UNESP – FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ADRIAN FELIPE NOGUEIRA BATISTA
ORIENTADOR: PROF. DR. FÁBIO BERTEQUINI LEÃO

DIAGNÓSTICO DE FALTAS INCIPIENTES EM TRANSFORMADORES DE POTÊNCIA BASEADO NA ANÁLISE DE GASES DISSOLVIDOS NO ÓLEO ISOLANTE EMPREGANDO REDES NEURAIS ARTIFICIAIS

ILHA SOLTEIRA – SP
2022

ADRIAN FELIPE NOGUEIRA BATISTA

DIAGNÓSTICO DE FALTAS INCIPIENTES EM TRANSFORMADORES DE POTÊNCIA BASEADO NA ANÁLISE DE GASES DISSOLVIDOS NO ÓLEO ISOLANTE EMPREGANDO REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de bacharelado em Engenharia Elétrica.

Nome do orientador

Fábio Bertequini Leão

ILHA SOLTEIRA – SP

2022

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

B333d

Batista, Adrian Felipe Nogueira.
Diagnóstico de faltas incipientes em transformadores de potência baseado na análise de gases dissolvidos no óleo isolante empregando Redes Neurais Artificiais / Adrian Felipe Nogueira Batista. -- Ilha Solteira: [s.n.], 2022
83 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2022

Orientador: Fábio Bertequini Leão
Inclui bibliografia

1. Transformadores. 2. Redes Neurais Artificiais. 3. Análise de gases dissolvidos no óleo.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos **onze dias do mês de julho do ano de dois mil e vinte e dois**, o discente ***Adrian Felipe Nogueira Batista,*** matriculado sob o nº 162054726, tendo como banca examinadora seu orientador, o *Prof. Dr. Fábio Bertequini Leão, a Profa. Doutora Anna Diva Plasencia Lotufo e a Mestranda Elis Caroline de Souza Trindade,* apresentou o Trabalho de Graduação intitulado **"Diagnóstico de Faltas Incipientes em Transformadores de Potência baseado na Análise de Gases Dissolvidos no Óleo Isolante empregando Redes Neurais Artificiais"** obtendo a nota 10 (DEZ) e conceito Aprovado.

Fábio Bertequini Leão
Prof. Dr. Fábio Bertequini Leão
- orientador -

Adrian Felipe Nogueira Batista
Adrian Felipe Nogueira Batista
- discente -

Profa. Dra. Anna Diva Plasencia Lotufo
- Membro da Banca -

Elis Caroline de Trindade
Mestranda Elis Caroline de Souza Trindade
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

AGRADECIMENTOS

A Deus, pela minha vida, e por me permitir ultrapassar todos os obstáculos encontrados ao longo da realização deste trabalho; aos meus pais Adailto Batista e Alessandra Vanessa Nogueira Batista e a toda minha família por todo o apoio e pela ajuda, que muito contribuíram para a realização deste trabalho.

Agradeço também, a todos os professores da UNESP de Ilha Solteira que guiaram meus estudos, principalmente ao professor Dr. Fábio Bertequini Leão que me orientou neste trabalho.

Por fim, gostaria de agradecer a todos meus colegas de turma e amigos que fiz ao longo deste caminho e com quem convivi intensamente durante os últimos anos, pelo companheirismo e pela troca de experiências que me permitiram crescer como pessoa.

RESUMO

A análise de gases dissolvidos no óleo de transformadores é uma metodologia empregada nos dias atuais para diagnosticar faltas em transformadores de potência. O objetivo deste trabalho é utilizar redes neurais artificiais para criar um método alternativo de análise de gases dissolvidos no óleo do transformador utilizando como base a norma técnica IEC 60559. Propõe-se, assim, analisar os dados obtidos utilizando-se diversas formas de treinamento da rede para verificar qual método de treinamento e qual configuração de rede apresenta melhor eficiência quando os resultados são comparados aos diagnósticos esperados. A partir dos resultados obtidos foi possível atingir uma eficiência de até 91% indicando que a metodologia empregada apresenta potencial para ser utilizada no diagnóstico de faltas em transformadores de potência.

Palavra-chave: Transformadores, Redes Neurais Artificiais, Análise de Gases Dissolvidos no Óleo.

ABSTRACT

The dissolved gasses analysis in transformer's oil is a methodology used in present days to diagnostic faults in power transformers. This work's objective is to use artificial neural networks to create an alternative method of dissolved gasses analysis in transformer's oil with technical standard IEC 60559 as base. Thereby, it is proposed to analyze the data obtained with several types of network training methods verify which method and which network configuration has better efficiency with the expected results. In that way, it was possible to achieve a 91% efficiency indicating that the used method has potential to be used in dissolved gasses analysis in power transformers.

Keyword: Transformers, Artificial Neural Networks, Dissolved Gasses Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Os transformadores de potência são componentes essenciais e de grande importância para os sistemas elétricos. Eles podem ser utilizados desde as usinas de geração de energia elétrica, onde o transformador eleva a tensão a níveis alto e extra alto sendo adequados para a transmissão econômica de potência (reduzindo a corrente e a queda de tensão no sistema) até os pontos de consumo onde a tensão é reduzida ao nível de média tensão para os sistemas de subtransmissão e distribuição, fornecendo energia as redes urbanas e rurais, e finalmente reduzida a níveis de baixa tensão adequados ao consumo de consumidores residenciais (FILHO, 2013), (FITZGERALD; K.; UMANS, 2003).

Estruturalmente o transformador de potência é basicamente composto por material ferromagnético laminado envolvido por bobinas de cobre isoladas com celulose e toda essa estrutura está imersa em óleo mineral, sintético ou vegetal, que tem a função de resfriar as bobinas e realizar o isolamento dielétrico para evitar arcos elétricos. O envelhecimento da isolação é considerado uma ameaça em potencial à segurança operacional dos transformadores, principalmente para as unidades submetidas a elevados gradientes de tensão. A gradual deterioração da pureza do óleo tem um importante papel no sistema de isolação destes equipamentos (NETO; ASSUNÇÃO; ASSUNÇÃO, 2009).

A análise da composição de gases dissolvidos no óleo isolante é uma das formas mais empregadas para realizar o diagnóstico, detectar e avaliar as condições de faltas[1] incipientes em transformadores que empregam óleo mineral. Faltas incipientes são aquelas que estão em um estágio inicial e promovem a decomposição do material isolante e, portanto, estão associadas às concentrações de gases formados no interior do transformador (IEC, 2015). Se as faltas incipientes não forem analisadas e tratadas através de ações corretivas elas podem evoluir para situações de sobreaquecimento, arco elétrico ou curto-circuito causando a desenergização do transformador pela atuação da proteção ou em casos mais graves a deterioração da isolação interna e degradação parcial ou total do equipamento quando da não detecção do defeito pela proteção (ANDERSON, 1999).

As técnicas convencionais de manutenção de transformadores requerem que estes sejam tirados de operação. Atualmente, com a evolução da tecnologia têm sido desenvolvidos diversos sistemas de monitoramento on-line, através de sensores e sistemas de aquisição de dados, que

[1] Ocorrência não planejada ou defeito em um componente que pode resultar em uma ou mais falhas do próprio componente ou componentes associados a ele (IEC, 2015).

possibilitam que os transformadores não tenham que ser retirados de operação para que seja feito um diagnóstico sobre possíveis faltas incipientes através da análise do óleo isolante (NETO; ASSUNÇÃO; ASSUNÇÃO, 2009).

As técnicas que utilizam a análise do óleo isolante são capazes de prover uma perspectiva única da condição atual do sistema de isolamento de transformadores. Tais informações são de importância fundamental para que os engenheiros possam tomar decisões quanto ao planejamento da operação e manutenção de tais equipamentos. O emprego do óleo isolante para realizar o diagnóstico da condição do equipamento tem suas vantagens, pois pode ser realizado com o transformador em operação. Os gases mais comumente encontrados no óleo de transformadores em operação, formados pela decomposição do óleo e do material isolante, são: oxigênio (O_2), hidrogênio (H_2), monóxido de carbono (CO), dióxido de carbono (CO_2), metano (CH_4), etano (C_2H_6), etileno (C_2H_4) e acetileno (C_2H_2) (NETO; ASSUNÇÃO; ASSUNÇÃO, 2009).

Vários métodos foram desenvolvidos para análise dos gases presentes no óleo isolante de modo a obter a indicação de sua real condição de operação. A natureza e a concentração dos gases indicam o tipo e a severidade da falta no transformador. Além disso as mudanças verificadas na produção de cada gás e a sua taxa de produção são fatores importantes na determinação do tipo e evolução de faltas nos transformadores. Os métodos mais comuns de diagnóstico de gases dissolvidos no óleo são: Método de Doernenburg, Método de Rogers, IEC 60599, Triângulo de Duval e Método dos gases chaves (NETO; ASSUNÇÃO; ASSUNÇÃO, 2009). Dentre esses métodos o mais atual e abrangente é o estabelecido na norma IEC 60599 (IEC, 2015), (DUVAL; DEPABLO, 2001) e (DUVAL, 2002).

Conforme a norma (IEC, 2015) a presença de gases dissolvidos no óleo isolante em determinadas proporções pode indicar seis tipos de faltas incipientes: descarga parcial (DP), descarga de baixa energia (D1), descarga de alta energia (D2), e três tipos de faltas térmicas (T1, T2 e T3) classificadas conforme a temperatura do óleo. As classificações baseadas nas proporções de gases encontrados no óleo são apresentadas na Tabela 1.

Tabela 1 - Diagnóstico baseado na proporção de gases dissolvidos no óleo em ppm

Falta	Característica da falta	$\frac{C_2H_2}{C_2H_4}$	$\frac{CH_4}{H_2}$	$\frac{C_2H_4}{C_2H_6}$
DP	Descarga parcial	NS[a]	<0,1	<0,2
D1	Descarga de baixa energia	>1	0,1 a 0,5	>1
D2	Descarga de alta energia	0,6 a 2,5	0,1 a 1	>2
T1	Falta térmica t<300ºC	NS	>1	<1
T2	Falta térmica 300ºC<t<700ºC	<0,1	>1	1 a 4
T3	Falta térmica t>700ºC	<0,2	>1	>4

[a]: quantidade não significante.

Fonte: (IEC, 2015)

Apesar da existência dos métodos convencionais a interpretação dos resultados é frequentemente complexa e deve ser sempre feita com cuidado, envolvendo pessoal experiente com conhecimento técnico na área. Desta forma a decisão final fica a cargo de um especialista visto que transformadores de diferentes fabricantes, nível de tensão e potência, estrutura, material utilizado no sistema de isolamento, tipo de carregamento e histórico de manutenção podem apresentar diferentes características quanto à produção de gases e, portanto, necessitam, na maioria dos casos, serem considerados de forma particular. Observa-se ainda que o nível e período de formação dos gases dependem da idade dos transformadores e também da localização, natureza e severidade das faltas a que são submetidos. Todos os fenômenos relacionados à formação de gases em transformadores e sua correlação com as faltas incipientes são caracterizados por imprecisões, incertezas nas medidas e não-linearidades não modeladas (DUARTE et al., 2020). Portanto, métodos convencionais de interpretação da análise dos gases combinados com métodos baseados em inteligência computacional, em especial as Redes Neurais Artificiais (RNAs), podem ser empregados de forma eficiente para o diagnóstico automático de faltas incipientes, e consequentemente na classificação dos transformadores em relação ao seu envelhecimento (NETO; ASSUNÇÃO; ASSUNÇÃO, 2009), (DAPONTE et al., 1996) e (GALDI et al., 2000).

A aplicação de RNAs para o diagnóstico de faltas incipientes em transformadores e consequentemente na classificação dos transformadores é particularmente interessante, pois, as RNAs são capazes de adquirir conhecimento diretamente dos dados e assim revelar relações não-lineares entre as entradas e saídas que ainda são desconhecidas pelos especialistas

possibilitando o diagnóstico em situações em que os métodos convencionais não conseguem classificar a falta.

Nos trabalhos de DAPONTE et al. (1996), GALDI et al. (2000) e DE FREITAS; DA SILVA; DE SOUZA (2002), os autores têm demonstrado a capacidade de RNAs para lidar adequadamente com relações não lineares e complexas, que são de difícil modelagem, empregando técnicas analíticas para a realização do monitoramento e diagnóstico de transformadores. Nestes artigos os autores propõem RNAs para o monitoramento das condições térmicas dos enrolamentos de transformadores de potência em condições de sobrecarga empregando variáveis ambientais.

Com o objetivo de realizar o diagnóstico de transformadores baseado na análise dos gases dissolvidos no óleo os autores do artigo ZHANG et al. (1996) propõem um método de dois passos baseado em RNAs. A RNA é projetada e treinada off-line e o diagnóstico é realizado on-line. Os autores empregam o algoritmo de aprendizagem por retropropagação para o treinamento da RNA na etapa off-line.

Em HELL; D`ANGELO; COSTA (2002) os autores propõem uma Rede Neural Fuzzy (RNF). O método de Rogers é empregado para construir a arquitetura inicial das funções membro do modelo Fuzzy. Os dados de entrada (proporção gasosa) são aplicados na camada Fuzzy e a saída alimenta uma RNA de uma única camada responsável pelo diagnóstico.

No trabalho de SUN; HUANG; HUANG (2012) os autores realizam uma revisão bibliográfica dos principais métodos computacionais baseados em inteligência artificial para o diagnóstico de faltas incipientes em transformadores. Os autores abordam principalmente métodos baseados em RNAs, Lógica Fuzzy e otimização evolucionária.

Em KHADE; MAHAJAN; CHAUDHARI (2016) os autores propõem uma RNA baseada no método de Rogers, e empregam dados reais para o treinamento e teste da rede. A RNA empregada é do tipo Perceptron de Multicamadas (MLP). Os autores evidenciam a capacidade da RNA realizar o diagnóstico, principalmente para os casos em que o método de Rogers não é capaz de diagnosticar a falta.

No artigo de WANG et al. (2021) é proposta uma RVFL (Random Vector Functional-Link Neural Network) de três camadas composta por quatro classificadores RVFLs que são responsáveis pela classificação dos tipos de faltas incipientes. Os autores apresentam diversos testes e resultados mostrando que os classificadores melhoram a eficiência global da RVFL proposta.

No artigo TAO et al. (2021) é proposta uma PNN (Probabilistic Neural Network) e um otimizador bio-inspirado. A PNN é empregada para o diagnóstico, enquanto o otimizador bio-

inpirado tem a função de otimizar a camada oculta da PNN, com o objetivo de melhorar a classificação da falta. São empregados dados reais e os resultados mostram que o método possui boa habilidade de aprendizado e elevada precisão no diagnóstico.

Baseado no que foi exposto anteriormente o objetivo deste trabalho de graduação é desenvolver uma RNA para o diagnóstico de faltas incipientes em transformadores de potência baseado na análise dos gases dissolvidos no óleo isolante. O método de diagnóstico que deve ser empregado como referência para análise e validação da RNA é o apresentado na norma IEC (IEC, 2015) cuja classificação das faltas é baseada nas relações dos gases conforme Tabela 1.

2 METODOLOGIA PARA O DIAGNÓSTICO DE FALTAS INCIPIENTES EM TRANSFORMADORES DE POTÊNCIA EMPREGANDO RNAS

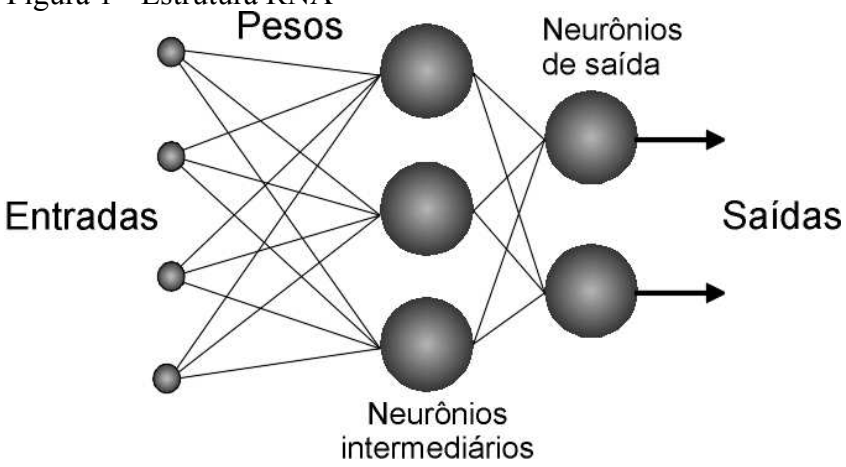
A metodologia utilizada neste projeto consiste em três partes: elaboração de um banco de dados de amostras de óleo de transformadores de potência, separando uma parte destes dados para o treinamento da rede e utilizando os demais dados para o posterior teste da rede; treinamento das redes neurais utilizando a ferramenta do Matlab e diferentes métodos e treinamento e; análise dos resultados obtidos com o banco de dados de teste da rede. Sendo necessário primeiramente, entender como funciona as redes neurais artificiais.

2.1 REDES NEURAIS ARTIFICIAIS

A Rede Neural Artificial (RNA) é um algoritimo matemático que utiliza uma estrutura de ligação baseada no funcionamento do cérebro humano. Possui a finalidade de definir uma saída numérica baseada em seus dados de entrada e nos pesos utilizados em sua rede. Estes pesos podem ser alterados com a utilização de métodos de aprendizado distintos, afim de permitir que a rede generalize para atender a maior quantidade de dados possível com a saída correta. Desta forma, a RNA pode ser utilizada para encontrar relações não-lineares. (HAYKIN, 2001).

Uma RNA é composta por diversos “neurônios” divididos em camadas, sendo a camada de saída obrigatória, com a quantidade de “neurônios” igual a quantidade de dados de saídas necessários. Além destas, camadas intermediárias (também conhecidas como camadas escondidas) podem ser inclusas, podendo ser variado o número e quantidade de “neurônios”. Na Figura 1 é ilustrado um exemplo genérico da estrutura de uma RNA.

Figura 1 - Estrutura RNA



Fonte: https://cerebromente.org.br/n05/tecnologia/image11.gif

No caso ilustrado na Figura 1, existem quatro dados de entrada, uma camada escondida com três "neurônios" e uma camada de saída com dois "neurônios", indicando que esta rede possui duas saídas. Redes com um número maior de "neurônios", além de serem mais "pesadas" também são mais demoradas para treinarem e, como veremos posteriormente, não produzem necessariamente um resultado melhor.

Cada dado é utilizado nas entradas de todos os "neurônios", por sua vez, a saída de cada "neurônio" é utilizado nas entradas de todos os "neurônios" da camada seguinte. Esse procedimento é repetido para cada camada da rede até a saída.

O "neurônio" de uma RNA é uma unidade de processamento básico que realiza uma soma ponderada entre as entradas e os pesos calculados, como mostrado na equação 1, e utiliza o resultado desta soma em uma função de ativação que irá gerar um resultado, como mostrado na equação 2. Esse resultado será propagado para os demais "neurônios". (SILVA, 2010).

$$u_k = \sum_{j=1}^{p} x_j * w_{kj} + \theta_k \quad (1)$$

$$y_k = \varphi(u_k) \quad (2)$$

Ou seja, a RNA possui uma estrutura simples, composta de operações matemáticas básicas como soma e multiplicação. Por isso, ao aplicar os dados em uma rede com os pesos já devidamente calculados, a resposta é gerada quase automaticamente considerando a capacidade computacional atual. Entretanto, a grande dificuldade desta técnica de solução é conseguir encontrar valores de pesos que garantam uma boa performance da rede, processo chamado de treinamento da rede.

O treinamento de RNAs é feito através de um banco de dados com amostras contendo as entradas da rede e os resultados esperados. Um método de treinamento é aplicado para ajustar os pesos com o objetivo de minimizar o erro entre a saída da rede e os resultados fornecidos. Existem vários métodos de treinamentos disponíveis. Na Tabela 2 são apresentados 12 métodos de treinamento e seus algoritmos disponíveis no MatLab (MATHWORKS, c2022).

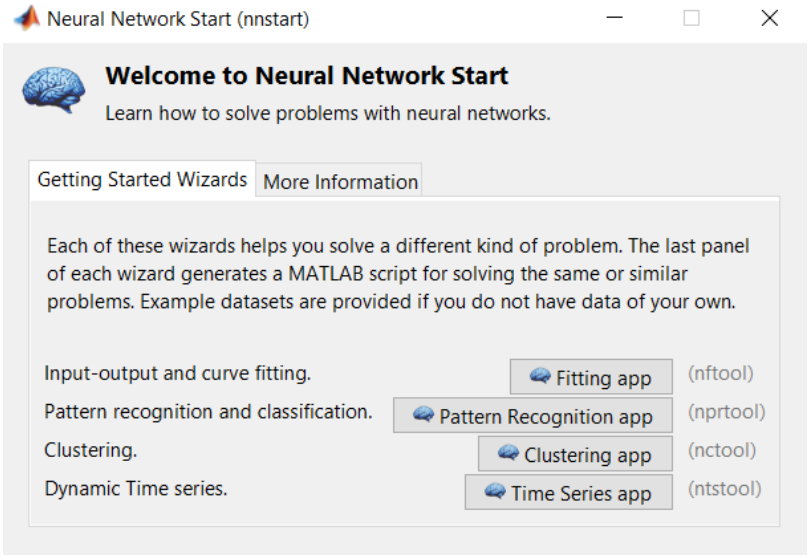
Tabela 2 - Métodos de Treinamento do MatLab

Método de Treinamento	Algoritmo
“trainlm”	Levenberg-Marquardt
“trainbr”	Bayesian Regularization
“trainbfg”	BFGS Quasi-Newton
“trainrp”	Resilient Backpropagation
“trainscg”	Scaled Conjugate Gradient
“traincgb”	Conjugate Gradient with Powell/Beale Restarts
“traincgf”	Fletcher-Powell Conjugate Gradient
“traincgp”	Polak-Ribiére Conjugate Gradient
“trainoss”	One Step Secant
“traingdx”	Variable Learning Rate Gradient Descent
“traingdm”	Gradient Descent with Momentum
“traingd”	Gradient Descent

Fonte: (MATHWORKS, c2022).

A utilização da ferramenta de criação de redes neurais artificiais no MatLab pode ser feita de duas maneiras: utilizando uma interface gráfica que o próprio MatLab disponibiliza ou através de comandos. Para utilizar a interface gráfica é necessário digitar “nnstart” na barra de comandos, para a janela mostrada na Figura 2 aparecer.

Figura 2 - Tela Inicial Ferramenta de Criação de RNA



Fonte: Matlab R2021a

Ao seguir as telas desta ferramenta, algumas informações serão requeridas, como os dados de treinamento que serão utilizados e a estrutura da rede (quantos neurônios serão utilizados em cada camada e qual o método de treinamento que será utilizado). Ao final, é oferecido a opção de gerar o código utilizado para a criação da rede.

2.2 BANCO DE DADOS

O banco de dados é fundamental no treinamento de redes neurais artificais, quanto mais dados são utilizados no treinamento de uma rede, mais esta rede será capaz de generalizar e, desta forma, possuirá uma maior eficiência em suas aplicações. Entretanto, conseguir um bom banco de dados pode se tornar um grande desafio em muitos casos. No caso deste trabalho, existe muito pouco material disponível na literatura com amostras de óleo retiradas de transformadores, fazendo-se necessário construir um banco de dados próprio. Os dados aqui utilizados foram minerados a partir de relatórios de análises cromatográficas do óleo de transformadores de potência, concedidos por uma empresa fabricante de transformadores, retirando-se destes relatórios apenas a quantidade de gases presentes no óleo, preservando quaisquer outros dados sensíveis da empresa. O processo de mineração destes dados será apresentado a seguir. Uma pequena amostra desses dados está disponível no Anexo 1.

2.2.1 Mineração de Dados

Para a realização deste projeto, foram concedidos mais de 11 (onze) mil relatórios em formato doc e docx.

A primeira dificuldade de extração de dados destes arquivos refere-se ao formato utilizado. Por se tratar de um formato proprietário da Microsoft, não é possível ler estes arquivos com linguagens de programação de forma simples. No Python (linguagem utilizada para realizar a mineração dos dados), existe uma biblioteca capaz de ler estes arquivos, entretanto os testes realizados com essa biblioteca neste projeto não obtiveram bons resultados. Na maior parte dos arquivos o programa não conseguiu ler a região onde os dados de interesse estavam. Para contornar este problema, foi desenvolvido um script para simular e automatizar a operação manual (mouse e teclado) como segue: abrir cada um dos arquivos; selecionar todo o seu conteúdo; copiar a seleção; fechar o arquivo ".doc ou .docx"; criar um novo arquivo com extensão ".txt"; abrir este arquivo ".txt"; colar o conteúdo e salvar. Desta forma, todos os arquivos foram convertidos para ".txt", os quais podem ser acessados e lidos empregando o Python.

Na segunda parte da construção do banco de dados, foi preciso encontrar padrões de apresentação das amostras dentro destes arquivos de forma a conseguir retirar o máximo de amostras possíveis. Por existirem arquivos de vários anos diferentes, nem todos possuíam a mesma formatação, então mais de um padrão foi necessário. Com esta etapa, as amostras encontradas foram salvas em um documento Excel com 5 colunas, sendo um para cada gás (H2, CH4, C2H6, C2H4, C2H2), e quase 6 (seis) mil linhas, cada linha correspondendo a uma cromatografia. Nenhum outro dado foi extraído dos relatórios.

2.2.1.1 Preparação do Ambiente para Conversão dos Arquivos

Antes de iniciar a conversão dos arquivos, foi necessário preparar o ambiente. Para preservar os arquivos fornecidos, evitando que fossem individamente apagados por algum erro, os arquivos foram copiados de sua localização original para uma pasta de backup. Todas as operações realizadas neste trabalho foram feitas nesta pasta com os arquivos copiados.

Estes arquivos estavam divididos em várias pastas e subpastas. Para facilitar a futura localização dos arquivos nos programas seguintes, foi criado um arquivo de comando em lotes do sistema operacional windows ".bat", para localizar os arquivos ".doc" e ".docx" dentro destas pastas e copiá-los para a raíz de uma outra pasta. O código fonte deste programa é apresentado no Apêndice 1. Ao final deste processo, haviam 11.593 arquivos ".doc" e ".docx" dentro da pasta "1. doc_data".

Com estes arquivos devidamente separados, uma nova pasta com o nome "2. txt_data" foi adicionada à pasta do projeto para armazenar os arquivos ".txt" que serão gerados.

2.2.1.2 Conversão de Arquivos de Word para Txt utilizando Python

Para realizar a conversão dos arquivos em formato do word para o formato ".txt", foi utilizada a biblioteca de código aberto Pyautogui do Python (Pyautogui, c2019), que permite simular ações empregando o mouse e teclado com linhas de comando. Para realizar este procedimento foi criado um arquivo em Python chamado "convert-pyautogui.py" disponível no Apêndice 2.

Este programa funciona da seguinte maneira: para cada documento dentro da pasta "1. doc_data", um novo arquivo com mesmo nome, porém com extensão ".txt" será adicionado dentro da pasta "2. txt_data"; após o arquivo ser gerado o programa abre o arquivo word que será processado e, com a biblioteca pyautogui, um comando para selecionar todo o texto do arquivo é dado ("Ctrl + T") e, em seguida, um comando para copiar este texto selecionado ("Ctrl + C"); com o texto selecionado, é dado um comando para fechar o arquivo atual ("Alt + F4"). Quando uma janela do Word é fechada com algum texto copiado, o Word mostra uma janela de pergunta para confirmar se a seleção copiada deve permanecer na memória, por isso, é necessário mais um comando para confirmar e fechar o documento ("Enter"). Com os arquivos copiados, o programa abre o arquivo ".txt" que foi gerado anteriormente; cola o texto copiado ("Ctrl+V"); salva o arquivo ("Ctrl+S") e fecha o arquivo ("Alt+F4").

Alguns comandos de tempo são colocados entre as ações para garantir que o dispositivo consiga realizar a terefa anterior antes de executar a próxima. Por fim, o programa deleta o arquivo do Word que foi processado. Este último comando é necessário para garantir que o processo continue de onde parou caso seja interrompido. Este programa possui a desvatagem de impossibilitar a utilização da máquina durante o processo de conversão dos arquivos e de

demorar muito tempo. Considerando que cada conversão durou 13 segundos (somatório de todos os comandos de tempo), e foram convertidos 11.593 arquivos, foi necessário aproximadamente 42 horas para converter todos os arquivos.

2.2.1.3 Extraindo Amostras de Arquivos em Txt

Com todos os arquivos convertidos para ".txt", é necessário extrair os dados de uma forma que seja de fácil apresentação. Para realizar essa tarefa foi criado o arquivo "extract.py" disponível no Apêndice 3. Seu funcionamento baseia-se em utilizar expressões regulares para pesquisar em cada linha do arquivo ".txt" um determinado padrão, sendo que dentro deste padrão deve conter o valor em ppm dos gases dissolvidos no óleo. Para cada arquivo dentro da pasta "2. txt_data", o programa irá ler o conteúdo deste arquivo e enviar este conteúdo para três funções que encontram esses padrões ("pattern1", "pattern2", "pattern3"), caso alguma destas funções retorne algum valor, o programa irá verificar se esses valores encontrados já não foram encontrados antes (para evitar amostras duplicadas) através de uma outra função ("allowSampleAppending") e adicionará esses valores ao vetor "gases". Neste programa foram criados três padrões diferentes. Com esses padrões foram retiradas 5835 amostras de óleo. Por fim, o programa salva esses dados em um arquivo Excel, utilizando a biblioteca Pandas.

2.2.2 Banco de Dados IEEE

Além das amostras adquiridas através da mineração dos arquivos de relatórios de análise cromatográfica, também foram utilizados neste projeto o banco de dados disponível no IEEEDataPort (IEEE, c2022). Este banco de dados possui 201 amostras, estas amostras foram adicionadas às amostras adquiridas no processo de mineração, resultando em um total de 6036 amostras.

2.2.3 Divisão do Banco de Dados

O banco de dados utilizado neste projeto foi dividido em duas partes, uma parte utilizada para o treinamento da rede e a outra utilizada para testar a eficiência da rede desenvolvida. Esta divisão foi feita considerando apenas as amostras que apresentam algum tipo de falha indicada na Tabela 1. Consequentemente, primeiro foi necessário remover do banco de dados as amostras que não apresentam nenhum tipo de falha e, em seguida, distribuir as amostras restantes para estas duas finalidades (treinamento e teste). As amostras de treinamento foram escolhidas aleatoriamente de modo a manter a mesma proporção entre cada tipo de falta, e todas as outras amostras foram utilizadas para o teste.

2.2.3.1 Aplicação da Norma IEC

A norma IEC classifica as faltas em seis tipos diferentes conforme a Tabela 1. Entretanto, existem casos em que a concentração dos gases de uma determinada amostra, não se encaixa em nenhuma das razões apresentadas na Tabela 1. Desta forma, para uma concentração qualquer de gases, há sete classificações possíveis:

- PD – Descarga Parcial
- D1 – Descarga de Baixa Energia
- D2 – Descarga de Alta Energia
- T1 – Falta Térmica $t < 300ºC$
- T2 – Falta Térmica $300ºC < t < 700ºC$
- T3 – Falta Térmica $t > 700ºC$
- NI – Não Identificado

Esta sétima categoria, representa amostras de transformadores que estão em bom funcionamento ou que este método não conseguiu identificar nenhuma falta. Como os dados foram obtidos de um monitoramento contínuo de vários transformadores que apresentam uma longa vida útil, a maior parte dos dados entram nesta categoria, como pode ser visto adiante neste trabalho.

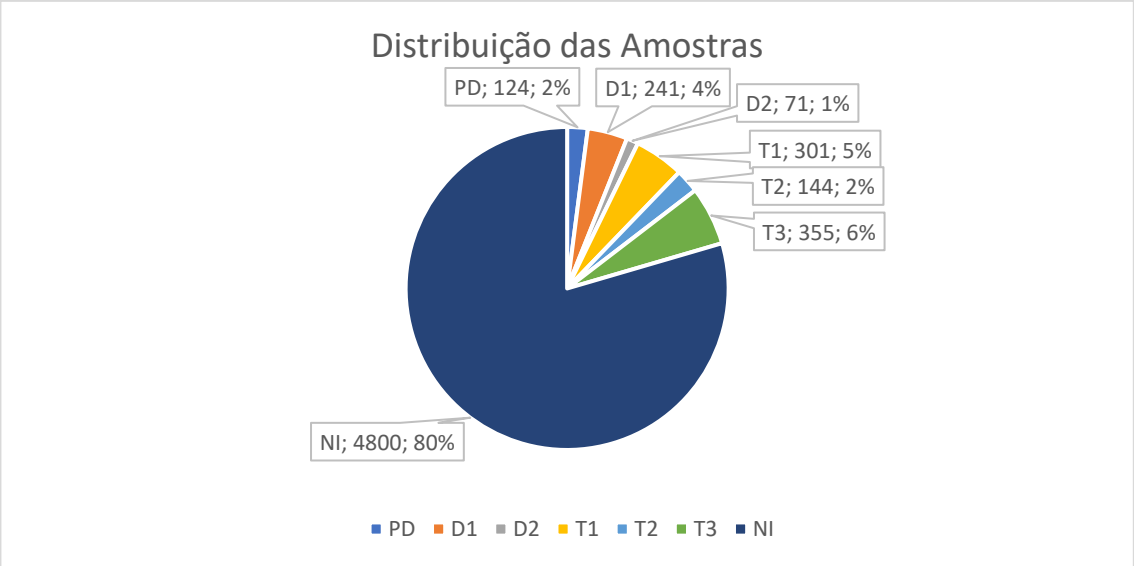
Uma das características da classificação apresentada na Tabela 1 é que, dependendo da amostra, pode haver mais de uma classificação possível, caso sejam da mesma categoria como, por exemplo, T1 e T2. Essa característica não é desejável neste trabalho pois, na saída da rede será aplicada uma função para arredondar os valores para zeros e um, sendo que somente uma classificação será possível. Por isso, na elaboração do código para aplicação da norma, foi realizado uma modificação para que somente uma classicação fosse possível.

Considerando os pontos apresentados, foi criado uma função em Matlab para realizar esta classificação, esta função está sendo apresentada no Apêndice 4. Este código recebe um vetor em linha com a concentração dos cinco gases e retorna um texto com a abreviatura de uma das sete classificações possíveis de falta. A ordem dos gases no vetor de entrada deve ser a seguinte: H2, CH4, C2H6, C2H4 e C2H2. Esta função foi nomeada como “applyIEC”.

2.2.3.2 Separação das Amostras com Faltas

Quando se aplica a função “applyIEC” em todas as amostras do banco de dados, é observado que a maior parte das amostras são classificadas como não identificadas. Essa divisão pode ser observada através do Gráfico 1.

Gráfico 1 – Distribuição Total das Amostras

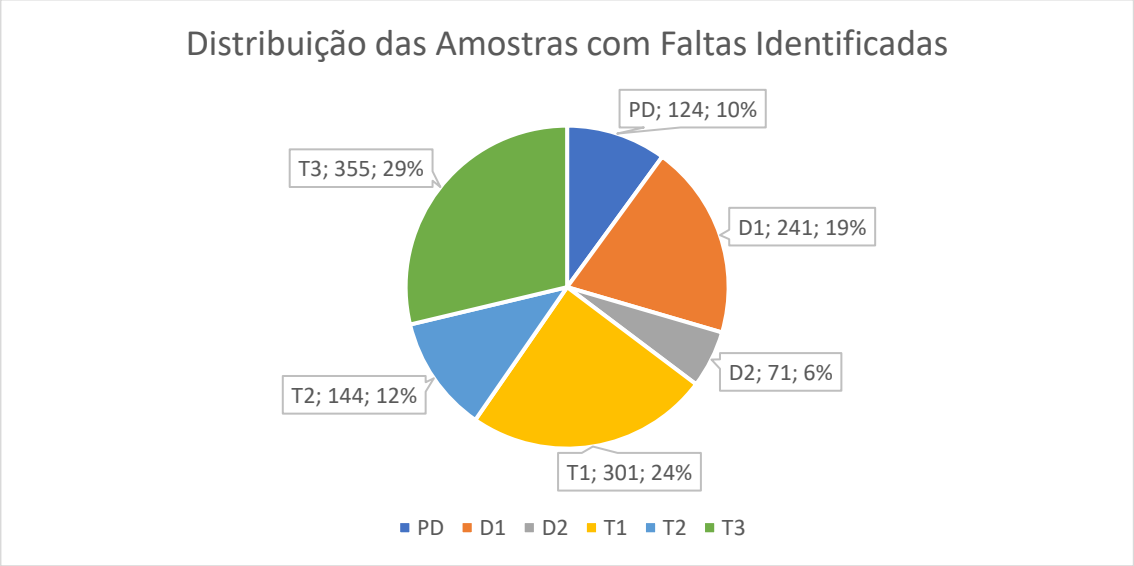


Fonte: Elaborado pelo Autor.

A partir dos dados conclui-se que de todas as 6036 amostras apenas 1236 apresentam algum tipo de falta identificada conforme o método IEC (Tabela 1). Para separar as amostras

que contém faltas das amostras que não contém nenhuma falta identificada, foi criado um programa no MatLab, apresentado no Apêndice 5. Este programa carrega os dados de uma variável chamada treinamento dentro de um arquivo chamado "Banco de Dados Completo.mat". Ao terminar, o programa salva novamente a variável treinamento, com os dados filtrados, dentro do arquivo "Banco de Dados Apenas Faltas.mat". Retirando esses dados não identificados, a nova distribuição do banco de dados resulta como apresentado no Gráfico 2.

Gráfico 2 - Distribuição das Amostras com Faltas Identificadas



Fonte: Elaborado pelo Autor.

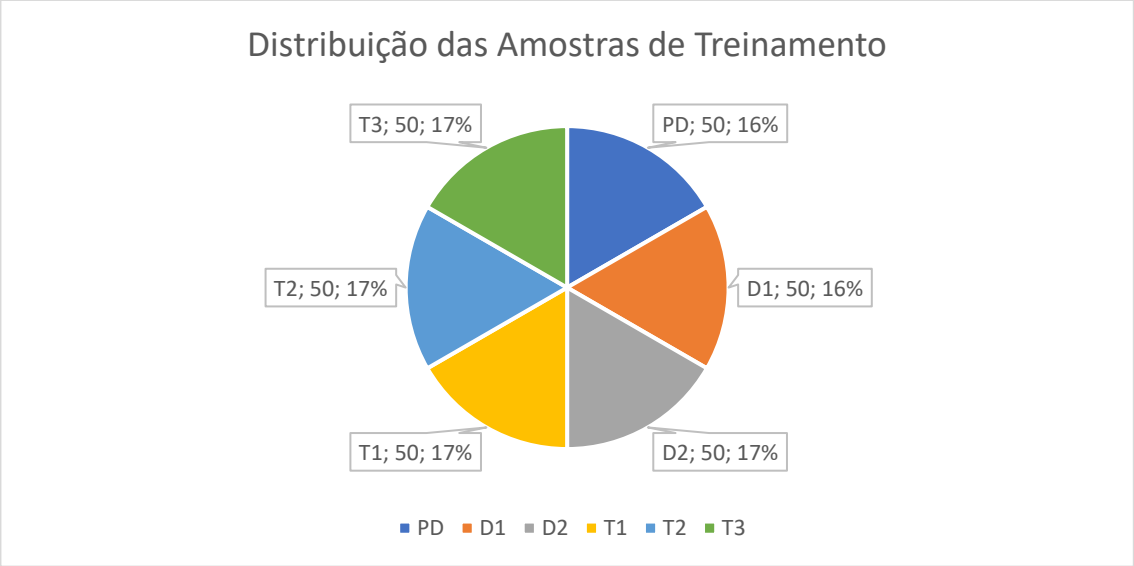
2.2.3.3 Divisão das Amostras para Treinamento e Testes

Podemos observar que a distribuição das amostras alcançada no Gráfico 2 apresenta proporções diferentes para cada tipo de falta. Para o tipo de falta T3 foram identificadas 355 ocorrências nas amostras, representando 29% do total. Entretanto, para o tipo de falta D2 foram identificadas apenas 71 ocorrências, representando apenas 6% do total.

Para as amostras de treinamento da rede, é interessante que haja uma proporção igualitária entre os resultados para, deste modo, evitar que a rede predomine uma determinada resposta. Desta forma, os dados de treinamento devem conter menos amotras para cada tipo de falta, do que a quantidade de amostras para a falta com menor quantidade, neste caso 71 amostras. Além disso, algumas destas amostras devem ser guardadas para os testes da rede.

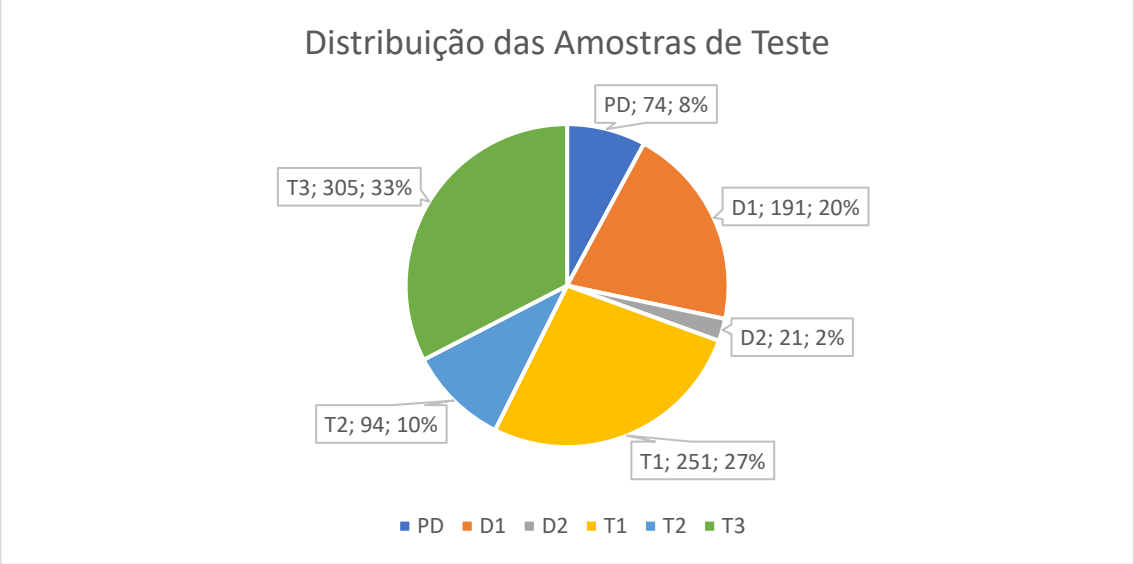
Neste trabalho optou-se por utilizar 50 amostras para cada tipo de falta para o treinamento da rede, resultando em 300 amostras no total. Totas as outras amostras não utilizadas são usadas para o teste das redes. As distribuições destas amostras estão sendo representadas no Gráfico 3 e no Gráfico 4, respectivamente.

Gráfico 3 - Distribuição das Amostras de Treinamento



Fonte: Elaborado pelo Autor.

Gráfico 4 - Distribuição das Amostras de Teste



Fonte: Elaborado pelo Autor.

Para realizar a divisão destes dados foi criado o programa em MatLab disponível no Apêndice 6. Este programa carrega o arquivo de banco de dados “Banco de Dados Apenas Faltas.mat” e gera quatro matrizes xr, yr, xt, yt, sendo que a matriz xr corresponde às amostras de treinamento e a matriz yr corresponde às suas respectivas saídas; e, da mesma forma, as

matrizes xt e yt correspondem às amostras e seus tipos de faltas para o teste. Essas quatro matrizes são salvas ao final em um arquivo chamado “Banco de Dados Separado.mat”.

2.3 MÉTODO DE AVALIAÇÃO DO DESEMPENHO DA REDE

Na avaliação da performance da rede são necessários três fatores: os dados de entrada, os dados de saída desejados e a rede neural a ser avaliada. Nesta etapa do processo, as redes ainda não foram criadas, mas a definição do metódo de avaliação já se faz necessário pois guiará o processo de seleção da rede durante o treinamento.

As redes utilizadas produzirão seis saídas, com valores entre zero e um, dependendo dos cinco valores da entrada. É necessária a conversão destes valores, de forma a se garantir uma resposta válida para qual falta aquela amostra é mais provável de pertencer. Sabe-se que quanto mais próximo de um o valor, maior é a probabilidade de a falta referente aquele valor seja a falta identificada.

Com essa informação, existem duas formas de lidar com este problema. A primeira forma seria definindo um valor limiar, em que qualquer valor acima desse limiar seria arredondado para um e qualquer valor abaixo seria arredondado para zero. Desta forma, qualquer valor acima desse limiar seria identificado como positivo para a falta. Este método possui a vantagem de identificar mais de uma falta para uma saída, o que realmente acontece quando aplicamos a Tabela 1 em alguma amostra (algumas amostras podem marcar faltas D1 e D2, T1 e T2 ou T2 e T3 ao mesmo tempo). Entretanto, escolher um valor de limiar para uma única rede já seria uma tarefa árdua que, não necessariamente, levaria a um bom resultado mesmo com muitos testes; para várias redes, como é o caso deste projeto, isso seria inviável. A segunda opção, é converter o valor mais próximo de um (maior valor) para um e os demais para zero. Esta forma possui a desvantagem de dar uma saída única.

Neste trabalho foi utilizado a opção de arredondar o maior número. O Apêndice 7 mostra o programa, feito no MatLab, para converter o resultado da rede. Este programa recebe um vetor coluna com valores entre zero e um e retorna um vetor coluna com uma das posições com o valor um e as demais com o valor zero, sendo esta posição do valor um o maior valor do vetor inicial. Este programa está em forma de função no MatLab para ser utilizado nos demais programas seguintes e foi renomeado como “roundBiggest.m”. Na Tabela 3 é apresentada como

essa conversão é realizada. Para o caso da amostra dessa tabela, seria identificada a falta como sendo do tipo D1.

Tabela 3 - Demonstração de Arredendamento da Saída da Rede

Falta	Saída da Rede	Saída Convertida
DP	0,0238	0
D1	0,9643	1
D2	0,0003	0
T1	0,0138	0
T2	0,0139	0
T3	0,0163	0

Fonte: Elaborado pelo Autor.

Com a configuração adequada da saída da rede neural, pode-se comparar estes valores com os valores presentes no vetor alvo/solução esperada. Para realizar esta comparação, foi criado o programa "compareOutput.m" presente no Apêndice 8. Esta também é uma função feita no MatLab e que será utilizada posteriormente dentro de outro algoritmo.

Esta função recebe duas matrizes como parâmetros. A primeira, denominada de "netOutput" representa a matriz de saída da rede e a segunda "target" representa os valores experados de saida. No caso deste programa, essas matrizes possuem seis linhas fixas e um número variável de colunas, dependendo do número de amostras. Todavia o programa suportaria outros casos com diferentes números de linhas. A única requisição para este programa funcionar é que a dimensão das duas matrizes sejam iguais. Para cada coluna da matriz "netOutput", o programa verificará para cada linha se o valor da linha é igual a um e se o valor da matriz "target" naquela mesma posição também possui o valor igual a um, se esta condição for verdadeira, então o programa colocará na matriz "output" o valor um, indicando que a indicação de falta desta amostra está correta.

Para realizar a análise dos resultados, é interessante que seja possível visualizar para cada um dos tipos de falta, qual foi o número de acertos, o número de amostras e a porcentagem de acertos. No Apêndice 9 é apresentado o programa desenvolvido para gerar o relatório de resultados por categoria de falta da RNA. Este programa recebe três parâmetros: o vetor de entrada da rede com cinco linhas e colunas dependendo do número de amostras; o vetor de saídas esperadas com seis linhas e colunas dependendo do número de amostras e, a rede que se deseja analisar. Este programa retorna uma matriz com seis linhas e três colunas, cada linha representando um tipo de falta na mesma sequência da Tabela 1. Na primeira coluna é

apresentado o número de acertos para aquele tipo de falta, na segunda coluna o número de amostras para aquele tipo de falta e na terceira coluna, a relação de acertos em porcentagem.

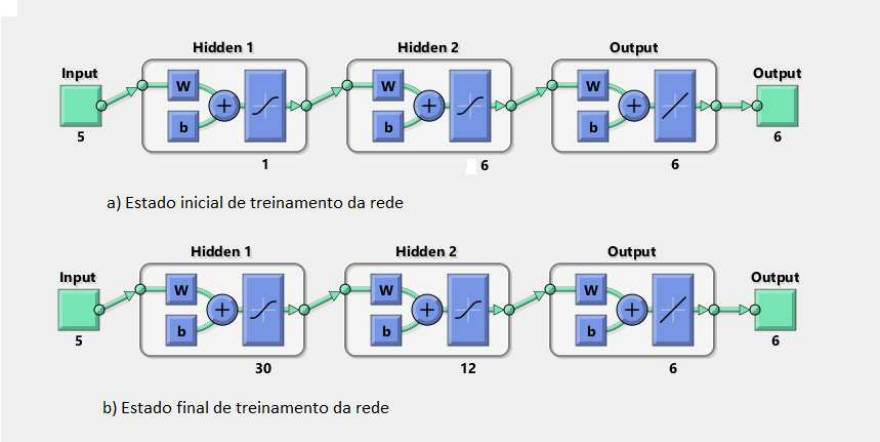
Por fim, com o resultado para cada tipo de erro para uma determinada rede, podemos calcular qual foi o resultado final desta rede fazendo uma média da terceira coluna do programa anterior. O programa disponível no Apêndice 10 retorna apenas o valor complementar dessa média para que se possa analisar a procentagem média de erros da rede. Os parâmetros de entrada são os mesmos do anterior pois a própria função anterior “resultCategory” é utilizada neste programa. Esta função foi denominada de “evaluateErro”.

2.4 MÉTODO DE TREINAMENTO DA REDE

Utilizando as bibliotecas prontas do MatLab, várias redes neurais artificiais foram treinadas variando a quantidade de neurônios na primeira e segunda camada, de modo a encontrar a melhor combinação. Além disso, também foram utilizados métodos de treinamentos diferentes para a rede. O MatLab oferece 12 tipos de métodos de treinamento conforme apresentado na seção 2.1 na Tabela 2.

Para cada um desses métodos, uma rede reural foi criada variando a quantidade de neurônios nas camadas intermediária para tentar otimizar os resultados obtidos, sendo utilizados 5 neurônios na primeira camada (correspondendo ao número de entradas) e 6 neurônios na camada de saída (correspondendo ao número de saídas). Os neurônios da primeira camada intermediária foram variados entre 1 a 30 e os neurônios da segunda camada intermediária foram variados entre 6 a 12. Desta forma, a estrutura da rede foi alterada do modelo apresentado na Figura 3 “a” até o modelo apresentado na Figura 3 “b”, totalizando 180 estruturas de rede utilizadas para cada método de treinamento. Como foram utilizados 12 métodos, então foram treinados ao todo 2160 redes. Por fim, cada rede foi treinada 2 vezes para diminuir os efeitos da aleatoriedade dos dados, o que totaliza 4320 treinamentos de redes.

Figura 3 – Estruturas inicial e final estabelecidas para as Redes



Fonte: Elaborado pelo Autor.

Cada treinamento foi realizado utilizando os dados treinamento “xr” e “yr” obtidos no capítulo 2.2.3.3. Após cada treinamento o desempenho da rede foi avaliado utilizando o programa “evaluateErro” do capítulo 2.3, usando os dados de teste “xt” e “yt”. Apenas a rede que obteve o menor erro foi salva para análise dos resultados para cada método. Para as demais redes, uma matriz contendo o valor do erro de cada rede foi salva, esta matriz possui 7 colunas (representando os neurônios de 6 a 12 da segunda camada intermediária) e 30 linhas (representando os neurônios da primeira camada intermediária). O programa utilizado para fazer o treinamento dessas redes está disponível no Apêndice 11.

Para cada método de treinamento, uma imagem representando a variação dos resultados para cada combinação de rede foi criada, nesta imagem é gerado um retângulo em amarelo que destaca qual foi a combinação de neurônios que apresentou o melhor resultado para a rede. Cada retângulo dessa imagem representa uma combinação, sendo que cada coluna representa uma quantidade de neurônios na primeira camada da rede (conforme representam os números na parte superior da imagem), e cada linha representa uma quantidade de neurônios na segunda camada da rede (conforme representam os números na parte lateral esquerda da imagem). Os tons de cinza representam os erros das RNAs, sendo que os tons mais escuros indicam maiores erros e os tons mais claros menores erros. O programa utilizado para a elaboração dessas imagens está disponível no Apêndice 12.

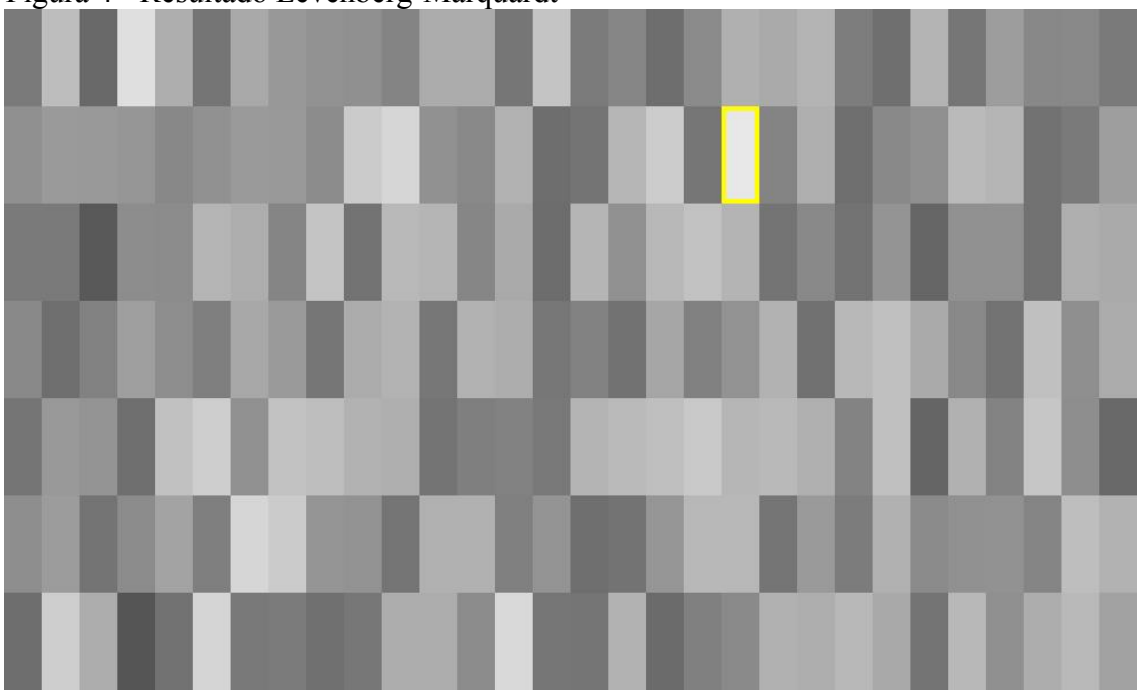
3 RESULTADOS

A seguir são demostrados os resultados obtidos para cada método de treinamento disponível na Tabela 2

3.1 LEVENBERG-MARQUARTD

Na Figura 4 a seguir é mostrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais claros, indicando que várias dessas combinações apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 20 neurônios na primeira camada e 7 neurônios na segunda camada.

Figura 4 - Resultado Levenberg-Marquardt



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 4 a seguir. Estes resultados produzem uma média de acertos de 91%.

Tabela 4 - Resultados Levenberg-Marquardt com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	44	50	88
D1	47	50	94
D2	49	50	98
T1	45	50	90
T2	49	50	98
T3	39	50	78

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 5 a seguir. Estes resultados produzem uma média de acertos de 55,60%.

Tabela 5 - Resultados Levenberg-Marquardt com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	47	74	63,51
D1	66	191	34,56
D2	9	21	42,86
T1	197	251	78,49
T2	83	94	88,30
T3	86	305	28,20

Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 73,3%.

3.2 BAYESIAN REGULARIZATION

Na Figura 5 a seguir é mostrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais claros, indicando que várias dessas combinações apresentaram bons resultados, com destaque para as combinações com poucos neurônios na camada principal (entre 4 a 9). A combinação que apresentou o melhor resultado (destacado em amarelo) possui 5 neurônios na primeira camada e 6 neurônios na segunda camada.

Figura 5 - Resultados Bayesian Regularization

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 6 a seguir. Estes resultados produzem uma média de acertos de 98%.

Tabela 6 - Resultados Bayesian Regularization com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	49	50	98
D1	50	50	100
D2	50	50	100
T1	49	50	98
T2	48	50	96
T3	38	50	96

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 7 a seguir. Estes resultados produzem uma média de acertos de 82,64%.

Tabela 7 - Resultados Bayesian Regularization com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	63	74	85,16
D1	185	191	96,86
D2	9	21	42,86
T1	243	251	96,81
T2	87	94	92,55
T3	249	305	81,64

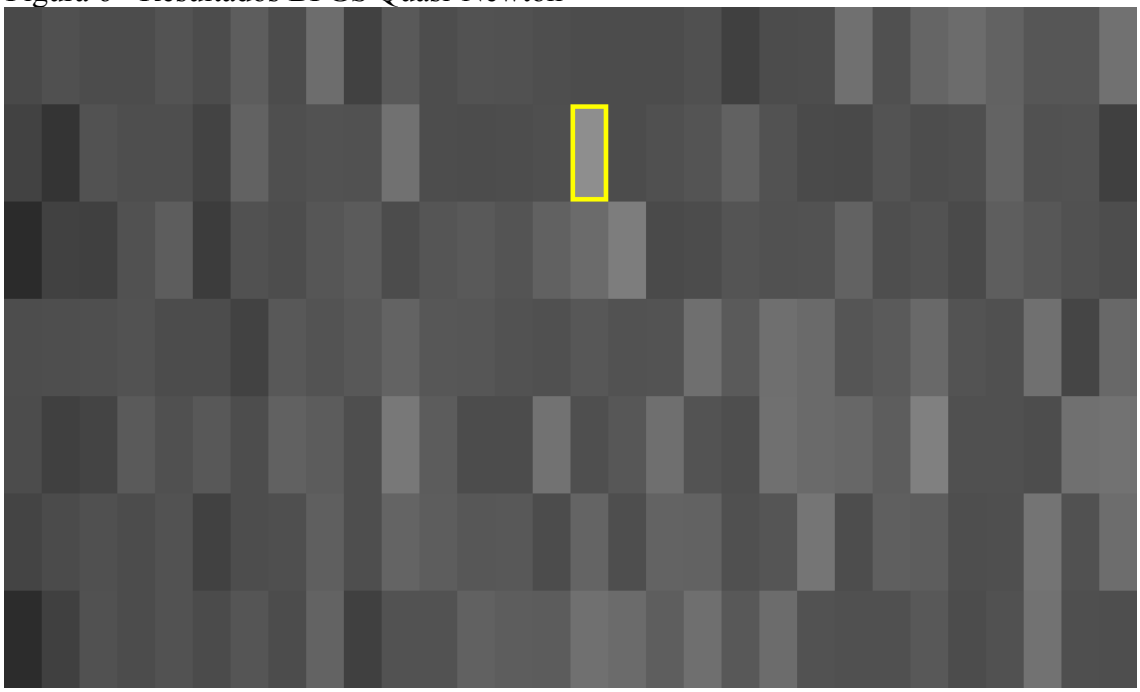
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 90,32%.

3.3 BFGS QUASI-NEWTON

Na Figura 6 a seguir é motrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 16 neurônios na primeira camada e 7 neurônios na segunda camada.

Figura 6 - Resultados BFGS Quasi-Newton



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 8 a seguir. Estes resultados produzem uma média de acertos de 55,67%.

Tabela 8 - Resultados BFGS Quasi-Newton com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	36	50	72
D1	21	50	42
D2	44	50	88
T1	0	50	0
T2	43	50	86
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 9 a seguir. Estes resultados produzem uma média de acertos de 30,43%.

Tabela 9 - Resultados BFGS Quasi-Newton com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	32	74	43,24
D1	48	191	25,13
D2	5	21	23,81
T1	0	251	0
T2	85	94	90,46
T3	0	305	0

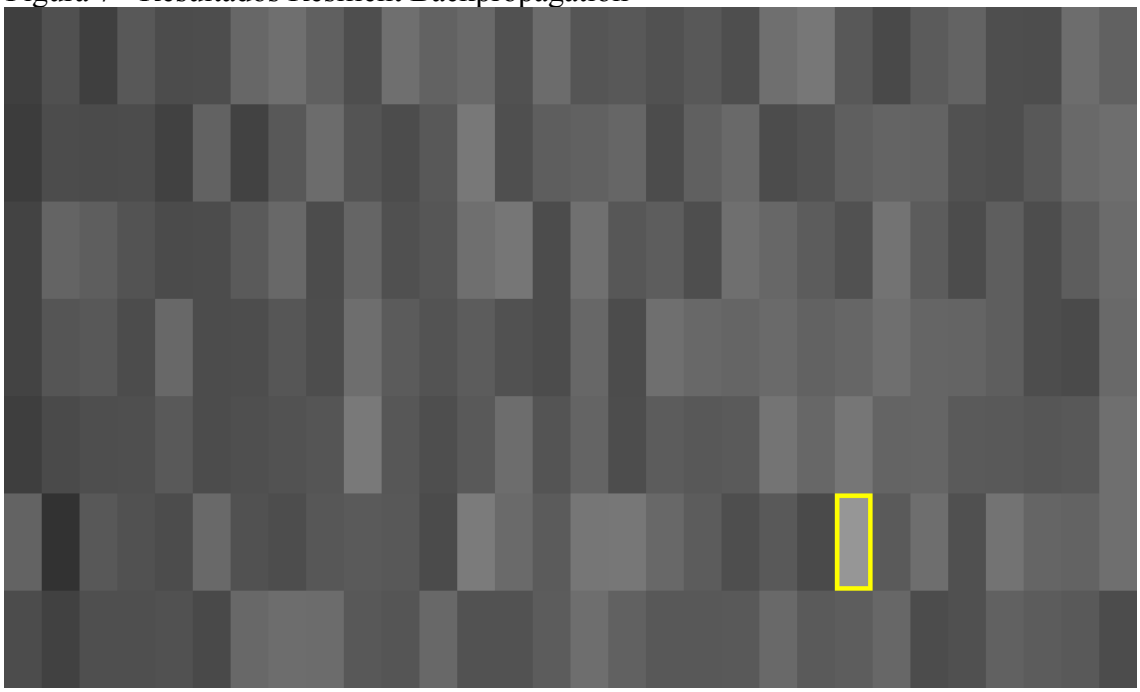
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 43,05%.

3.4 RESILIENT BACKPROPAGATION

Na Figura 7 a seguir é mostrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 23 neurônios na primeira camada e 11 neurônios na segunda camada.

Figura 7 - Resultados Resilient Backpropagation



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 10 a seguir. Estes resultados produzem uma média de acertos de 58,67%.

Tabela 10 - Resultados Resilient Backpropagation com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	41	50	82
D1	31	50	62
D2	41	50	82
T1	0	50	0
T2	40	50	80
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 11 a seguir. Estes resultados produzem uma média de acertos de 31,42%.

Tabela 11 - Resultados Resilient Backpropagation com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	43	74	58,11
D1	60	191	31,41
D2	5	21	23,81
T1	1	251	0,40
T2	70	94	74,47
T3	1	305	0,33

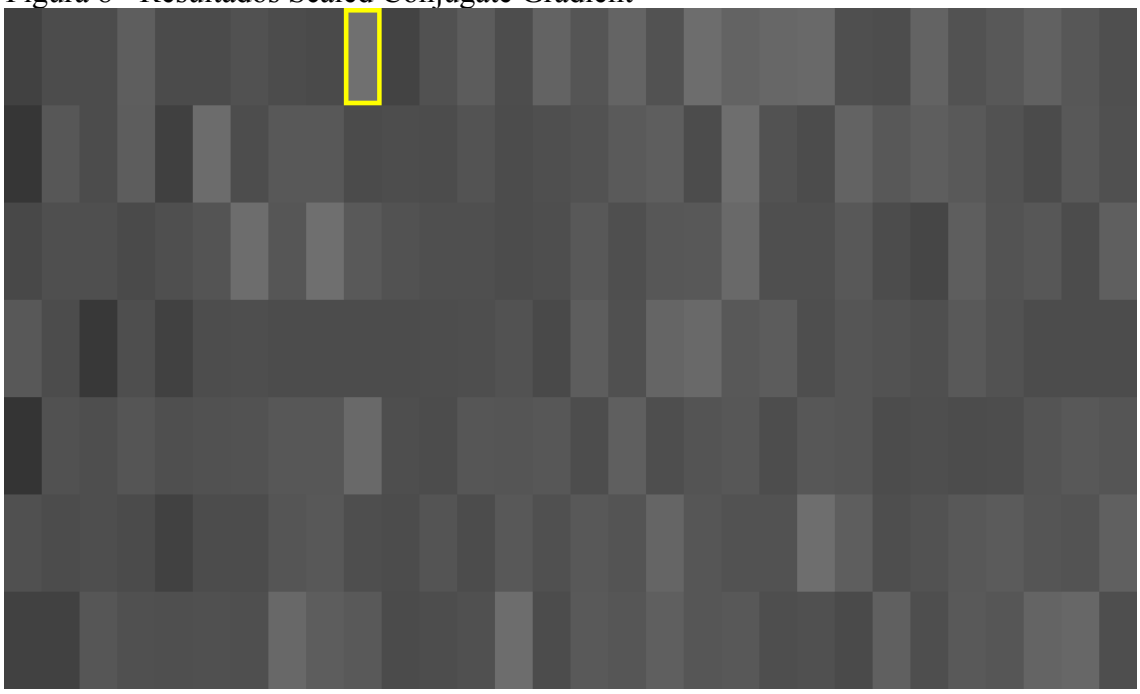
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 45,04%.

3.5 SCALED CONJUGATE GRADIENT

Na Figura 8 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 10 neurônios na primeira camada e 6 neurônios na segunda camada.

Figura 8 - Resultados Scaled Conjugate Gradient



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 12 a seguir. Estes resultados produzem uma média de acertos de 44%.

Tabela 12 - Resultados Scaled Conjugate Gradient com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	40	50	80
D1	18	50	36
D2	0	50	0
T1	49	50	98
T2	2	50	4
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 13 a seguir. Estes resultados produzem uma média de acertos de 29,67%.

Tabela 13 - Resultados Scaled Conjugate Gradient com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	41	74	55,41
D1	47	191	24,61
D2	0	21	0
T1	246	251	98,01
T2	0	94	0
T3	0	305	0

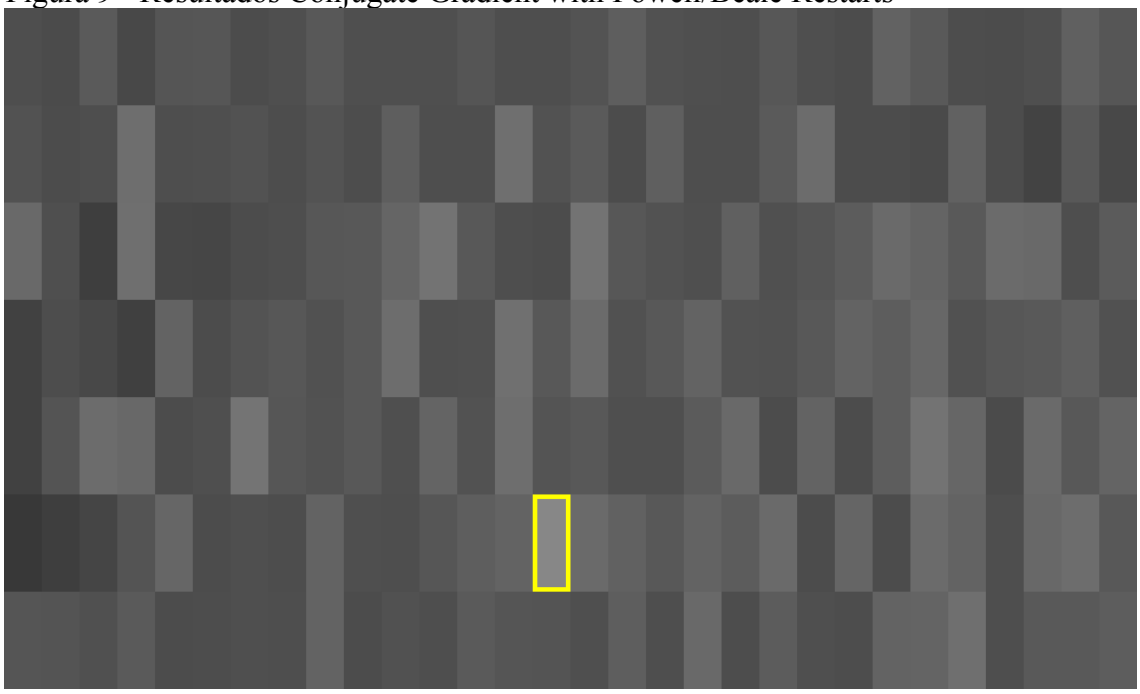
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 36,84%.

3.6 CONJUGATE GRADIENT WITH POWELL/BEALE RESTARTS

Na Figura 9 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 15 neurônios na primeira camada e 11 neurônios na segunda camada.

Figura 9 - Resultados Conjugate Gradient with Powell/Beale Restarts



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 14 a seguir. Estes resultados produzem uma média de acertos de 53%.

Tabela 14 - Resultados Conjugate Gradient with Powell/Beale Restarts com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	37	50	74
D1	17	50	34
D2	45	50	90
T1	35	50	70
T2	2	50	4
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 15 a seguir. Estes resultados produzem uma média de acertos de 28,60%.

Tabela 15 - Resultados Conjugate Gradient with Powell/Beale Restarts com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	34	74	45,95
D1	42	191	21,99
D2	6	21	28,57
T1	170	251	67,73
T2	6	94	6,38
T3	3	305	0,98

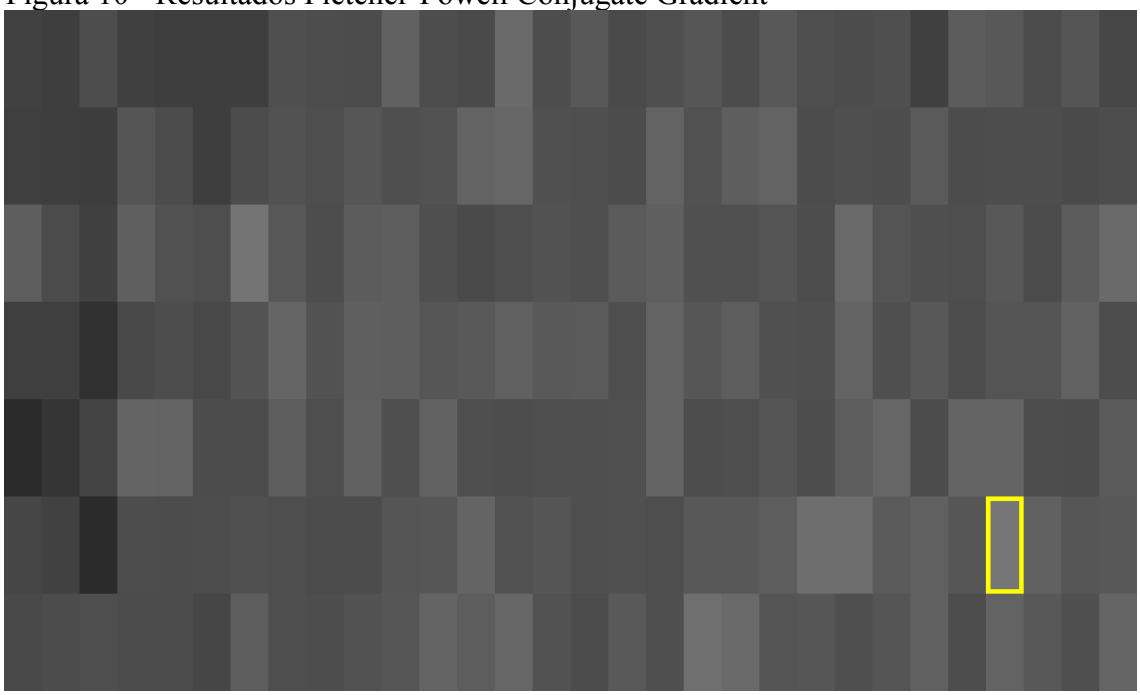
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 40,80%.

3.7 FLETCHER-POWELL CONJUGATE GRADIENT

Na Figura 10 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 27 neurônios na primeira camada e 11 neurônios na segunda camada.

Figura 10 - Resultados Fletcher-Powell Conjugate Gradient



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 16 a seguir. Estes resultados produzem uma média de acertos de 46,33%.

Tabela 16 - Resultados Fletcher-Powell Conjugate Gradient com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	37	50	74
D1	23	50	46
D2	45	50	90
T1	1	50	2
T2	10	50	20
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 17 a seguir. Estes resultados produzem uma média de acertos de 20,77%.

Tabela 17 - Resultados Fletcher-Powell Conjugate Gradient com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	34	74	45,95
D1	44	191	23,04
D2	5	21	23,81
T1	31	251	12,35
T2	18	94	19,15
T3	1	305	0,38

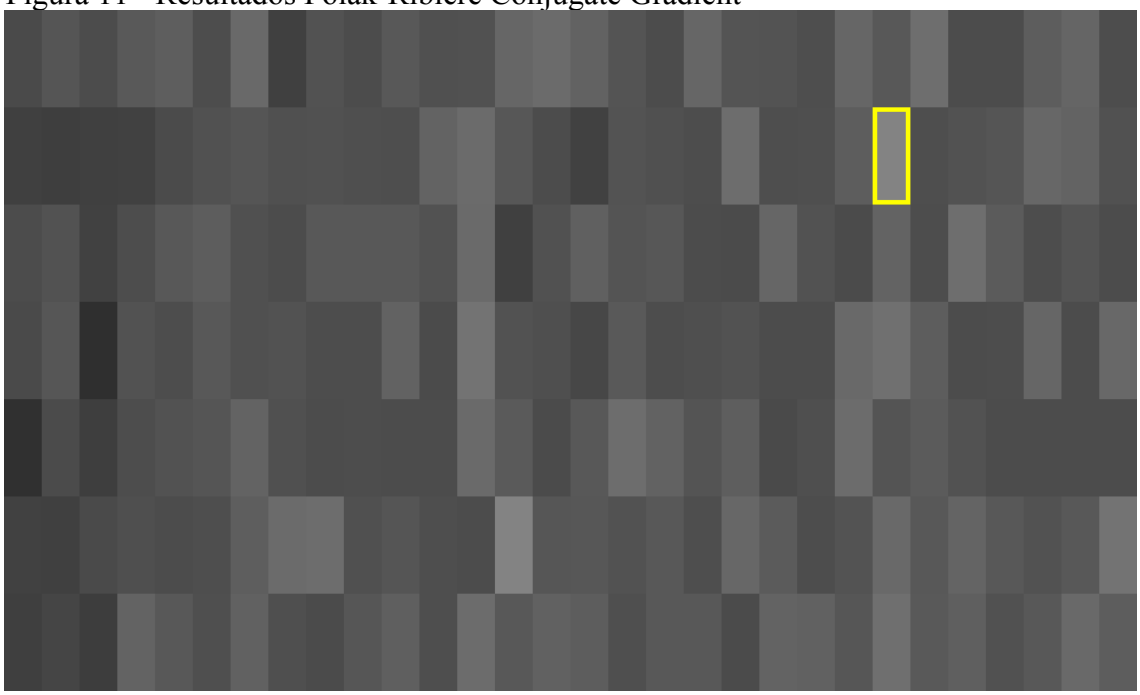
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 33,55%.

3.8 POLAK-RIBIÉRE CONJUGATE GRADIENT

Na Figura 11 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 24 neurônios na primeira camada e 7 neurônios na segunda camada.

Figura 11 - Resultados Polak-Ribiére Conjugate Gradient



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 18 a seguir. Estes resultados produzem uma média de acertos de 51,33%.

Tabela 18 - Resultados Polak-Ribiére Conjugate Gradient com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	37	50	74
D1	17	50	34
D2	11	50	22
T1	43	50	86
T2	23	50	46
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 19 a seguir. Estes resultados produzem uma média de acertos de 33,43%.

Tabela 19 - Resultados Polak-Ribiére Conjugate Gradient com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	34	74	45,95
D1	46	191	24,08
D2	2	21	9,52
T1	213	251	84,86
T2	34	94	36,17
T3	0	305	0

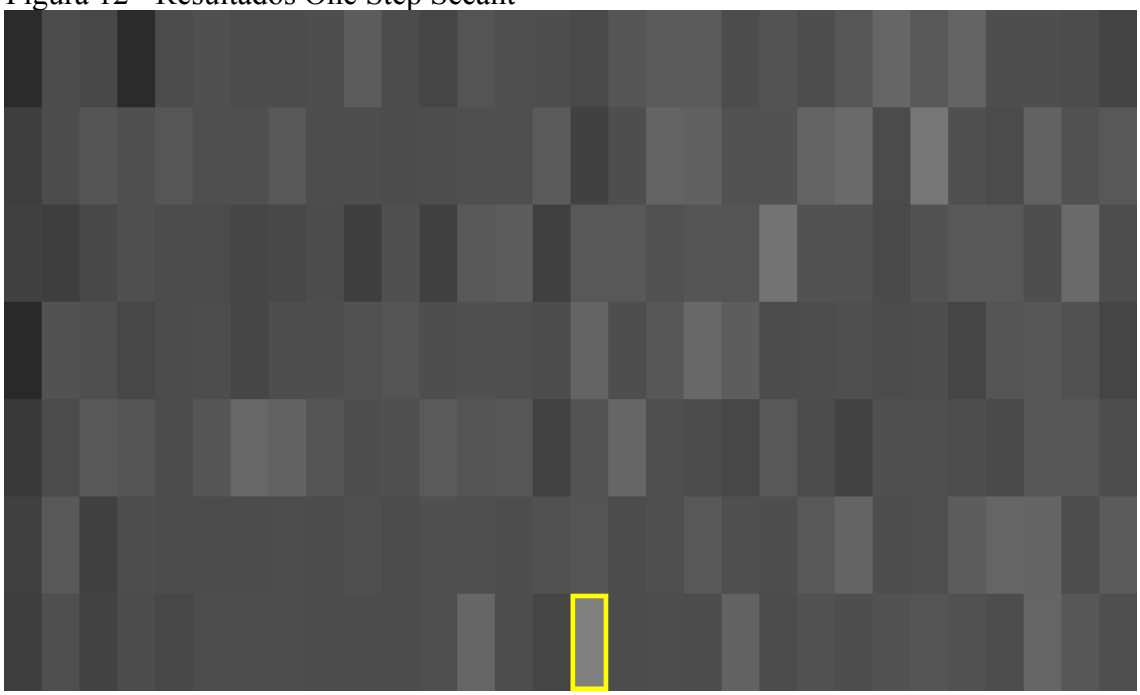
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 42,38%.

3.9 ONE STEP SECANT

Na Figura 12 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 16 neurônios na primeira camada e 12 neurônios na segunda camada.

Figura 12 - Resultados One Step Secant



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 20 a seguir. Estes resultados produzem uma média de acertos de 50%.

Tabela 20 - Resultados One Step Secant com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	35	50	70
D1	18	50	36
D2	38	50	76
T1	1	50	2
T2	35	50	70
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 21 a seguir. Estes resultados produzem uma média de acertos de 24,83%.

Tabela 21 - Resultados One Step Secant com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	30	74	40,54
D1	47	191	24,61
D2	3	21	14,29
T1	1	251	0,40
T2	65	94	69,15
T3	0	305	0

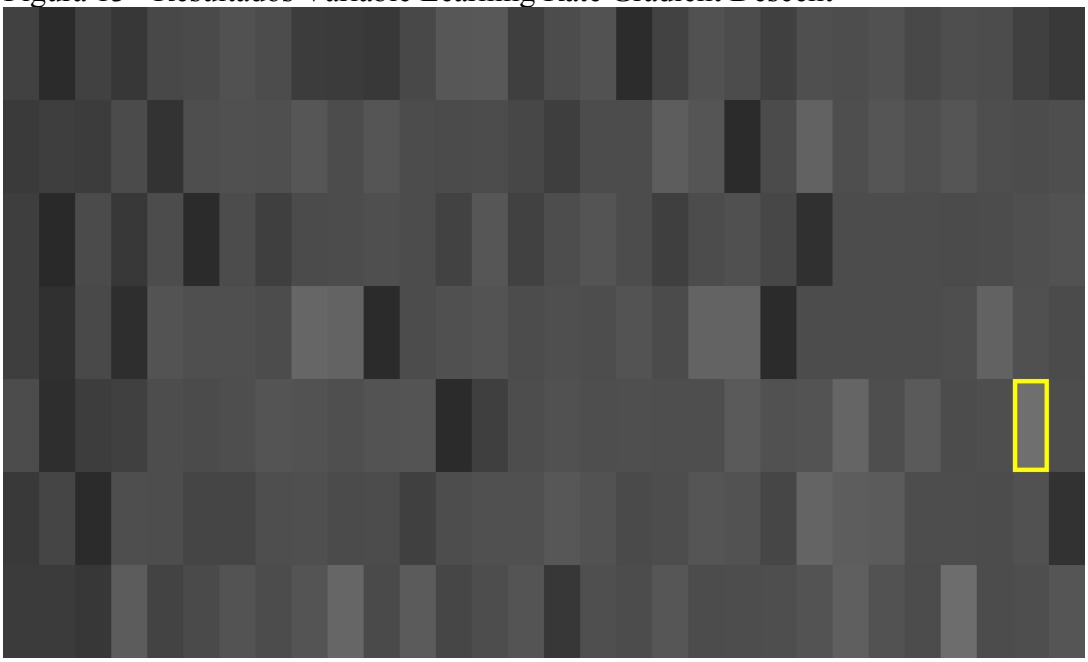
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 37,42%.

3.10 VARIABLE LEARNING RATE GRADIENT DESCENT

Na Figura 13 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 29 neurônios na primeira camada e 10 neurônios na segunda camada.

Figura 13 - Resultados Variable Learning Rate Gradient Descent



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 22 a seguir. Estes resultados produzem uma média de acertos de 43,67%.

Tabela 22 - Resultados Variable Learning Rate Gradient Descent com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	35	50	70
D1	17	50	34
D2	0	50	0
T1	15	50	30
T2	41	50	82
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 23 a seguir. Estes resultados produzem uma média de acertos de 31,93%.

Tabela 23 - Resultados Variable Learning Rate Gradient Descent com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	34	74	45,95
D1	38	191	19,90
D2	1	21	4,76
T1	114	251	45,42
T2	71	94	75,53
T3	0	305	0

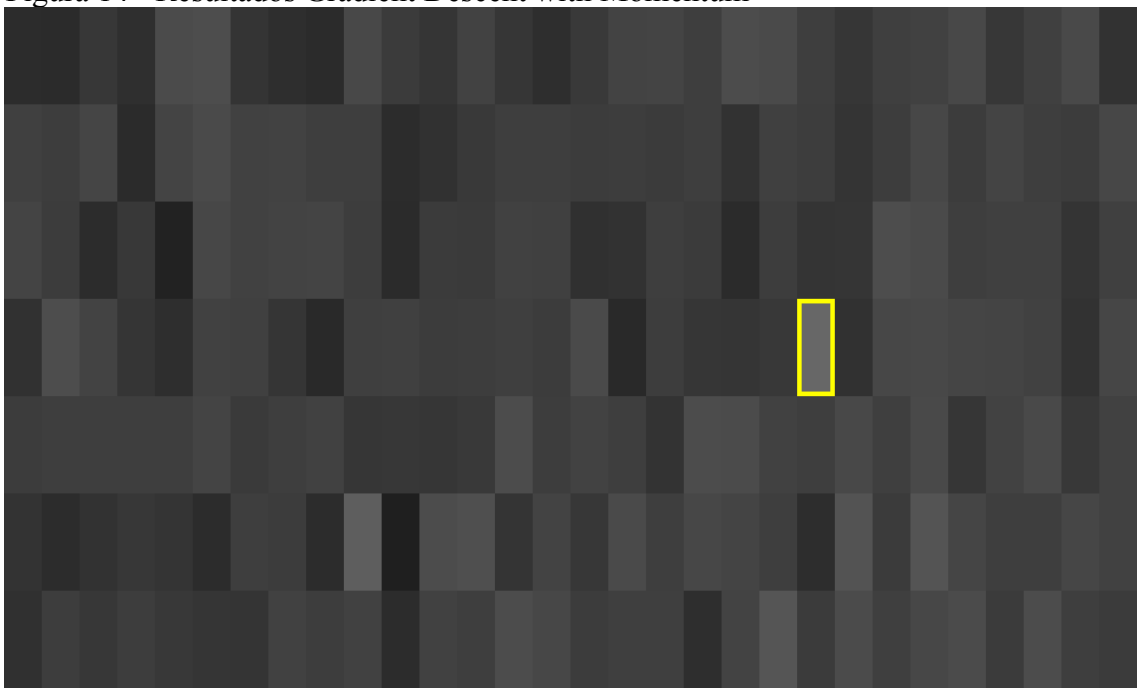
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 37,80%.

3.11 GRADIENT DESCENT WITH MOMENTUM

Na Figura 14 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 22 neurônios na primeira camada e 9 neurônios na segunda camada.

Figura 14 - Resultados Gradient Descent with Momentum



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 24 a seguir. Estes resultados produzem uma média de acertos de 40,33%.

Tabela 24 - Resultados Gradient Descent with Momentum com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	40	50	80
D1	14	50	28
D2	44	50	88
T1	0	50	0
T2	0	50	0
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 25 a seguir. Estes resultados produzem uma média de acertos de 16,67%.

Tabela 25 - Resultados Gradient Descent with Momentum com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	48	74	64,86
D1	21	191	10,99
D2	5	21	23,81
T1	0	251	0
T2	0	94	0
T3	1	305	0,33

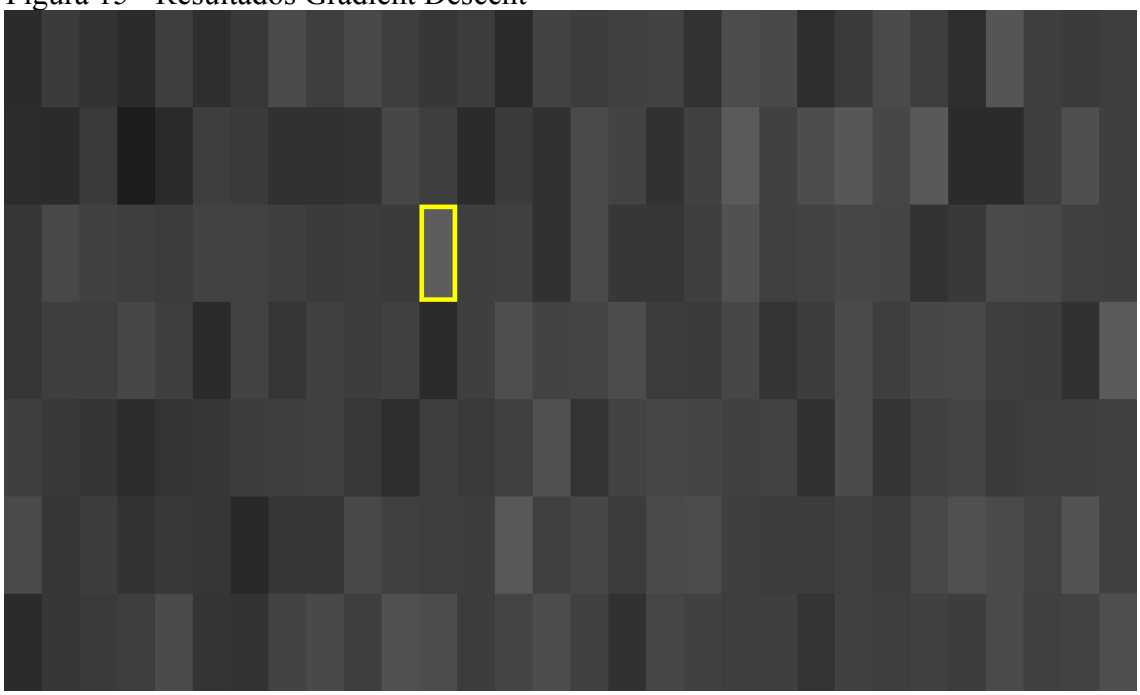
Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 28,50%.

3.12 GRADIENT DESCENT

Na Figura 15 é ilustrado como foi o desempenho geral deste método de treinamento. Boa parte das combinações possuem tons mais escuros, indicando que várias dessas combinações não apresentaram bons resultados. A combinação que apresentou o melhor resultado (destacado em amarelo) possui 12 neurônios na primeira camada e 8 neurônios na segunda camada.

Figura 15 - Resultados Gradient Descent



Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para treinamento da rede, são mostrados na Tabela 26 a seguir. Estes resultados produzem uma média de acertos de 36,67%.

Tabela 26 - Resultados Gradient Descent com Amostras de Treinamento

Falta	Acertos	Amostra	Acertos (%)
DP	0	50	0
D1	43	50	86
D2	0	50	0
T1	0	50	0
T2	44	50	88
T3	23	50	46

Fonte: Elaborado pelo Autor.

Os resultados para essa rede de melhor desempenho, utilizando os dados para teste da rede, são mostrados na Tabela 27 a seguir. Estes resultados produzem uma média de acertos de 25,25%.

Tabela 27 - Resultados Gradient Descent com Amostras de Teste

Falta	Acertos	Amostra	Acertos (%)
DP	0	74	0
D1	109	191	57,07
D2	1	21	4,76
T1	0	251	0
T2	84	94	89,36
T3	1	305	0,33

Fonte: Elaborado pelo Autor.

Desta forma, combinando os resultados obtidos para as amostras de treinamento e para as amostras de teste, este método de treinamento obteve uma taxa média de acertos de 30,96%.

3.13 RESULTADO GERAL ORDENADO

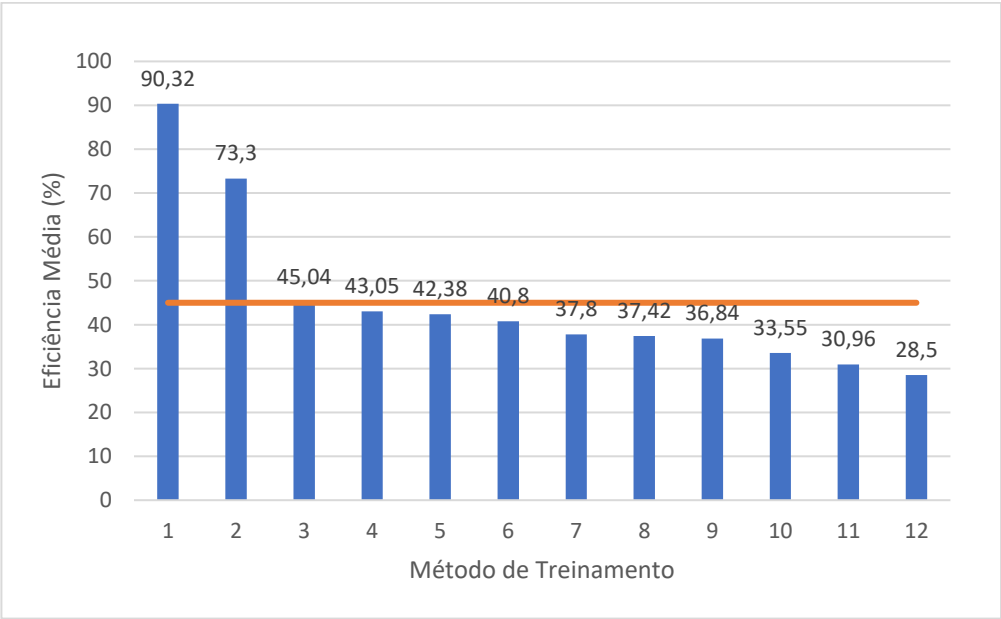
Para facilitar a visualização dos resultados, a Tabela 28 e o Gráfico 5 forão criados com os resultados ordenados desde o melhor até o pior desempenho dos métodos de treinamento empregados.

Tabela 28 - Resultado Geral Ordenado

Nº	Método de Treinamento	Eficiência Média (%)
1	Bayesian Regularization	90,32
2	Levenberg-Marquardt	73,30
3	Resilient Backpropagation	45,04
4	BFGS Quasi-Newton	43,05
5	Polak-Ribiére Conjugate Gradient	42,38
6	Conjugate Gradient with Powell/Beale Restarts	40,80
7	Variable Learning Rate Gradient Descent	37,80
8	One Step Secant	37,42
9	Scaled Conjugate Gradient	36,84
10	Fletcher-Powell Conjugate Gradient	33,55
11	Gradient Descent	30,96
12	Gradient Descent with Momentum	28,50

Fonte: Elaborado pelo Autor.

Gráfico 5 - Eficiência Média dos Métodos de Treinamento



Fonte: Elaborado pelo autor.

Pode-se observar que o método de treinamento que obteve o melhor resultado foi o Bayesian Regularization com 90,32%, precisando de poucos neurônios para conseguir atingir esse resultado se comparado com as outras RNAs. E o método de treinamento que obteve o pior resultado foi o Gradient Descent with Momentum com apenas 28,50%. A eficiência média geral foi de 45% e apenas os dois primeiros métodos ficaram acima desse valor.

4 CONCLUSÃO

Os transformadores de potência podem apresentar diversos problemas durante sua vida útil que podem gerar gases pela decomposição do óleo e de sua isolação. Desta forma, existem vários métodos disponíveis no mercado e em normas técnicas que tentam analisar os gases gerados para reversamente, tentar encontrar e/ou prevenir falhas nestes equipamentos.

As redes neurais artificiais são modelos matemáticos que podem ser utilizados em problemas de classificação complexos e não-lineares como a análise de gases dissolvidos no óleo do transformador. Por isso, o emprego de RNAs com o objetivo de substituir os métodos de analises tradicionais é um processo natural. Para a comparação entre o método de análise utilizado pelo IEC com a capacidade da RNA, vários métodos de treinamento e estruturas da rede foram testados neste trabalho.

Com base nos resultados apresentados, pode-se concluir que o método de treinamento e o número de "neurônios" em cada camada da RNA são muito relevantes para a eficiência final da metodologia de diagnóstico. Entre os resultados obtidos, a rede treinada utilizando o método de treinamento "Bayesian Regularization" com 5 neurônios na primeira camada e 6 neurônios na segunda camada apresentou o melhor resultado com 90,32% de eficiência, e a rede treinada utilizando o método de treinamento "Gradient Descent with Momentum" com 22 neurônios na primeira camada e 9 neurônios na segunda camada apresentou o pior resultado com 28,50% de eficiência, totalizando uma diferença percentual de eficiência de 61,82% entre estas duas redes.

Além disso, é possível observar que redes que possuem mais "neurônios" não necessariamente produzem um resultado melhor que redes com menos "neurônios". A rede de melhor resultado deste trabalho, possui apenas 5 "neurônios" na primeira camada e 6 "neurônios" na segunda camada e apresenta uma eficiência melhor que suas versões com mais "neurônios".

Como trabalhos futuros pode-se citar a realização de testes adicionais com outras configurações de rede para tentar melhorar a eficiência da RNA. Nota-se que o mais interessante seria usar dados com resultados confirmados por especialistas para comparar qual método é mais eficiente na análise de falhas no transformador.

5 REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, P. M. **Power System Protection**. New York: IEEE: Willey-Interscience, 1999.

DAPONTE, P.; GRIMALDI, D.; PICCOLO, A.; VILLACCI, D. A neural diagnostic system for the monitoring of transformer heating. **Measurement: Journal of the International Measurement Confederation**, [s. l.], v. 18, n. 1, p. 35–46, 1996.

DE FREITAS, A. A. C.; DA SILVA, I. N.; DE SOUZA, A. N. **Aplicação de redes neurais na estimação da temperatura interna de transformadores de distribuição imersos em óleo**. Revista Controle & Automação, [s.l.], v.13, n. 3, 2002.

DUARTE, L. J.; PINHEIRO, A. P.; NUNES, W. D. C.; FERREIRA, D. O. Modelagem Térmica de Transformadores de Distribuição e Estimação de Parâmetros. In: VIII SIMPÓSIO BRASILEIRO DE SISTEMAS ELÉTRICOS (SBSE 2020) 2020, Virtual. **Anais**... Virtual: Sociedade Brasileira de Automática (SBA), 2020.

DUVAL, M. A review of faults detectable by gas-in-oil analysis in transformers. **IEEE Electrical Insulation Magazine**, [s. l.], v. 18, n. 3, p. 8–17, 2002.

DUVAL, M.; DEPABLO, A. Interpretation of gas-in-oil analysis using new IEC publication 60599 and IEC TC 10 databases. **IEEE Electrical Insulation Magazine**, [s. l.], v. 17, n. 2, p. 31–41, 2001.

FILHO, J. M. **Manual de Equipamentos Elétricos**. 4ª ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 2013.

Fitnet. **MATHWORKS**. c2022. Disponível em: https://www.mathworks.com/help/deeplearning/ref/fitnet.html;jsessionid=3c2ed896d63afbe13122405d3d7f. Acesso em: 11 jun. 2022.

FITZGERALD, A. E.; K., J. C.; UMANS, S. D. **Electric machinery**. 6ª Ed. ed. [s.l.] : McGraw-Hill series in electrical engineering, 2003.

GALDI, V.; IPPOLITO, L.; PICCOLO, A.; VACCARO, A. Neural diagnostic system for transformer thermal overload protection. **IEEE Proceedings-Electric Power Applications**, [s. l.], v. 147, n. 5, p. 415–421, 2000.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2ª ed. São Paulo: Bookman Companhia Editora, 2001.

HELL, M.; D`ANGELO, M.; COSTA, P. Power Transformer Fault Diagnosis Based on Dissolved Gas Analysis Using a Fuzzy Neural Network Approach in a Real Data Base. In: PROCEEDINGS OF THE WORLD SCIENTIFIC AND ENGINEERING ACADEMY AND SOCIETY INTERNATIONAL CONFERENCES, WSEAS ’02 2002, Anais... [s.l: s.n.]

IEC. **IEC 60599 - Mineral oil-filled electrical equipment in service - Guidance on the interpretation of dissolved and free gases analysis**, IEC - International Electrotechnical Commission, 2015.

KHADE, P. S.; MAHAJAN, G. K.; CHAUDHARI, A. P. Artificial Neural Network Approach to Dissolved Gas Analysis for Interpretation of Fault in Power Transformer. International Journal of Scientific …, [s. l.], v. 7, n. 3, p. 373–377, 2016.

NETO, A. D.; ASSUNÇÃO, T. C. B. N.; ASSUNÇÃO, J. T. Classificação Dos Transformadores De Potência Empregando a Análise Dos Gases Dissolvidos No Óleo Isolante.

In: IX CONGRESSO BRASILEIRO DE REDES NEURAIS 2009, **Anais**... : SBIC Sociedade Brasileira de Inteligência Computacional, 2009.

SILVA, M. P. **Aplicação de Redes Neurais Artificiais no Diagnóstico de Falhas de Turbinas a Gás**. p. 33-42, 2010. Disponível em: https://www.maxwell.vrac.puc-rio.br/16580/16580_4.PDF. Acesso em: 14 jul 2022.

SUN, H. C.; HUANG, Y. C.; HUANG, C. M. Fault diagnosis of power transformers using computational intelligence: A review. **Energy Procedia**, [s. l.], v. 14, p. 1226–1231, 2012.

TAO, L.; YANG, X.; ZHOU, Y.; YANG, L. A novel transformers fault diagnosis method based on probabilistic neural network and bio-inspired optimizer. **Sensors**, [s. l.], v. 21, n. 11, p. 20, 2021.

WANG, Q.; WANG, S.; SHI, R.; LI, Y. A Power Transformer Fault Diagnosis Method Based on Random Vector Functional-Link Neural Network. **Mathematical Problems in Engineering**, [s. l.], v. 2021, n. 6656061, p. 1–18, mar. 2021.

Welcome to PyAutoGUI's documentation!. PYAUTOGUI. c2019. Disponível em: https://pyautogui.readthedocs.io/en/latest/#. Acesso em: 11 jun 2022.

ZHANG, Y.; DING, X.; LIU, Y.; GRIFFIN, P. J. An Artificial Neural Network Approach to Transformer Fault Diagnosis. **IEEE Transaction on Power Delivery**, [s. l.], v. 11, n. 4, p. 1836–1841, 1996.

APÊNDICE 1 AGRUPAMENTO DE ARQUIVOS BAT

```
@echo off

for /f "delims=|" %%i in ('dir *.doc /s /b') do (
echo %%i
xcopy "%%i" "1. doc_data" /y
)

Pause
```

APÊNDICE 2 CONVERSÃO DE ARQUIVOS WORD PARA TXT PYTHON

```
import os
import time
import pyautogui
doc_directory = os.path.join(os.getcwd(), "1. doc_data")
txt_directory = os.path.join(os.getcwd(), "2. txt_data")
word_directory = "C:\\Program Files\\Microsoft Office\\root\\Office16\\WINWORD.exe"
notp_directory = "C:\\Windows\\system32\\notepad.exe"
for process_file in  os.listdir(doc_directory):
    file, extension = os.path.splitext(process_file)
    dest_file = file + '.txt'
    process_file_path = os.path.join(doc_directory, process_file)
    dest_file_path = os.path.join(txt_directory, dest_file)
    with open(dest_file_path, "w") as text_file:
        text_file.write("")
        text_file.close()
    os.system('start /max "' + word_directory + '" "' + process_file_path + '"')
    time.sleep(2.5)
    pyautogui.hotkey('ctrl', 't')
    time.sleep(0.4)
    pyautogui.hotkey('ctrl', 'c')
    time.sleep(0.4)
    pyautogui.hotkey('alt', 'f4')
    time.sleep(0.4)
    pyautogui.hotkey('enter')
    time.sleep(5)
    os.system('start /max "' + notp_directory + '" "' + dest_file_path + '"')
    time.sleep(1.5)
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(0.4)
    pyautogui.hotkey('ctrl', 's')
    time.sleep(0.4)
```

```
pyautogui.hotkey('alt', 'f4')
time.sleep(1)
print("Arquivo convertido: " + process_file + " --> " + dest_file)
time.sleep(1)
os.remove(process_file_path)
```

APÊNDICE 3 EXTRAÇÃO DE DADOS DE ARQUVOS TXT

```
import os
import re
import pandas as pd
def pattern1(filecontent):
pattern=re.compile(r"^.*\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?
\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)\t(\d+,?\d*|ND)$")
    gases = []
    for line in filecontent:
        if re.match(pattern, line):
            matched = re.match(pattern, line).groups()
            gases_line = [matched[i] for i in range(9)]
            nulos = 0
            for i in range(9):
                if gases_line[i] == 'ND':
                    gases_line[i] = 0
                elif gases_line[i] == '':
                    nulos = nulos + 1
                else:
                    gases_line[i] = float(str(gases_line[i]).replace(',', '.'))
            if nulos < 9:
                gases.append(gases_line)
    return gases
def pattern2(filecontent):
pattern=re.compile(r"^\w+.*ppm\t(\d+,?\d*|ND)\t(\d*,?\d*|ND)\t(\d*,?\d*|ND)\t(\d*,?\d*|ND)
\t+$")
    gases = []
    gases_test = []
    for line in filecontent:
        if re.match(pattern, line):
            gases_test.append(re.match(pattern, line).groups())
    if len(gases_test)==9:
```

```
        for j in range(4):
            aux = []
            nulos = 0
            for i in range(9):
                if gases_test[i][j] == 'ND':
                    aux.append(0)
                elif gases_test[i][j] == '':
                    aux.append('')
                    nulos = nulos + 1
                else:
                    aux.append(float(str(gases_test[i][j]).replace(',', '.')))
            if nulos < 9:
                gases.append(aux)
    return gases
def pattern3(filecontent):
    pattern=re.compile(r"^[H|O|N|M|D|E|A].*PPM\t(\d+,?\d*|ND|-)\t(\d+,?\d*|ND|-
)\t(\d+,?\d*|ND|-).*$")
    gases = []
    gases_test = []
    for line in filecontent:
        if re.match(pattern, line):
            gases_test.append(re.match(pattern, line).groups())
    if len(gases_test)==9:
        for j in range(2):
            aux = []
            nulos = 0
            for i in range(9):
                if gases_test[i][j] == 'ND':
                    aux.append(0)
                elif gases_test[i][j] == '' or gases_test[i][j] == '-':
                    aux.append('')
                    nulos = nulos + 1
                else:
                    aux.append(float(str(gases_test[i][j]).replace(',', '.')))
```

```
            if nulos < 9:
                gases.append(aux)
    return gases
def allowSampleAppending(sample, sampleList):
    if sample in sampleList:
        return False
    else:
        return True
txt_directory = os.path.join(os.getcwd(), "2. txt_data")
gases = [['H2', 'O2', 'N2', 'CH4', 'CO', 'CO2', 'C2H4', 'C2H6', 'C2H2']]
for process_file in  os.listdir(txt_directory):
    process_file_path = os.path.join(txt_directory, process_file)
    with open(process_file_path, encoding="utf8") as text_file:
        content = text_file.readlines()
        if(pattern1(content)):
            for amostra in pattern1(content):
                if allowSampleAppending(amostra, gases):
                    gases.append(amostra)
        if(pattern2(content)):
            for amostra in pattern2(content):
                if allowSampleAppending(amostra, gases):
                    gases.append(amostra)
        if(pattern3(content)):
            for amostra in pattern3(content):
                if allowSampleAppending(amostra, gases):
                    gases.append(amostra)
df = pd.DataFrame(gases)
writer = pd.ExcelWriter('data.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='amostras', index=False)
writer.save()
```

APÊNDICE 4 FUNÇÃO PARA APLICAÇÃO DA NORMA IEC

```
function output = applyIEC(input)
%  Função utilizada para aplicação da Norma IEC.
%  O vetor de entrada deve ser do tipo de linha
% com a concentração dos seguintes gases: H2, CH4,
% C2H6, C2H4 e C2H2, nesta ordem.
label = ["PD"; "D1"; "D2"; "T1"; "T2"; "T3"; "NI"];
output = label(7);
r1 = input(5)/input(4);
r2 = input(2)/input(1);
r3 = input(4)/input(3);
if (r2 < 0.1) && (r3 < 0.2)
  output = label(1);
end
if (r1 > 1) && (r2 >= 0.1) && (r2 <= 0.5) && (r3 > 1)
  output = label(2);
elseif (r1 >= 0.6) && (r1 <= 2.5) && (r2 >= 0.1) && (r2 <= 1) && (r3 > 2)
  output = label(3);
end
if (r2 > 1) && (r3 < 1)
  output = label(4);
elseif (r1 < 0.1) && (r2 > 1) && (r3 >= 1) && (r3 <= 4)
  output = label(5);
elseif (r1 < 0.2) && (r2 > 1) && (r3 > 4)
  output = label(6);
end
end
```

APÊNDICE 5 SEPARAR AMOSTRAS COM FALTAS

```
clear;
load('Banco de Dados Completo.mat');
clc;
input = treinamento;
label = ["PD"; "D1"; "D2"; "T1"; "T2"; "T3"; "NI"];
samples = size(input, 2);
output = zeros(7, samples);
xf = zeros(5, samples);
count = 0;
for i=1:samples
  gases = input(:, i)';
  fault_type = applyIEC(gases);
  fault_pos = find(strcmp(label, fault_type));
  output(fault_pos, i) = 1;
  if (fault_pos < 7)
     count = count + 1;
     xf(:, count) = gases;
  end
end
xf = xf(:, 1:count);
treinamento = xf;
save("Banco de Dados Apenas Faltas", "treinamento");
```

APÊNDICE 6 SEPARAR AMOSTRAS DE TREINAMENTO E TESTE

```
clear;
load('Banco de Dados Apenas Faltas.mat');
clc;
label = ["PD"; "D1"; "D2"; "T1"; "T2"; "T3"];
output_count = [0; 0; 0; 0; 0; 0];
output_count2 = [0; 0; 0; 0; 0; 0];
count_max = 50;
input = treinamento;
samples = size(input, 2);
xr = zeros(5, samples);
yr = zeros(6, samples);
xt = zeros(5, samples);
yt = zeros(6, samples);
ri = 0;
ti = 0;
for i=1:samples
  gases = input(:, i)';
  fault_type = zeros(1, 6);
  output = applyIEC(gases);
  fault_pos = find(strcmp(label, output));
  fault_type(fault_pos) = 1;
  output_count(fault_pos) = output_count(fault_pos) + 1;
  if output_count(fault_pos) <= count_max
     ri = ri + 1;
     xr(:, ri) = gases';
     yr(:, ri) = fault_type';
     output_count2(fault_pos) = output_count2(fault_pos) + 1;
  else
     ti = ti + 1;
     xt(:, ti) = gases';
     yt(:, ti) = fault_type';
```

```
  end
end
xr = xr(:, 1:ri);
yr = yr(:, 1:ri);
xt = xt(:, 1:ti);
yt = yt(:, 1:ti);
save("Banco de Dados Separado", "xr", "yr", "xt", "yt");
```

APÊNDICE 7 ARREDONDAR MAIOR NÚMERO

```
function [output] = roundBiggest(input)
%  Essa função arredonda o maior elemento
% de uma coluna para 1 e os restantes dos
% elementos para 0. Os valores da matriz
% devem estar entre 0 e 1.


for j=1:size(input, 2)
   biggest = 0;
   index = 1;
   for i=1:size(input, 1)
      if input(i, j) > biggest
         biggest = input(i, j);
         input(index, j) = 0;
         index = i;
         input(i, j) = 1;
      else
         input(i, j) = 0;
      end
   end
end

output = input;

end
```

APÊNDICE 8 COMPARAR SAÍDA DA REDE COM VETOR ALVO

```
function output = compareOutput(netOutput,target)
%  Verifica se o vetor 1 apresentou
% as mesmas saídas do vetor 2.

output = zeros(size(netOutput));
for j=1:size(netOutput, 2)
   for i=1:size(netOutput, 1)
      if netOutput(i, j) == 1 && target(i, j) == netOutput(i, j)
         output(i, j) = 1;
      else
         output(i, j) = 0;
      end
   end
end
```

APÊNDICE 9 AVALIAÇÃO DE RESULTADO POR CATEGORIA

```
function output = resultCategory(input, target, net)

input = compareOutput(roundBiggest(net(input)), target);
output = zeros(size(input, 1), 3);
for i=1:size(input, 1)
   output(i, :) = [sum(input(i, :)) sum(target(i, :)) (sum(input(i, :))/sum(target(i, :)))*100];
end

end
```

APÊNDICE 10 AVALIAÇÃO DE ERRO DA REDE

```
function output = evaluateErro(input,target, net)

% Esta função retorna a porcentagem de amostras com erro.
result = resultCategory(input, target, net);
output = 100-mean(result(:, 3));

end
```

APÊNDICE 11 TREINAMENTO DAS REDES

```
clear;
load('Banco de Dados.mat');
clc;

erros_index = zeros(1, 3);
erros_matriz = zeros(20, 6);
erros_min = 100;
erros_count = 0;

trainFcn = ["trainlm", "trainbr", 'trainbfg', 'trainrp', 'trainscg', 'traincgb', 'traincgf', 'traincgp',
'trainoss', 'traingdx', 'traingdm', 'traingd'];

for t = 1:12
   for n2=6:12
      for n1=1:30
         erro_min_local = 100;
         for r=1:2
            hiddenLayerSize = [n1 n2];
            net = fitnet(hiddenLayerSize, trainFcn(t));

            net.input.processFcns = {'removeconstantrows','mapminmax'};
            net.output.processFcns = {'removeconstantrows','mapminmax'};

            net.divideFcn = 'dividerand';
            net.divideMode = 'sample';
            net.divideParam.trainRatio = 70/100;
            net.divideParam.valRatio = 15/100;
            net.divideParam.testRatio = 15/100;

            net.performFcn = 'mse';
```

```
            [net,tr] = train(net,xr,yr);

            erros = evaluateErro(xr, yr, net);

            if erros < erro_min_local
               erro_min_local = erros;
               best_local_net = net;
            end
            erros_matriz(n1, n2 - 5) = erro_min_local;
         end

         if erro_min_local < erros_min
            erros_min = erro_min_local;
            erros_count = erros_count + 1;
            erros_index(erros_count, :) = [n1 n2 erro_min_local];
            best_net = best_local_net;
         end
      end
   end

   net = best_net;
   save(strcat("res_",trainFcn(t)), "erros_index", "erros_matriz", "net");
end

clearvars -except erros_index erros_matriz net;
load('Banco de Dados.mat');
clc;
```

APÊNDICE 12 IMAGEM PARA ANÁLISE DOS RESULTADOS

```
clear;
clc;
px = 1000;
py = 600;
trainFcn = ["trainlm", "trainbr", 'trainbfg', 'trainrp', 'trainscg', 'traincgb', 'traincgf', 'traincgp', 'trainoss', 'traingdx', 'traingdm', 'traingd'];
for t=1:12
    load(strcat('res_', trainFcn(t), '.mat'));
    erros_matriz = erros_matriz';
    ex = size(erros_matriz, 2);
    ey = size(erros_matriz, 1);
    qx = floor(px/ex);
    qy = floor(py/ey);
    sx = ex*qx;
    sy = ey*qy;
    em = 1-mat2gray(erros_matriz, [0 100]);
    img = zeros(sy, sx, 3);
    b = 3;
    for i=1:ex
        for j=1:ey
            qi = i + (i - 1) * (qx - 1);
            qj = j + (j - 1) * (qy - 1);
            img(qj:qj+qy-1, qi:qi+qx-1, :) = em(j, i);
            if j == (erros_index(size(erros_index, 1), 2)-5) && i == erros_index(size(erros_index, 1), 1)
                img(qj:qj+qy-1, qi:qi+b, 1) = 1;
                img(qj:qj+qy-1, qi:qi+b, 2) = 1;
                img(qj:qj+qy-1, qi:qi+b, 3) = 0;
                img(qj:qj+qy-1, qi+qx-1-b:qi+qx-1, 1) = 1;
                img(qj:qj+qy-1, qi+qx-1-b:qi+qx-1, 2) = 1;
                img(qj:qj+qy-1, qi+qx-1-b:qi+qx-1, 3) = 0;
```

```
            img(qj:qj+b, qi:qi+qx-1, 1) = 1;
            img(qj:qj+b, qi:qi+qx-1, 2) = 1;
            img(qj:qj+b, qi:qi+qx-1, 3) = 0;
            img(qj+qy-1-b:qj+qy-1, qi:qi+qx-1, 1) = 1;
            img(qj+qy-1-b:qj+qy-1, qi:qi+qx-1, 2) = 1;
            img(qj+qy-1-b:qj+qy-1, qi:qi+qx-1, 3) = 0;
        end
    end
  end
  imwrite(img, strcat('hor_res_', trainFcn(t), '.png'));
end
clearvars em ex ey i j px py qi qj qx qy sx sy amostras
```

ANEXO 1 PARTE DAS AMOSTRAS DOS DADOS EMPREGADOS

Na Tabela 29 é ilustrado uma pequena parte dos dados utilizados. Cada linha corresponde a uma amostra.

Tabela 29 - Amostra de Dados Utilizados

H2	O2	N2	CH4	CO	CO2	C2H4	C2H6	C2H2
119	1974	14039	27,3	420	1253	2	2,8	0
101	9342	43167	34,4	497	1774	2,2	3,3	0
86	2933	20648	30,4	505	1686	1,9	2,4	0
10	1368	5666	2,2	125	717	0,4	0,3	0,6
10	2119	8356	2,8	152	918	0,6	0,5	0,7
9	1223	6627	2,4	162	935	0,2	0,2	0,3
11	1394	6415	3,2	170	899	0,5	0,5	1
13	6706	26702	3,4	181	1141	0,5	1,5	0,4
2943	23439	55018	317,5	762	1432	350	84,8	2890
2404	25378	61435	298,4	817	1419	344,6	76,9	2932,2
2984	18661	49006	361,1	999	1462	372	65,3	1966,1
4086	17324	66749	866,3	2059	2582	811,9	102	4274,1
2269	19840	48440	216,1	580	1121	240	45	1766,5
2790	17203	41664	349,2	872	1279	361,1	53,9	1832,2
5606	12247	54424	556,3	1499	1957	502,1	86,5	3605,6
76	1076	10379	33	436	1727	1,2	1	0
77	1582	7447	7	146	752	0,6	0,6	0
68	18345	71006	14,8	304	2229	1,3	1,4	0
71	2381	12247	8,3	210	944	0,6	0,5	0
8	29439	69514	1,2	70	1042	6,1	0,2	0
10	13193	63854	1,5	122	1145	6,7	0,3	0
10	23689	60524	2,1	87	1122	7,6	0,3	0
3	1160	13349	12,3	231	1977	0,9	2,8	0
14	10163	45182	15,4	257	2874	0,8	4,5	0,1
10	1739	19404	17,4	307	2785	0,8	4,5	0
5	1502	13484	17,2	246	2464	0,8	4,7	0
2	627	14130	15,8	225	2373	0,7	4,5	0
3	575	16292	13,8	275	2661	0,8	5,1	0
17	31566	68656	1,2	170	828	5,6	0,3	0
20	33186	74100	1,3	168	812	5,6	0,3	0
18	25304	56480	1,2	84	747	5,5	0,3	0
11	17000	68013	10,4	909	3429	14,2	1	0
11	16403	67149	11	1010	4193	16,9	1	0
15	18093	82471	13,8	1317	4825	17,7	1	0
13	15963	73609	12,3	1182	4356	16,4	1	0
9	14943	67589	11,5	1158	3878	16,5	0,9	0
9	16533	76909	14,9	1236	4362	19,7	1,2	0

8	12984	62828	10,4	917	3358	14,6	0,9	0
28	21537	68488	8,1	383	4523	10,3	4,8	0
66	14634	73260	10,7	558	5263	12	5,6	0
51	16651	61538	10,7	440	4543	12,2	5,4	0
5	2210	19593	4	388	1190	0,2	0,2	0
5	2016	17881	4,3	407	1244	0,3	0,2	0
3	7803	35495	5,4	329	705	0,2	0,2	0
5	3002	21453	4,8	360	901	0,2	0,2	0
0	30645	63425	1,2	9	403	1,8	0,1	0,1
1	31002	64170	1,3	10	412	1,7	0,1	0,1
1	30346	62826	1,2	10	410	1,9	0,2	0,1
2	30567	63129	1,2	11	421	1,9	0,1	0,1
59	1630	29971	50,1	118	1754	4,8	24	0
65	7240	52574	57,4	138	2209	5,8	24,2	0
101	3503	30152	42,3	116	2289	6	24,4	0
50	1125	27213	63,7	113	2646	5,9	19,1	0
52	1729	26345	50,1	94	2342	4,3	11,3	0,2
13441	10099	49980	1380,2	529	2937	161,6	13,6	1623,4
11532	96870	64416	1414,3	726	4955	183,6	11	1215,8
8609	6299	55363	854,6	613	4014	1196,9	176,9	6669,8
19	24121	60974	3,9	277	1731	5,8	1,6	0,2
12	26617	60756	6,5	202	1239	5,7	1,4	0,1
16	14719	53912	3,9	206	2425	5,1	2,2	0
74	1072	32814	90,4	714	5577	40,7	9,9	3,2
13	697	2975	6,8	39	732	3,2	0,5	0,5
23	1657	6982	17,8	103	1429	9,4	1,5	0,6
23	731	4821	21,2	100	1403	11,1	1,6	0,7
18	1181	5986	17	85	1203	8,2	1,3	0,3
1	1885	46420	23	948	4850	0	0	0
4	1487	40741	20	812	3342	0	0	0
1	2879	19688	12	378	3163	0	0	0
3	5098	29780	12	367	3570	0	0	0
1	1248	16371	10	294	2860	0	0,6	0
5	1442	20368	12	352	3685	0	0,9	0
19	34197	73856	1,3	59	1106	5,5	0,6	0,3
18	27465	72596	4,4	284	1595	6,2	1,5	0,3
11	25682	57558	1,5	89	1098	5,2	0,3	0
17	2031	20541	68,2	82	2953	41,7	27,4	0
5	1543	8428	3,9	16	422	2,7	0,7	0
6	2139	20742	7,9	24	621	5,4	1,3	0
10	1867	10344	11,2	23	697	7,6	1,1	0
8	851	7171	9	18	508	5,5	1,6	0
8	26145	63582	2,8	146	1027	4,4	1,1	0,1
1	1035	3100	0,1	4	12	0	0	0
4	1148	4501	0,2	17	43	0	0	0
1	2602	6094	0,1	1	19	0	0	0
1	2399	6342	0,1	2	19	0	0	0
1	2546	6596	0,1	1	17	0	0	0

1	3846	10888	0,1	2	23	0	0	0
0	1491	4070	0	0	7	0	0	0
23	1511	3809	0,1	1	20	0	0	0
1	961	2448	0,1	2	44	0	0	0
2	1434	4055	0,4	7	60	0	0	0
1	1282	2820	0,1	2	19	0	0	0
2	13903	46274	0,5	41	130	0	0	0
8	5139	16654	0,5	68	134	0	0	0
16	9293	43943	4,4	452	981	0	0	0
16	9695	45495	4,7	475	1076	0	0	0
181	2177	70800	510	291	10326	376,4	247,7	0
10	5652	19980	17,1	20	1074	13,6	13,3	0
5	20595	65519	2	17	180	2,4	0	0
8	19638	69624	0,2	3	70	0	0	0
3	16891	61097	0,3	7	68	0	0	0
0	1049	3300	0,3	4	23	0	0	0
8	17845	64308	0,4	10	102	0	0	0
1	2656	8457	0	0	14	0	0	0
2	2904	9495	0,1	3	22	0	0	0
1	1250	3262	0,1	1	15	0	0	0
3	2080	8700	0,1	5	36	0	0	0
1	2794	7856	0,2	3	26	0,1	0	0
8	4798	72217	36,9	981	3687	52,2	4,6	0
8	4144	62383	40,6	1105	4271	55,9	4,9	0
6	5123	72958	38,4	1105	4279	55,2	4,6	0,6
9	3072	82107	48,7	1257	5030	65	6,4	1,8
7	2687	72484	36,9	994	4103	48,3	4,9	1,1
3	3954	12132	0,1	0	26	0	0	0
3	1520	3600	0,1	1	22	0	0	0
12	3646	1146	0,1	4	29	0	0	0
0	1546	3074	0,1	2	33	0	0	0
2	2744	8074	0,3	14	94	0	0	0
2	1673	5427	0,1	3	20	0	0	0
2	2676	25027	0,3	42	211	0	0	0
0	1615	4423	0,1	3	35	0	0	0
2	3372	32383	0,3	35	245	0	0	0
0	1411	3945	0,2	4	60	0	0	0
0	1137	2459	0	1	20	0	0	0
0	4325	8432	0,1	2	60	0	0	0
0	1678	3650	0,1	1	27	0	0	0
2	4013	11824	0,1	4	35	0	0	0
2	2161	5654	0,1	0	21	0	0	0
0	2043	4811	0,1	0	26	0	0	0
0	2462	6269	0,1	1	32	0	0	0
0	1591	3911	0,1	2	28	0	0	0
90	4508	85124	72,1	1587	7309	234,5	15	15,3
4	2829	9594	0	1	19	0	0	0
1	1353	3498	0,1	3	51	0,1	0	0

1	1840	5105	0,6	15	236	0,7	0	0,3
0	2174	5868	0,1	1	26	0	0	0
1	3314	9623	0	1	15	0	0	0
37	5177	21461	2,4	110	1336	0,7	0	7,5
36	4850	21816	2,4	112	1252	0,7	0	7
3	1443	3696	0,1	1	20	0	0	0
4	3184	10919	0	0	16	0	0	0
20	1903	4169	0,1	2	38	0	0	0
1	2595	7374	0,1	0	24	0	0	0
0	1726	3677	0,6	2	209	0	0	0
0	1787	5272	0,2	1	47	0	0	0
0	2040	3870	0	1	36	0	0	0
1	3164	6127	0,1	1	55	0	0	0
0	2320	4989	0	0	36	0	0	0
2	1862	6441	0,4	20	168	0	0	0
1	3838	12948	0,3	22	207	0	0	0

Fonte: Elaborado pelo Autor.