, o S6, até onde vai o conhecimento dos autores deste trabalho, é a única abordagem para detectar engano, emoção ou estresse usando a técnica de Neuroevolução, embora seja um problema de detecção de estresse multiclasse.

As abordagens atuais para detectar enganos com redes neurais profundas têm resultados promissores. A melhor precisão foi de 80,85% alcançada por uma rede neural profunda usando ConvBiLSTM e BiLSTM com uma conexão de salto (S5). Esses resultados foram obtidos usando um conjunto de dados próprio. Em termos de F1-score, um BiLSTM com mecanismo de *Attention* teve o melhor resultado com 74,39% (S3). Nesse caso, a rede neural profunda foi treinada usando o conjunto de dados DDDM. Além disso, o trabalho utilizando uma abordagem semi-supervisionada apresentou resultados interessantes com 62,78% de acurácia no corpus CSC com apenas 18% de dados rotulados (S4).

Dois trabalhos recentes merecem destaque por abordarem a detecção de estresse e de engano de maneiras diferentes. O primeiro é o S10, que propôs um modelo de detecção de estresse baseado em CNN. O modelo alcançou uma acurácia de 61% na tarefa de classificação de estresse. Embora a acurácia ainda não seja ideal, a abordagem com CNN pode ser uma alternativa promissora para a detecção de estresse em futuras pesquisas. Já o segundo é o S11, que utilizou a técnica de meta-learning para a detecção de engano. O modelo proposto alcançou um F1-score de 38,57% em um conjunto de dados criado pelos autores do trabalho. Embora a

performance não seja melhor do que outras abordagens mencionadas anteriormente, o uso de meta-learning pode ser uma alternativa interessante para o problema de detecção de engano em cenários com poucos dados rotulados.

Há também pesquisas usando diferentes tipos de conjuntos de variáveis, o que pode implicar que não há consenso sobre qual conjunto de variáveis apresenta os melhores resultados. Assim, é difícil determinar se um trabalho pode ser aplicado em cenários do mundo real porque todos esses trabalhos foram realizados com dados sintéticos. Além disso, apesar de alguns trabalhos utilizarem mais de um idioma, não há fortes evidências de que um modelo treinado em um idioma possa ter resultados semelhantes em outros idiomas. Nesse caso, diferenças culturais e múltiplos sotaques podem influenciar os resultados.

3.2 TRABALHOS RELACIONADOS

Com a expansão do poder computacional, novas abordagens surgiram nos últimos anos para detectar enganos. Um dos campos mais promissores é o aprendizado profundo. Nesse contexto, vários trabalhos foram propostos nos últimos anos para prever o engano por meio da fala. O presente capítulo está organizado em seções conforme as abordagens destacadas por este trabalho, que incluem *Multilayer perceptron* (3.2.1), Redes Neurais Convolucionais (3.2.2), *Long short-term memory* (3.2.3), Redes Neurais Híbridas (3.2.4) e Neuroevolução (3.2.5).

3.2.1 Multilayer perceptron

Em seu trabalho (KOPEV et al., 2019) propôs uma arquitetura de rede perceptron de multicamada, treinada usando dados acústicos e textuais. O conjunto de dados foi criado usando debates políticos do mundo real, onde cada afirmação extraída foi rotulada como verdadeira, meia-verdade e falsa. A rotulação foi realizada pelos autores com base em organizações de verificação de fatos. O conjunto de treinamento contém 22 afirmativas verdadeiras, 24 meio verdadeiras e 48 falsas, enquanto o conjunto de teste tem 71 verdadeiras, 39 meio verdadeiras e 82 falsas, indicando um conjunto de dados desbalanceado. O conjunto de variáveis é composto por categorias psicologicamente significativas extraídas com o software LIWC, indicativo de importância de n-gramas das palavras através do TF-IDF (JONES, 1972), e token do BERT-Base (DEVLIN et al., 2018) como variáveis textuais pré-treinadas extraídas da transcrição de áudio. O INTERSPEECH 2013 ComParE (SCHULLER et al., 2013) e o i-vector (ALI et al., 2015) foram usados como variáveis acústicas. Cada tipo de variável foi treinado por uma rede totalmente conectada com uma camada oculta de tamanho 16 e a saída foi combinada em uma camada oculta totalmente conectada de tamanho 32, ambas usando ReLu como função de ativação. O modelo foi treinado por 512 épocas com otimizador *Stochastic Gradient Descent* (SGD), *learning rate* de 0,005, *dropout* de 0,5 e *cross-entropy* como função de perda. O resultado foi de 51,04% de acurácia e 45,07% para F1-score.

3.2.2 Redes Neurais Convolucionais

CNNs são amplamente utilizadas em aplicações de imagem, mas também podem ser utilizadas para processar outros tipos de dados, como séries temporais e sinais de áudio. Essa é a proposta do estudo realizado por (HILMY et al., 2021) cujo objetivo encontrar um método não invasivo para detectar estresse. Os próprios autores do trabalho criaram o conjunto de dados utilizado através de entrevistas com estudantes da Universidade Islâmica Internacional da Malásia. Dos 150 áudios coletados, 112 foram usados como dados de treinamento, enquanto os 38 restantes foram usados como dados de teste. Os dados de teste possuem uma distribuição de 22 amostras rotuladas como estresse e 16 amostras rotuladas como sem estresse. Como variáveis, os autores utilizaram MFCCs, extraídos da fala dos participantes. Redes neurais convolucionais foram utilizadas para classificação de estresse. Contudo, não foram disponibilizados mais detalhes com relação à arquitetura da rede. O modelo proposto conseguiu detectar estresse com 61% de acurácia.

3.2.3 Long short-term memory

LSTM é um tipo de RNN onde cada célula tem a entrada, o estado anterior e a memória. Devido a essa arquitetura, redes LSTM são úteis para processar dados sequenciais. (NEITERMAN; BITAN; AZARIA, 2020) desenvolveu um modelo MLP multilíngue, usando falas em inglês e hebraico. Devido à falta de dados, os autores criaram um conjunto de dados através de um jogo de cartas onde os participantes foram instruídos a blefar para ganhar o jogo. No total, a fase de coleta de dados forneceu 637 amostras rotuladas. Dessas, 395 são declarações verdadeiras e 242 são declarações falsas. Os autores usaram uma camada LSTM com 64 neurônios ocultos e *batch size* de 64 amostras. O processo de treinamento foi feito usando 300 épocas. Os autores também utilizaram uma técnica de validação cruzada com 5 *k-fold*. Os melhores resultados foram alcançados com o conjunto de dados de treino e teste misturando as duas línguas. A acurácia foi de 60% e 46,47% de F1-score.

Marcolla propõe um trabalho para detectar mentiras usando o estresse das falas (MARCOLLA; SANTIAGO; DAZZI, 2020). O conjunto de dados foi coletado por meio de entrevistas na língua portuguesa do Brasil, onde os sujeitos foram orientados a responder perguntas com afirmações verdadeiras e falsas. Foram coletadas 220 amostras sendo 110 rotuladas como verdade e 110 como mentira. Das 220 amostras, 180 foram utilizadas como dados de treino e o restando como teste. Seu método usa voz e as variáveis são um conjunto de características do MFCC extraídas do áudio. O melhor modelo para detectar o estresse na fala foi uma rede LSTM de 3 camadas com 300 neurônios ocultos cada e um *batch size* de 64 amostras. Além disso, a função de perda utilizada foi a *cross-entropy* binária com softmax como função de ativação. O modelo foi treinado usando o otimizador Adam com 150 épocas e 0,01 como *learning rate*. Por fim, o número de MFCCs utilizado foi 13. A acurácia deste modelo foi de 72,5%.

Em seu trabalho, os autores de (HAN; BYUN; KANG, 2018) propuseram uma aborda-

gem baseada em LSTM para detectar estresse através da fala. O conjunto de dados foi coletado por uma simulação de processo de entrevista de emprego onde os participantes foram expostos a situações de estresse e não estresse. Os níveis de estresse foram medidos ao nível de cortisol, que aumenta em situações estressantes. O trabalho relacionado apresenta os tempos de gravação (em horas) para as categorias "Estressado" e "Não estressado" nos conjuntos de treinamento e teste, com 1,60h e 0,28h registradas para "Estressado" no conjunto de treinamento e teste, respectivamente, e 1,52h e 0,28h registradas para "Não estressado" no conjunto de treinamento e teste, respectivamente. A variável utilizada para treinamento foi o coeficiente de banco de filtros na escala Mel com 40 dimensões. Essas características foram passadas para o modelo proposto, que é um LSTM de duas camadas para capturar o valor médio da sequência de saída e a saída do último nível de quadro. Posteriormente, a saída foi passada para uma camada totalmente conectada e depois para o classificador, a fim de determinar se a declaração atual é uma declaração de estresse ou não. Dois classificadores foram testados, uma camada adicional com softmax como função de ativação e um SVM para obter a localização da sentença no espaço vetorial. Em termos de acurácia, o melhor resultado foi obtido pelo modelo LSTM-SVM utilizando o valor médio com 66,4% de acurácia.

3.2.4 Redes Neurais Híbridas

A arquitetura proposta por (CHOU; LIU; LEE, 2019), é um conjunto de redes BiLSTM com camadas densas e um mecanismo de *attention*. Em seu trabalho, os autores usaram o conjunto de dados DDDM (HUANG et al., 2019) e as variáveis extraídas foram F0, energia, volume, MFCC, probabilidade de voz, frequências espectrais de 8 linhas (LSF), taxa de cruzamento no ponto zero, coeficientes de regressão delta, duração, diferença de duração, adição de duração, razão de duração, razão enunciado-duração, razão silêncio-duração, razão silêncio-enunciado, tempo de hesitação, tempos de *backchannel* e tempos de silêncio. O modelo utilizado neste trabalho é composto por cinco camadas. A primeira e a quarta camadas estão totalmente conectadas com 16 unidades ocultas usando ReLU como função de ativação. Essas camadas também têm um *dropout* de 0,5. A segunda camada é um BiLSTM com 8 neurônios ocultos e um *dropout* de 0,5. Em seguida, está a camada *attention* com 0,5 de *dropout*. Finalmente, a última camada é a camada de saída com 1 unidade oculta. Para o treinamento, os autores usaram um *batch size* de 32, *learning rate* de 0,0005 e uma estratégia de interrupção antecipada para interromper o treinamento do modelo quando o desempenho parar de melhorar. Além disso, foi utilizada uma técnica de validação cruzada com um *k-fold* de 10. Foi utilizado o otimizador Adamax e o modelo foi treinado com 300 épocas. Os melhores resultados foram 74,71% de acurácia e 74,39% de F1-score.

O método proposto por (XIE et al., 2018) consiste em ConvBiLSTM, BiLSTM e camadas totalmente conectadas. Dois modelos foram treinados usando um conjunto de dados proposto e o CSC (HIRSCHBERG et al., 2005). O conjunto de dados CSC possui 5412 amostras onde 2209 são rotuladas como mentiras e 3203 como verdade. Os dados de teste consiste

em 542 amostras. O conjunto de dados proposto possui 7864 amostras sendo 3272 rotuladas como mentira e 4592 como verdade. Aproximadamente, 10% das amostras foram utilizadas como teste. O conjunto de variáveis consiste em taxa de cruzamento zero, raiz quadrada média da energia do quadro, F0, centroide espectral, dispersão espectral, entropia espectral, fluxo espectral, *roll-off* espectral, MFCCs, MFCCs delta, MFCCs delta de segunda ordem e coeficientes lineares de previsão. As duas primeiras camadas do modelo são um ConvBiLSTM 60 unidades ocultas e forma de kernel de (60,2,4). As próximas duas camadas são BiLSTM com 1024 e 512 unidades ocultas. Na sequência, uma camada totalmente conectada vem em seguida com 512 células e outra totalmente conectada com 2 células. Após o segundo ConvBiLSTM e o segundo BiLSTM, os autores usaram uma normalização em lote e um *dropout* de 0,6. Uma conexão de salto foi usada após a normalização do primeiro lote e o *dropout* através da primeira camada totalmente conectada. O treinamento foi realizado em 60.000 etapas com *learning* rate de 0,0001, *batch size* de 128 e recorte de 32. O modelo obteve 80,85% de acurácia no conjunto de dados da pesquisa e 68,4% no CSC.

Transcrever o áudio em texto, extrair recursos textuais e combinar recursos acústicos e de texto. Essa é a proposta do (MENDELS et al., 2017). O conjunto de dados utilizado nesse trabalho foi o CXD Corpus (LEVITAN et al., 2015). Os autores realizaram uma segmentação das amostras do conjunto de dados utilizando pausas nas falas de pelo menos 50ms. Com isso, 76%, 4% e 20% foram os percentuais de amostras utilizadas nos conjuntos de treino, validação e teste, respectivamente. O conjunto de variáveis foi composto pelo descritor de baixo nível (LLD) do conjunto de recursos básicos do ComParE Challenge (SCHULLER et al., 2013), como *pitch*, F0, energia, espectral, MFCC, duração, qualidade de voz, harmonia espectral e nitidez espectral psicoacústica. Além disso, os MFCCs foram extraídos gerando 13 coeficientes cepstral por cada janela de 256 *frames* com um passo de 100 *frames*. As variáveis textuais utilizados neste trabalho foram unigramas, bigramas e trigramas entre discursos enganosos e verdadeiros. Assim, *Word Embeddings* (WE) foram extraídos de transcrições de áudio usando vetores de palavras pré-treinados GloVe (PENNINGTON; SOCHER; MANNING, 2014). O modelo com melhor desempenho foi uma combinação de 1) BiLSTM com 256 unidades ocultas e um softmax como função de ativação, usando os *embeddings* de palavras pré-treinados como variáveis e 2) MLP com 6 camadas e 1095 unidades ocultas usando o conjunto de variáveis do desafio emocional Interspeech 2009 (IS09). A função de ativação foi ReLu e uma normalização em lote foi aplicada. Também foi utilizado um *dropout* de 0,497 com regularização L2 de 0,2. Finalmente, o MLP foi treinado usando o otimizador SGD e *learning rate* de 0,00134. Para equilibrar os pesos das duas redes, os autores aplicaram uma camada auxiliar de previsão softmax ao BiLSTM. O F1-score foi de 63,90%.

Um dos principais problemas para detectar enganos é a falta de dados. Para resolver este problema, (FU et al., 2019) propôs uma abordagem usando uma técnica semi-supervisionada composta por um codificador, um decodificador e um classificador. Em seu trabalho, os autores usaram um conjunto de dados proposto e o corpus CSC. O conjunto de variáveis é composto por 16×2 LLDs que consistem em ZCR, harmônicos para ruído (HNR),

MFCCs, raiz quadrada média (RMS), e 12 funções de descrição com média, máxima e mínima, erro quadrático médio, entre outras. Portanto, o vetor de variáveis total por bloco contém $16 \times 2 \times 12 = 384$ atributos. A rede proposta é um autoencoder de ruído aditivo semi-supervisionado (SS-ANE). O modelo é composto por um codificador para extrair características de alta ordem dos dados originais com um conjunto de camadas contendo pesos, processo de normalização em lote para acelerar o treinamento, Elu como função de ativação, que é uma variação da função ReLu, e um decodificador para mapear as características para a saída para reconstruir os dados e classificar as informações em recursos codificados. A acurácia alcançada foi de 61,81% e 63,89% no conjunto de dados proposto, e 59,52% e 62,78% no corpus CSC, com 500 e 1000 exemplos rotulados, respectivamente.

O estudo realizado por (MANSBACH; AZARIA, 2023) propõe o uso de meta-aprendizagem para detectar mentiras em discursos. Os dados foram coletados pelos autores do trabalho através de um jogo de cartas de turno chamado de "*Cheat-Game*" onde os jogadores foram instruídos a mentir para ganhar o jogo. Ao todo foram coletadas 10788 amostras sendo 70,3% classificadas como verdade e 29.7% como mentira. O modelo proposto chama-se CHAML e usa meta-aprendizagem para detecção de engano de poucas amostras com base na fala. Cada pessoa é definida como uma "tarefa de meta-aprendizagem" e seu conjunto relacionado contém quatro exemplos aleatórios, dois por classe, para treinamento. O classificador consiste em três camadas totalmente conectadas com função de ativação ReLU e uma camada de saída softmax com duas classes. Os resultados obtidos com o método proposto mostram uma acurácia de 61,34% e um F1-score de 0,3857.

3.2.5 Neuroevolução

Os autores de (YERIGERI; RAGHA, 2019) propuseram um classificador de rede neural de reconhecimento de emoção com otimização de peso, combinando algoritmos de otimização como BAT, algoritmo genético, organização de enxame de partículas e recozimento simulado. A ideia chave desta abordagem é substituir os algoritmos baseados em gradiente como retropropagação por uma abordagem de otimização para encontrar os pesos da rede. O conjunto de variáveis usado neste trabalho foi *Gammatone Wavelet Cepstral Coefficient* GWCC, MFCC, tom e energia. A rede neural utilizada neste trabalho é uma MLP com 2 camadas e 2 unidades ocultas em cada camada. O *learning rate* é de 0,001 e a rede foi treinada por 300 épocas. Diversas combinações de algoritmos de otimização foram utilizadas, e os melhores resultados foram alcançados por BAT + Optimização por enxame de partículas (PSO). A acurácia foi de 85,53% no banco de dados SUSAS (HANSEN; BOU-GHAZALE, 1997) e 84,2% no CMU Marathi (WAGHMARE et al., 2014a; WAGHMARE et al., 2014b). Vale destacar que, apesar do método utilizar Neuroevolução, a abordagem visa otimizar os pesos da rede neural, diferente deste trabalho atual cujo objetivo é otimizar a topologia da rede.

A diversidade de métodos utilizados inclui LSTM, BiLSTM, SS-ANE, MLP, Neuroevolução, ConvBiLSTM, CNN e meta-aprendizado. As performances variam significativamente

entre os métodos, com as melhores performances alcançando cerca de 85% de acurácia (acc) e 74,39% de F1-score (F1) na detecção de engano. Além disso, não há um consenso sobre qual conjunto de dados utilizar, sugerindo que a escolha do conjunto de dados pode ter influência nas performances obtidas.

4 MÉTODO DE NEUROEVOLUÇÃO PARA REDES LSTM

O método proposto por este trabalho, ilustrado na Figura 7, consiste em dois módulos. O primeiro é responsável pelo treinamento da rede neural, que recebe como entrada uma configuração de topologia de LSTM e realiza o treinamento para detectar enganos através da fala, retornando o valor de *fitness*. O segundo módulo é o algoritmo genético, utilizado para gerar e evoluir arquiteturas de redes LSTM observando o valor de *fitness* das redes treinadas em relação ao problema de detecção de engano através da fala.

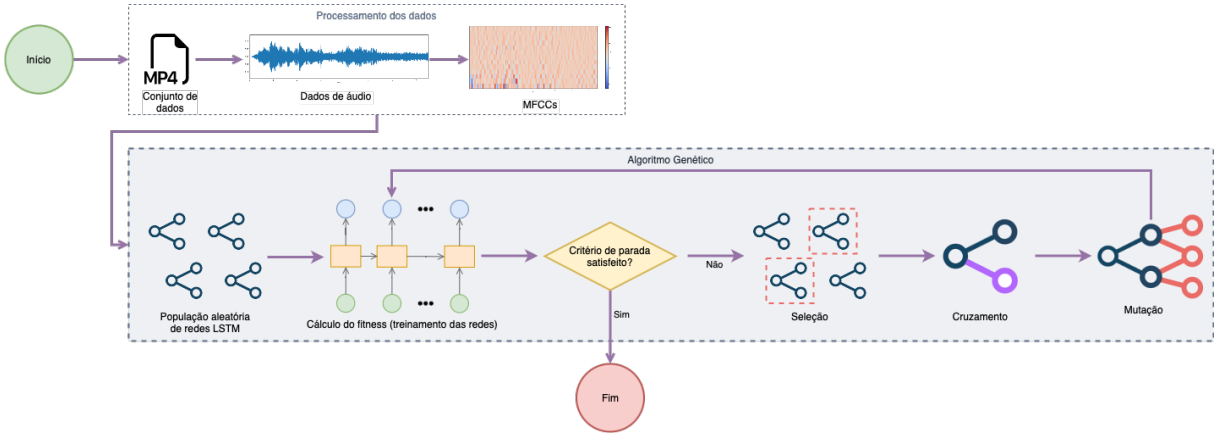

Figura 7 – O processo de uso de aprendizado de máquina automático para otimizar redes LSTM para detecção de engano utilizando a fala é retratado. A estrutura é separada em duas seções: processamento de dados e algoritmo genético.

4.1 CONJUNTO DE DADOS UTILIZADO E PREPARAÇÃO

O conjunto de dados utilizado em nosso estudo é o "*Bag-of-Lies*" (GUPTA et al., 2019), que é coletado em um cenário que espelha situações da vida real. Ele é composto por 35 participantes distintos que forneceram 325 declarações, com um equilíbrio entre declarações verdadeiras (163) e falsas (162). O conjunto de dados também examina as vantagens da incorporação de diversas formas de análise, como áudio, vídeo, sinais fisiológicos e outras, para aprimorar a precisão e robustez dos modelos de detecção de mentiras. O objetivo é utilizar um conjunto de dados que possa ser utilizado para treinar e avaliar modelos de detecção de mentiras mais relevantes para cenários do mundo real.

Optou-se por utilizar esse conjunto de dados devido às restrições de acesso a outras fontes de dados. Diversos conjuntos de dados disponíveis exigiam acesso exclusivo por parte das instituições acadêmicas responsáveis por sua criação, limitando assim a generalização dos resultados. Ademais, certos conjuntos de dados impunham um ônus financeiro considerável para sua obtenção, o que se mostrava inviável para o escopo do estudo em questão. Portanto, a seleção do conjunto de dados "*Bag-of-Lies*" foi motivada por sua adequação como alternativa viável. Embora outros trabalhos utilizam esse conjunto de dados, elas se concentram na detec-

ção de enganos em vídeos, diferindo do enfoque deste estudo. Por esse motivo, não há nenhum trabalho relacionado que utiliza esse conjunto de dados.

Na abordagem adotada, a fase inicial consiste em realizar o pré-processamento dos dados no conjunto de dados. Para separar o áudio do vídeo, é utilizada a biblioteca *Python MoviePy* para edição de vídeo. O processo inclui a definição do vídeo de entrada e do formato do arquivo de áudio de saída desejado. Optou-se pelo formato pcm_s16le, que é um padrão amplamente aceito e oferece alta qualidade. O *MoviePy* lê o arquivo de vídeo e salva o áudio no arquivo de saída especificado. Esse processo é automatizado para todo o conjunto de dados.

Após extrair o áudio dos vídeos, são extraídos os MFCCs dos arquivos de áudio. Os MFCCs são um conjunto de características que representam as características espectrais do sinal de áudio. Eles são comumente usados em tarefas de processamento de fala e áudio, como reconhecimento de fala e verificação de alto-falante. Para extrair os MFCCs, é utilizada a biblioteca *Python Librosa* para análise de áudio. Esse processo recebe os arquivos de áudio como entrada, realiza uma série de etapas de processamento de sinal, como janelamento, transformada de Fourier e filtragem de frequência de Mel, e produz os MFCCs como um vetor de características.

Após extrair os MFCCs, os dados são divididos em conjuntos de treinamento e teste. Essa é uma etapa importante na avaliação da performance do modelo. É utilizada uma proporção de 80/20 para a divisão treinamento/teste, em que 80% dos dados são utilizados para treinamento e os 20% restantes são usados para teste. Isso permite avaliar o desempenho do modelo em dados não vistos anteriormente.

Após dividir os dados, eles são normalizados para garantir que todas as características estejam na mesma escala. A normalização dos dados ajuda a evitar que qualquer característica domine o processo de tomada de decisão do modelo, seja por valores muito grandes ou pequenos. Foi utilizada a normalização min-max para normalizar os dados.

4.2 PERSISTÊNCIA DOS DADOS

O projeto possui um módulo para persistir os dados responsável por salvar e carregar as informações relacionadas as execuções. Essas informações incluem os conjunto de dados de treino e teste, metadados da execução corrente e os resultados dos treinamentos das redes neurais. As informações são armazenadas no disco e uma cópia é armazenada em um *bucket* no serviço *Amazon Simple Storage Service* (Amazon S3) da nuvem da *Amazon Web Services* (AWS).

Antes de realizar a preparação do conjunto de dados, o módulo de persistência dos dados irá verificar se existem conjunto de dados salvos no disco. Esses conjuntos de dados são armazenados no formato npy que são arquivos criados pela biblioteca *NumPy* do *Python* e que armazenam tensores utilizadas no projeto. Se os conjunto de dados existirem, esse processo irá carregar os seguintes arquivos:

- x_train: Conjunto de dados utilizado para treinamento
- y_train: A coluna alvo de todos os dados presentes em x_train
- x_test: Conjunto de dados utilizado para teste
- y_test: A coluna alvo de todos os dados presentes em x_test

Após o processo de preparação dos dados, o algoritmo genético é executado e o primeiro no passo o módulo de persistência dos dados irá verificar se existe alguma execução que não foi concluída. Caso positivo, os metadados da execução não concluída serão carregados e o processo será retomado do ponto em que foi interrompido.

4.3 CONFIGURAÇÕES PRELIMINARES DAS REDES NEURAIS

O módulo da rede neural é utilizado para criação e treinamento da rede neural. Uma configuração de topologia da rede é esperada pelo módulo, para ser utilizado para a construção da rede e posteriormente o treinamento. Isso permite que as soluções encontradas pelo algoritmo genético sejam enviadas para o módulo de rede neural. As configurações aceitas pelo módulo são o número de camadas e número de unidades por camada. O *framework* utilizado para a criação e treinamento da rede é o Tensorflow através do módulo Keras.

A Figura 8 representa uma rede LSTM gerada pelo algoritmo. A rede é construída utilizando camadas LSTM onde a saída da primeira camada até a penúltima retorna a sequência completa para a camada subsequente. A saída da última camada LSTM é um único vetor, pois é o formato esperado pela camada densa ou camada de saída. A única camada da topologia, que é fixa é a camada de saída. Essa camada possui um único neurônio que utiliza a sigmoide como função de ativação. Dessa forma, a predição é classificada como verdade quando o valor de saída da rede é maior que $0,5$, do contrário é mentira.

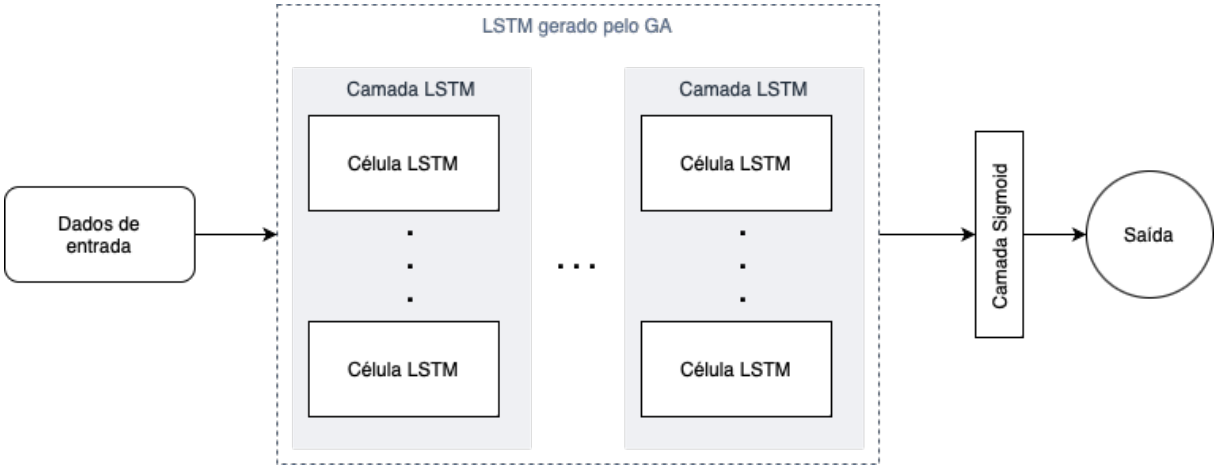

Figura 8 – Representação de uma rede LSTM. Os dados de entrada são introduzidos nas camadas LSTM geradas pelo algoritmo genético. A camada de saída é fixa e determina o valor de saída da rede

4.3.1 Otimizador da rede

A rede neural utiliza o otimizador Adam. A principal vantagem de usar o Adam é que ele é um algoritmo adaptativo, o que significa que ele ajusta a taxa de aprendizagem dinamicamente durante o processo de treinamento, permitindo uma melhor convergência e tempo de treinamento mais rápido. Além disso, o Adam é computacionalmente eficiente, o que significa que requer menos memória e processamento em comparação com outros algoritmos de otimização. Outro aspecto positivo, é que ele também pode reduzir a velocidade de treinamento quando necessário, a fim de realizar uma busca mais completa do espaço de parâmetros, levando a um melhor desempenho (KINGMA; BA, 2014). No geral, o otimizador Adam é uma ótima escolha para redes neurais, pois não apenas melhora o tempo de treinamento, mas também aumenta as chances de encontrar o mínimo global.

4.3.2 Função de perda

A função de perda *binary crossentropy* é utilizada neste trabalho por ser uma função amplamente adotada e altamente eficaz para estimativa de probabilidade. Uma das principais vantagens de usar a *binary crossentropy* é que ela é adequada para problemas de classificação binária, em que a saída do modelo é uma probabilidade de uma instância pertencer a uma determinada classe. Além disso, é uma função de perda amplamente utilizada no campo de aprendizagem profunda, e tem se mostrado muito eficaz em muitas aplicações. A função é computacionalmente eficiente e pode fornecer gradientes precisos para retropropagação, permitindo um treinamento mais rápido e eficiente do modelo (RAMOS et al., 2018).

4.3.3 Learning rate

Mesmo utilizando o método de otimização Adam, o *learning rate* é um hiperparâmetro e precisa ser ajustado, pois a queda do valor de *learning rate* geralmente funciona melhor do que não fazê-lo. Durante a fase dos experimentos diversos parâmetros foram testados e essas configurações foram escolhidas por permitem com que o algoritmo genético busque a melhor configuração conforme as demais configurações da rede. Um dos parâmetros avaliados foi o *learning rate*. Ao permitir que esse parâmetro fosse otimizado pelo algoritmo genético, notou-se que o algoritmo genético estava demorando muito tempo para encontrar a melhor solução possível e nem sempre as soluções eram boas.

Neste trabalho, *Cyclical Learning Rate* (CLR) foi utilizado como método para encontrar automaticamente o valor ótimo do *learning rate* para otimizadores baseados em gradiente (SMITH, 2017). CLR é uma técnica amplamente utilizada que permite que o *learning rate* varie ciclicamente entre um limite inferior e superior durante o processo de treinamento, o que pode auxiliar o modelo a explorar uma ampla gama de valores de *learning rate* e convergir mais rapidamente e melhor. Embora a implementação de CLR seja bem conhecida e amplamente uti-

lizada no *framework fast.ai*, ela não está disponível nativamente no Tensorflow. Portanto, este trabalho realizou a adaptação do algoritmo com base no código de (Bering Limited, 2020) para atender aos requisitos específicos deste projeto. Nos experimentos, as configurações utilizadas para encontrar um bom valor de *learning rate* foram o otimizador Adam com um intervalo de taxa de aprendizado entre 0,0001 e 1, 5 épocas e *batch size* de 260. Os valores de *learning rate* podem ser considerados baixos ou altos de acordo com o modelo. Esse intervalo foi escolhido observando estratégias utilizadas ao desenvolver um modelo através de tentativa e erro (SMITH, 2018). A quantidade de épocas foi escolhida através de observações onde foram avaliados intervalos de 2-10. O *batch size* para encontrar o *learning rate* foi o mesmo utilizado pelo treinamento das redes, conforme descrito adiante. O método para encontrar o *learning rate* retorna uma lista de valores testados e as respectivas perdas associadas aos valores. Para esse projeto o objetivo é utilizar o *learning rate* associado ao menor valor de perda.

4.3.4 Batch size

Neste trabalho, um *batch size* de 260 foi utilizado tanto para encontrar um *learning rate* ótimo quanto para treinar a rede neural. Esse valor foi determinado por meio de uma série de experimentos e testes. Primeiramente, um conjunto de topologias aleatórias de rede neural foi gerado. Em seguida, essas redes foram treinadas com um *batch size* crescente, iniciando em 1 e terminando em 260 (que é o número de amostras no conjunto de dados de treinamento), com um incremento de 5 em cada treinamento. O desempenho da rede foi avaliado analisando os gráficos de perda em cada época. Os resultados mostraram que a rede geralmente teve um desempenho melhor com um batch size de 260, independentemente da topologia. No entanto, é importante notar que isso não é uma garantia e que não há uma maneira automatizada, até onde vai o conhecimento do autor desse trabalho, de encontrar o melhor *batch size* para cada topologia, ou um valor *batch size* ótimo para todas as topologias.

4.3.5 Avaliação de performance

O desempenho da rede é avaliado usando a métrica F1-score macro, que fornece uma avaliação equilibrada e abrangente do desempenho do modelo. O F1-score macro é uma média do F1-score de cada classe. Por sua vez, o F1-score é uma média harmônica de precisão e *recall*, o que garante que ambas as métricas tenham o mesmo peso e contribuição para a avaliação geral. Um valor mais alto de F1-score indica que o modelo tem um melhor equilíbrio de precisão e *recall*, o que significa que ele tem menos falsos positivos e falsos negativos. A precisão mede a proporção de instâncias positivas corretamente identificadas, enquanto o *recall* mede a proporção de instâncias positivas reais que foram corretamente identificadas pelo modelo.

Além disso, a escolha de utilizar o F1-score como métrica de avaliação de desempenho é vantajosa por ser uma métrica robusta, independentemente se o conjunto de dados está balanceado ou não. Enquanto a acurácia é uma métrica boa para conjuntos de dados com classes

balanceadas, ela pode ser enganosa quando as classes estão desbalanceadas. Por outro lado, o F1-score considera tanto a precisão quanto o recall, o que o torna uma medida mais confiável para problemas em que as classes têm tamanhos diferentes. Assim, A métrica F1-score permite uma avaliação precisa e holística do desempenho da rede, fornecendo *insights* valiosos sobre sua capacidade de classificar corretamente as instâncias, independentemente do conjunto de dados e a sua distribuição das classes.

4.4 ALGORITMO GENÉTICO

O módulo do algoritmo genético executa o processo para encontrar as melhores topologias de redes para o problema de detecção de enganos através da fala. Os parâmetros de entrada do módulo são a função que calcula o *fitness*, a quantidade de gerações, o tamanho da população, a taxa de mutação, quantidade máxima camadas da rede e quantidade mínima e máxima de unidades por camada.

A Figura 9 descreve a representação da população, os indivíduos e os genes. A população consiste em um conjunto de topologias de redes LSTM, em que cada indivíduo representa uma topologia específica. O tamanho de cada indivíduo indica o número de camadas da rede LSTM. Por sua vez, cada gene do indivíduo indica o número de unidades por camada. Por exemplo, considerando o primeiro indivíduo da Figura $I = [4, 8, 7]$, isso representa uma rede LSTM com três camadas, em que a primeira camada possui 12 unidades, a segunda camada possui 4 unidades e a terceira camada possui 16 unidades.

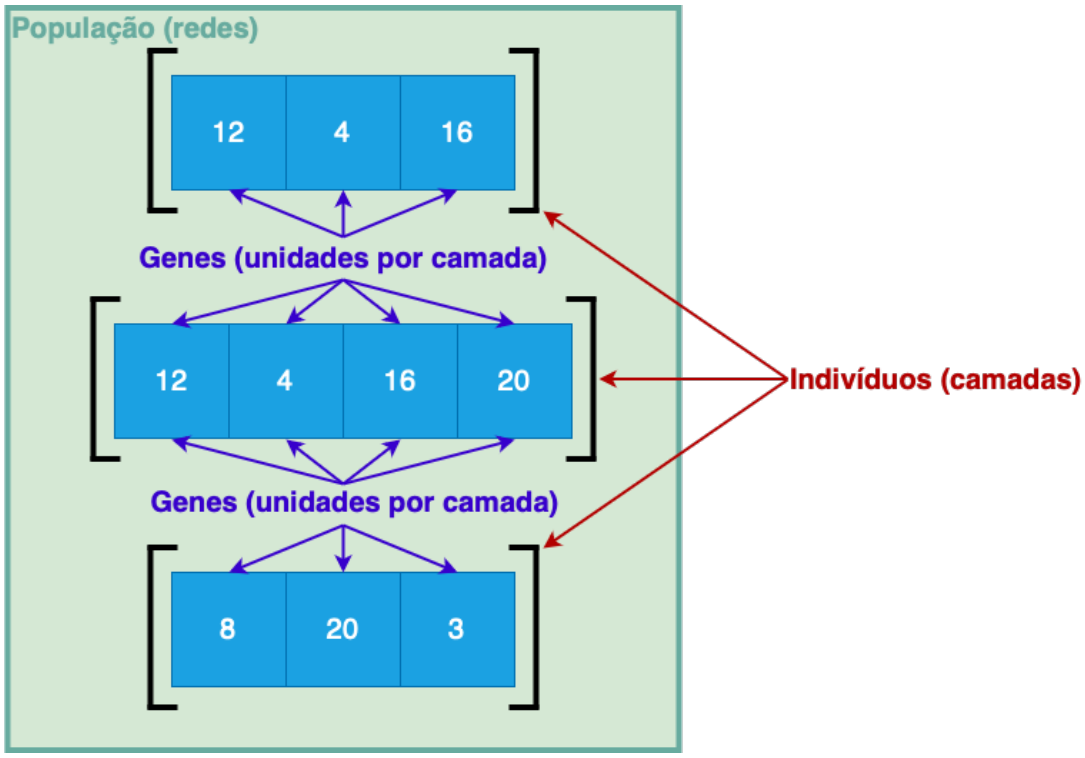

Figura 9 – Representação da população, os indivíduos e os genes.

A primeira tarefa do módulo consiste em verificar se existe alguma execução que não foi finalizada. Todas as execuções do sistema proposto realizam o salvamento das informações da execução atual ao final de cada geração. Através desses metadados, o algoritmo consegue verificar se existe alguma execução pendente. Caso positivo, o módulo irá retomar a execução a partir da geração em que a execução anterior parou e utilizando os mesmos indivíduos. Se não houver meta dados de execuções anteriores, ou se a última execução estiver com status de completado, uma nova execução é iniciada.

O algoritmo 1 mostra o pseudocódigo da função *ga* do módulo do algoritmo evolutivo, responsável por executar o algoritmo genético para encontrar a melhor solução possível para o problema proposto. Primeiramente, a função cria uma nova população. Na sequência o algoritmo irá iterar sobre as gerações. Em cada geração a função irá calcular o *fitness* de cada indivíduo através da função de *fitness*. A chamada dessa função é executada em paralelo conforme a quantidade de CPUs disponíveis na máquina que está executando a solução.

Algorithm 1: Função que executa o algoritmo genético do processo evolutivo.

```
1  Function ga(pop_size, num_generation, num_best: int, mut_rate: float,
     fitnessFn: callable) : int
2      P ← initPop(pop_size);
3      for i ← 0 to num_generation do
4          P ← fitnessFn(P);
5          P ← sortByFitness(P);
6          if best is null OR P[0].score > best.score then
7              best ← P[0]
8          N ← ∅;
9          for _ ← 0 to ⌊(pop_size − num_best)/4⌋ do
10             while C is null OR C ∩ (P[0..num_best] ∪ N) ≠ ∅ do
11                 p1, p2 = selection(P[num_best..P.length]);
12                 C ← crossover(p1, p2)
13             for c ∈ C do
14                 m ← mutation(c, mut_rate);
15                 N ← N ∪ m
16         P ← P[0..num_best] ∪ N
17     return best
```

Uma vez que o *fitness* de todos os indivíduos foram calculados, o algoritmo genético ordena a lista pelo valor de *fitness* de forma reversa. O próximo passo é verificar se o primeiro indivíduo possui a melhor performance até então. Caso positivo, o mesmo será armazenado. Caso contrário, o melhor indivíduo continua sendo de alguma execução anterior.

4.4.1 Fitness e função de fitness

O *fitness* é o escore de cada indivíduo. Nesse trabalho o valor utilizado como valor de *fitness* é o resultado da função linear representado pela equação 4.1. Quanto maior o valor de *fitness* melhor é o indivíduo. A variável x_0 representa o valor negativo do cálculo da diferença absoluta entre às perdas de treino e validação ao final do treinamento da rede. É utilizado o valor negativo das diferenças das perdas, pois o objetivo é minimizar esse valor, então quanto menor é a diferença, melhor é a solução. Por sua vez, x_1 é o valor de F1-score que calculado através das predições realizadas pelo modelo no conjunto de dados de teste. β_0 e β_1 representa os pesos aplicados a x_0 e x_1, respectivamente. Os pesos são definidos de forma fixa antes da execução do algoritmo. Consequentemente, quanto menor é a diferença entre as perdas de treino e validação e maior é o F1-score da rede, melhor é considerada a solução.

$$y = \beta_0 x_0 + \beta_1 x_1 \tag{4.1}$$

Para calcular o valor de *fitness* dos indivíduos, a função de *fitness* recebe um indivíduo da população como parâmetro e retorna o F1-score da rede, que no caso é o valor de *fitness*. O primeiro passo da função é verificar se já foi realizado algum treinamento utilizando a topologia da rede de questão, representada pelo indivíduo. Caso positivo, o valor de F1-score é retornado pelo função *fitness*. Caso contrário, as informações do indivíduo são passadas para o módulo de rede neural e a rede é treinada. Após o treinamento, o módulo de rede neural irá retornar o valor de F1-score, esse valor será então armazenado para não ser necessário treinar novamente a mesma rede, caso houver a necessidade. Finalmente a função de *fitness* retorna o valor do F1-score.

4.4.2 Seleção

O algoritmo executa o processo de seleção (linha 11) múltiplas vezes conforme a quantidade de indivíduos na população, menos o número dos melhores indivíduos que serão levados para a próxima geração. Esse valor então é divido e arredondado para baixo por 4, que é a quantidade de filhos gerados pelo processo de cruzamento (linha 12). Por exemplo, em uma população de 60 indivíduos, onde 4 é o *num_best*, os processos de seleção e cruzamento serão executados no mínimo 14 vezes. Esses mesmos processos ainda podem ser executados mais vezes por conta da validação da linha 10 do algoritmo, que descarta o resultado do cruzamento, caso algum dos indivíduos gerados já existam na lista que contém os novos indivíduos da próxima geração (N).

O processo de seleção consiste em selecionar dois indivíduos para que possam gerar novas soluções. O método para realizar a seleção é o método da roleta. A seleção pelo método da roleta realiza uma soma do *fitness* de todos os indivíduos. Na sequência, para obter as probabilidades de seleção, o valor de *fitness* de cada indivíduo dividido pela soma total de *fitness*.

Finalmente, dois indivíduos da população são selecionados de forma randômica utilizando as probabilidades como peso para a seleção. Outra característica é não haver repetição, ou seja, os dois indivíduos selecionados não podem ser iguais.

Para decidir usar o método da roleta, foram realizados experimentos preliminares. Neles, diferentes métodos de seleção foram testados e os aspectos considerados para a escolha do método da roleta foram a qualidade das soluções, a quantidade de gerações necessárias para obter as melhores soluções e também a diversidade de indivíduos para as próximas gerações. No contexto de diversidade populacional, quanto menos repetições de indivíduos, mais soluções diferentes são testadas e é maior a chance de encontrar soluções melhores.

4.4.3 Cruzamento

O próximo passo é realizar o cruzamento dos dois indivíduos selecionados no método de seleção, conforme mostrado na Figura 10. Esse processo utiliza uma parte de cada indivíduo para gerar quatro novas soluções. Ambos os indivíduos $p1$ e $p2$ são divididos exatamente na metade e o resultado do cruzamento são quatro novos indivíduos $c1$, $c2$, $c3$ e $c4$.

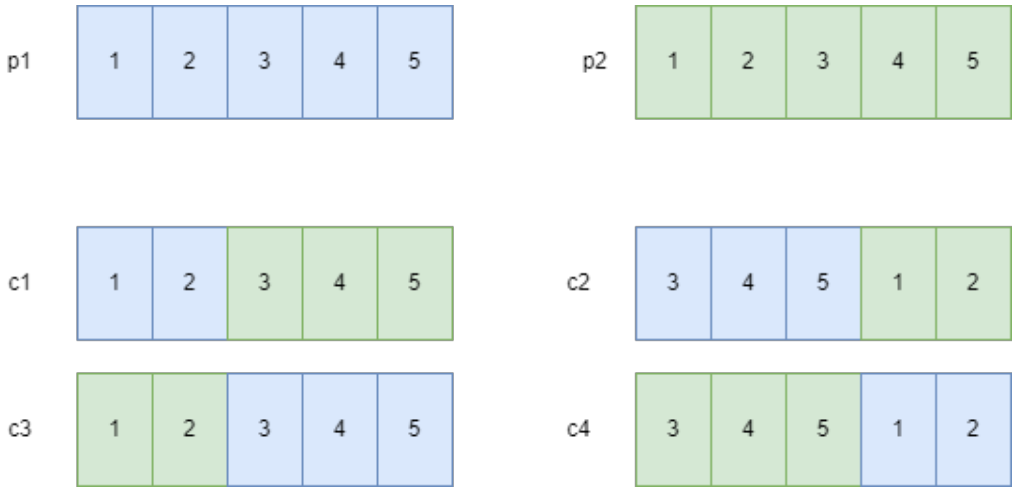

Figura 10 – Ilustração do processo de cruzamento para obter novas soluções.

Os indivíduos resultantes do processo de cruzamento são selecionados para serem utilizados na próxima geração. Durante os experimentos foi observado que o processo de cruzamento estava gerando um grande números de indivíduos iguais após algumas gerações. Para evitar esse tipo de problema foi utilizado um mecanismo que verifica se $c1$, $c2$, $c3$ e $c4$ estão na lista P, que armazena os indivíduos para a próxima geração. Caso positivo, o algoritmo irá realizar novamente os processos de seleção selecionando assim diferentes $p1$ e $p2$, e cruzamento até obter indivíduos únicos em relação a P.

4.4.4 Mutação

O processo de mutação é realizado para os indivíduos resultantes do cruzamento. Tradicionalmente, esse processo é baseado em populações de indivíduos com tamanhos iguais. No entanto, este estudo propõe uma abordagem que considera indivíduos com tamanhos diferentes. Cada indivíduo tem uma probabilidade de 5% de passar pela mutação. Após observar a probabilidade do indivíduo de passar pela mutação, o processo irá adicionar ou remover novas camadas na rede. Em situações onde o indivíduo possui apenas uma camada, é adicionado uma nova camada com um número aleatório de unidades observando os limites mínimo e máximo de unidades por camada. Por outro lado, a última camada do indivíduo é removida se o indivíduo possuir uma quantidade de camadas igual ao limite do algoritmo. Atualmente esse limite está configurado para cinco camadas. Finalmente, em situações onde o indivíduo possui de duas até quatro camadas, uma nova camada será adicionada ou removida na última posição de forma aleatória.

A probabilidade de 5% de passar pela mutação foi obtida baseado na observação dos experimentos preliminares. A taxa foi testada começando em 5% e aumentando em incrementos de 5% até atingir um máximo de 30%. Com uma probabilidade de mutação muito baixa foi observado que era necessário mais gerações parar obter melhores resultados. Uma alternativa foi manter a probabilidade de mutação baixa e aumentar a quantidade de indivíduos na população. Apesar dessa abordagem ter levado menos gerações para obter bons resultados, o tempo para processar cada geração aumentou significativamente, por ser necessário obter o *fitness* de um grande número de indivíduos. Por outro lado, utilizando uma probabilidade de mutação muito alta resultou em um grau de aleatoriedade muito grande nos indivíduos e a não convergência do algoritmo.

4.4.5 Nova população

Para gerar a nova população, o algoritmo reserva uma quantidade dos indivíduos com maior *fitness* e os novos indivíduos gerados através dos passos descritos anteriormente. Os demais indivíduos da nova população são resultantes do processo de cruzamento. Com uma população de 60 indivíduos, 56 novos indivíduos são gerados para processo de cruzamento. Visto que o processo de cruzamento gera 4 indivíduos novos, isso que dizer que os passos de seleção, cruzamento e mutação são executados pelo menos 14 vezes pelo algoritmo.

A quantidade dos melhores indivíduos foi determinada através dos experimentos preliminares. Ao reservar um número grande de indivíduos, após algumas gerações foi observado uma grande quantidade de redes iguais. Isso acontece, pois a probabilidade dos indivíduos com maior *fitness* serem selecionados no processo de seleção é maior. Sendo assim, para uma população com 60 indivíduos, apenas os 4 melhores estão sendo adicionados na nova população.

Finalmente, uma nova população N é gerada utilizando os novos indivíduos criados pelo processo de cruzamento somados aos indivíduos com o maior valor de *fitness*. Com isso, a

geração atual será terminada e uma nova irá começar executando os mesmos procedimentos descritos acima, e observando o número máximo de gerações definidas pelo parâmetro *num_gen*. Após a execução da última geração, o indivíduo com o maior *fitness* será retornado como resposta pelo algoritmo.

4.5 CONSIDERAÇÕES SOBRE MÉTODO PROPOSTO

Resumindo, o conjunto de dados é pré-processado, com a separação do áudio do vídeo e a extração dos MFCCs dos arquivos de áudio. Após isso, os dados são divididos em conjuntos de treinamento e teste e normalizados. O projeto possui um módulo para persistir dos dados e as informações são armazenadas no disco e na nuvem. O módulo da rede neural é utilizado para criação e treinamento da rede neural, e uma configuração de topologia da rede é esperada pelo módulo para que seja utilizado para a construção da rede e posteriormente o treinamento. As redes são treinadas utilizando o optimizador Adam, a função de perda *binary cross entropy*, a técnica de CLR para encontrar o *learning rate* ótimo e a performance da rede é avaliada em termos de F1-score.

O algoritmo genético é utilizado para encontrar as melhores topologias de redes para resolver o problema de detecção de enganos através da fala. Os parâmetros de entrada incluem a função que calcula o *fitness*, a quantidade de gerações, tamanho da população, a taxa de mutação, a quantidade máxima de camadas da rede e a quantidade mínima e máxima de unidades por camada. Em cada geração, é calculado o valor de *fitness* de cada indivíduo que consiste em uma função linear dos valores de F1-score e do negativo da diferença absoluta entre as perdas de treinamento e validação. O método para realizar a seleção é o método da roleta. O cruzamento utiliza parte a metade de cada indivíduo combinadas para gerar novos indivíduos inéditos. Cada indivíduo tem uma probabilidade de 5% de passar pelo processo de mutação.

5 EXPERIMENTOS E ANÁLISES

Este capítulo fornece uma visão geral abrangente dos resultados e uma análise dos resultados. Para analisar os efeitos do algoritmo no desempenho do algoritmo genético, foram realizados uma série de experimentos em uma tarefa de detecção de mentiras, utilizando o conjunto de dados *Bag-of-Lies* como referência. Nesta seção, será descrita a configuração experimental e apresentados os resultados do estudo.

5.1 CONFIGURAÇÕES DO ALGORITMO GENÉTICO E DAS REDES LSTM

No presente estudo, foram conduzidos uma série de experimentos para determinar os parâmetros arquiteturais ótimos para o treinamento de redes LSTM. Através de análises empíricas, conforme descritos na seção 4.4, foi estabelecido um intervalo de valores adequados para o número de camadas e o número de unidades por camada. Os resultados indicam que, dada a complexidade da tarefa e a quantidade de dados disponíveis, um mínimo de 1 camada e um máximo de 5 camadas são apropriados. Da mesma forma, o número mínimo de unidades por camada foi estabelecido em 2 e o máximo em 40. Esses intervalos foram estabelecidos por meio de observação, análise do desempenho do modelo e frequência de *overfitting*.

Para determinar o critério de parada ótimo, foram testados diferentes números de gerações do algoritmo genético, começando em 50 e aumentando em incrementos de 20 até chegar a um máximo de 150. A análise empírica mostrou que o critério de parada ideal para o algoritmo genético utilizado neste estudo é de 100 gerações. Além desse ponto, observou-se um retorno diminuído em termos de desempenho, indicando que o algoritmo atingiu um nível satisfatório de convergência.

Em seguida, foi variado o tamanho da população, começando em 20 e aumentando em incrementos de 20 até atingir um máximo de 100. Descobriu-se que os melhores resultados foram obtidos quando o tamanho da população foi definido como 60. Além desse ponto, observou-se uma diminuição na capacidade do algoritmo de convergir para uma solução ótima, indicando que um tamanho de população maior não era benéfico para este problema em particular.

5.2 CONFIGURAÇÕES DE FUNÇÃO DE FITNESS

A Figura 11 mostra os valores de *fitness* ao longo do tempo de acordo com o experimento. Para os pesos de perda e F1-score (β_0 e β_1, respectivamente), foram realizados experimentos para avaliar o impacto da alteração dos valores de peso do F1-score e peso de perda no desempenho do GA. Foi iniciado com um valor de 0,1 para β_0 e 0,9 para β_1. Nos experimentos seguintes, incrementou-se 0,1 em β_0 e decrementou-se 0,1 em β_1 a cada vez. No experimento 5, quando ambos os pesos atingiram 0,5. A escolha dos pesos foi baseada em ajuste experimental e análises empíricas.

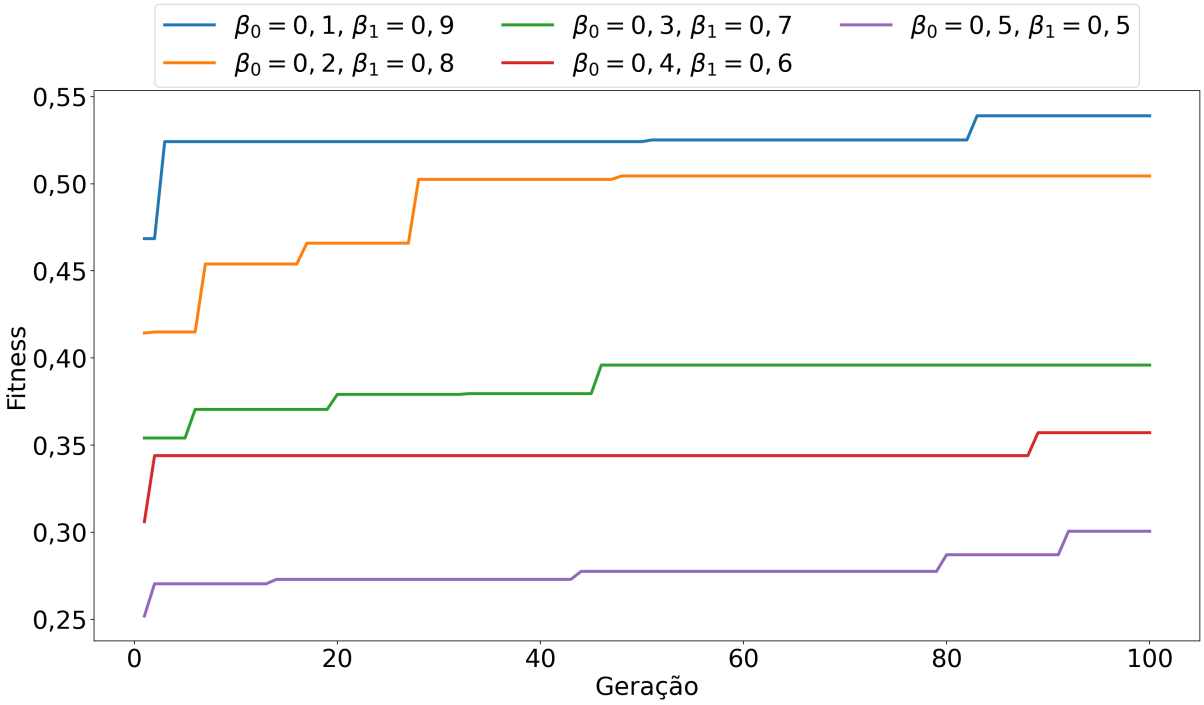

Figura 11 – Valor máximo de *fitness* ao longo de gerações para cada experimento. O eixo x representa as gerações e o eixo y representa o valor máximo de *fitness*.

Na Figura 12, são apresentados o F1-score e a diferença absoluta entre as perdas de treinamento e a perda de validação de cada experimento ao longo das gerações. Na Tabela 2, é realizada uma análise comparativa dos melhores indivíduos em termos de *fitness*, F1-score e diferença de perda de cada experimento. O primeiro experimento registrou um valor de *fitness* máximo de 0,538983 na geração 83. O indivíduo ótimo foi caracterizado por um F1-score de 0,61 e uma diferença absoluta entre as perdas de treinamento e validação de -0,11. Pelo gráfico é possível observar que o melhor indivíduo teve uma melhora no valor de F1-score justamente no momento em que a diferença entre as perdas aumentou. Isso pode ser explicado pelo peso do F1-score ser muito mais relevante do que o peso aplicado a diferença das perdas. Este indivíduo apresenta *overfitting*, o que pode ser atribuído a uma alocação desproporcional de peso em relação ao F1-score em relação à perda.

Tabela 2 – Comparação entre os melhores indivíduos em termos de *fitness*, F1-score e diferença absoluta entre as perdas de treinamento e validação para cada experimento

Experimento	Geração	Topologia	Fitness	F1-score	Dif. de Perda
$\beta_0 = 0,1$, $\beta_1 = 0,9$	83	[5, 17, 3]	0,538983	0,612079	-0,118886
$\beta_0 = 0,2$, $\beta_1 = 0,8$	48	[3, 30, 32, 32]	0,504438	0,630682	-0,000539
$\beta_0 = 0,3$, $\beta_1 = 0,7$	46	[3, 20, 3, 20]	0,395924	0,566667	-0,002475
$\beta_0 = 0,4$, $\beta_1 = 0,6$	89	[19, 19, 4, 8]	0,357114	0,623552	-0,042543
$\beta_0 = 0,5$, $\beta_1 = 0,5$	92	[14, 4, 18, 18, 4]	0,300551	0,612079	-0,010977

No segundo experimento, foi observado um alto F1-score e uma baixa diferença ab-

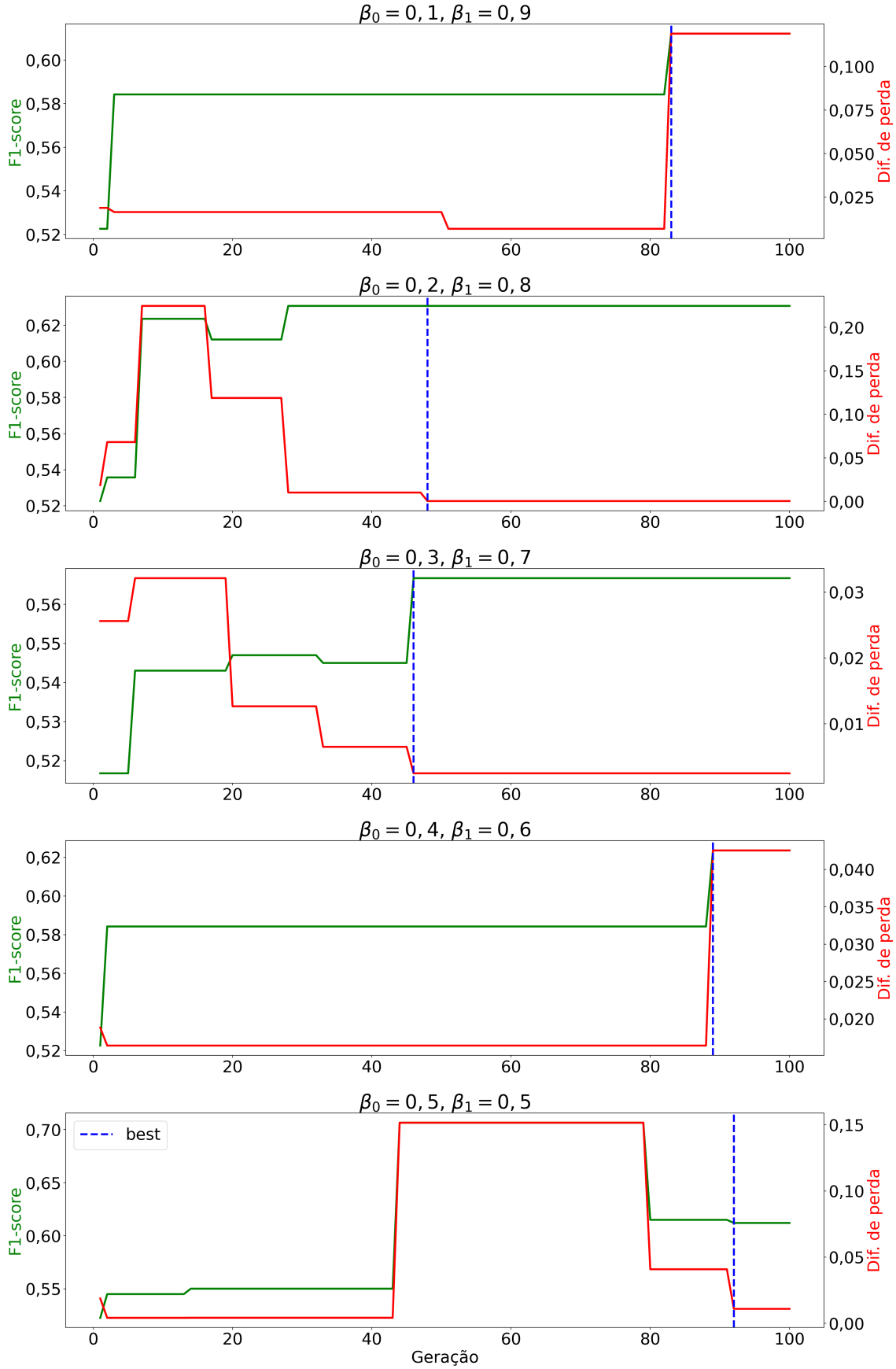

Figura 12 – Comparação de F1-score e a diferença absoluta entre perda de treinamento e perda de validação para cada experimento. Essa visualização ajuda a avaliar o desempenho do modelo e a capacidade de generalizar para novos dados e detectar *overfitting*.

soluta entre as perdas de treinamento e validação. Os resultados ótimos foram atingidos na geração 48, com o melhor indivíduo apresentando um valor de *fitness* de 0,504438, um F1-score de 0,630682 e o negativo da diferença absoluta entre as duas perdas de -0,000539. Dos experimentos realizados, esse foi o melhor indivíduo em termos de F1-score e diferença das perdas muito próximo de zero.

O terceiro experimento iniciou com os valores de F1-score baixos e *overfitting* pronunciado entre indivíduos com o maior *fitness* em cada geração. Ao longo das gerações, o algoritmo genético progrediu em direção à busca de soluções ótimas, onde os valores de F1-score melhoraram e a diferença entre as perdas diminuiu. Os melhores resultados foram obtidos na geração 46, com o melhor indivíduo possuindo um valor de *fitness* de 0,395924, um F1-score de 0,566667 e o negativo da diferença de perdas de -0,002475.

Os resultados do quarto experimento mostraram que, embora isso tenha levado a um valor de *fitness* menor em comparação ao segundo experimento, o melhor indivíduo ainda alcançou um F1-score de 0,623552. Embora esse escore seja ligeiramente menor do que o F1-score do segundo experimento, ainda foi considerado um desempenho aceitável. Os resultados do quarto experimento também revelaram um ligeiro aumento no *overfitting*, em comparação com o segundo experimento. O F1-score de 0,623552 foi alcançado na geração 89 e o valor de *fitness* correspondente foi 0,357114.

Os resultados do quinto experimento indicaram uma tendência semelhante ao quarto experimento, com um valor de *fitness* ligeiramente menor de 0,300551, alcançado na geração 92. No entanto, o F1-score de 0,61208 também foi menor do que o F1-score alcançado no quarto experimento, apesar de ter sido alcançado em uma geração posterior. Um aspecto interessante desse experimento foi que o valor de F1-score alcançado pelo melhor indivíduo não foi o maior valor de F1-score durante o experimento. O melhor valor de F1-score desse experimento foi de 0,706581 entre as gerações 44 até 79. Consequentemente, nesse intervalo observa-se o pior valor da diferença entre as perdas. Esses resultados sugerem que, embora aumentar o peso da diferença absoluta entre a perda de treinamento e a perda de validação e diminuir o peso do F1-score possa levar a um valor de F1-score mais baixo, pode resultar em um valor de *fitness* mais alto devido ao desbalanceamento dos pesos.

5.3 ANÁLISE DOS RESULTADOS

Uma comparação dos métodos relacionados com o método deste trabalho é apresentada na Tabela 3, que destaca os melhores resultados. A tabela demonstra que não existe um método único ideal para todos os cenários e cada um dos resultados pode variar de acordo com o conjunto de dados e o problema em questão.

Os resultados do método proposto foram avaliados em um conjunto de dados chamado *Bag-of-lies*, com um F1-score de 63,06% sendo alcançado. Embora essa performance não seja a mais alta entre os métodos descritos na literatura relacionada, deve-se notar que o método é único em seu uso de uma abordagem de autoaprendizagem de máquina que evita o processo de

Tabela 3 – Comparação dos métodos de detecção de engano e estresse e as suas performances com a abordagem deste trabalho.

Referência	Problema	Método	Conjunto de dados	Performance
(NEITERMAN; BITAN; AZARIA, 2020)	Detecção de engano	LSTM	Próprio	46,47% (F1)
(MARCOLLA; SANTIAGO; DAZZI, 2020)	Detecção de estresse	LSTM	Próprio	72,5% (acc)
(CHOU; LIU; LEE, 2019)	Detecção de engano	BiLSTM-DNN	DDDM	74,39% (F1)
(FU et al., 2019)	Detecção de engano	SS-ANE	CSC Próprio	62,78% (acc) 63,89% (acc)
(KOPEV et al., 2019)	Detecção de engano	MLP	Próprio	45,07% (F1)
(YERIGERI; RAGHA, 2019)	Detecção de estresse	Neuroevolution	SUSAS CMU Mahathi	85,53% (acc) 84,2% (acc)
(XIE et al., 2018)	Detecção de engano	CovBLSTM w/ Skip	CSC Próprio	68,4% (acc) 80,85% (acc)
(HAN; BYUN; KANG, 2018)	Detecção de estresse	LSTM	Próprio	65,2% (acc)
(MENDELS et al., 2017)	Detecção de engano	LSTM + MLP	CXD	63,90% (F1)
(HILMY et al., 2021)	Detecção de estresse	CNN	Próprio	61% (acc)
(MANSBACH; AZARIA, 2023)	Detecção de engano	meta-learning	Próprio	38,57% (F1)
Este trabalho	Detecção de engano	LSTM + GA	Bag-of-lies	63,06% (F1)

tentativa e erro tipicamente associado à seleção manual do modelo.

Vale ressaltar que o F1-score de 63,06% alcançado pelo método representa um nível razoável de desempenho para essa tarefa desafiadora, e fornece evidências para a validade da abordagem. Além disso, a abordagem de autoaprendizagem de máquina utilizada neste trabalho tem o potencial de ser mais amplamente aplicável em outros domínios, devido à sua capacidade de generalizar melhor do que os métodos tradicionais. Isso destaca o potencial do método para contribuir para o avanço do campo e oferece uma direção promissora para futuras pesquisas.

5.4 MELHOR INDIVÍDUO

A Figura 13 mostra a matriz de confusão do indivíduo com melhor desempenho descoberto durante os experimentos. Das 65 amostras usadas no conjunto de dados de treinamento, 33 são verdadeiras e 32 são classificadas como falsas. A rede classificou com precisão 20 amostras como verdadeiras, e essas amostras eram realmente verdadeiras. Em relação às amostras falsas, a rede classificou 21 amostras como falsas, que eram de fato falsas. Ou seja, das 65 amostras, 41 foram classificadas corretamente e 24 foram classificadas incorretamente. Os resultados mostram que 11 amostras foram classificadas como verdadeiras, mas na realidade, eram falsas, e 13 amostras foram classificadas como falsas, mas na realidade, eram verdadeiras.

Em relação às métricas desse indivíduo, a precisão foi de 0,645161, o *recall* foi de 0,606060 e a acurácia foi de 0,630769. A precisão de 0,645161 indica que de todas as amostras classificadas como verdadeiras, 64,52% eram realmente verdadeiras. O *recall* de 0,606060 indica que a rede identificou corretamente 60,60% de todas as amostras verdadeiras no conjunto de dados. A acurácia de 0,630769 indica que a rede classificou corretamente 63,07% de todas as amostras do conjunto de dados.

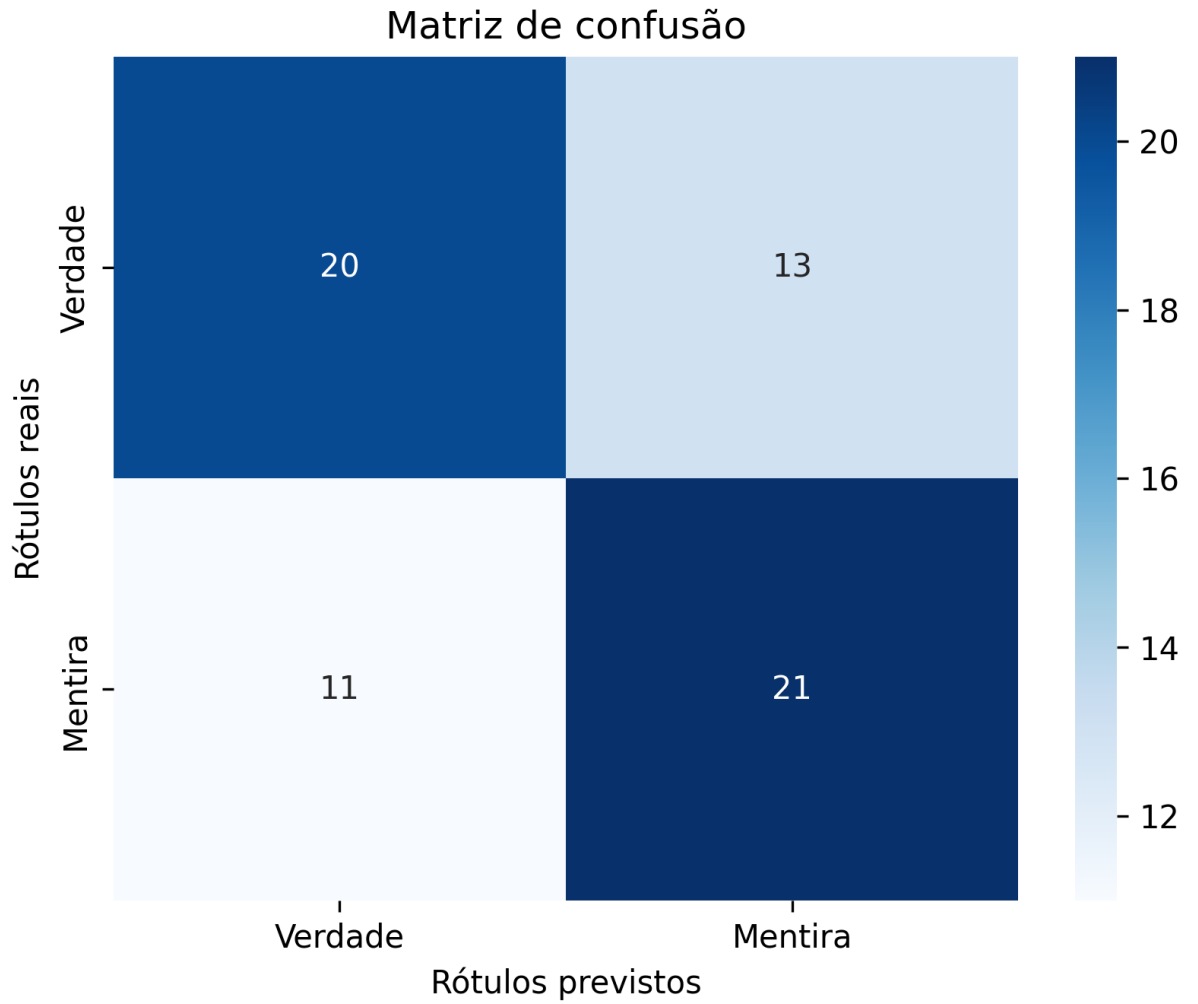

Figura 13 – Matriz de confusão mostrando a distribuição de rótulos previstos e reais, com verdadeiros positivos (TP) no canto superior esquerdo, falsos positivos (FP) no canto inferior esquerdo, verdadeiros negativos (TN) no canto inferior direito e falsos negativos (FN) no canto superior direito.

O gráfico da Figura 14 mostra a perda de treinamento e perda de validação ao longo das épocas durante o processo de treinamento. Ambas as linhas começam com alta variação de perda antes da época 40, indicando que o modelo ainda estava aprendendo e se ajustando aos dados de treinamento. No entanto, após a época 40, os valores de perda começam a se estabilizar e ambas as linhas se aproximam, indicando que o modelo está melhorando sua capacidade de generalização para novos dados.

Essa convergência das duas linhas sugere que o modelo não apresenta *overfitting* aos dados de treinamento, pois a perda de validação também está diminuindo com o tempo. A perda de validação decrescente indica que o modelo está aprendendo com sucesso os padrões subjacentes nos dados e pode generalizar bem para dados não vistos.

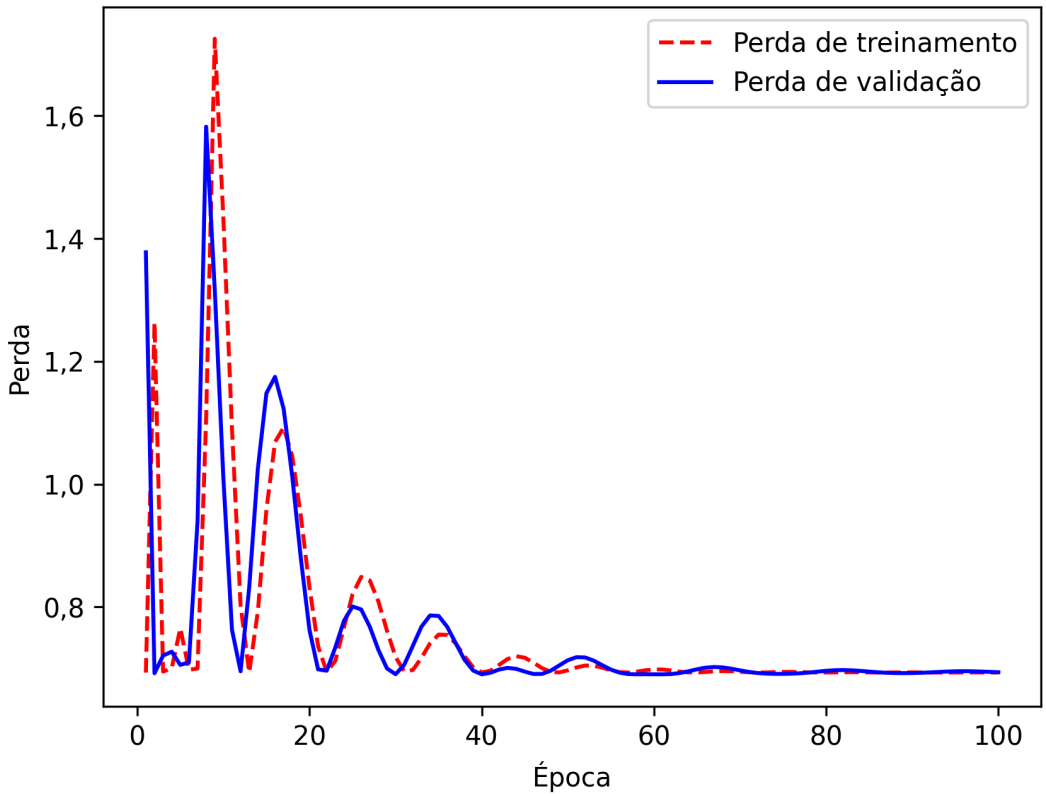

Figura 14 – Perdas de treinamento e validação ao longo das épocas de treinamento. A linha pontilhada vermelha indica a perda de treinamento. A linha contínua azul indica a perda de validação.

6 CONCLUSÕES

Este trabalho apresentou uma abordagem de aprendizado de máquina automatizada baseada em algoritmo genético para otimizar Redes Neurais do tipo *Long short-term memory* para detecção de mentiras em dados de voz. O objetivo deste estudo foi encontrar a topologia ótima da rede que resultasse em uma alta pontuação de F1-score e baixo *overfitting*. Os resultados mostraram que o método proposto foi eficaz em atingir esse objetivo. O algoritmo genético pôde identificar um compromisso ótimo entre maximizar a pontuação de F1-score e minimizar o *overfitting*, resultando em melhor desempenho na tarefa de detecção de enganos.

Voltando à pergunta inicial da pesquisa — "Poderia uma abordagem de Neuroevolução alcançar melhores resultados em comparação com os modelos de aprendizado profundo do estado da arte, na detecção de declarações enganosas por meio da fala?" — conclui-se que a abordagem de autoaprendizado de máquina empregada em nosso método não atingiu o valor de F1-score mais alto em comparação com outros métodos descritos nos trabalhos relacionados. No entanto, o presente método tem a vantagem distinta de evitar o processo exaustivo de tentativa e erro, que pode ser demorado e intensivo em recursos. Além disso, o método tem o potencial de generalizar melhor, uma vez que reduz o risco de *overfitting*, que é um desafio comum em abordagens tradicionais de aprendizado de máquina. Esses recursos exclusivos do método apresentado tornam-no uma abordagem promissora para futuras pesquisas na área. Além disso, há espaço para ajustar parâmetros, o que pode levar a superação dos resultados obtidos pelos métodos relacionados.

O método proposto foi eficaz em encontrar uma topologia da rede que resultou em um compromisso ideal entre maximizar a pontuação de F1-score e minimizar o *overfitting*. O melhor desempenho em termos de F1-score foi de 0,630682. Vale destacar que esse resultado não possui incidência de *overfitting*, cumprindo o objetivo de encontrar um modelo com uma alta performance capaz de generalizar e obter resultados no processo de inferência tão bom quanto os resultados de treinamento. Outra contribuição do método proposto é que o algoritmo conseguiu encontrar a topologia ótima da rede em poucas horas, economizando muito tempo em comparação com o processo exaustivo e demorado de tentativa e erro tipicamente associado à seleção manual de topologia de rede neural.

Em suma, os resultados deste estudo fornecem evidências de que algoritmos genéticos podem ser uma abordagem eficaz para otimizar modelos de aprendizado profundo para detecção de mentiras em dados de voz. O método proposto tem o potencial de ser estendido e melhorado para abordar as limitações deste estudo e encontrar os hiperparâmetros ótimos para modelos LSTM.

6.1 TRABALHOS FUTUROS

Este estudo avaliou apenas a topologia ótima para modelos LSTM. Para pesquisas futuras, seria interessante melhorar o algoritmo genético para também encontrar valores ideais de

hiperparâmetros para os modelos LSTM. Além disso, há a possibilidade de utilizar outras arquiteturas de redes neurais, como BiLSTM, por exemplo. Isso poderia resultar em uma melhoria adicional no desempenho e fornecer um processo de otimização mais abrangente.

Outro trabalho futuro poderia ser a incorporação de uma abordagem multiobjetivo (como NSGA), com o objetivo de maximizar não apenas a pontuação de F1-score, mas também outras métricas relevantes, como a precisão, recall, acurácia, entre outras. Além disso, pode-se utilizar validação cruzada para medir o desempenho do modelo e obter as métricas desejadas utilizando o conjunto de dados de validação. Isso pode ajudar a encontrar um equilíbrio ótimo entre várias métricas e permitir que o modelo atenda a várias necessidades ao mesmo tempo.

Além disso, este estudo foi limitado a dados de voz para detecção de decepção. Pesquisas futuras poderiam explorar a generalização do método proposto para outros tipos de dados, como dados de texto ou séries temporais. Além disso, o método proposto poderia ser utilizado a outras aplicações onde a aprendizagem de máquina automatizada possa ser usada para otimizar modelos complexos.

REFERÊNCIAS

ABBASI, H. et al. Comparison of trial and error and genetic algorithm in neural network development for estimating farinograph properties of wheat-flour dough. **Nutrition and Food Sciences Research**, Nutrition and Food Sciences Research, v. 2, n. 3, p. 29–38, 2015.

ALGORITHMS, G. Computer programs that"evolve"in ways that resemble natural selection can solve complex problems even their creators do not fully understand. **Holland in Scientific American**, p. 66–72, 1992.

ALI, A. et al. Automatic dialect detection in arabic broadcast speech. **arXiv preprint arXiv:1509.06928**, 2015.

ASIMOV, I. **I, Asimov: a memoir**. [S.l.]: Bantam, 2009.

BAE, S. H.; CHOI, I. K.; KIM, N. S. Acoustic scene classification using parallel combination of lstm and cnn. In: **DCASE**. [S.l.: s.n.], 2016. p. 11–15.

BAHMANI, V.; NASAB, M. Providing cloud security by combining authentication methods. **Journal of Fundamental and Applied Sciences**, v. 8, n. 2S, p. 2761–2773, 2016.

Bering Limited. **lrfinder**. 2020. https://github.com/beringresearch/lrfinder. Accessed: 2022-12-10.

BISHOP, C. M. et al. **Neural networks for pattern recognition**. [S.l.]: Oxford university press, 1995.

BRÅTEN, S. **The intersubjective mirror in infant learning and evolution of speech**. [S.l.]: John Benjamins Pub., 2009.

CADORET, R. J.; CAIN, C. A.; CROWE, R. R. Evidence for gene-environment interaction in the development of adolescent antisocial behavior. **Behavior Genetics**, Springer, v. 13, n. 3, p. 301–310, 1983.

CHEN, X.-W.; LIN, X. Big data deep learning: challenges and perspectives. **IEEE access**, Ieee, v. 2, p. 514–525, 2014.

CHISHOLM, R. M.; FEEHAN, T. D. The intent to deceive. **The journal of Philosophy**, v. 74, n. 3, p. 143–159, 1977.

CHOU, H.-C.; LIU, Y.-W.; LEE, C.-C. Joint learning of conversational temporal dynamics and acoustic features for speech deception detection in dialog games. In: IEEE. **2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)**. [S.l.], 2019. p. 1044–1050.

CHOWDHURY, A.; ROSS, A. Fusing mfcc and lpc features using 1d triplet cnn for speaker recognition in severely degraded audio signals. **IEEE transactions on information forensics and security**, IEEE, v. 15, p. 1616–1629, 2019.

CLONINGER, C. R.; REICH, T.; GUZE, S. B. Genetic-environmental interactions and antisocial behavior. **Psychopathic behavior: Approaches to research**, John Wiley & Sons New York, p. 225–237, 1978.

COLE, T. Lying to the one you love: The use of deception in romantic relationships. **Journal of Social and Personal Relationships**, Sage Publications Sage UK: London, England, v. 18, n. 1, p. 107–129, 2001.

COOK, L. G.; MITSCHOW, L. C. Beyond the polygraph: Deception detection and the autonomic nervous system. **Federal Practitioner**, Frontline Medical Communications, v. 36, n. 7, p. 316, 2019.

DARWISH, A.; BATAINEH, E. Eye tracking analysis of browser security indicators. In: IEEE. **2012 International Conference on Computer Systems and Industrial Informatics**. [S.l.], 2012. p. 1–6.

DASGUPTA, D.; MCGREGOR, D. R. Designing application-specific neural networks using the structured genetic algorithm. In: IEEE. **[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks**. [S.l.], 1992. p. 87–96.

DENG, M. et al. Heart sound classification based on improved mfcc features and convolutional recurrent neural networks. **Neural Networks**, Elsevier, v. 130, p. 22–32, 2020.

DEPAULO, B. M. et al. Lying in everyday life. **Journal of personality and social psychology**, American Psychological Association, v. 70, n. 5, p. 979, 1996.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DIKE, C. C.; BARANOSKI, M.; GRIFFITH, E. E. Pathological lying revisited. **Journal of the American Academy of Psychiatry and the Law Online**, Citeseer, v. 33, n. 3, p. 342–349, 2005.

DONG, X. et al. Dynamical hyperparameter optimization via deep reinforcement learning in tracking. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 43, n. 5, p. 1515–1529, 2019.

FERNÁNDEZ, R. G. La mentira. un arte con historia. **Aposta. Revista de Ciencias Sociales**, Luis Gómez Encinas ed., n. 26, p. 1–17, 2006.

FLOREANO, D.; DÜRR, P.; MATTIUSSI, C. Neuroevolution: from architectures to learning. **Evolutionary intelligence**, Springer, v. 1, p. 47–62, 2008.

FORNACIARI, T.; POESIO, M. Automatic deception detection in italian court cases. **Artificial intelligence and law**, Springer, v. 21, n. 3, p. 303–340, 2013.

FU, H. et al. Improved semi-supervised autoencoder for deception detection. **PloS one**, Public Library of Science San Francisco, CA USA, v. 14, n. 10, p. e0223361, 2019.

GALVÁN, E.; MOONEY, P. Neuroevolution in deep neural networks: Current trends and future challenges. **IEEE Transactions on Artificial Intelligence**, IEEE, v. 2, n. 6, p. 476–493, 2021.

GONG, D.; YAN, J.; ZUO, G. A review of gait optimization based on evolutionary computation. **Applied Computational Intelligence and Soft Computing**, Hindawi, v. 2010, 2010.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GOODFELLOW, I. et al. Generative adversarial networks. **Communications of the ACM**, ACM New York, NY, USA, v. 63, n. 11, p. 139–144, 2020.

GRACIARENA, M. et al. Combining prosodic lexical and cepstral systems for deceptive speech detection. In: IEEE. **2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings**. [S.l.], 2006. v. 1, p. I–I.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. **Neural networks**, Elsevier, v. 18, n. 5-6, p. 602–610, 2005.

GRUBIN, D. Commentary: Getting at the truth about pathological lying. **Journal of the American Academy of Psychiatry and the Law Online**, Citeseer, v. 33, n. 3, p. 350–353, 2005.

GUPTA, V. et al. Bag-of-lies: A multimodal dataset for deception detection. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2019. p. 0–0.

HAN, H.; BYUN, K.; KANG, H.-G. A deep learning-based stress detection algorithm with speech signal. In: **proceedings of the 2018 workshop on audio-visual scene understanding for immersive multimedia**. [S.l.: s.n.], 2018. p. 11–15.

HAN, W. et al. An efficient mfcc extraction method in speech recognition. In: IEEE. **2006 IEEE International Symposium on Circuits and Systems (ISCAS)**. [S.l.], 2006. p. 4–pp.

HANSEN, J. H.; BOU-GHAZALE, S. E. Getting started with susas: A speech under simulated and actual stress database. In: **Fifth European Conference on Speech Communication and Technology**. [S.l.: s.n.], 1997.

HARTSHORNE, H.; MAY, M. A. Studies in the nature of character,[part] 1: Studies in deceit: book 1, general methods and results; book 2, statistical methods and results. Macmillan Co, 1928.

HASTINGS, E. J.; GUHA, R. K.; STANLEY, K. O. Automatic content generation in the galactic arms race video game. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v. 1, n. 4, p. 245–263, 2009.

HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. **Knowledge-Based Systems**, Elsevier, v. 212, p. 106622, 2021.

HEBB, D. O. **The organization of behavior: A neuropsychological theory**. [S.l.]: Psychology Press, 2005.

HERCHONVICZ, A. L. **Implementação do trabalho Neuroevolução de Redes LSTM: uma aplicação na detecção de engano via voz**. 2023. https://github.com/andreylh/ga-automl. [Online; Acessado 06-Julho-2023].

HERCHONVICZ, A. L.; SANTIAGO, R. de. Deep neural network architectures for speech deception detection: A brief survey. In: SPRINGER. **Progress in Artificial Intelligence: 20th EPIA Conference on Artificial Intelligence, EPIA 2021, Virtual Event, September 7–9, 2021, Proceedings 20**. [S.l.], 2021. p. 301–312.

HILMY, M. S. H. et al. Stress classification based on speech analysis of mfcc feature via machine learning. In: IEEE. **2021 8th International Conference on Computer and Communication Engineering (ICCCE)**. [S.l.], 2021. p. 339–343.

HIPPEL, W. V.; TRIVERS, R. The evolution and psychology of self-deception. **Behavioral and brain sciences**, Cambridge University Press, v. 34, n. 1, p. 1, 2011.

HIRSCHBERG, J. B. et al. Distinguishing deceptive from non-deceptive speech. 2005.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOLLAND, J. H. et al. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

HUANG, C.-H. et al. Acoustic indicators of deception in mandarin daily conversations recorded from an interactive game. In: **INTERSPEECH**. [S.l.: s.n.], 2019. p. 1731–1735.

IACONO, W. G. Accuracy of polygraph techniques: Problems using confessions to determine ground truth. **Physiology & Behavior**, Elsevier, v. 95, n. 1-2, p. 24–26, 2008.

INNES, B. **The history of torture**. [S.l.]: Amber Books Ltd, 2012.

JEBARI, K.; MADIAFI, M. Selection methods for genetic algorithms. **International Journal of Emerging Sciences**, Springfield Publishing Corporation, v. 3, n. 4, p. 333–344, 2013.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. **Journal of documentation**, MCB UP Ltd, 1972.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

JR, C. F. B.; DEPAULO, B. M. Accuracy of deception judgments. **Personality and social psychology Review**, SAGE Publications Sage CA: Los Angeles, CA, v. 10, n. 3, p. 214–234, 2006.

JUVELA, L. et al. Speech waveform synthesis from mfcc sequences with generative adversarial networks. In: IEEE. **2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2018. p. 5679–5683.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KOPEV, D. et al. Detecting deception in political debates using acoustic and textual features. In: IEEE. **2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)**. [S.l.], 2019. p. 652–659.

KOZEL, F. A. et al. Detecting deception using functional magnetic resonance imaging. **Biological psychiatry**, Elsevier, v. 58, n. 8, p. 605–613, 2005.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Communications of the ACM**, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

LAMPERT, J. Lie detectors-industrial use of the polygraph. **DePaul L. Rev.**, HeinOnline, v. 13, p. 287, 1963.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LEDELL, E.; POIRIER, S. H2o automl: Scalable automatic machine learning. In: **Proceedings of the AutoML Workshop at ICML**. [S.l.: s.n.], 2020. v. 2020.

LEVINE, T. R.; KIM, R. K.; HAMEL, L. M. People lie for a reason: Three experiments documenting the principle of veracity. **Communication Research Reports**, Taylor & Francis, v. 27, n. 4, p. 271–285, 2010.

LEVITAN, S. I. et al. Cross-cultural production and detection of deception from speech. In: **Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection**. [S.l.: s.n.], 2015. p. 1–8.

LEWIS, M. The development of deception. The Guilford Press, 1993.

MAJDI, A.; BEIKI, M. Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses. **International Journal of Rock Mechanics and Mining Sciences**, Elsevier, v. 47, n. 2, p. 246–253, 2010.

MANSBACH, N.; AZARIA, A. Meta learning based deception detection from speech. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 13, n. 1, p. 626, 2023.

MARCOLLA, F. M.; SANTIAGO, R. de; DAZZI, R. L. Novel lie speech classification by using voice stress. In: **ICAART (2)**. [S.l.: s.n.], 2020. p. 742–749.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **The MIT License**. 1988. Disponível em: https://opensource.org/licenses/MIT.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MEDSKER, L. R.; JAIN, L. Recurrent neural networks. **Design and Applications**, v. 5, p. 64–67, 2001.

MEGHANANI, A.; ANOOP, C.; RAMAKRISHNAN, A. An exploration of log-mel spectrogram and mfcc features for alzheimer's dementia recognition from spontaneous speech. In: IEEE. **2021 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.], 2021. p. 670–677.

MELE, A. R. **Self-deception unmasked**. [S.l.]: Princeton University Press, 2000.

MENDELS, G. et al. Hybrid acoustic-lexical deep learning approach for deception detection. In: **INTERSPEECH**. [S.l.: s.n.], 2017. p. 1472–1476.

MINSKY, M.; PAPERT, S. A. **Perceptrons: An introduction to computational geometry**. [S.l.]: MIT press, 2017.

MOFFITT, T. E. The new look of behavioral genetics in developmental psychopathology: Gene-environment interplay in antisocial behaviors. **Biosocial Theories of Crime**, Routledge, p. 183–204, 2017.

MUZINIC, L.; KOZARIC-KOVACIC, D.; MARINIC, I. Psychiatric aspects of normal and pathological lying. **International journal of law and psychiatry**, Elsevier, v. 46, p. 88–93, 2016.

NEITERMAN, E. H.; BITAN, M.; AZARIA, A. Multilingual deception detection by autonomous agents. In: **Companion Proceedings of the Web Conference 2020**. [S.l.: s.n.], 2020. p. 480–484.

O'MARA, S. **Why torture doesn't work: The neuroscience of interrogation**. [S.l.]: Harvard University Press, 2015.

PANOVA, M.; POSTOLOV, K. Unconventional methods for selecting human resources-polygraph and honesty (integrity) tests. **Economic Development/Ekonomiski Razvoj**, v. 17, n. 3, 2015.

PAPAVASILEIOU, E.; CORNELIS, J.; JANSEN, B. A systematic literature review of the successors of "neuroevolution of augmenting topologies". **Evolutionary Computation**, MIT Press, v. 29, n. 1, p. 1–73, 2021.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: PMLR. **International conference on machine learning**. [S.l.], 2013. p. 1310–1318.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

PÉREZ-ROSAS, V. et al. Verbal and nonverbal clues for real-life deception detection. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2015. p. 2336–2346.

PLACE, V. **The Guilt Project: Rape, Morality, and Law**. [S.l.]: Other Press, LLC, 2010.

RAMOS, D. et al. Deconstructing cross-entropy for probabilistic binary classifiers. **Entropy**, MDPI, v. 20, n. 3, p. 208, 2018.

REINACH, L.; LOUW, D. The relationship between anxiety and polygraph results. **Acta Criminologica: African Journal of Criminology & Victimology**, Criminological and Victimological Society of Southern Africa (CRIMSA), v. 15, n. 3, p. 56–67, 2002.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

SALIMANS, T. et al. Evolution strategies as a scalable alternative to reinforcement learning. **arXiv preprint arXiv:1703.03864**, 2017.

SAMEK, W.; WIEGAND, T.; MÜLLER, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. **arXiv preprint arXiv:1708.08296**, 2017.

SCHULLER, B. et al. The interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism. In: **Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France**. [S.l.: s.n.], 2013.

SHAFIQUE, R. et al. A novel approach to railway track faults detection using acoustic analysis. **Sensors**, MDPI, v. 21, n. 18, p. 6221, 2021.

SILVA, F. et al. odneat: An algorithm for decentralised online evolution of robotic controllers. **Evolutionary Computation**, MIT Press, v. 23, n. 3, p. 421–449, 2015.

SMITH, L. N. Cyclical learning rates for training neural networks. In: IEEE. **2017 IEEE winter conference on applications of computer vision (WACV)**. [S.l.], 2017. p. 464–472.

SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. **arXiv preprint arXiv:1803.09820**, 2018.

STANLEY, K. O.; BRYANT, B. D.; MIIKKULAINEN, R. Real-time neuroevolution in the nero video game. **IEEE transactions on evolutionary computation**, IEEE, v. 9, n. 6, p. 653–668, 2005.

STANLEY, K. O.; D'AMBROSIO, D. B.; GAUCI, J. A hypercube-based encoding for evolving large-scale neural networks. **Artificial life**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 15, n. 2, p. 185–212, 2009.

STANLEY, K. O.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. **Evolutionary computation**, MIT Press, v. 10, n. 2, p. 99–127, 2002.

SUCH, F. P. et al. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. **arXiv preprint arXiv:1712.06567**, 2017.

SUNG, M.; PENTLAND, A. Pokermetrics: Stress and lie detection through non-invasive physiological sensing. **Tech. Rep., MIT Media Lab**, 2005.

SWEDIA, E. R. et al. Deep learning long-short term memory (lstm) for indonesian speech digit recognition using lpc and mfcc feature. In: IEEE. **2018 Third International Conference on Informatics and Computing (ICIC)**. [S.l.], 2018. p. 1–5.

VERBANCSICS, P.; HARGUESS, J. Image classification using generative neuro evolution for deep learning. In: IEEE. **2015 IEEE winter conference on applications of computer vision**. [S.l.], 2015. p. 488–493.

WAGHMARE, V. B. et al. Development of isolated marathi words emotional speech database. **International Journal of Computer Applications**, Foundation of Computer Science, v. 94, n. 4, p. 19–22, 2014.

WAGHMARE, V. B. et al. Emotion recognition system from artificial marathi speech using mfcc and lda techniques. In: **Fifth International Conference on Advances in Communication, Network, and Computing–CNC**. [S.l.: s.n.], 2014.

WARING, J.; LINDVALL, C.; UMETON, R. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. **Artificial intelligence in medicine**, Elsevier, v. 104, p. 101822, 2020.

WILLIAMS, K. C. et al. The business of lying. **Journal of Leadership, Accountability and Ethics**, North American Business Press, p. 1, 2009.

XIE, Y. et al. Convolutional bidirectional long short-term memory for deception detection with acoustic features. **IEEE Access**, IEEE, v. 6, p. 76527–76534, 2018.

YERIGERI, V. V.; RAGHA, L. Meta-heuristic approach in neural network for stress detection in marathi speech. **International Journal of Speech Technology**, Springer, v. 22, n. 4, p. 937–957, 2019.

YU, X.; GEN, M. **Introduction to evolutionary algorithms**. [S.l.]: Springer Science & Business Media, 2010.

ZHENG, F.; ZHANG, G.; SONG, Z. Comparison of different implementations of mfcc. **Journal of Computer science and Technology**, Springer, v. 16, p. 582–589, 2001.