UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA – CENTRO TECNOLÓGICO

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

João Victor Laitano Coelho Silva

MACHINE LEARNING PARA DETECÇÃO DE FALTAS EM UM SISTEMA ELÉTRICO

Florianópolis - SC

2019

João Victor Laitano Coelho Silva

MACHINE LEARNING PARA DETECÇÃO DE FALTAS EM UM SISTEMA ELÉTRICO

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Miguel Moreto, Dr. Eng.

Florianópolis - SC

2019

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

Silva, João victor
Machine Learning para detecção de faltas em um sistema elétrico / João victor Silva ; orientador, Miguel Moreto, 2019.
79 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2019.

Inclui referências.

1. Engenharia Elétrica. 2. Machine Learning. 3. Redes Neurais. 4. Faltas. 5. Sistema Elétrico. I. Moreto, Miguel . II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

João Victor Laitano Coelho Silva

Machine Learning para Detecção de Faltas em um Sistema Elétrico

Este Trabalho foi julgado adequado como parte dos requisitos para obtenção do Título de Bacharel em Engenharia Elétrica e aprovado, em sua forma final, pela Banca Examinadora

Florianópolis, 10 de dezembro de 2019.

Prof. Jean Vianei Leite, Dr.
Coordenador do Curso de Graduação em Engenharia Elétrica

Banca Examinadora:

Miguel Moreto, Dr
Orientador
Universidade Federal de Santa Catarina

Prof. Aguinaldo Silveira e Silva, Ph.D.
Universidade Federal de Santa Catarina

Eng. Eletricista Dante Edson Sifuentes Quiroz, Me.
Universidade Federal de Santa Catarina

Este trabalho é dedicado a minha companheira e aos meus familiares.

AGRADECIMENTOS

Gostaria de agradecer sobretudo a minha companheira Paula pelo suporte em todos os momentos. À minha família pelo apoio incondicional. Aos meus amigos pelas discussões e conversas inspiradoras. Meus professores e colegas por tudo aquilo que me foi ensinado.

“Seek freedom and become captive of your desires.
Seek discipline and find your liberty.”
(Frank Herbert)

RESUMO

Neste trabalho é documentada uma metodologia para se criar um sistema inteligente para detecção, localização e classificação de curto-circuitos de um sistema de potência elétrico e por conta disto realizado um estudo das questões técnicas utilizadas nesta metodologia. A maioria das técnicas utilizadas para localização de faltas utilizadas nos dias atuais são voltadas para sistemas de transmissão e não para distribuição e mesmo estas técnicas possuem algumas limitações. As técnicas propostas aqui dependem apenas da aquisição dos sinais da rede para serem usados como dados de treinamento e para avaliar se está ocorrendo a falta. É com este intuito que este trabalho elabora uma rede neural artificial recorrente utilizando os sinais de uma rede simulada que funciona em três módulos um para detecção de falta que denota se está ocorrendo alguma falta no sistema, outro que localiza se a falta ocorre na linha entre dois sensores e por último um módulo que identifica qual que o tipo de falta que está ocorrendo na linha.

Palavras-chave: Redes Neurais. Faltas. Sistema Elétrico.

ABSTRACT

This work documents a methodology to create an intelligent system for detecting, locating and classifying short circuits of an electric power system and, as a result, a study is performed of the technical issues used in this methodology. Most fault locating techniques used today are geared towards transmission systems rather than distribution and even these techniques have some limitations. The techniques proposed here depend only on the acquisition of power grid signals to be used as training data and to assess whether the fault is occurring. It is with this aim that this work elaborates a recurrent artificial neural network using the signals of a simulated network that works in three modules: one for fault detection that denotes if there is a fault in the system, another that finds if the fault occurs in the line between two sensors and lastly a module that identifies what type of fault is occurring on the line.

Keywords: Neural Networks. Faults. Power Grid.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

Um futuro com energia sustentável depende de um sistema de transmissão e distribuição, i.e. sistema elétrico, eficiente, confiável e inteligente. Define-se a *smart grid* como um sistema elétrico automatizado que monitora e controla atividades da rede, garantindo um fluxo nos dois sentidos tanto de energia quanto de informação entre os geradores, consumidores e qualquer ponto intermediário. A *smart grid* do futuro terá que atuar de forma eficiente para satisfazer a crescente capacidade de demanda e ao mesmo tempo usar o máximo da rede atual para manter os custos baixos (WU *et al.*, 2011).

A partir dos sistemas de informação da *smart grid* uma quantidade de dados significante pode ser obtida e como ferramenta de análise de dados uma opção interessante seria o aprendizado de máquina . Cada vez mais áreas como ciência dos dados, *machine learning* e inteligência artificial estão se tornando atuantes no mercado, na engenharia e no meio acadêmico. Os benefícios que as técnicas destas áreas podem trazer ao sistema elétrico ainda são pouco conhecidos e estudados. Entretanto, pode-se dizer que eles poderiam ir desde detecção de padrões até a otimização de resultados desejados de um sistema o que resultaria em um sistema mais eficiente, confiável e inteligente. Aliado a isso existe um forte incentivo econômico a este tipo de tecnologia e tendência de crescimento de investimentos da indústria nesta área (IDC, 2019).

Em sistemas elétricos de potência muitos alarmes são transmitidos ao centro de controle após a ocorrência de distúrbios provocados por diferentes tipos de faltas. Os aparatos de proteção são responsáveis por detectar e isolar somente a parte defeituosa do sistema. A bibliografia sugere a utilização de sistemas inteligentes na tarefa de diagnóstico de faltas a fim de deixar todo o processo mais inteligente e mais fácil de ser automatizado (CARDOSO JR; ROLIM; ZÜRN, 2004).

Este trabalho propõe fazer exatamente isto, uma pesquisa quanto às inovações que algoritmos de *machine learning* que foram propostos nos últimos anos podem trazer elaborando uma metodologia utilizando redes neurais artificiais a fim de fazer uma detecção de faltas robusta, localizar faltas apenas pelos sinais dos sensores e diagnosticar os tipos de faltas que ocorrem por reconhecimento de padrões nos sinais do sistema elétrico. Tudo isto é claro pressupondo que os dados se encontram a disposição. Explorando este problema com estas técnicas pode fazer iluminar um pouco mais o problema de automação do sistema e fazer com que atuemos de maneira

proativa ao invés de reativa ao lidarmos com o sistema.

1.2 REVISÃO DO ESTADO DA ARTE

A maioria das técnicas de localização de faltas foi desenvolvida para os sistemas de transmissão. Um destas técnicas é baseada na teoria das ondas viajantes na qual ondas são injetadas na linha e através das descontinuidades geradas pelas faltas na linha é possível calcular a distância da falta pelo do tempo de reflexão das ondas. No entanto, existem alguns problemas que podem ocorrer ao utilizar a técnica das "ondas viajantes"como quando a falta ocorre próximo ao ângulo de incidência de zero graus o que faz que não sejam gerados componentes significativos de ondas viajantes. Outros problemas com esta técnica é a limitação da banda de passagem dos transdutores utilizados que por conta disso se tornam imprecisos e não confiáveis e a necessidade de equipamentos especiais de custo elevado (OLESKOVICZ, 2001) (MORETO, 2005).

No entanto as técnicas mais utilizadas são as chamadas técnicas baseadas em medições de fasores de frequência fundamental. Estes métodos baseiam-se na medição da impedância aparente através de fasores de tensão e corrente em regime permanente obtidos na frequência fundamental. A distância de uma determinada referência até o ponto de ocorrência da falta é calculada a partir da impedância da linha sob falta, podendo ser utilizados para o cálculo desta impedância registros oscilográficos obtidos de um terminal ou, a fim de aumentar a precisão da tarefa de localização da falta, de dois ou mais terminais do trecho protegido (LIMA, 2013).

Um problema que pode surgir na detecção, localização e classificação de faltas é a existência de uma impedância de contato de valor elevado. Uma impedância de falta de valor grande pode fazer a falta passar despercebida pelos módulos de detecção convencionais (BAQUI *et al.*, 2011).

Uma solução alternativa aos métodos tradicionais é a utilização de técnicas de inteligência artificial que são particularmente boas quando se dispõe de um grande volume de dados os quais tornam possível fazer sistemas criados com estas técnicas capazes de resolver problemas que ainda não se tem conhecimento específico sem necessariamente utilizar uma matemática complexa. Isto acontece através do reconhecimento de padrões entre os dados e as respostas desejadas. Além disso estes sistemas inteligentes são capazes de se adaptar ao longo do tempo tornando-os robustos. Entre as diversas técnicas de IA(inteligência artificial), destacam-se os sistemas especialistas, as redes neurais, a lógica difusa e os algoritmos genéticos (CARDOSO JR; ROLIM; ZÜRN, 2004).

O uso de redes neurais artificiais(RNA) para detecção de faltas tem tido crescente aumento de interesse na literatura ao longo do tempo como alguns trabalhos proeminentes pode-se citar (OLESKOVICZ, 2001) que já elabora um esquema de proteção que envolve módulos de detecção, classificação e localização através de tensões e correntes amostradas no período de pós-falta. Em (MORETO, 2005) é feito um estudo de localização de faltas de alta impedância com uma metodologia que resultou em um percentual de erro menor do que 1,5%.

Muitos outros trabalhos de pesquisa voltados para aplicação de redes neurais artificiais para a detecção de faltas em sistemas de potência demonstraram resultados encorajadores e levaram ao desenvolvimento deste trabalho, apesar que não tenham sido mencionados aqui.

1.3 OBJETIVOS

Este trabalho visa modelar e validar técnicas de ciência dos dados e aprendizado de máquina para sistemas de potência com a finalidade de comprovar a hipótese de que é possível detectar falhas em uma microrrede. Como sub-objetivos específicos pode-se listar:

- Criação de um modelo de sistema elétrico de uma microrrede.
- Tratamento dos dados e sinais fornecidos pela simulação.
- Elaboração de um modelo matemático que possa prever falhas.
- Implementação do modelo através de algoritmos.
- Criação de um modelo que identificasse a falta apenas entre os sensores de uma linha. Qualquer outra falta fora desta linha ou até mesmo a ausência de faltas seria reconhecida como um exemplo negativo(ausência de falta nela). Assim poderíamos ter uma localização de onde foi a falta.
- Detecção de faltas de alta impedância.
- Indentificar qual foi o tipo de falta.
- Avaliação de possíveis aplicações e melhorias no processo.

Ao final da conclusão destes objetivos será obtido um sistema inteligente como mostra a figura 1.

Figura 1 – Diagrama de um sistema inteligente para lidar com faltas, composto dos módulos de detecção, localização e classificação de faltas. Fonte: Autoria Própria

1.4 ESTRUTURA DO TRABALHO

O referencial teórico e um pouco da bibliografia sobre redes neurais é apresentado no capitulo 2 de forma bem didática. O capítulo inicia com uma visão geral do conceito de redes neurais e, posteriormente, é apresentado o neurônio biológico, o modelo de neurônio artificial, as arquiteturas de rede, os paradigmas de aprendizagem. Por fim, são detalhadas as redes neurais recorrentes e alguns outros conceitos essenciais para seu entendimento.

No capítulo 3 é apresentada uma breve revisão da literatura e teoria de conceitos básicos de sistemas de potência com mais enfase sobre faltas em linhas de transmissão. São apresentados os tipos de curtos-circuitos, como eles ocorrem e sua modelagem.

No capítulo 4 são apresentadas as metodologias e ferramentas utilizadas de forma detalhada, quais foram as técnicas utilizadas, linguagens, os softwares, bibliotecas e por fim o computador utilizado. Os resultados obtidos através dos experimentos, através de gráficos, tabelas, figuras de mérito todos embasados são apresentados no capítulo 5. A conclusão do trabalho bem como sugestões de possíveis futuras melhorias do projeto são apresentadas no capítulo 6

2 REDES NEURAIS

2.1 INTRODUÇÃO

A natureza sempre foi para o homem uma fonte de inspiração e epifanias, pássaros nos inspiraram a voar e a natureza também nos inspirou em outras invenções. Assim como nossa própria estrutura nervosa é um sistema de processamento de informação altamente complexo, não linear e potente nada mais justo que na procura de construir uma máquina inteligente olhássemos para o cérebro como inspiração. Está foi a principal ideia que inspirou o conceito das RNAs. Porém assim como aviões que foram inspirados em pássaros não tem que bater asas, similarmente as RNAs se tornaram bem diferente dos seus primos biológicos. Alguns pesquisadores e especialistas até sugerem que esqueçamos desta analogia biológica de uma vez para chamar os neurônios de unidades (GÉRON, 2017).

RNAs são o cerne do Deep Learning que é o sub-campo mais popular no momento pertencente ao campo da ciência da computação denominado Inteligência Artificial (IA). As RNAs possuem alta versatilidade, poder e escalabilidade fazendo delas a ferramenta ideal para problemas de Machine Learning altamente complexos, como classificação de imagens, reconhecimento de voz, recomendação de produtos que usados todos os dias ou até mesmo derrotando o campeão mundial do jogo de tabuleiro Go examinando milhões de jogos passados jogando com si mesmo (CHANG *et al.*, 2016).

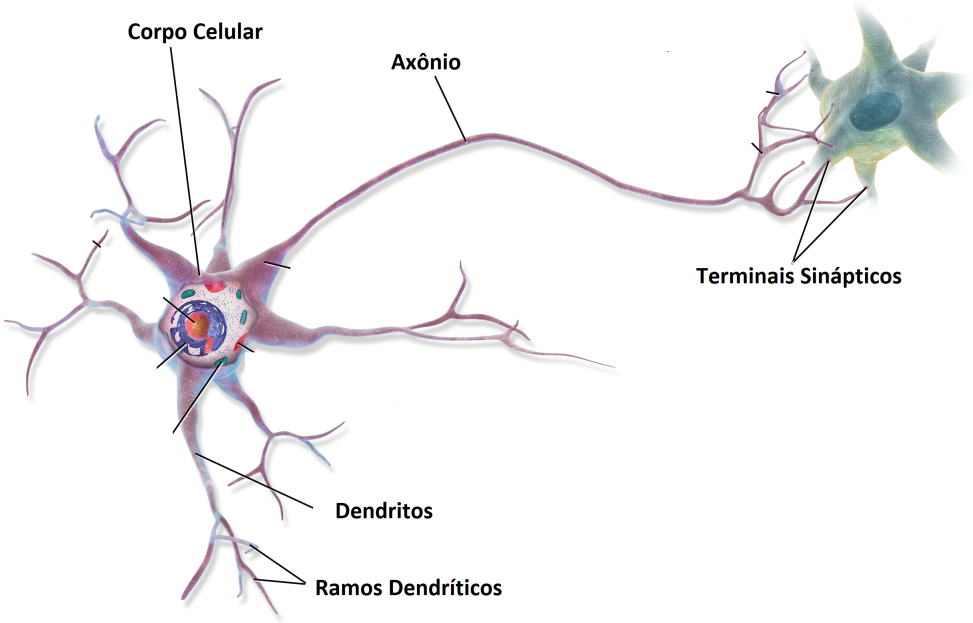
Neste capítulo será introduzido todo o embasamento teórico, citando ocasionalmente fatos históricos em alguns momentos, de um neurônio até uma rede neural e por fim a uma rede neural recorrente.

2.2 O NEURÔNIO BIOLÓGICO

Antes de olharmos para para a unidade básica de processamento de uma rede neural artificial, o neurônio artificial, é imprescindível saber um pouco sobre sua fonte de inspiração o neurônio biológico ilustrado na figura 2[1]. Os conceitos chave que precisamos saber sobre um neurônio para entendermos melhor sobre RNAs são listados a seguir:

- Corpo celular também chamado de *somma* contém o núcleo e todos os componentes complexos das células. Quando o corpo celular recebe um sinal forte suficiente que ultrapassa um certo limiar ele "dispara"um sinal que por sua vez é transmitido a outros neurônios.
- *Dendritos* ramos de entrada do neurônio, receptores que se ligam ao axônios.
- *Axônios* são os canais que transportam o sinal de um neurônio a outro se ramificando em diversos filamentos terminais chamados de telodendros.
- *Sinapses* são os sinais eletroquímicos que os neurônios enviam uns para os outros.

Figura 2 – Neurônio Biológico. Fonte: Adaptado de BruceBlaus. wikimedia.org

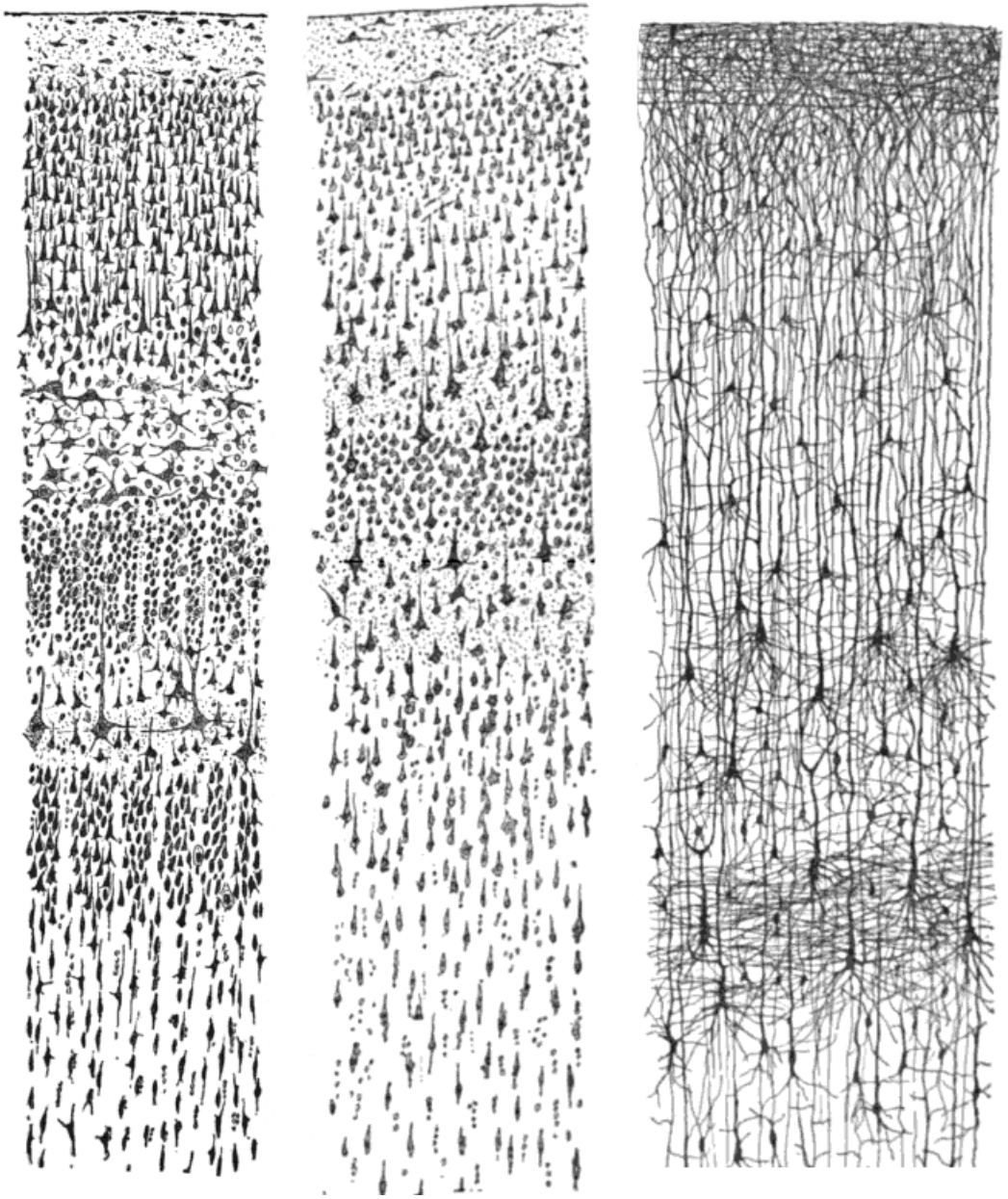


Dessa forma os neurônios parecem se comportar de maneira relativamente simples, porém como eles são organizados em redes de bilhões de outros neurônios,

[1] Acesso no site https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png em 01/11/2019

cada neurônio conectado a milhares de outros, eles acabam sendo capazes de funções altamente complexas. Estas arquiteturas ainda estão sendo estudadas mas há indícios que em muitos lugares elas lembrem redes consecutivas como a figura 3 abaixo[2].

Figura 3 – Camadas de Neurônios desenhadas a partir de um recorte de um cérebro. Fonte: Santiago Ramon y Cajal. wikimedia.org



[2] Acesso no site https://en.wikipedia.org/wiki/Cerebral_cortex#/media/File:Cajal_cortex_drawings.png em 02/11/2019

2.3 MODELO DO NEURÔNIO ARTIFICIAL

O neurônio artificial é uma ideia que foi introduzida em 1943 pelo neurofisiologista Warren McCulloch e o matemático Walter Pitts em seu artigo que entrou para a historia (MCCULLOCH; PITTS, 1943). Lá eles propuseram um modelo computacional simplificado de como um neurônio biológico deveria funcionar por lógica proposicional.

2.3.1 O PERCEPTRON

O perceptron é o mais simples tipo de neurônio das arquiteturas de RNAs foi desenvolvido por Frank Rosenblatt (ROSENBLATT, 1958) inspirado pelo neurônio de McCulloch e Pitss. Seu comportamento depende de entradas binárias de 0s e 1s que multiplicadas cada uma respectivamente pelo seus pesos geram uma saída binária de 0 e 1 também, isto dependendo se a somatório dos pesos e entradas for maior que um determinado limiar. Um diagrama de um perceptron pode ser observado na figura 5. Colocando de forma mais precisa em termos algébricos conforme a equação (1):

$$\text{saída} = \begin{cases} 0 \text{ se } \sum_{n=1}^{j} w_j x_j \leq \text{limiar} \\ 1 \text{ se } \sum_{n=1}^{j} w_j x_j > \text{limiar} \end{cases} \tag{1}$$

Onde:

x_j é um sinal de entrada do neurônio;
w_j é o "peso sináptico"do mesmo sinal de entrada, que é um número real que representa a importância daquele sinal.

É possível simplificar a equação (1) ainda mais com $\mathbf{w}^t \cdot \mathbf{x} \equiv \sum_{n=1}^{j} w_j x_j$ e definindo o *bias* como $b \equiv -$limiar na forma de (2):

$$\text{saída} = \begin{cases} 0 \text{ se } \mathbf{w}^t \cdot \mathbf{x} + b \leq 0 \\ 1 \text{ se } \mathbf{w}^t \cdot \mathbf{x} + b > 0 \end{cases} \tag{2}$$

Pode-se generalizar este modelo definindo-se (3) como a saída do somatório e (4) como uma função degrau. Podemos generalizar $g(z)$ como uma função de ativação qualquer mas no caso do perceptron ela tem a forma de uma função degrau.

$$z = \mathbf{w}^t \cdot \mathbf{x} \tag{3}$$

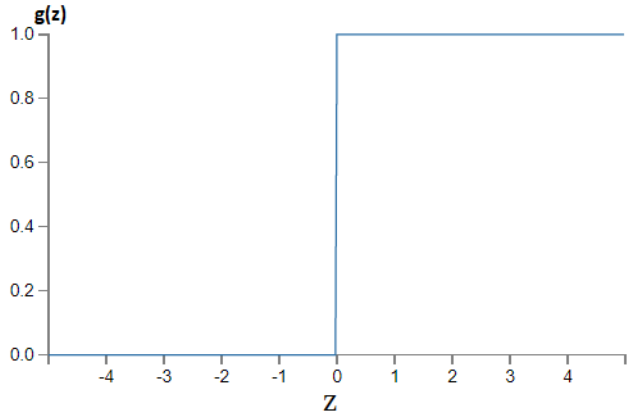
Onde: $\mathbf{x^t} = \begin{bmatrix} x_0 \dots x_n \end{bmatrix}$ é o vetor de sinais de entrada e $x_0 = 1$;

$\mathbf{w^t} = \begin{bmatrix} w_0 \dots w_n \end{bmatrix}$ é o vetor de pesos com w_0 sendo o bias.

$$g(z) = step(z) \tag{4}$$

Assim o sinal de saída com $b = 0$ fica mostrado na figura 4:

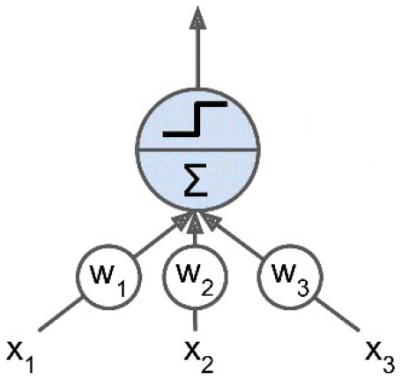
Figura 4 – Sinal de sáida com $b = 0$. Fonte adaptado de (NIELSEN, 2015)



2.4 FUNÇÕES DE ATIVAÇÃO

Segundo (KARLIK; OLGAC, 2011) as funções de ativação são funções matemáticas onde que transformam a ativação de um neurônio no de saída do mesmo neurônio. São preferíveis funções não lineares, pois pode-se provar que duas camadas da rede se tornam um aproximador universal de funções (CYBENKO, 1992).

Figura 5 – Perceptron, modelo de neurônio de Rosemblatt. Fonte: adaptado de (GÉRON, 2017)



São exemplificados nesta seção as funções mais clássicas bem como aquelas que foram usadas neste trabalho.

2.4.1 FUNÇÃO DEGRAU

A função degrau, também chamada de função de Heaviside, foi a primeira função a ser usada em perceptrons (ROSENBLATT, 1958). Ela é ilustrada na figura 4 e obedece a expressão:

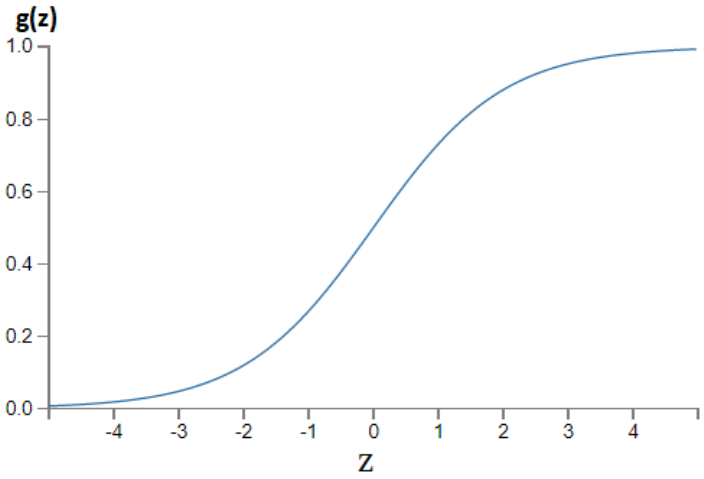
$$\text{g(z)} = \begin{cases} 0 \text{ , } z \leq 0 \\ 1 \text{ , } z > 0 \end{cases} \tag{5}$$

2.4.2 FUNÇÃO SIGMÓIDE

A função sigmoide também conhecida por função logística, tem sua expressão descrita conforme (6). Para valores muito altos ou baixos de z a saída muda muito pouco assim ela apresenta o problema do *vanishing gradient* o que faz que o treinamento não convirja para uma solução boa (GOODFELLOW; BENGIO; COURVILLE, 2016). Esta função é ilustrada na figura 6.

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-}z} \tag{6}$$

Figura 6 – Gráfico da função sigmoide. Fonte: adaptado de (NIELSEN, 2015)

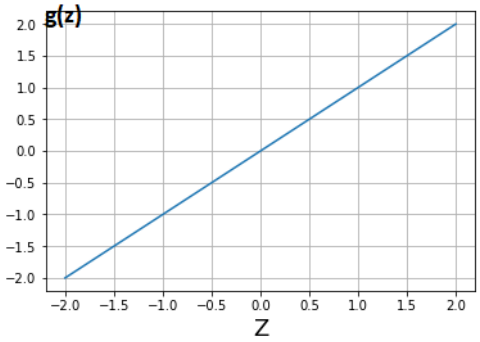


2.4.3 FUNÇÃO IDENTIDADE

Função linear que não limita a saída do neurônio. Como é uma função linear as camadas formadas por ela não conseguem generalizar qualquer função (CYBENKO, 1992). A expressão da identidade fica descrita na equação (7) e um gráfico dela pode ser visto na imagem 7.

$$g(z) = z \tag{7}$$

Figura 7 – Gráfico da função identidade. Fonte: Autoria própria



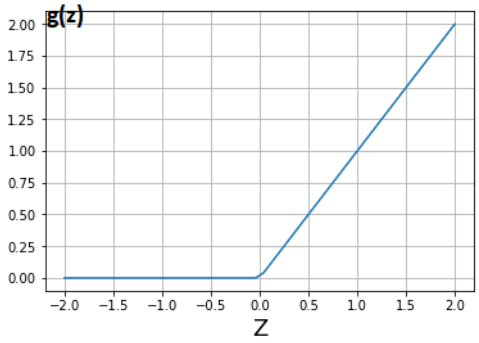
2.4.4 FUNÇÃO RELU

Em redes neurais modernas é a função *default* a ser usada ela é definida pela equação (8) e ilustrada na figura 8. A função RELU é uma função linear por partes e

também é uma função que aplicada uma entrada linear causa uma transformação não-linear. Assim por ela ser quase linear ela preserva as propriedades que fazem modelos lineares generalizarem bem e serem fáceis de otimizar com métodos baseados em gradientes segundo os autores (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$g(z) = max(0, z) \tag{8}$$

Figura 8 – Gráfico da função RELU. Fonte: Autoria própria



2.4.5 FUNÇÃO TANGENTE HIPERBÓLICA

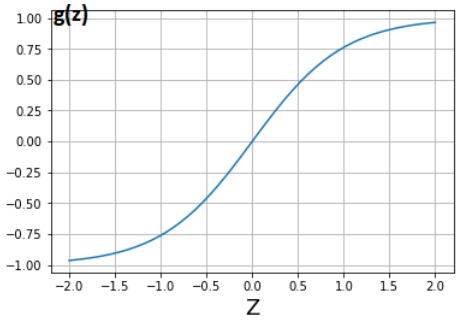
A função de ativação tanh, tangente hperbólica, tem bastante proximidade com a sigmoide uma vez que $tanh(z) = 2\sigma(2z) - 1$. Geralmente quando é necessário uma ativação sigmoidal a tanh tem desempenho melhor que sigmoide por ser mais próxima de uma função linear (GOODFELLOW; BENGIO; COURVILLE, 2016). Sua função é descrita matematicamente pela equação (9) com um gráfico na figura 9.

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{9}$$

2.5 PARADIGMAS DE APRENDIZAGEM

Um algoritmo de machine learning é um algoritmo que é capaz de aprender através de dados e melhorar seu desempenho. Isto pode ser feito através de um processo iterativo de ajustes aplicado aos pesos sinápticos da rede, chamado de treinamento. O

Figura 9 – Gráfico da função tanh. Fonte: Autoria própria



aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma determinada classe de problemas (MORETO, 2005).

A seguinte definição é dada por (MITCHELL, 1997) "Um computador aprende com a experiencia **E** com respeito a um conjunto de tarefas **T** e métrica de performance **P**, se sua performance na tarefa **T**, medida por **P**, melhora com a experiência **E**".

Sistemas de machine learning podem ser classificados de acordo com a quantidade de supervisão que eles recebem durante o treinamento. Existem quatro categorias principais, aprendizado supervisionado, não supervisionado, semi-supervisionado e por reforço, no trabalho só serão abordados os aprendizados supervisonado e não-supervisonado pois os outros fogem do escopo deste trabalho.

2.5.1 APRENDIZADO SUPERVISIONADO

No aprendizado supervisionado os dados já incluem as soluções desejada chamadas de *labels*. Uma típica tarefa de aprendizado supervisionado é a classificação aonde os labels são as classes a que pertencem as observações.

A diferença entre o valor desejado e o valor calculado pela rede gera um sinal de erro, que é utilizado para ajustar os pesos sinápticos da rede, de modo que a resposta da rede seja mais próxima possível da esperada (OLESKOVICZ, 2001).

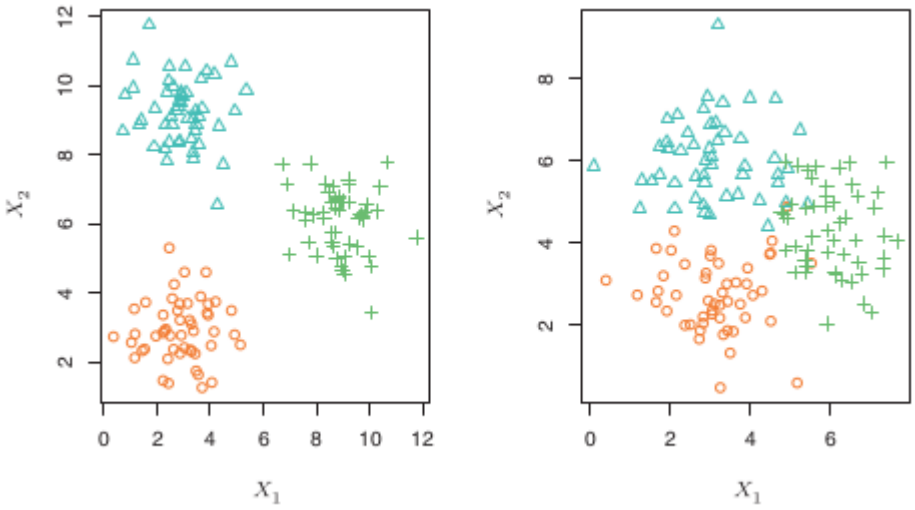
Um exemplo prático já usado há muito tempo é o filtro de spam dos emails que são treinados junto com suas classes(spam ou não-spam), para aprender a classificar novos emails.

2.5.2 APRENDIZADO NÃO SUPERVISIONADO

Em aprendizagem não supervisionada os dados não possuem as soluções desejadas ou seja não possuem *labels*. O sistema tenta aprender sem professor. A rede é responsável por agrupar os dados de entrada conforme algum critério de similaridade que ela define (OLESKOVICZ, 2001).

Geralmente em machine learning é necessário aprender a distribuição de probabilidade dos dados, como por exemplo em uma detecção de anomalias é útil saber quais as observações são *outliers*. Já em outros tipos de problema o algoritmo divide as observações, detectando padrões em suas variaveis, em diferentes *clusters* que nada mais são que aglomerados de exemplos semelhantes, como ilustrado na figura 10.

Figura 10 – Exemplo de dados clusterizados a equerda nota-se uma clara divisão, já a direita é possível observar que os clusters se sobrepõem um pouco. Fonte: Adaptado de (JAMES *et al.*, 2013)



2.6 ARQUITETURAS DE REDES NEURAIS ARTIFICIAIS

Um único neurônio é capaz apenas de uma saída que representa a intensidade de cada entrada após passar por uma função de ativação e portanto é insuficiente para representar qualquer resposta mais complexa. Por isso redes com grandes números de neurônios são frequentemente utilizadas.

Os neurônios das redes neurais são agrupados em camadas e a arquitetura de uma rede neural é definida pela quantidade de camadas de neurônios e pelo tipo de ligação entre eles. Existem três tipos principais de grupos de redes neurais as redes diretas, redes recorrentes e redes convolucionais. Neste trabalho serão abordadas

apenas as redes diretas e as recorrentes pois a rede convolucional foge do foco dos objetivos.

2.6.1 Redes diretas de camada única

Este tipo de arquitetura de RNA, tem-se apenas uma camada de neurônios, como se observa na figura 11. Elas são divididas em duas partes a camada de entrada que são as variáveis de cada observação ou sinal, e a camada de saída que é a representação da resposta. Se for um problema de classificação binário será apenas um neurônio já em uma classificação multi-classe mais de um neurônio. Estas redes são chamadas de redes de camada única pois possuem somente uma camada de neurônios, como pode ser visto na figura 11. Por conta de como a informação é passada direto para a camada seguinte sem haver nenhuma realimentação de sáida as estas redes levam são chamadas de diretas.

A camada de saída, e seus neurônios através do treinamento serão responsáveis por mapear a saída pelas entradas ajustando os pesos e os *bias* de cada neurônio. Uma ressalva importante é que apesar de apenas uma única camada ser suficiente para representar qualquer função a quantidade de neurônios na camada pode se tornar extremamente grande, fazer com que não seja possível o aprendizado e ela não seja capaz de generalizar (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.6.2 Redes neurais diretas de múltiplas camadas

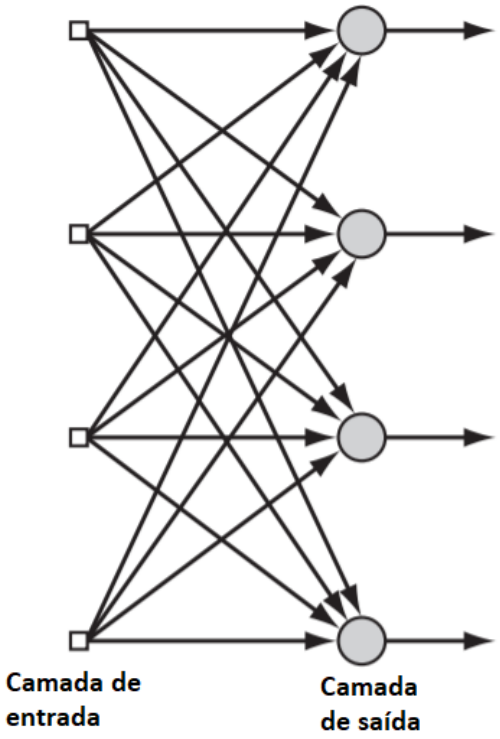
As redes de múltiplas camadas, chamadas também de redes profundas ou *Multi-layer Perceptron*(MLP), se diferenciam das redes de camada única por possuírem outras camadas de neurônios entre a camada de saída e a de entrada, conhecidas como camadas ocultas.

Adicionando mais camadas e mais unidades, uma rede neural consegue representar funções de maior complexidade através das camadas ocultas. Durante o processo de treinamento o algoritmo decidirá como serão usadas esta camadas porém não é possível prever qual será seu comportamento no final. Uma rede Neural é ilustrada na figura 12

2.6.3 Redes Neurais Recorrentes

Redes neurais recorrentes, em inglês *recurrent neural networks*(rnns), são a família de redes neurais usadas para processar dados sequênciais. As rnns são muito parecidas com as redes convencionais com a diferença que a saída é retro-alimentada como uma outra entrada, exemplificado na figura 13.

Figura 11 – Rede neural de uma única camada. Fonte: (HAYKIN, 2010)



Cada neurônio recorrente possui um par de pesos: um para suas entradas $\mathbf{x_{(t)}}$ e o outro para a saída no instante de tempo prévio $\mathbf{y_{(t-1)}}$, denotando estes vetores de pesos como $\mathbf{w_{(x)}}$ e $\mathbf{w_{(y)}}$. Considerando toda a camada recorrente ao invés de um só neurônio e colocando os vetores de pesos em duas matrizes de pesos $\mathbf{W_{(x)}}$ e $\mathbf{W_{(y)}}$.

Agora com isso o vetor de saída da camada no instante de tempo t pode ser expresso conforme a equação (10). Aonde $\mathbf{b}$ é o vetor de *bias* da camada e $g(\cdot)$ é a função de ativação. Uma camada de neurônios recorrentes é ilustrada na figura 14.

$$\mathbf{y_t} = g(\mathbf{W_x}^T \cdot \mathbf{x_{(t)}} + \mathbf{W_{(y)}}^T \cdot \mathbf{y_{(t-1)}} + \mathbf{b}) \tag{10}$$

2.7 PROCESSOS DA APRENDIZAGEM DA REDE

2.7.1 Propagação Direta

Para que a rede neural faça uma previsão é necessário realizar o que é chamado de propagação direta, em inglês *forward pass*. Isto é similar ao que foi feito na subseção 2.3.1, apenas expandido para mais camadas e por conseguinte para toda a rede.

Figura 12 – Rede neural de múltiplas camadas. Fonte: (HAYKIN, 2010)

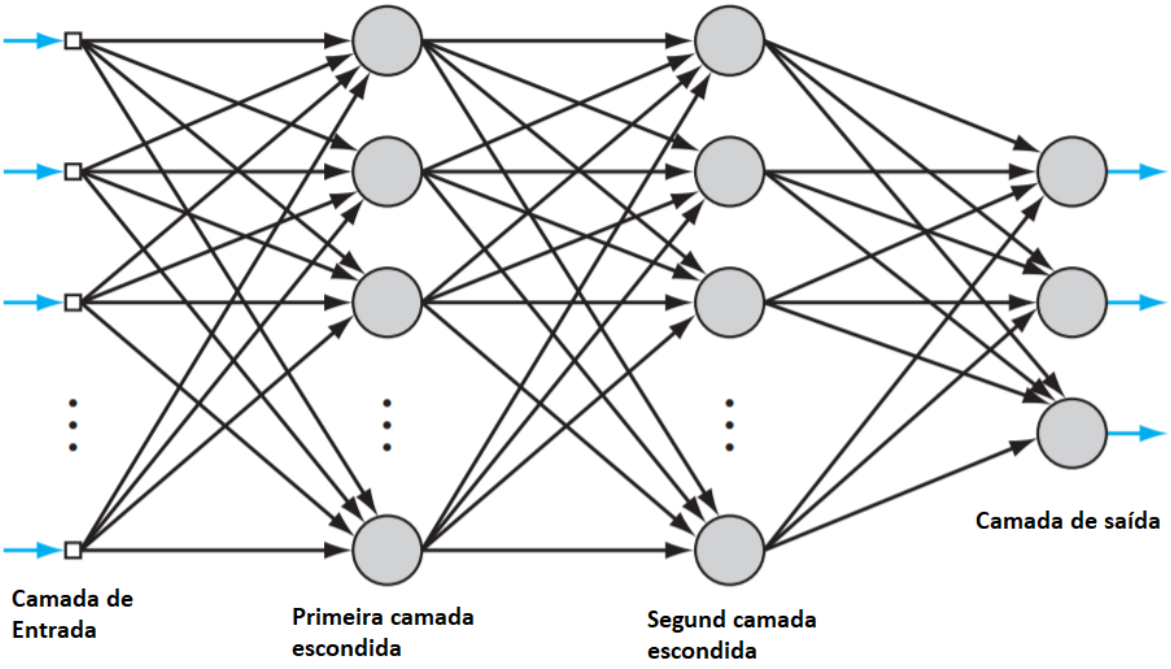
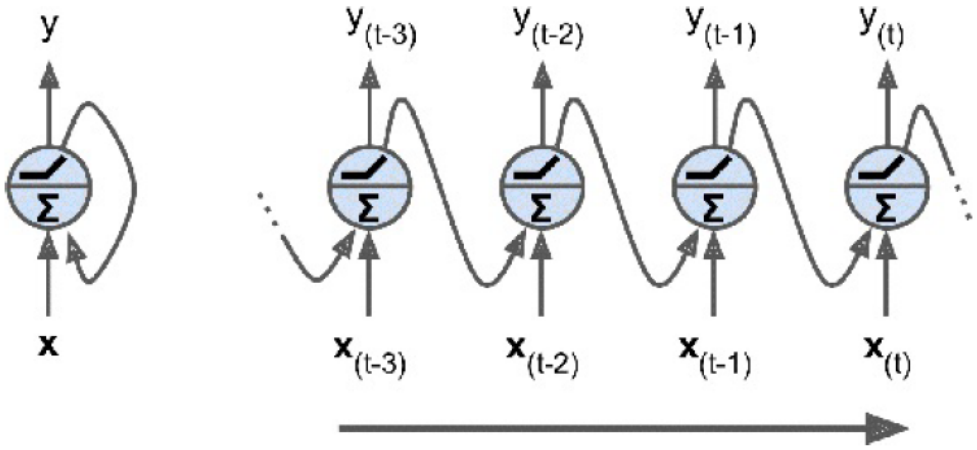


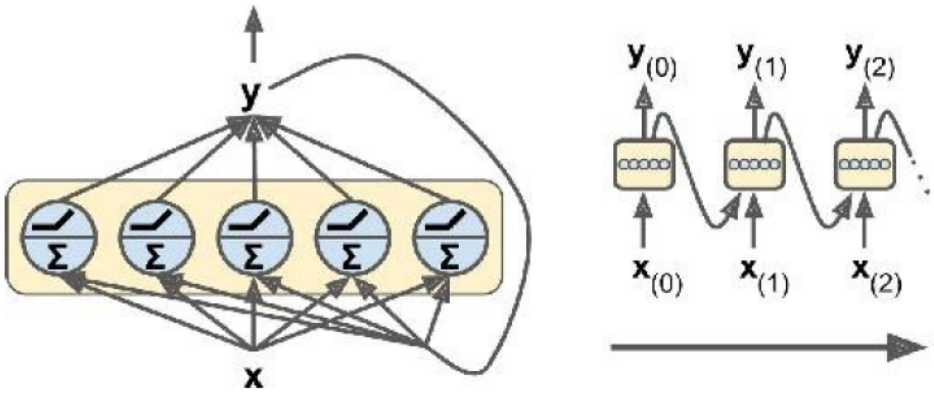
Figura 13 – Exemplo de um neurônio recorrente, a direita como seria o neurônio "desenrolado"no tempo. Fonte: (GÉRON, 2017)



A equação (11) é o vetor de ativação da camada e (12) é a saída da camada

$$\mathbf{z}^{[l]} = \mathbf{W}^{[1]} \cdot \mathbf{y}^{[l-1]} + \mathbf{b} \tag{11}$$

Figura 14 – Exemplo de uma camada recorrente, a direita como seria a camada "desenrolada"no tempo. Fonte: (GÉRON, 2017)



$$\mathbf{y}^{[l]} = g(\mathbf{z}^{[l]}) \tag{12}$$

Onde l denota o número da camada, $\mathbf{W}^{[\mathbf{l}]}$ é a matriz de pesos pertencente a camada e $\mathbf{b}$ é o vetor de *bias* da camada. $\mathbf{y}^{[l]}$ é a saída da camada e $g(\cdot)$ é a função de ativação.

2.7.2 Função Custo

Quando treinamos uma rede o objetivo é achar os parâmetros, pesos e *bias*, que façam a saída da rede se aproximar o máximo possível de $y(x)$, saída esperada, para todo x. Para quantificar o quão próximas as saídas previstas pela rede estão das saídas reais é utilizada o que é chamada de função custo, função perda ou até mesmo função objetivo.

Abaixo na equação (13) um exemplo de uma função custo chamada de erro médio quadrado, em inglês *Mean Square Error* MSE. Nos algoritmos de aprendizagem são estas funções que são otimizadas e que acabam acarretando um desempenho melhor do modelo durante o treinamento.

Em 13 $J(w)$ é a função custo, n o número de exemplos, y a saída esperada do exemplo x e $\hat{y}$ é a saída prevista pela rede.

$$J(x) = \frac{1}{2n} \sum_{x} \|y(x) - \hat{y}\|^2 \tag{13}$$

2.7.3 Algoritmo *backpropagation*

Por muitos anos os pesquisadores tiveram muita dificuldade para treinar MLPs. Até que em 1986 D. E. Rumelhart et. al. publicaram um artigo que mudou isso (RUMELHART; HINTON; WILLIAMS, 1985). Neste artigo eles descreveram como um algoritmo chamado de *backpropagation*, que já havia sido introduzido nos anos 70, funcionava bem mais rápido que métodos anteriores. Isto tornou possível as redes neurais resolverem problemas que anteriormente não podiam ser resolvidos. Hoje em dia o algoritmo de *backpropagation* é o algoritmo central de aprendizagem em RNAs.

O algoritmo funciona da seguinte maneira; cada observação de treinamento é alimentada para a rede e calculam-se todas as saídas de cada neurônio em todas as camadas. Esta é a propagação direta(*forward pass*) usada para fazer previsões. Então o erro da saída da rede é calculado, diferença entre a saída que deveria ter ocorrido e a saída da rede, assim é calculado quanto cada neurônio contribui para o erro na camada de saída, camada por camada até chegar na camada de entrada.

A equação (14) representa o erro δ^L, o índice L representa a última camada, entre a saída e a saída prevista pela rede. Onde $\hat{y}^L$ é a sáida prevista pela rede da última camada, y é a sáida esperada, $g'(\cdot)$ é a derivada da função de ativação, z^L é a ativação na última camada e $\odot$ é conhecido como operador de Haddamard é a multiplicação elemento a elemento.

$$\delta^L = (\hat{y}^L - y) \odot g'(z^L) \tag{14}$$

Na equação (15) o índice l representa a camada, o que significa que w^{l+1} é a matriz de pesos da próxima camada e δ^{l+1} é o erro da camada posterior.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot g'(z^l) \tag{15}$$

A equação (16) representa a variação da função custo J em relação ao peso w_{jk} da camada l que depende de ${y_k}^{l-1}$, saída da camada anterior e δ^l o elemento j do vetor de erro desta camada. Já a equação (17) é a variação da função custo em relação ao *bias*.

$$\frac{\partial J}{\partial {w_{jk}}^l} = {y_k}^{l-1} {\delta_j}^l \tag{16}$$

$$\frac{\partial J}{\partial {b_j}^l} = {\delta_j}^l \tag{17}$$

No final os pesos e *bias* são atualizados segundo as regras (18) e (19):

$$w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^l((w^{l+1})^T \delta^{x,l+1}) \odot g'(z^{x,l})(y^{x,l-1})^T \tag{18}$$

$$b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^l((w^{l+1})^T \delta^{x,l+1}) \odot g'(z^{x,l}) \tag{19}$$

2.7.4 *Batch Size*

O *batch size*, também chamado de *minibatch*, é o número de exemplos de treinamento que são utilizados em cada iteração do algoritmo de otimização, o que causa os *updates* de parâmetros.

2.7.5 Épocas

Por convenção uma época é considerada quando o algoritmo de otimização passa por todos os exemplos de treinamento(GÉRON, 2017). Em uma aprendizagem todo o conjunto de treinamento é passado muitas vezes pela rede o que caracteriza muitas épocas.

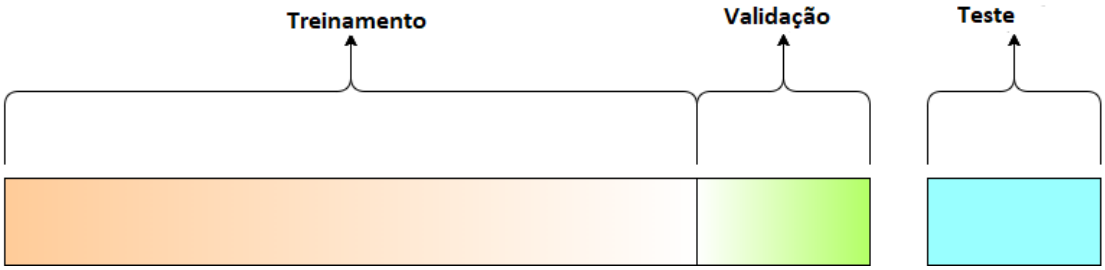
2.7.6 Hiper parâmetros e conjunto de validação

A maioria dos algoritmos de *machine learning* possuem diversas configurações que podem ser modificadas para ajustar o comportamento do modelo. Estas configurações ajustáveis, chamadas de hiper parâmetros, são os parâmetros que não podem ser treinados durante o aprendizado.

É possível otimizar estes parâmetros utilizando algum algoritmo de otimização. Por isso na hora de avaliar diversos modelos com hiper parâmetros diferente se utiliza um conjunto de validação e com base em alguma figura de mérito deste conjunto os parâmetros são otimizados. Para avaliar a capacidade de generalização dos algoritmos é feito um outro conjunto chamado de conjunto de teste. Esta distinção é feita entre conjunto de teste e validação para que não ocorra *overfitting*. Na figura 15[3] é ilustrado um esquema de partição dos dados em conjuntos uma prática corriqueira que tem como objetivo separas os dados para o treinamento, teste e validação.

[3] Acessado no site https://tarangshah.com/ em 08/11/2019.

Figura 15 – Partição dos dados em conjuntos de treinamento teste e validação. Fonte:Tarang Shah, tarangshah.com



3 ANÁLISE DE CURTO CIRCUITO

Em um Sistema Elétrico de Potência (SEP) a transmissão é a área responsável por transportar a energia gerada pelas usinas para os centros de distribuição e consumo, com o mínimo de perdas. As linhas de transmissão (LT) são os principais componentes da Transmissão, e são compostas por torres, condutores, isoladores e para-raios (PINTO, 2014).

Por serem de grande porte e operarem em elevadas tensões, as LT são os componentes do SEP mais susceptíveis a defeitos. Por conseguinte, o foco dos sistemas de proteção é isolar as faltas no menor tempo possível para que a integridade da linha e a estabilidade do sistema não sejam ameaçadas (OLIVEIRA, 2005).

Essas faltas podem ser de vários tipos, envolvendo um ou mais elementos do sistema de potência, sendo geralmente classificadas em ordem decrescente de frequência de ocorrência. Nos sistemas de transmissão de Alta Tensão, esta ordem geralmente é a seguinte (SALGADO, 2018):

1. falta fase-terra ($\phi - \frac{\perp}{=}$)
2. falta fase-fase (2ϕ)
3. falta fase-fase-terra ($2\phi - \frac{\perp}{=}$)
4. falta trifásica (3ϕ)

Em linhas de transmissão, a detecção e localização de faltas são possíveis por meio da utilização de métodos como, por exemplo, os baseados em ondas viajantes e pelo uso de relés de distância. Esses métodos apresentam precisão bastante aceitável(PETITE; SANTOS; ASANO, 2017).

Já em sistemas de distribuição, tais métodos se mostram ineficazes devido às diferentes topologias da rede, existência ou não de geradores distribuídos (GD), diferentes níveis de carga, seções da rede com condutores de diferentes bitolas e falta de conhecimento da exata impedância equivalente do sistema (TRINDADE, 2013).

3.1 TIPOS DE FALTA

Serão aprofundados nesta seção os tipos de falta que podem ocorrer nas LTs de um sistema elétrico.

3.1.1 Curto-circuito Fase-Terra

A falta mais comum em sistemas de potência com incidência de 63% entre as faltas é também a menos grave (KINDERMANN, 2010), ocorre quando há contato entre uma fase e o terra. O curto é dito franco quando não existe a resistência de falta entre a fase e a terra(SALGADO, 2018). Na figura 16 é ilustrado um curto circuito monofásico e na (17) seu circuito equivalente.

Figura 16 – Curto-circuito monofásico. Fonte: (ZANETTA JR, 2006)

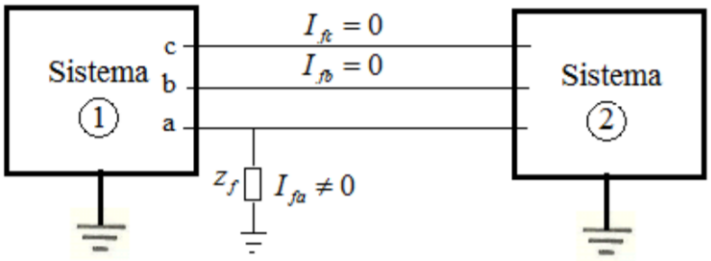
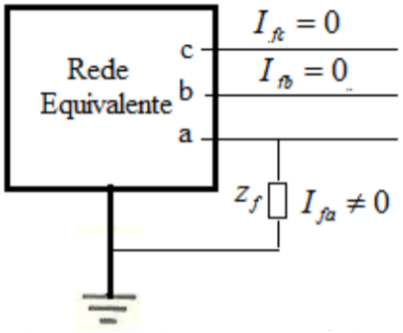


Figura 17 – Circuito equivalente do Curto-circuito monofásico. Fonte: (ZANETTA JR, 2006)



Segundo (ZANETTA JR, 2006) a corrente e a tensão da falta são descritas por(20) e (21):

$$\frac{1}{3} \cdot \dot{\mathbf{I}}_{fa} = \dot{\mathbf{I}}_{a0} = \dot{\mathbf{I}}_{a1} = \dot{\mathbf{I}}_{a2} \tag{20}$$

$$\dot{\mathbf{V}}_{fa} = \dot{\mathbf{V}}_{a0} + \dot{\mathbf{V}}_{a1} + \dot{\mathbf{V}}_{a2} = \dot{\mathbf{Z}}_f \cdot 3 \cdot \dot{\mathbf{I}}_{a0} \tag{21}$$

Onde:

$\dot{\mathbf{I}}_{fa}$ é a corrente de falta;

$\dot{\mathbf{V}}_{fa}$ é a tensão de falta.;

$\dot{\mathbf{V}}_{a0}$, $\dot{\mathbf{V}}_{a1}$ e $\dot{\mathbf{V}}_{a2}$ são as tensões de sequência zero, positiva e negativa respectivamente;

$\dot{\mathbf{I}}_{a0}$, $\dot{\mathbf{I}}_{a1}$ e $\dot{\mathbf{V}}\mathbf{I}_{a2}$ são as correntes de sequência zero, positiva e negativa respectivamente;

$\dot{\mathbf{Z}}_f$ é a impedância de falta.

3.1.2 Curto-circuito Fase-Fase

As faltas deste tipo geralmente ocorrem quando dois condutores da linha de fases distintas entram em contato um com o outro. Na figura 18 é ilustrado um curto-circuito bifásico e seu circuito equivalente na figura 19.

Abaixo nas equações (22) são descritas as corrente de falta de uma falta do tipo fase-fase:

$$\begin{aligned} \dot{\mathbf{I}}_{fa} = \dot{\mathbf{I}}_{a0} + \dot{\mathbf{I}}_{a1} + \dot{\mathbf{I}}_{a2} = 0 \\ \dot{\mathbf{I}}_{a0} = 0 \\ \dot{\mathbf{I}}_{a0} = -\dot{\mathbf{I}}_{a1} \end{aligned} \tag{22}$$

Figura 18 – Curto Circuito bifásico. Fonte: (ZANETTA JR, 2006)

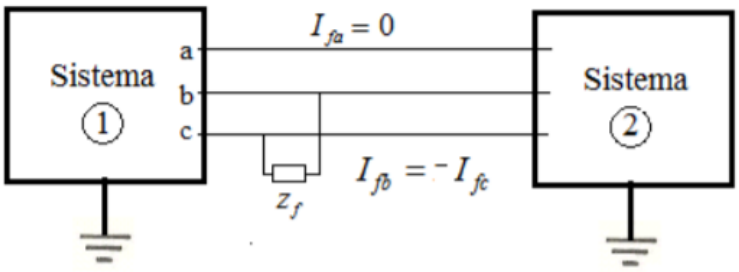
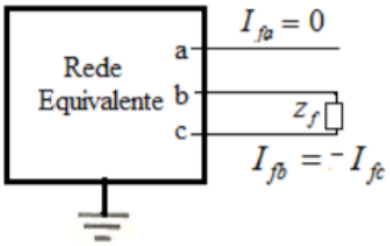


Figura 19 – Circuito equivalente do Curto-circuito bifásico. Fonte: (ZANETTA JR, 2006)



3.1.3 Curto-circuito Fase-Fase-Terra

Este tipo de falta que geralmente ocorre quando dois condutores da linha de fases diferentes entram em contato com o solo. Junto com a falta fase-fase tem uma ocorrência de 16% (KINDERMANN, 2010). Nas figuras 20 e 21 são mostrados um curto-circuito fase-fase-terra e seu circuito equivalente.

As condições que descrevem analiticamente a falta fase-fase-terra entre a fase a e b estão descritas em (23) abaixo.

$$\begin{aligned} \mathbf{V}_b &= \mathbf{V}_b = 0 \\ \mathbf{I}_a &= 0 \\ \frac{\mathbf{V}_a}{3} &= \mathbf{V}_{a0} + \mathbf{V}_{a1} + \mathbf{V}_{a2} \\ \mathbf{I}_a &= \mathbf{I}_{a0} + \mathbf{I}_{a1} + \mathbf{I}_{a2} \\ \mathbf{I}_{fa} &= \mathbf{I}_b + \mathbf{I}_c = 3\mathbf{I}_{a0} \end{aligned} \tag{23}$$

Figura 20 – Curto Circuito bifásico em contato com o terra. Fonte: (ZANETTA JR, 2006)

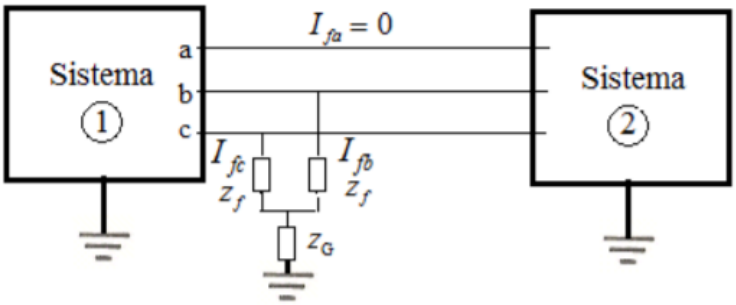
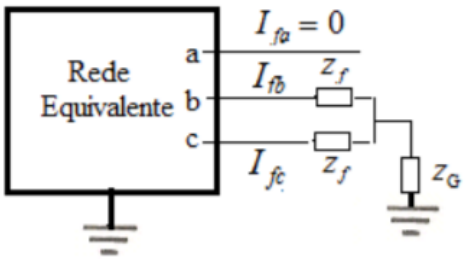


Figura 21 – Circuito equivalente do Curto-circuito bifásico em contato com o terra. Fonte: (ZANETTA JR, 2006)



3.1.4 Curto-circuito trifásico

Esta falta é caracterizada quando os condutores das três fases entram em contato entre si podendo ou não entrar em contato com o solo. Dos tipos de falta é o mais raro ocorrendo apenas 6% das vezes, porém é o mais grave de todos (KINDERMANN,

2010). Na figura 22[1] são ilustradas faltas sem contato com o terra e com contato com o terra, já na figura 23[2] é mostrada uma falta com impedâncias iguais.

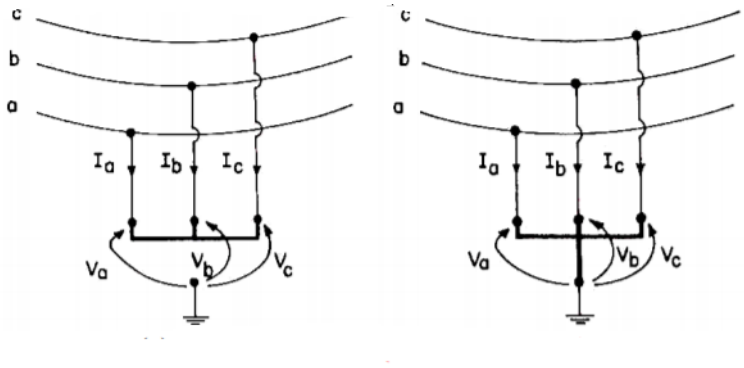
Este tipo de falta é caracterizado pelas seguintes equações (24):

$$\begin{aligned} \mathbf{V}_a = \mathbf{V}_b = \mathbf{V}_c = 0 \\ \mathbf{I}_a + \mathbf{I}_b + \mathbf{I}_c = 0 \end{aligned} \tag{24}$$

Em termos de componentes de sequência (25):

$$\begin{aligned} \mathbf{V}_{a0} = \mathbf{V}_{a1} = \mathbf{V}_{a2} = 0 \\ \mathbf{I}_{a0} = 0 \\ \mathbf{I}_{a1} = \mathbf{I}_a \\ \mathbf{I}_{a2} = 0 \end{aligned} \tag{25}$$

Figura 22 – Curto Circuito trifásico. Fonte: Prof. Carlos Medeiros, https://sites.google.com/site/cx3medeiros



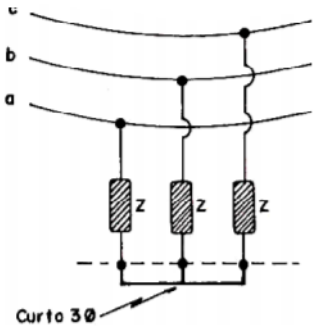
3.2 IMPEDÂNCIA DE FALTA

A impedância no local da falta $\mathbf{Z_f}$ está sempre presente no curto circuito, no entanto nem sempre é fácil se obter valores típicos para ela e seu cálculo é difícil. Assim é relativamente aceitável considerar seu valor como zero, um curto solidamente aterrado. Isso implica fazer o projeto para correntes maiores que as reais e assim aumentar o nível de segurança.

Podemos citar alguns exemplos no quais a impedância de falta tem sua origem como resistência do arco elétrico entre o condutor e a terra ou entre dois condutores,

[1] Acesso ao site https://sites.google.com/site/cx3medeiros em 09/10/2019
[2] Acesso ao site https://sites.google.com/site/cx3medeiros em 09/10/2019

Figura 23 – Falta trifásica com cargas iguais Fonte: Prof. Carlos Medeiros, https://sites.google.com/site/cx3medeiros

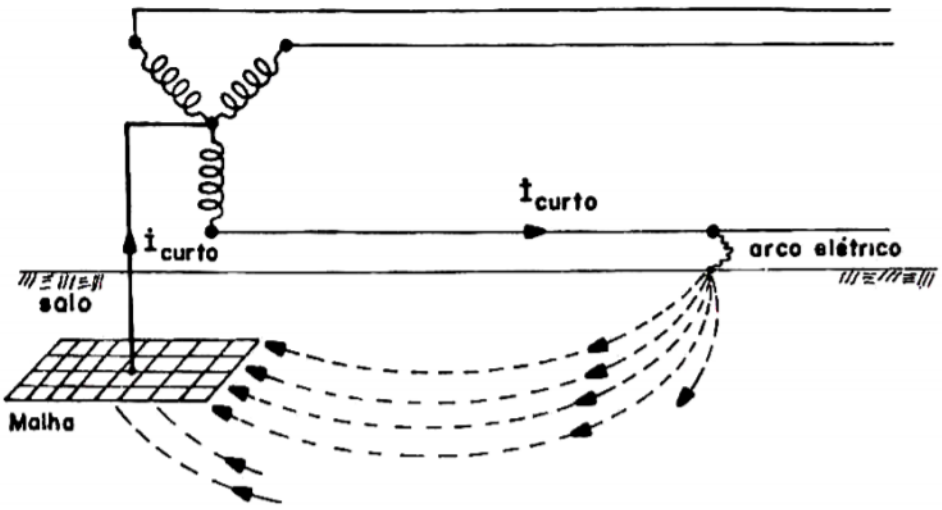


resistência de contato devido a oxidação do local, resistência da camada mais superficial do solo, resistência de terra no local, etc (KINDERMANN, 2010). Na figura 24 abaixo é ilustrada uma falta que resulta de um arco elétrico.

Existe porém a chance da impedância da falta ter valor elevado, de modo que não possa ser considerada zero. Devido à alta impedância, a falta não gera corrente o suficiente para acionar os relés de sobrecorrente. Essas faltas são muito difíceis de serem detectadas, por causa da baixa magnitude da corrente, que é facilmente confundida com uma simples variação de carga (MORETO, 2005).

As faltas de alta impedância são normalmente causadas pelo contato dos condutores com superfícies e objetos de alta impedância, como solo seco, concreto, cascalho, galhos de árvores, etc. Como há grande diferença de tensão entre o condutor e um objeto de alta impedância em contato com o solo pode ocorrer um arco elétrico. Apesar de este tipo de falha não ser muito grave para o sistema elétrico ela traz grande risco a segurança humana (GHADERI; GINN III; MOHAMMADPOUR, 2017).

Figura 24 – Arco elétrico formado por uma falta, https://sites.google.com/site/cx3medeiros



4 METODOLOGIA

4.1 RECURSOS UTILIZADOS

Para a criação de um sistema elétrico de potência foram utilizados neste trabalho o software MATLAB em conjunto com o ambiente de simulação *Simulink* e neste o *toolbox power systems*.

No desenvolvimento dos modelos e no tratamento dos dados foram utilizadas, as seguintes ferramentas e linguagens:

- Python: É uma linguagem de programação de alto nivel, interpretada e orientada a objetos . Possui estruturas de dados altamente dinâmicas e uma sintaxe muito clara. Bastante difundida na ciência e industria e especialmente *data science*. É multiplataforma, sendo possível executa-la em Unix, Mac e Windows (VAN ROSSUM *et al.*, 2007).
- Pandas: É uma biblioteca de Python utilizada para manipulação e análise de dados. É um software livre. Oferece um conjunto completo de ferramentas para trabalhar com conjuntos de dados estruturados conhecidos como *dataframes* (MCKINNEY, 2011).
- NumPy: É uma biblioteca de python de análise numérica e funções matemáticas que opera com vetores e matrizes sendo implementada em linguagem C para permitir cálculos computacionais eficientes (OLIPHANT, 2006).
- Scikit-Learn: É um pacote da linguagem Python que fornece um grande conjunto de ferramentas que auxiliam processos de machine learning. Possui código fonte aberto (PEDREGOSA *et al.*, 2011).
- Tensorflow: É uma biblioteca de código aberto para computação numérica de alta performance para aplicações de *machine learning*. Permite computação paralela tanto em *Central Processing Unit* (CPU) quanto em *Graphics Processing Unit* (GPU). (ABADI *et al.*, 2016).
- Keras: É uma *Application Programing Interface* (API) de alto-nível para desenvolvimento de redes neurais e pode utilizar o Tensorflow como *backend*. Foi desenvolvida em Python com o foco de proporcionar ao usuário uma rápida experimentação através da criação de redes em camadas (CHOLLET *et al.*, 2015).

Tabela 1 – Tabela com as versões de cada uma das ferramentas. Fonte: Autoria Própria

Ferramenta	Versão	Ferramenta	Versão
Python	3.6.3	Keras	2.1.5
Pandas	0.24.2	MaplotLib	2.2.2
NumPy	1.16.4	Anaconda	4.7.5
Scikit-Learn	0.20.0	Matlab	9.4
Tensorflow	1.8.0	Simulink	9.1

Tabela 2 – Tabela com as especificações das máquinas utilizadas. Fonte: Autoria Própria

Sistema operacional	Versão	Processador	Memória
Windows	10	i7 2.0GHZ	16 GB
Ubuntu	18.10	i3 2.3 GHZ x 4	3,7 GB

- Matplotlib: É uma biblioteca para plotar gráficos da linguagem python de rápida implementação. Parte do principio que vai se incrementando características ao gráfico até ele ter a aparência desejada (HUNTER, 2007).

Para a realização dos experimentos e protótipos tanto na construção de modelos quanto no tratamento de dados em Python foram utilizados:

- Anaconda: É uma plataforma de distribuição Python, R e outras linguagens voltadas a ciências dos dados que visa simplificar o gerenciamento de pacotes, visto que em alguns sistemas operacionais como o Windows isto pode ser especialmente complicado. Possui código fonte aberto, é multiplataforma e possui uma fácil instalação (WANG, 2012);
- Jupyter Notebook: É um ambiente web interativo de programação que permite que os código sejam executados em células. Também é possível criar células em markdown e dentro destas utilizar formatação LaTeX. Além disso é possivel plotar os gráficos no próprio notebook. Todas estas características tornam as tarefas de programar protótipos, fazer análises e documentar muito mais rápidas. (KLUYVER *et al.*, 2016). Abaixo na figura 25[1] um exemplo de um notebook no jupyter que utiliza markdown, LaTeX, códigos interativos e gráficos.

A fim de garantir a replicabilidade deste trabalho a tabela 1 explicita quais foram as versões de cada uma das ferramentas utilizadas. Já a tabela 2 mostra as especificações das maquinas utilizadas neste trabalho.

[1] Acesso no site https://jupyter.org/ em 11/11/2019

Figura 25 – Exemplo de um Notebook no Jupyter,. Fonte: jupyter.org

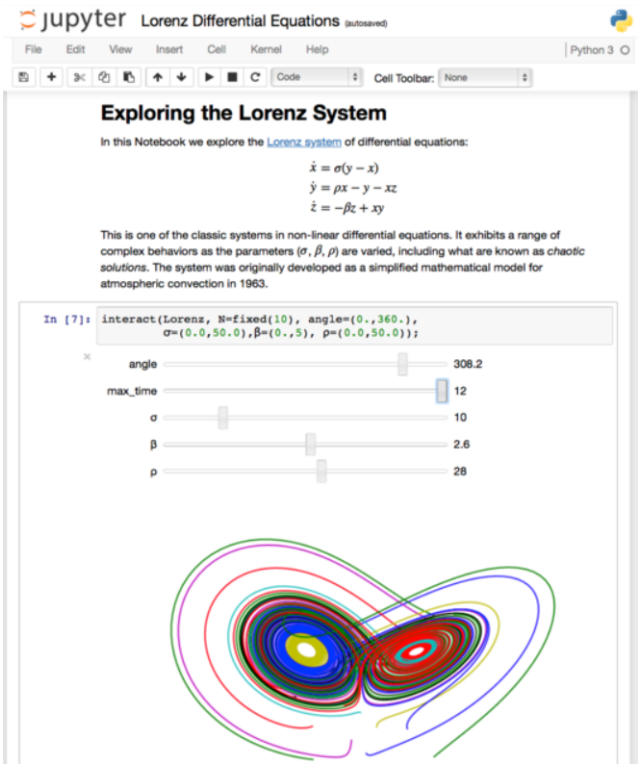


Tabela 3 – Especificações dos elementos do sistema de potência. Fonte: Autoria Própria

Elemento	Parâmetros	Elemento	Parâmetros
G_1	Y: 20kV	T_1	500kVA,Δ20kV:Y400V
G_2	Y: 20kV	T_2	500kVA,Δ20kV:Y400V
Z_L	$0,017 + j0,095$	L_1	110kVA

4.2 SIMULAÇÃO DA LINHA DE TRANSMISSÃO

O sistema utilizado foi um sistema simples, como pode ser visto na figura 26, com o intuito de facilitar a tarefa de simular uma falta e obter as medidas dos sinais para detecções de faltas e outros experimentos semelhantes. Na tabela 3 se encontram os parâmetros dos elementos do sistema.

Para a simulação do sistema de potência este trabalho utilizou o ambiente de simulação do MATLAB chamado simulink, sendo utilizados principalmente os blocos do *Power Systems Toolbox*. Uma imagem do sistema simulado no simulink pode ser vista na figura 27.

Para a falta foi utilizado o bloco *three-Phase Fault* visto na figura 28, que permite simular faltas de todos os tipos e também ajustar a impedância de falta. Como parâmetros principais este bloco possui as opções de definir o intervalo de tempo que

Figura 26 – Diagrama do sistema Elétrico utilizado, Fonte: Autoria Própria

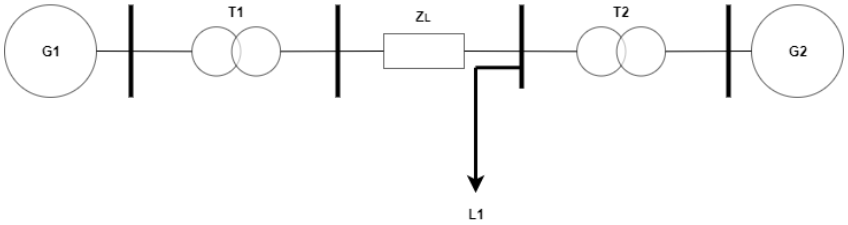
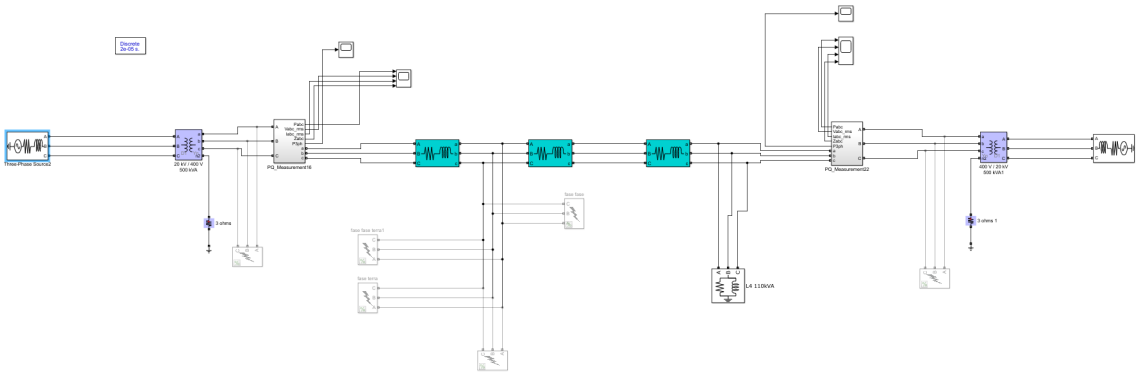


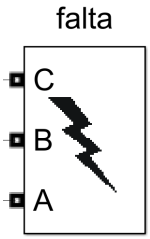
Figura 27 – Sistema elétrico simulado no simulink, Fonte: Autoria Própria



a falta ocorria, quais as conexões da falta entre as fases e o terra. Aliado a tudo isso é possível colocar este bloco em diferentes locais da linha.

Desta forma foi possível ter como variáveis do experimento de falta: local, tipo de falta, intervalo de falta e impedância de falta.

Figura 28 – Bloco de falta do simulink, Fonte: Autoria Própria



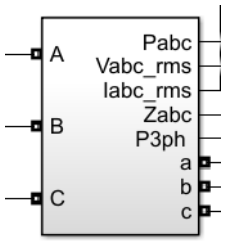
4.3 OBTENÇÃO DOS DADOS E EXPORTAÇÃO

Para adquirir os dados do simulink foi utilizado um bloco, figura 29, aonde são calculados os sinais de V_{rms}, I_{rms} e P das três fases a, b e c. Estes blocos salvam estes sinais em arquivos objeto *.mat*. Foi criado um script de MATLAB que automatizou o processo de passar estes arquivos para variáveis séries de tempo, permitindo que se salvassem estas variáveis em arquivos *.csv* e movessem estes arquivos para a pasta de dados do experimento.

Já em Python foi implementada também uma função que obtem estes arquivos *.csv* e colocando-os em *dataframes* do Pandas com as colunas nomeadas, isto para cada arquivo de sinais dos sensores. E finalmente permitindo que cada um destes *dataframes* ficasse armazenado em uma estrutura de dados do tipo dicionário.

É possível notar na figura 27 que os sensores estão posicionados entre a linha de transmissão modelada por três impedâncias e um bloco de carga. Isto foi feito pois entre os objetivos deste trabalho está localizar uma falta que ocorra dentro da linha, entre os sensores, ou fora dela.

Figura 29 – Bloco sensor de sinais, Fonte: Autoria Própria



4.4 TRATAMENTO DOS DADOS

4.4.1 Janelamento

Tendo os dados é necessário formata-los de modo que ele possam ser alimentados no modelo. Para fazer o treinamento de uma rede neural recorrente é necessário que os dados tenham o formato de um *array* 3D com cada dimensão sendo [quantidade de janelas, amostras de tempo, número variáveis].

Assim foi necessário fazer o "janelamento"dos dados. Como nas simulações do simulink as amostras dos sinais eram tomadas a cada 1 mili-segundo e um período da rede é de 16,67 mili-segundos foi decidido que cada janela teria 17 instantes de tempo o que seria aproximadamente igual um ciclo de rede.

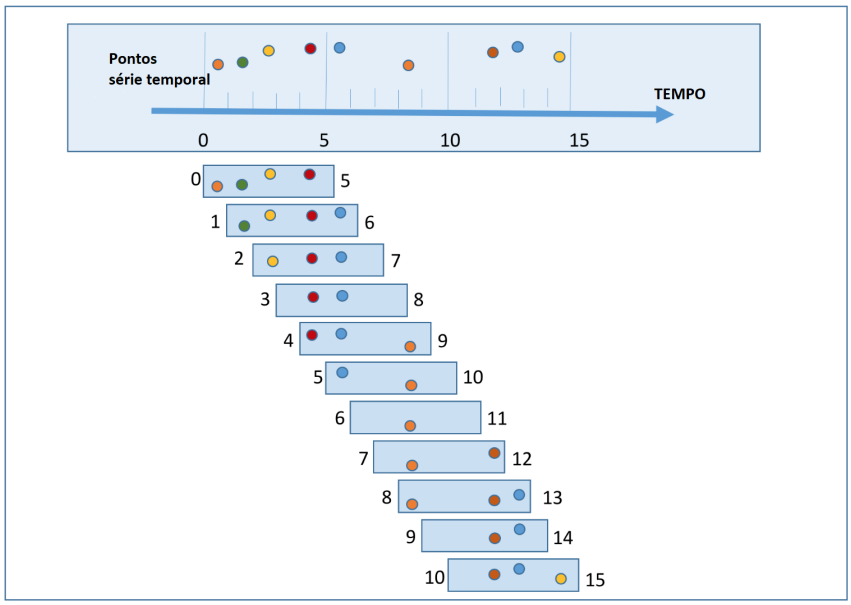
Estas janelas são janelas móveis o que significa que cada janela seguinte esta perderia o instante de tempo mais antigo e teria como instante de tempo novo o instante seguinte. Na figura 30[2] fica ilustrado com mais clareza o comportamento de janelas móveis.

4.4.2 *Feature Scaling*

Durante este trabalho notou-se uma melhora significativa no desempenho dos modelos quando foi feito o que se chama de *feature scaling* nos dados. Esta técnica

[2] Acesso no site https://docs.wavefront.com/query_language_windows_trends.html em 11/11/2019

Figura 30 – Funcionamento de uma janelas móveis, Fonte: Adaptado de docs.wavefront.com



consiste em pegar os dados de uma variável(*feature*) e "normalizar"cada ponto através de (26):

$$x' = \frac{x - \overline{x}}{\sigma} \tag{26}$$

onde x é cada ponto da variável, $\overline{x}$ é a média daquela variável e σ é o desvio padrão da variável.

A razão disso é que com poucas exceções a maioria dos algoritmos de machine learning não tem uma performance muito boa quando as variáveis não estão na mesma escala (GÉRON, 2017). Isto se deve ao fato de o otimizador não convergir tão bem ao mínimo da função custo em variaveis de escalas diferentes. A figura 31 ilustra isto de forma mais simples.

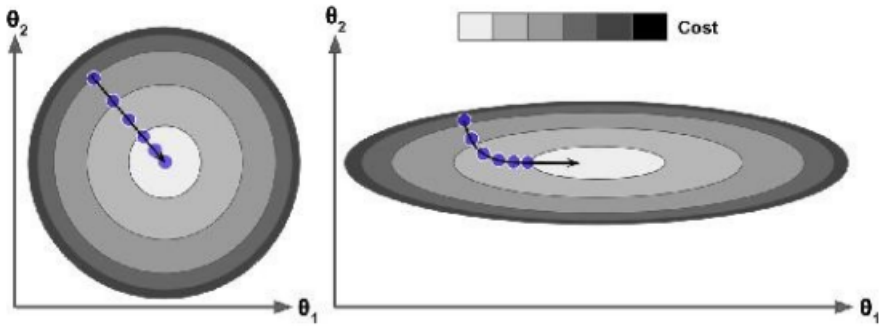
4.5 IMPLEMENTAÇÃO DA REDE NEURAL E AJUSTES

Para implementar a rede neural foi utilizado o objeto da biblioteca keras *sequential model()*. Nele as camadas da rede são acrescentadas uma por vez como por exemplo:

Listing 4.1 – Python example

```
model = Sequential()
```

Figura 31 – Otimização da função custo com e sem *feature scaling*, Fonte:(GÉRON, 2017)




```
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
```

O código acima faz uma rede neural de duas camadas internas com a primeira contendo 32 neurônios ativação relu, e na segunda um único neurônio e a ativação sendo uma sigmoide.

Os modelos em cada experimento possuem algumas diferenças mas no geral eles compartilharam características em comum.As próximas sub-seções explicitam as estrategias e razões das escolhas dos hiper-parâmetros.

4.5.1 Otimizador

O algoritmo otimizador é responsável por fazer o modelo aprender minimizando a função custo. Apesar de o gradiente descendente ser o mais conhecido hoje existe uma família de algoritmos conhecidos como algoritmos com taxas de aprendizado adaptativas(GOODFELLOW; BENGIO; COURVILLE, 2016) que tendem a ser mais rápidos.

Apesar de alguns autores (GOODFELLOW; BENGIO; COURVILLE, 2016) concluírem que não há prova de um algoritmo desses ser superior aos outros, a escolha *default* é o algoritmo ADAM (GÉRON, 2017).

4.5.2 Taxa de aprendizagem

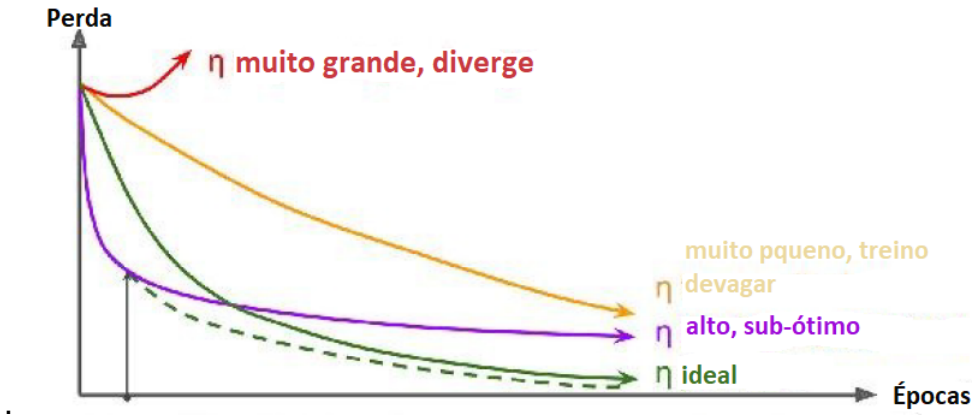
A variação que o modelo sofre durante cada "passo"que o algoritmo de otimização dá ao minimizar a função custo é chamada de taxa de aprendizado, simbolizada muitas vezes como η. Concretamente η é um escalar positivo entre 0 e 1.

Muitos pesquisadores consideram que se há apenas um hiper parâmetro a

ser ajustado este hiper parâmetro deve ser a taxa de aprendizado (GOODFELLOW; BENGIO; COURVILLE, 2016), (BENGIO, 2012). Apesar de não ser possível saber a priori qual a melhor taxa de aprendizado é recomendado que o valor seja maior que 10^{-6} e menor que 1 e adotar $0,01$ como valor inicial (BENGIO, 2012) e ir alterando o valor até uma melhora no desempenho.

Uma boa maneira de ir ajustando a taxa de aprendizado é sempre plotar um gráfico chamado de curva de aprendizado, que coloca as épocas do treinamento no eixo das abcissas e a perda oriunda da função custo no eixo das ordenadas. Conforme a figura 32 é possível ver que um η muito alto pode fazer os hiper-parâmetros divergirem já um η baixo faz com que o treinamento se torne muito devagar. Portanto é interessante plotar a curva de aprendizado e ir ajustando o η até que se tenha um gráfico com um comportamento bom o suficiente.

Figura 32 – Diferentes curvas de aprendizado dependendo do valor de η. Fonte: Adaptado de (GÉRON, 2017)

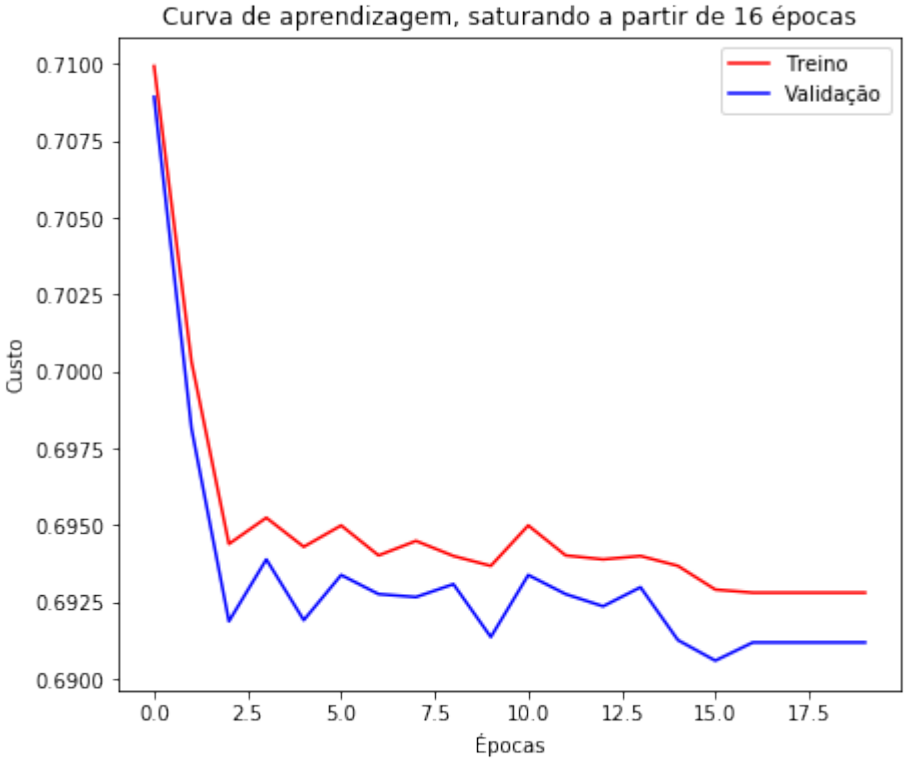


4.5.3 Número de Épocas

Como já dito anteriormente o número de épocas é o número de passagens que todo o conjunto de treinamento adentra as redes neurais durante a aprendizagem. O número de épocas ideal é aquele em que a função custo para de diminuir durante o aprendizado, após este número de épocas o modelo vai estar sendo treinado em vão.

Para ajustar o número de épocas basta fazer um gráfico da curva de aprendizagem e observar até próximo de qual época a função custo para de diminuir. Em algumas bibliotecas de *machine learning* é possível fazer isto de maneira automática utilizando o *learning rate scheduling* que a partir de um certo número de épocas sem mudança significativa na função custo automaticamente interrompe o treinamento. Como é possível ver na figura 33 a partir de certa época já não é necessário mais o treinamento.

Figura 33 – Curva de aprendizado saturando nas últimas épocas. Fonte: Autoria própria



4.5.4 Funções de Ativação

Para saber qual a melhor função de ativação foi utilizado o conjunto de validação com 20% dos dados durante o trabalho. O resultado obtido está ilustrado na figura 34. Como se pode ver a função de ativação com melhor desempenho foi a relu, como esperado segundo (LECUN *et al.*, 2012). Apesar de as funções logística e tangente hiperbólica serem semelhantes, a tanh obteve um resultado superior não somente a ela porém à softmax também.

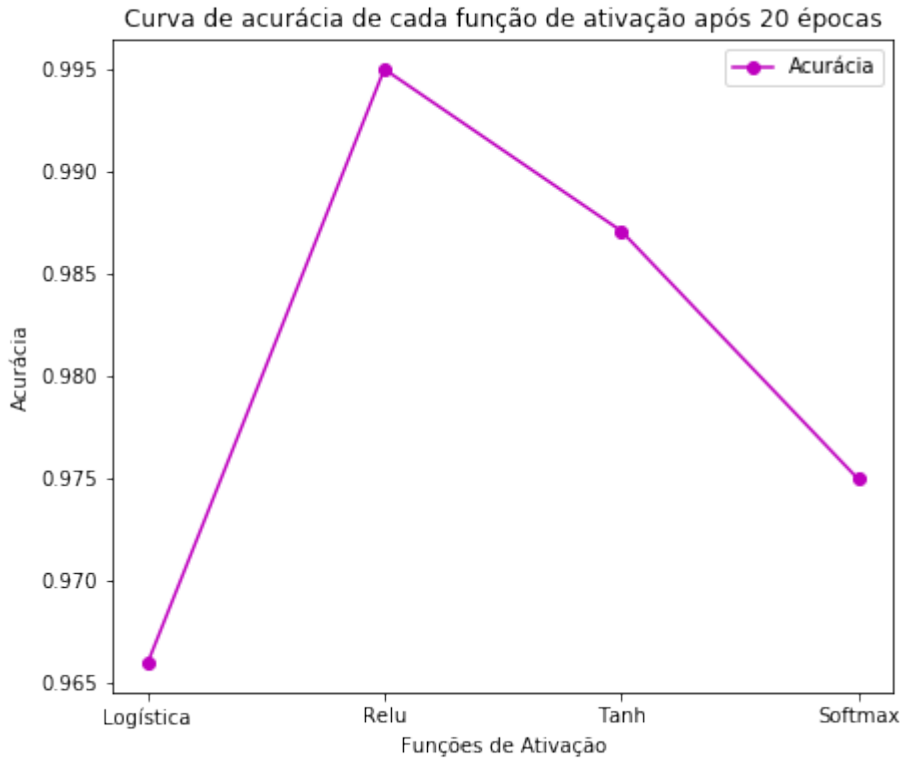
Surpreendentemente na classificação multi-classe a função logística apresentou melhor desempenho que a tanh como será visto no capitulo de simulações.

4.5.5 Número de neurônios e camadas ocultas

Segundo (GOODFELLOW; BENGIO; COURVILLE, 2016) um aumento do número de neurônios e camadas aumenta a capacidade de representação do modelo, porém ao mesmo tempo aumenta o tempo de treinamento e o custo de memoria computacional além de deixar o modelo mais propenso ao *overfitting*.

Como estratégia é recomendado que aumente-se o número de camadas e neurônios até que o modelo sinalize de estar próximo de *overfitting* (GÉRON, 2017)

Figura 34 – Comparação das funções de ativação. Fonte: Autoria própria



dentro do conjunto de treinamento. Na maioria dos experimentos com poucas camadas e neurônios o modelo já estava tendo uma acurácia bem alta. Por esta razão os neurônios e as camadas não foram tão ajustados.

4.5.6 Função Perda

Como funções perda foram utilizadas duas funções a função entropia cruzada binária que é dada pela equação (27) pois é uma das mais recomendadas no caso de classificações binárias pela maioria dos autores e o erro médio quadrático, equação apresentada em (28), pois foi a que apresentou melhores resultados na parte de classificações multi-classe (GÉRON, 2017), (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$J = -\sum_{i}^{m}(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{27}$$

$$J(x) = \frac{1}{2n}\sum_{x} \|y(x) - \hat{y}\|^2 \tag{28}$$

4.6 MÉTRICAS DE DESEMPENHO

Ao avaliar um classificador algumas métricas além da acurácia são necessárias para saber a qualidade do modelo. Nesta seção as principais figuras de mérito utilizadas neste trabalho são apresentadas.

Para isso é preciso determinar uma terminologia de classificações:

- **P**: Quantidade total de positivos, tanto falsos quanto verdadeiros.
- **N**: Quantidade total de negativos, verdadeiros e falsos.
- **TP**: Verdadeiro positivo, em inglês *True Positive*, quantos casos positivos foram aceitos corretamente.
- **TN**: Verdadeiro negativo, *True Negative* em inglês, quantos negativos foram rejeitados corretamente.
- **FP**: Falso positivo, *False Positive* em inglês, quantos negativos passaram por positivos pelo classificador, alarme falso.
- **FN**: False negativo, *False Negative* em inglês, quantos positivos foram erroneamente classificados.

4.6.1 Acurácia

A acurácia é a métrica mais simples para avaliar um modelo, ela é definida por (29). A acurácia acaba não sendo uma boa métrica de desempenho quando a distribuição das classes é muito desigual. Se uma classe ocorrer 99,999% das vezes e a outra 0,001% se um modelo cego sempre escolher a classe que ocorre mais vezes ele vai ter uma acurácia de 99,999% porém será um modelo ruim.

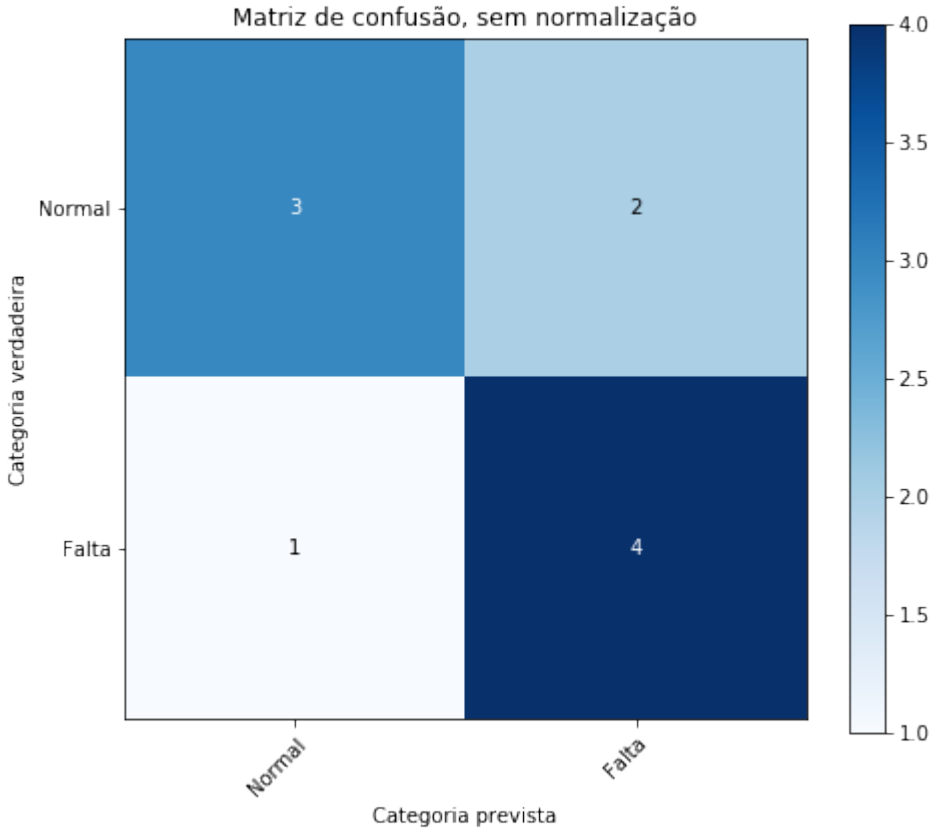
$$\text{Acurácia} = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{P} + \mathbf{N}} \tag{29}$$

4.6.2 Matriz de Confusão

Uma maneira muito melhor de avaliar a performance de um classificador é olhar para a matriz de confusão. A ideia da matriz de confusão é contar a quantidade de vezes que uma classe foi classificada corretamente ou não e colocar na coluna e linha apropriada.

Na figura 35 é possível ver como a matriz funciona de maneira melhor. As linhas representam os valores reais das classes e as colunas os valores previstos. Dela é possível tirar de forma direta **TP**, **TN**, **FP** e **FN**.

Figura 35 – Matriz de confusão de um classificador binário. Fonte: Autoria própria



4.6.3 Precisão e Recall

Outras duas métricas interessantes e mais concisas que podem ser retiradas da matriz de confusão são a precisão e o *recall*. A precisão, expressa na equação (30), mostra quanto daquilo que foi sinalizado como positivo realmente era positivo.

$$\text{Precisão} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}} \tag{30}$$

Já o *recall*, qual é proporção do que foi classificado como positivo entre o que realmente era positivo, descrito pela equação (31). Com a matriz de confusão é possível obter essas métricas facilmente.

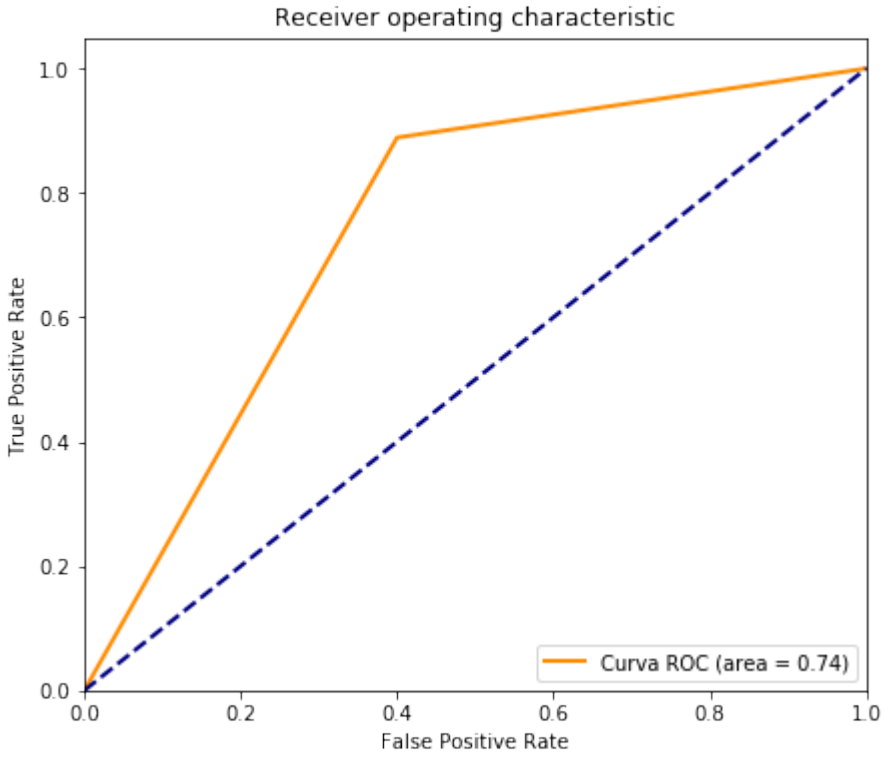
$$\textit{Recall} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} \tag{31}$$

4.6.4 Curva ROC e AUC

A curva ROC é a curva traçada a partir do *recall* e **FPR**, em inglês *false positive rate* dado pela equação (32), levando em conta varios limiares. Uma métrica muito importante que vem da curva ROC é o AUC a área sob a curva, *area under the curve*, que permite dizer se o classificador está classificando de forma totalmente aleatória AUC $= 0,5$ ou classificando de forma perfeita AUC $= 1,0$. Na figura 36 é possível ver uma curva ROC e seu AUC no canto inferior direito. A curva ROC foi uilizada neste trabalho como meio de obter o AUC, métrica usada para avaliar o desempenho do modelo.

$$\textbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}} \tag{32}$$

Figura 36 – Curva ROC de um classificador binário com AUC no canto inferior direito. Fonte: Autoria própria



5 SIMULAÇÕES

Neste capitulo são descritas as simulações realizadas e as conclusões obtidas a partir delas. No decorrer deste trabalho quando um resultado satisfatório foi obtido dentro de cada experimento foram elaborados experimentos com maior complexidade com o intuito de tentar resolver problemas cada vez maiores.

5.1 DETECÇÃO DE FALTA

O primeiro problema atacado por este trabalho é o de saber se um modelo de *machine learning* conseguiria detectar uma falta. As condições de classificação foram sendo pioradas aos poucos.

5.1.1 Detecção de um tipo de falta

O primeiro experimento realizado foi comprovar a hipótese de que era possível criar um modelo que identificasse se houve um tipo específico de falta a partir de um único sinal da rede. Para este tipo de modelo era necessário apenas um sinal de um dos dois sensores da rede que foi elaborada, ver figuras 26 e 27.

5.1.1.1 Falta trifásica

Na tabela 4 abaixo são fornecidos os dados da primeira simulação. A figura 37 mostra os sinais do sistema neste experimento. A tabela 5 mostra os hiper-parâmetros da rede neural. Para obtenção destes resultados os dados dos sinais foram normalizados, utilizando *feature scaling*. A Primeira camada, cujas entradas são os sinais utilizados podendo ser a tensão, corrente ou potência do sistema, em todos os experimentos é uma RNN simples de também 60 neurônios gerando 60 saídas, na biblioteca keras *SimpleRNN*. Já as outras camadas são camadas normais *dense* na biblioteca keras.

Duração da simulação (s)	Período de Falta (s)
2,15	1,15 - 2,15
tipo de falta	impedância de falta (Ω)
3ϕ	0,001
Quantidade de amostras	Tamanho conjunto de treinamento
1985	1191
Tamanho do conjunto de validação	Tamanho do conjunto de teste
397	397

Tabela 4 – Dados da primeira simulação. Fonte: Autoria Própria

Figura 37 – Sinais de potência, tensão e corrente da fase a, falta trifásica em 1,15 segundos. Fonte: Autoria Própria

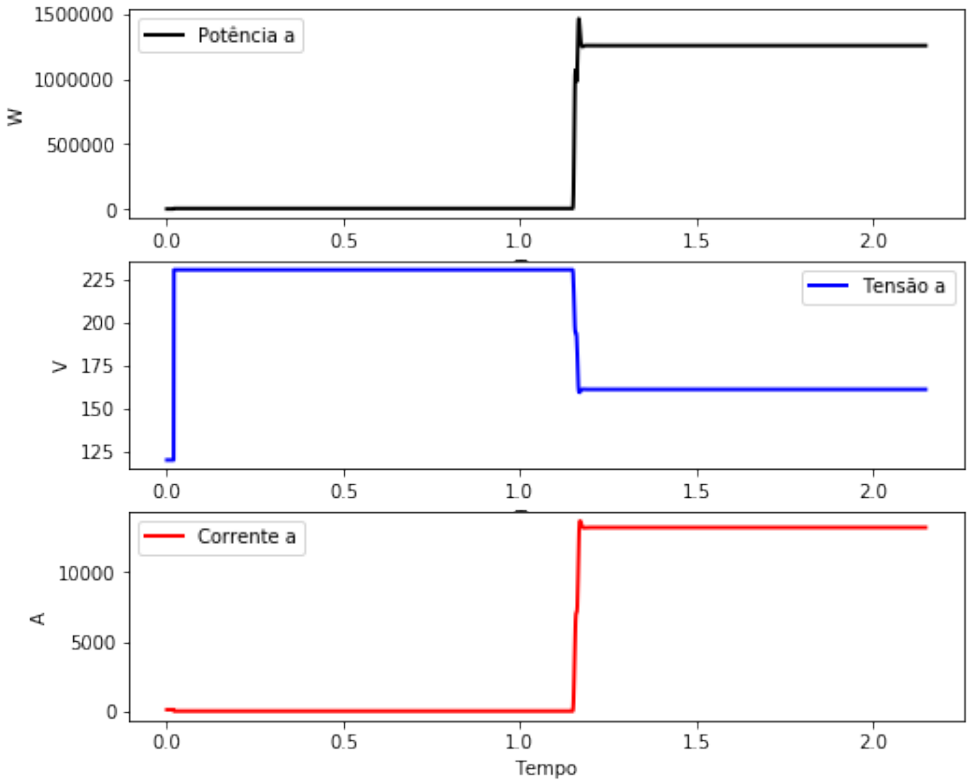


Tabela 5 – Hiper-parâmetros da rede neural do primeiro experimento. Fonte: Autoria Própria

Taxa de aprendizagem	0.0006
Ativação	relu
Camadas internas	3
Épocas	20
Neurônios	60

É possível observar na tabela 6 que os resultados tanto de acurácia quanto de AUC foram todos bem altos. Porém observando a figura 37 é evidente que há uma diferença bem grande entre quando ocorre uma falta e o contrário. Por esta razão a rede teve um desempenho tão bom.

Em seguida serão feitas simulação praticamente idênticas exceto pelo tipo de falta para verificar se a rede mantém os mesmos resultados.

5.2.0.1 Falta monofásica

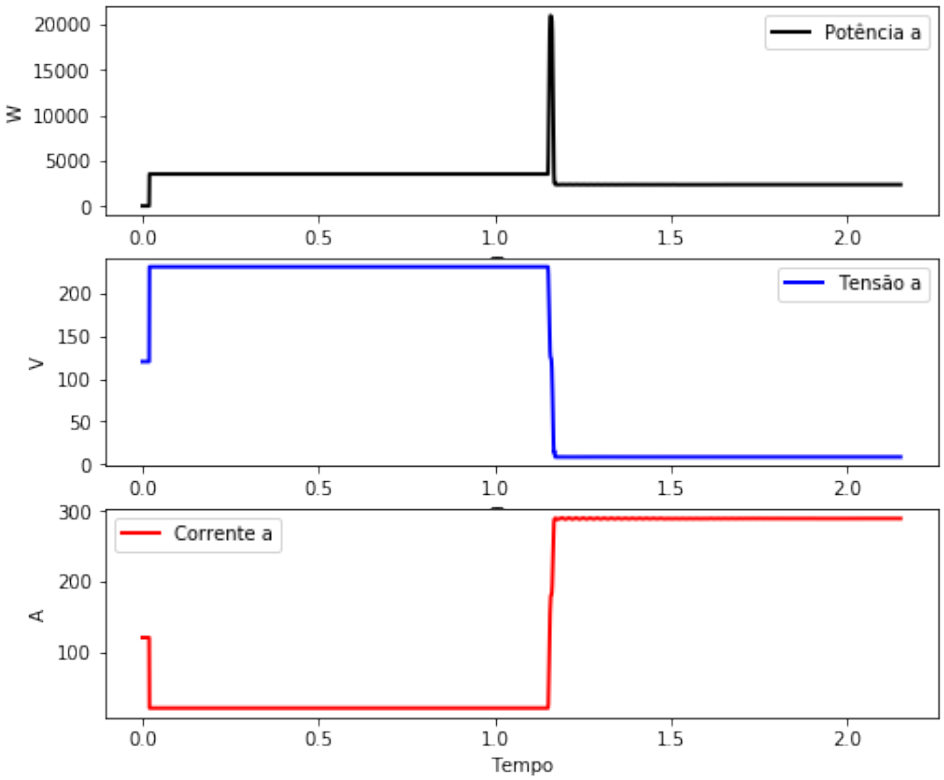
Nesta simulação a falta que ocorre é a monofásica na fase a. Não estão expostos aqui os resultados de outras fases pois os resultados de uma fase podem ser generalizados para as outras. Na figura 38 os sinais do sistema de uma falta mo-

Tabela 6 – Resultados do experimento 1 com falta trifásica. Fonte: Autoria Própria

Sinal	Acurácia	AUC
P_a	99,74%	99,75%
P_b	99,75%	99,73%
P_c	99,24%	99,21%
V_a	97,98%	99,92%
V_b	99,75%	99,75%
V_c	97,98%	97,92%
I_a	99,50%	99,48%
I_b	99,24%	99,22%
I_c	99,74%	99,75%

nofásica, nota-se que logo quando ocorre a falta o sinal de potência tem um rápido *overshoot* e depois se torna mais baixo do que em regime permanente.

Figura 38 – Sinais de potência, tensão e corrente da fase a, falta monofásica em 1,15 segundos. Fonte: Autoria Própria



Na tabela 7 observa-se que além da previsão de faltas na fase a ter tido um desempenho bom, nas fases b e c o mesmo ocorreu. Isso se deve ao fato de uma falta em uma fase ter gerado uma diferença entre o pré-falta e o pós-falta dos outros sinais das outras fases.

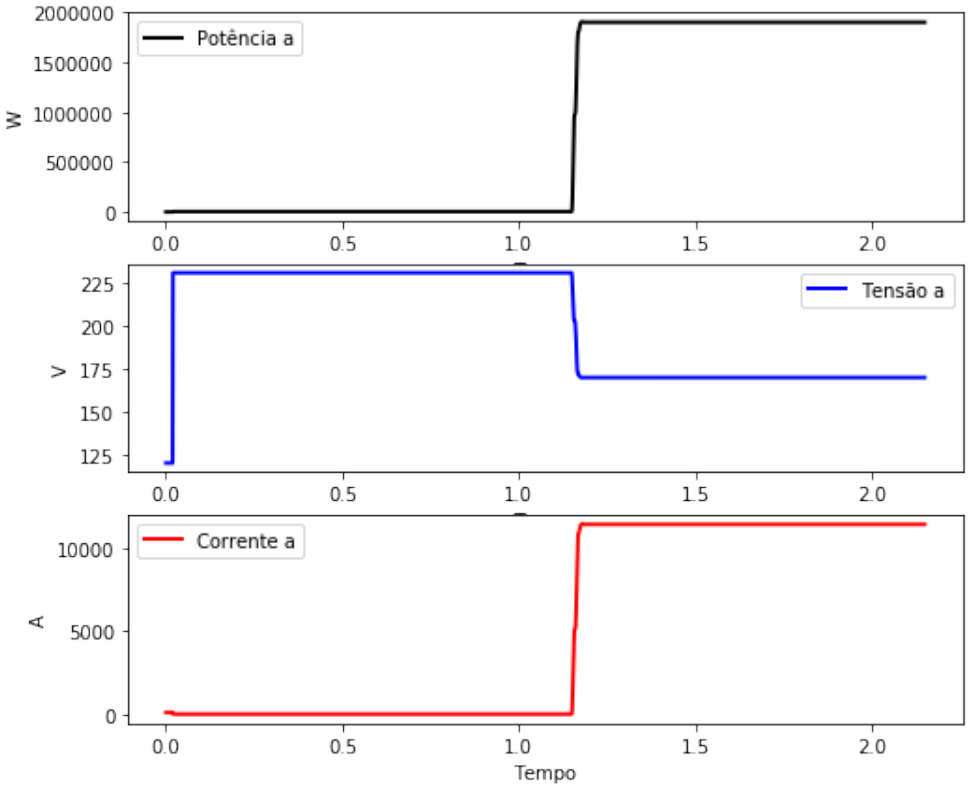
Tabela 7 – Resultados do experimento 1 com falta monofásica. Fonte: Autoria Própria

Sinal	Acurácia	AUC
P_a	99,5%	99,45%
P_b	100%	100%
P_c	99,50%	99,49%
V_a	99,49%	99,48%
V_b	99,49%	99,54%
V_c	99,5%	99,52%
I_a	99,74%	99,76%
I_b	98,74%	98,83%
I_c	99,74%	99,76%

5.2.0.2 Falta fase fase

Para a simulação de uma falta fase fase foram utilizadas as fases a e b para o curto-circuito de falta. Nas figuras 39 e 40 são mostrados os sinais de falta utilizados.

Figura 39 – Sinais de potência, tensão e corrente da fase a, falta bifásica em 1,15 segundos. Fonte: Autoria Própria



Na tabela 8 os resultados para a fase a e b como esperado atingiram valores altos de acurácia e AUC. Já para a fase c não, uma vez que houve apenas uma pequena diferença entre os sinais quando em falta e quando não.

Figura 40 – Sinais de potência, tensão e corrente da fase b, falta bifásica em 1,15 segundos. Fonte: Autoria Própria

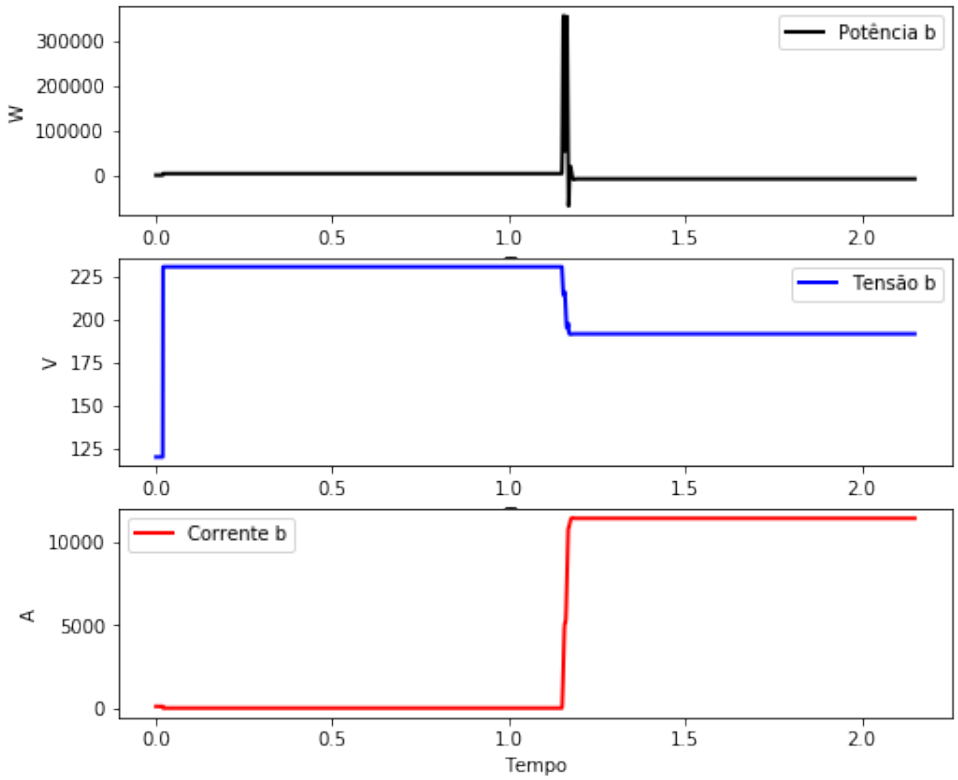


Tabela 8 – Resultados do experimento 1 com falta fase fase. Fonte: Autoria Própria

Sinal	Acurácia	AUC
P_a	100%	100%
P_b	97,73%	97,70%
P_c	47,10%	50,00%
V_a	99,74%	99,76%
V_b	100%	100%
V_c	46,34%	50,00%
I_a	100%	100%
I_b	98,74%	98,86%
I_c	49,87%	50,00%

5.2.0.3 fase fase terra

Nas figuras 41, 42 e 43 são mostrados os sinais de uma falta fase fase terra nos sinais das três fases.

Na tabela 9 observa-se que é possível detectar uma falta fase fase terra de todas as fases através de qualquer sinal da rede. Além disso nota-se que ao contrário dos resultados da tabela 8 aqui o classificador conseguiu generalizar para todas as fases, isto ocorre pois há uma diferença significativa entre o sinal pré-falta e pós-falta

Figura 41 – Sinais de potência, tensão e corrente da fase a, falta bifásica terra em 1,15 segundos. Fonte: Autoria Própria

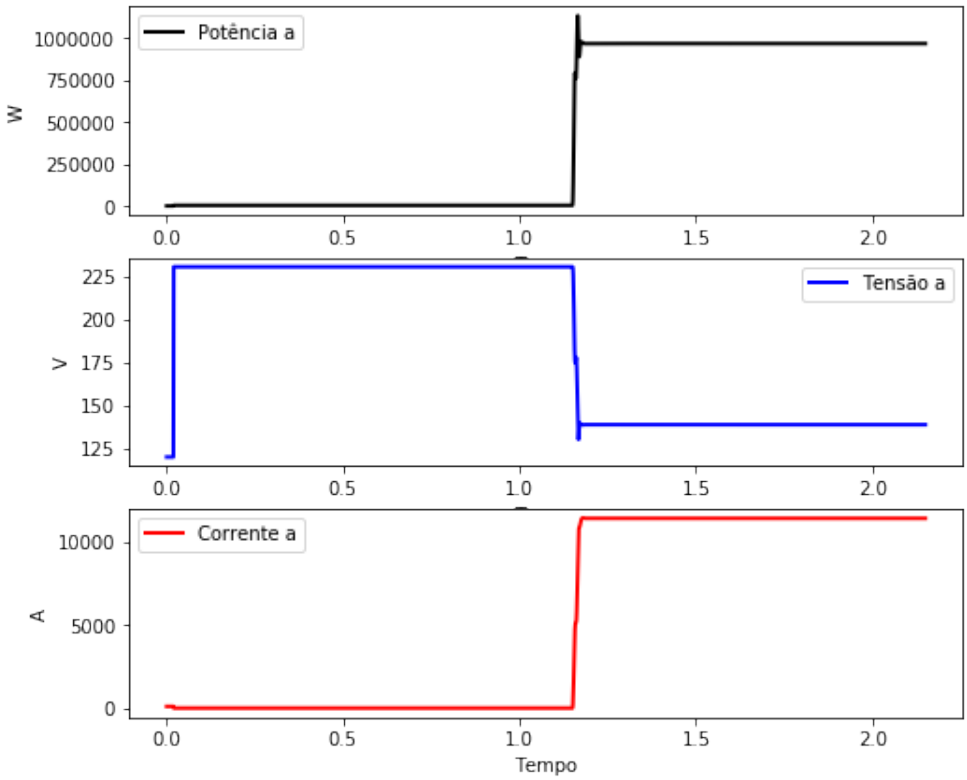


Tabela 9 – Resultados do experimento 1 com falta fase fase terra. Fonte: Autoria Própria

Sinal	Acurácia	AUC
P_a	100%	100%
P_b	100%	100%
P_c	100%	100%
V_a	99,75%	99,74%
V_b	99,24%	99,23%
V_c	99,24%	99,29%
I_a	99,74%	99,76%
I_b	99,74%	99,76%
I_c	98,74%	98,86%

nos sinais das outras fases.

5.2.1 Falta de qualquer tipo

Apesar dos resultados acima tiverem obtido um desempenho bem satisfatório eles são classificadores de apenas um tipo de falta. Nesta parte do trabalho o experimento será tentar criar um modelo que identifique qualquer tipo de falta.

Figura 42 – Sinais de potência, tensão e corrente da fase b, falta bifásica terra em 1,15 segundos. Fonte: Autoria Própria

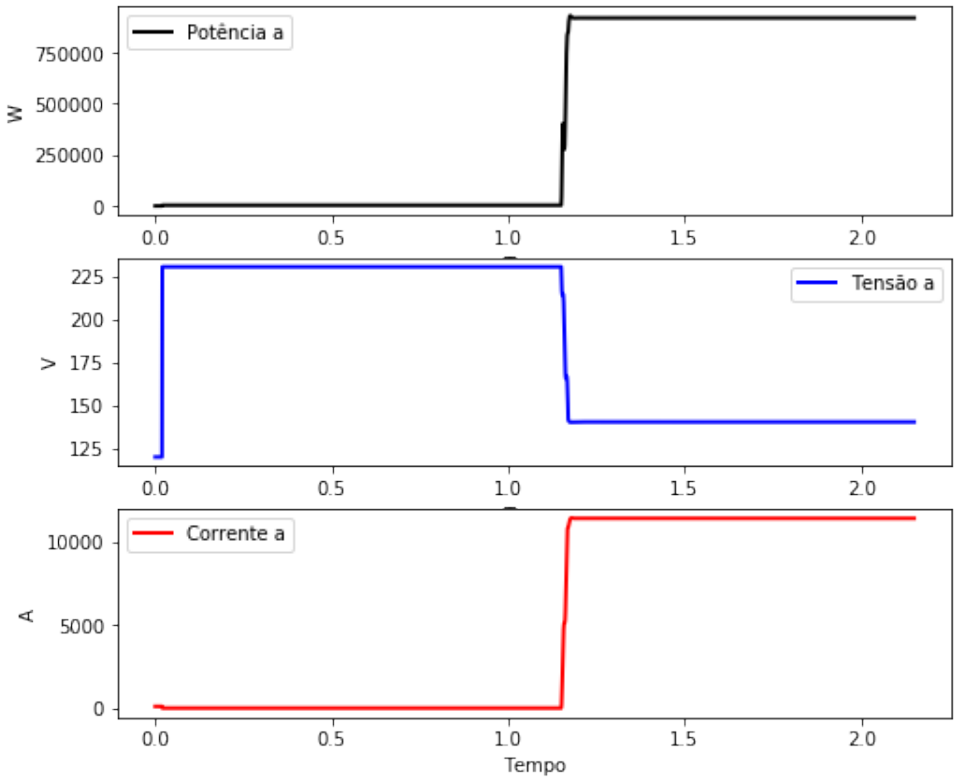


Tabela 10 – Dados do segundo experimento. Fonte: Autoria Própria

Duração da simulação (s)	Período de Falta (s)
10,15	5,15 - 10,15
tipo de falta	impedância de falta (Ω)
1ϕ - 2ϕ - 2ϕ terra - 3ϕ	0,001
Quantidade de amostras	Tamanho conjunto de treinamento
9985	5991
Tamanho do conjunto de validação	Tamanho do conjunto de teste
1997	1997

5.2.1.1 Detectar diversos tipos de falta com apenas um sinal

A tabela 10 mostra os dados utilizados nesta primeira parte do experimento, após 5,15s são simulados todos os 10 tipos de faltas uma em seguida da outra. Será treinada uma rede com apenas um sinal para detectar diversos tipos de falta, os sinais do sistema estão na figura 44. A rede manteve a arquitetura que está descrita pela tabela 5.

Através da tabela 11 verifica-se que os resultados foram bem mais inconstantes do que quando só se simulou uma falta. Para obter um resultado melhor serão utili-

Figura 43 – Sinais de potência, tensão e corrente da fase a, falta bifásica terra em 1,15 segundos. Fonte: Autoria Própria

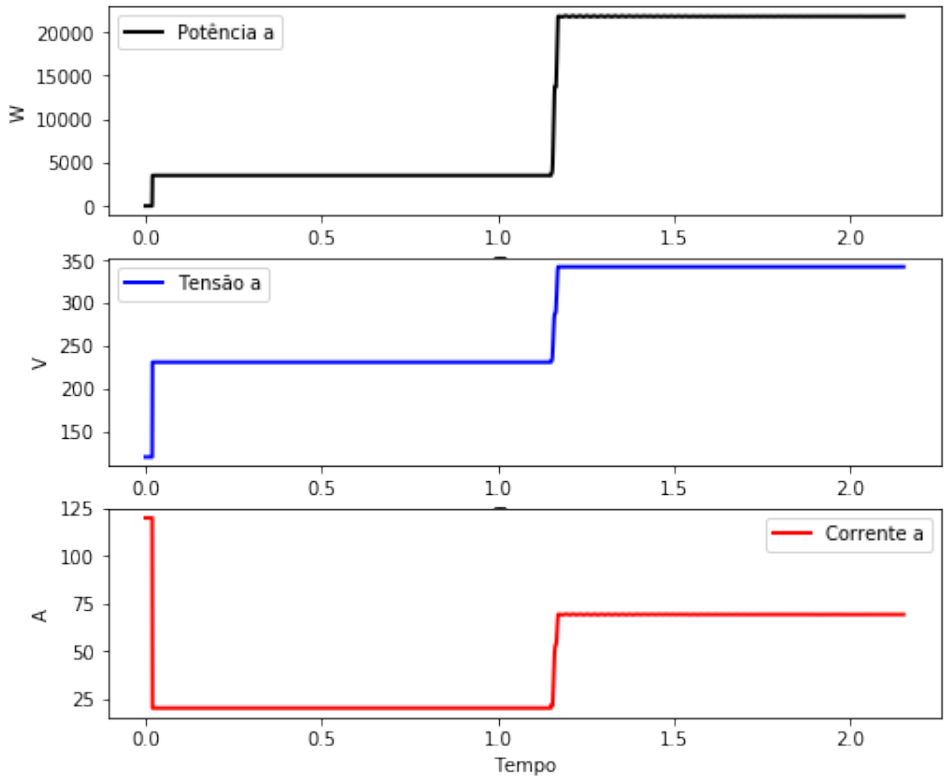


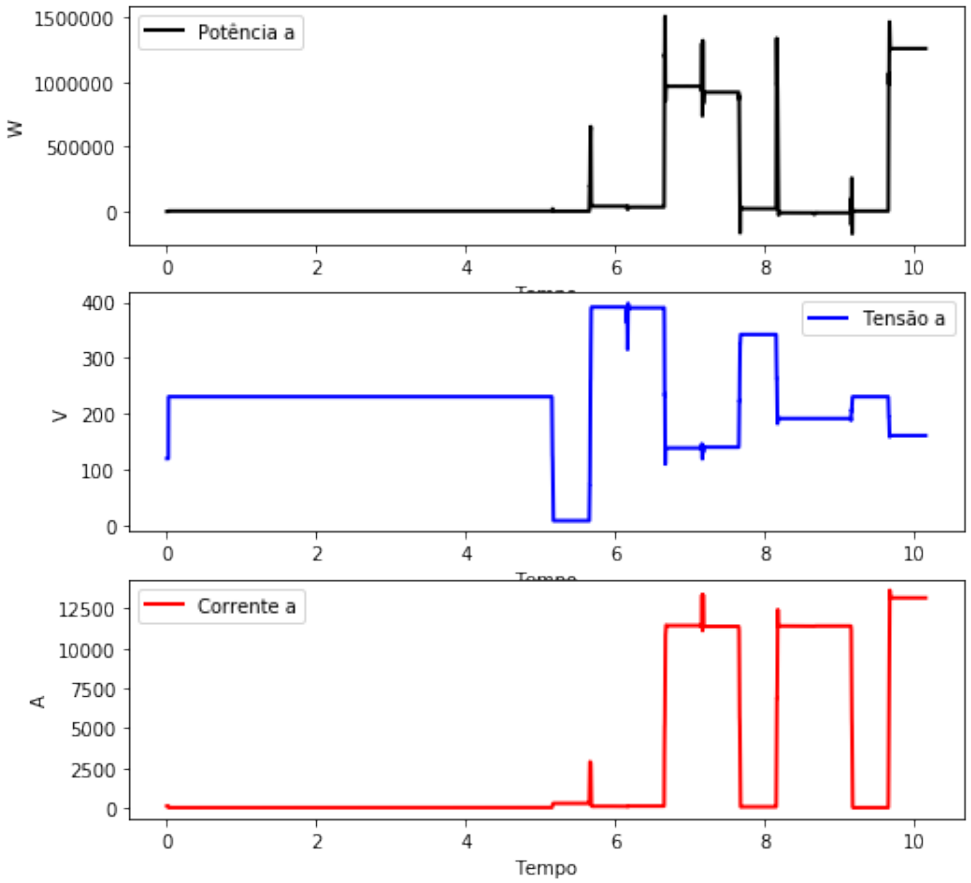
Tabela 11 – Resultados do experimento 2 usando apenas um sinal para identificar a falta. Fonte: Autoria Própria

Sinal	Acurácia	AUC
P_a	86,82%	87,56%
P_b	87,58%	87,75%
P_c	60,73%	60,78%
V_a	100%	100%
V_b	100%	100%
V_c	99,75%	99,75%
I_a	51,96%	50,00%
I_b	86,32%	87,01%
I_c	50,94%	50,00%

zados mais sinais do sistema como entrada de dados para o treinamento das redes neurais. Serão utilizados os sinais de potência, tensão eficaz e corrente eficaz de todas a fases.

O resultado obtido foi de 99,88% de acurácia e 99,87% de AUC para os sinais da figura 44. Em seguida o mesmo modelo sera utilizado para classificar 10 tipos de falta oriundas de sinais diferentes dos utilizados para treinamento e teste que ocorrem nos intervalos entre 1,15 e 2,15 segundos.

Figura 44 – Sinais de potência, tensão e corrente da fase a, faltas iniciam em 5,15 segundos. Sinal usado para o experimento de detecção de diversos tipos de falta. Fonte: Autoria Própria



A partir da tabela 12 é possível observar que o modelo consegue generalizar suas previsões para outros sinais com configurações diferentes de faltas.

5.2.2 Faltas com diferentes impedâncias de falta

Uma outro parâmetro que pode ser alterado nas faltas a fim de verificar se a rede neural é capaz de generalizar mais é variar as impedâncias de falta. Foram treinadas algumas redes neurais com diferentes intervalos com os 10 tipos diferentes de faltas e com cada uma dessas faltas possuindo valores diferentes de impedância.

Para este teste foi feita uma simulação 10 segundos aonde as faltas passaram a ocorrer a partir dos 5 segundos.

Na tabela 13 são mostrados os resultados obtidos deste experimento. As impedâncias foram variadas de forma linear, isto é de um intervalo de 1 a 100 as impedâncias foram sendo variadas de 10 em 10. A partir dos resultados conclui-se que quanto maior o intervalo das impedâncias pior foi o desempenho do modelo.

Tabela 12 – Resultados utilizando a mesmo modelo treinado usando os sinais da figura 44 para identificar se houve falta ou não usando os dez sinais diferentes listados abaixo experimento(5.1.2.1). Fonte: Autoria Própria

Falta	fase(s)	Acurácia	AUC
3ϕ	abc	100%	100%
2ϕ-T	ab	99,75%	99,75%
2ϕ-T	ac	99,49%	99,50%
2ϕ-T	bc	96,82%	87,56%
2ϕ	ab	98,74%	98,86%
2ϕ	ac	99,50%	99,50%
2ϕ	bc	99,73%	99,76%
1ϕ	a	94,54%	94,47%
1ϕ	b	95,19%	94,48%
1ϕ	c	99,75%	99,75%

Tabela 13 – Resultados do experimento 3 utilizando faltas com diferentes impedâncias. Fonte: Autoria Própria

Intervalo de impedância (Ω)	Acurácia	AUC
0.0001-100	94,42%	93,96%
0.0001-1	97,21%	96,58%
1-100	99,80%	99,80%
0.0001-10000	85,32%	83,98%
0.00001-100000	75,46%	75,35%

5.2.3 Detectar faltas em diferentes locais da linha

Como último parâmetro de falta para validar a capacidade de generalização do modelo de *machine learning* serão simuladas faltas em diferentes locais da linha. Como se pode observar na figura 27 a linha foi modelada com 3 blocos de impedâncias para simular a impedância ao longo de uma linha. Desta forma é possível ter quatro locais diferentes de falta.

Foram simuladas 4 configurações de 10 tipos diferentes de faltas com impedâncias variando de 0,001 a 100 Ω cada uma dessas configurações teve faltas em locais diferentes. Na figura 45 são mostrados alguns dos sinais de falta que foram simulados.

Com tabela 14 conclui-se que a rede neural é capaz de generalizar com relação ao tipo, impedância e local da falta ainda mantendo acurácia e AUC elevados Levando em consideração que essas métricas dizem respeito a janelas de 17 mili-segundos e com base nas evidências dos experimentos acima pode se afirmar que redes neurais são capazes de detectar faltas em simulações de um sistema elétrico.

Figura 45 – Sinais de potência tensão e corrente falta com variação em local, impedância e tipo de falta. Fonte: Autoria Própria

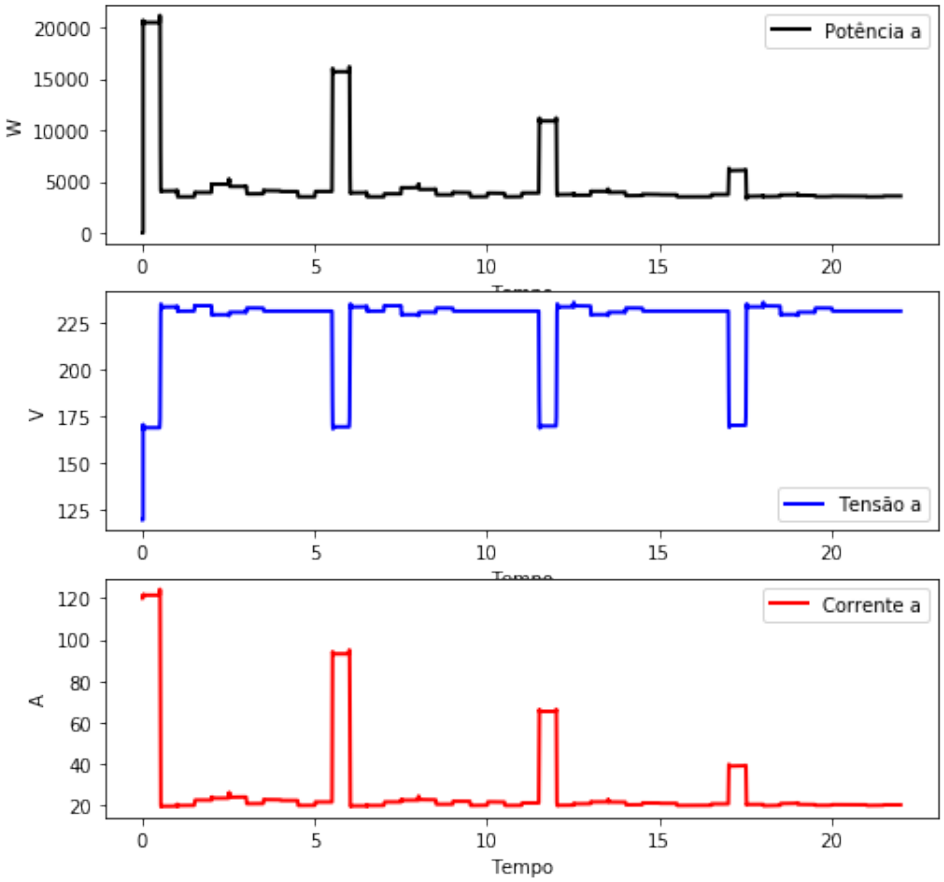


Tabela 14 – Resultados do experimento 2, variação do local da falta, usando sinais diversos de falta. As configurações que vão de um à 4 são os quatro lugares possíveis de simular uma falta dentro da linha simulada no simulink. Através da figura 27 é possível ver os quatro possíveis locais de falta, as configurações foram numeradas da esquerda para a direita. 1 seria o local mais à esquerda e 4 o local mais à direita Fonte: Autoria Própria

Configuração	Acurácia	AUC
1	99,80%	99,79%
2	90,23%	89,62%
3	99,57%	99,60%
4	84,02%	84,29%
média	93,40%	93,33%

5.3 LOCALIZAÇÃO DE UMA FALTA

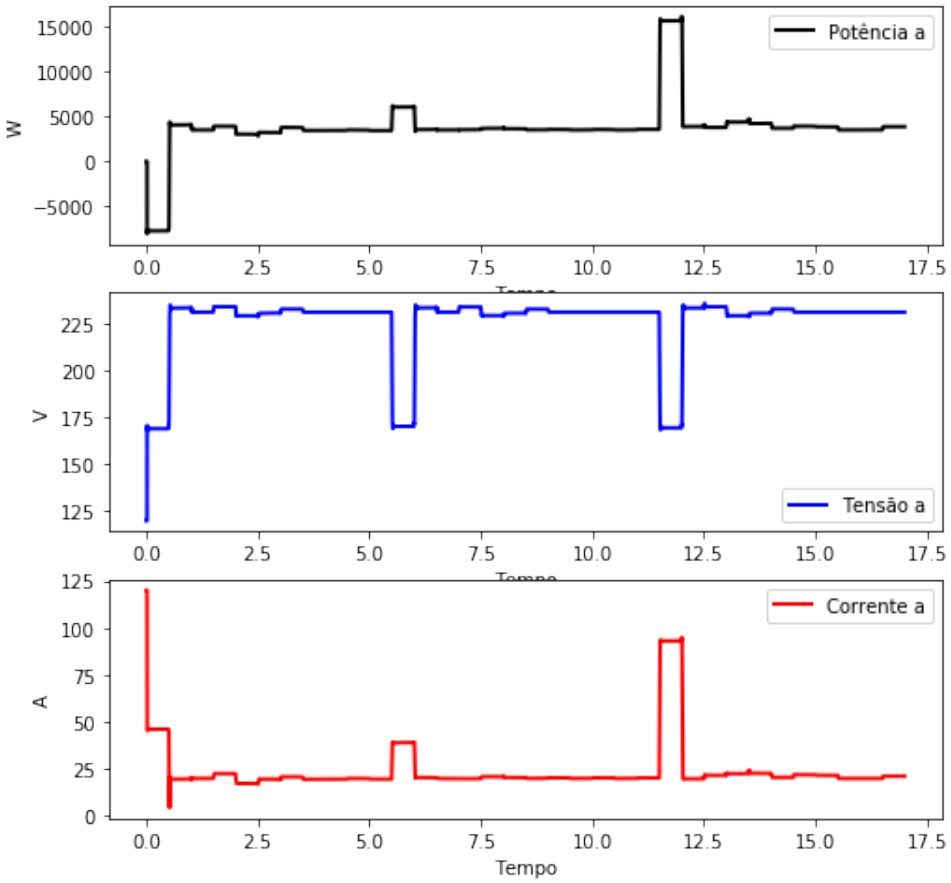
Para a tarefa de localização de falta no sistema elétrico usado neste trabalho, foi definida como linha de transmissão os elementos da simulação que se encontravam entre os dois sensores. Neste sistema utilizado eram as linhas e as impedâncias das linhas, ver figura 27. Todas as faltas que ocorressem fora desta área entre os sensores

não seriam considerados uma falta na linha.

Pensando em um sistema real, se fosse detectada uma falta no sistema, cada par de sensores entre as linhas de transmissão iria fazer essa verificação. Portanto para treinar o modelo os sinais necessários devem conter somente faltas. Além disso os sinais utilizados foram o dos dois sensores dando um total de 18 sinais pois são três fases e três grandezas. Alguns dos gráficos dos sinais medidos pelos sensores da esquerda e direita são mostrados nas figuras 46 e 47 onde as faltas dentro fora da linha ocorrem antes de 11,5 segundos e as faltas dentro ocorrem depois.

As simulações foram feitas de acordo com a tabela 15. Foram simuladas faltas nos dois locais fora da linha e sempre em um dos 4 locais dentro da linha, através da figura 27 é possível identificar estes locais dentro do modelo de simulação do simulink.

Figura 46 – Sinais de potência, tensão e corrente da simulação de localização de falta medidos pelo sensor a esquerda da linha, a falta dentro da linha ocorre a partir de 11,5 segundos. Fonte: Autoria Própria



Na tabela 16 são mostrados os resultados nas quatro localizações dentro da linha, os valores de impedância foram gerados aleatoriamente entre 1 e 100Ω. Analisando os valores de acurácia e AUC, que denotam a precisão do classificador em identificar faltas que ocorrem dentro e fora da linha, verifica-se que é possível localizar

Figura 47 – Sinais de potência, tensão e corrente da simulação de localização de falta medidos pelo sensor a direita da linha, a falta dentro da linha ocorre a partir de 11,5 segundos. Fonte: Autoria Própria

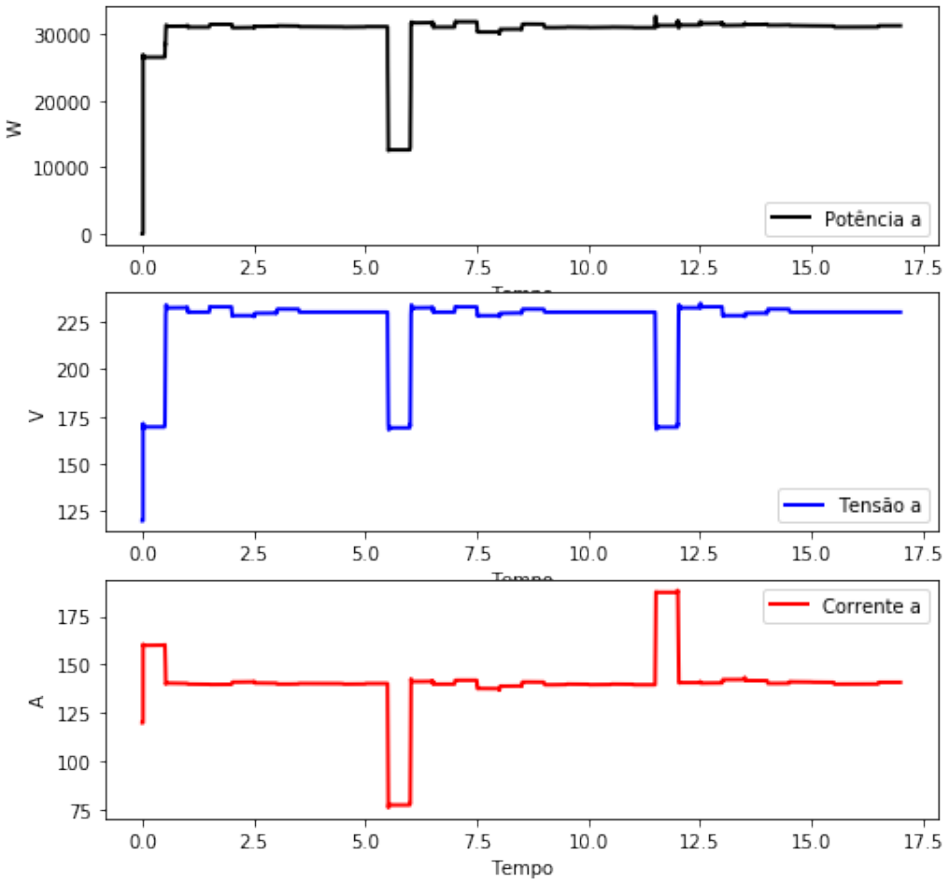


Tabela 15 – Dados da simulação de localização de faltas. Fonte: Autoria Própria

Duração da simulação (s)	Período de Falta dentro da linha (s)
17,5	11,5 - 17,5
tipos de falta	Intervalo impedância de falta (Ω)
todas	1 - 100
Quantidade de amostras	Tamanho conjunto de treinamento
16985	10191
Tamanho do conjunto de validação	Tamanho do conjunto de teste
3397	3397

uma falta através da distinção entre as faltas que ocorrem dentro e fora da linha de transmissão.

5.4 CLASSIFICAÇÃO DE FALTAS

Sabendo que houve uma falta e em qual linha que ela ocorre o que resta saber é qual o tipo da falta. Nesta seção serão mostrados os resultados das tentativas de

Tabela 16 – Resultados do experimento de localização de faltas usando sinais diversos de falta. Fonte: Autoria Própria

Configuração	Acurácia	AUC
1	97,44%	96,03%
2	96,94%	95,18%
3	91,23%	86,61%
4	96,70%	94,70%
média	95,56%	93,13%

Tabela 17 – Hiper-parâmetros da rede utilizada para a classificação de faltas multi-classe. Fonte: Autoria Própria

Taxa de aprendizagem	0.03
Ativação	sigmoide
Camadas Internas	4
Épocas	60
Neurônios	200
Função Custo	mse

Tabela 18 – Resultados da identificação de apenas 4 tipos diferentes de falta envolvendo a fase a e com variação de impedância. Fonte: Autoria Própria

Intervalo de impedância de falta	Acurácia	AUC
0,001 - 1	98,87%	99,22%
1 - 100	96,70%	94,70%
0,001 - 100	92,26%	91,13%

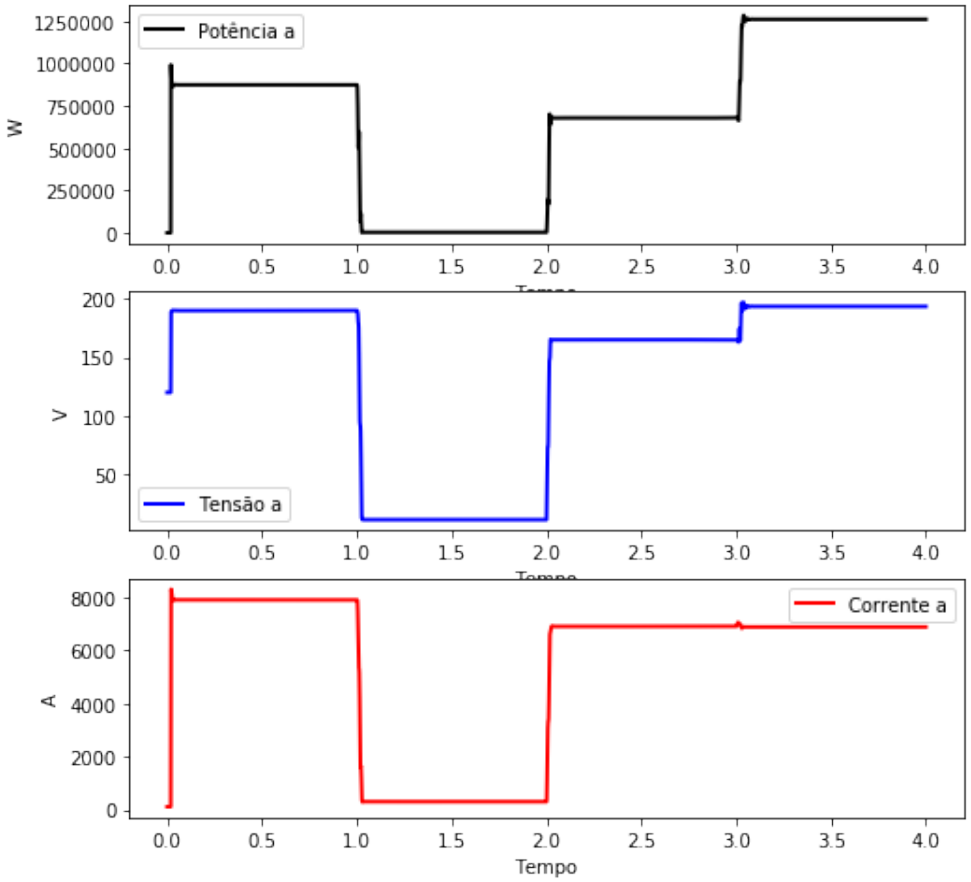
identificação de faltas. As simulações pressupõem que as faltas já ocorreram e foram separadas do sinal aonde não ocorre falta.

Para esta tarefa de classificação multi-classe durante os experimentos foi percebido que a rede teve melhor desempenho modificando alguns de seus hiper-parâmetros conforme mostra a tabela 17 a função sigmoide teve melhor desempenho do que as funções tanh e relu, além disso a função custo foi modificada para a função mse descrita na equação (13). Outro resultado curioso foi que ao contrário de todos os experimentos passados ao fazer a normalização das variáveis os resultados tiveram uma piora drástica nas métricas de AUC e acurácia.

5.4.1 Identificação de 4 faltas

Como objetivo preliminar de classificar qual é a falta que está ocorrendo na linha de transmissão, foi feita a tentativa de identificar 4 tipos diferentes de faltas, todas as faltas sempre envolvendo a fase a. Na figura 48 é mostrada a sequência de faltas simuladas no sistema.

Figura 48 – Sinais de potência, corrente e tensão da fase a de falta trifásica, monofásica, bifásica terra e bifásica em sequência. Fonte: Autoria Própria



A tabela 18 mostra os resultados para a mesma sequência de faltas dos sinais da figura 48 com uma variação das impedâncias de falta de cada falta sendo escolhidas aleatoriamente entre os intervalos da tabela 18. A tabela 48 mostra ser possível criar um algoritmo que diferencie as 4 faltas diferentes com alto grau de precisão e generalização em relação à impedância e o tipo de falta.

5.4.2 Identificação de 10 tipos de falta

Nesta subseção foram utilizados todos os dez tipos faltas diferentes porém eles foram agrupados em quatro classes que só diferenciam o tipo de falta e não o tipo de falta e as fases envolvidas. Como exemplo as faltas monofásicas das fases a, b e c foram incluídas na mesma classe de falta monofásica. Na figura 49 está ilustrado o sinal com 10 tipos de falta usado para a classificação de 4 faltas diferentes.

Pelos resultados na tabela 19 é possível observar que com uma impedância fixa de 0,001Ω o modelo teve alto AUC e acurácia. Já generalizando um pouco mais, com cada falta tendo impedâncias diferentes dentro dos intervalos de impedância entre 0,001 - 1 e 0,001 e 100 Ω o modelo teve uma piora mas mesmo assim tem acurácia e

Figura 49 – Sinais de potência, tensão e corrente da fase a. Sequência das faltas: 0 à 1,5 segundos faltas monofásicas das 3 fases, de 1,5 a 3 segundos faltas fase-fase-terra, 3 a 4,5 segundos faltas fase-fase e de 4,5 a 6 segundos falta trifásica. Fonte: Autoria Própria

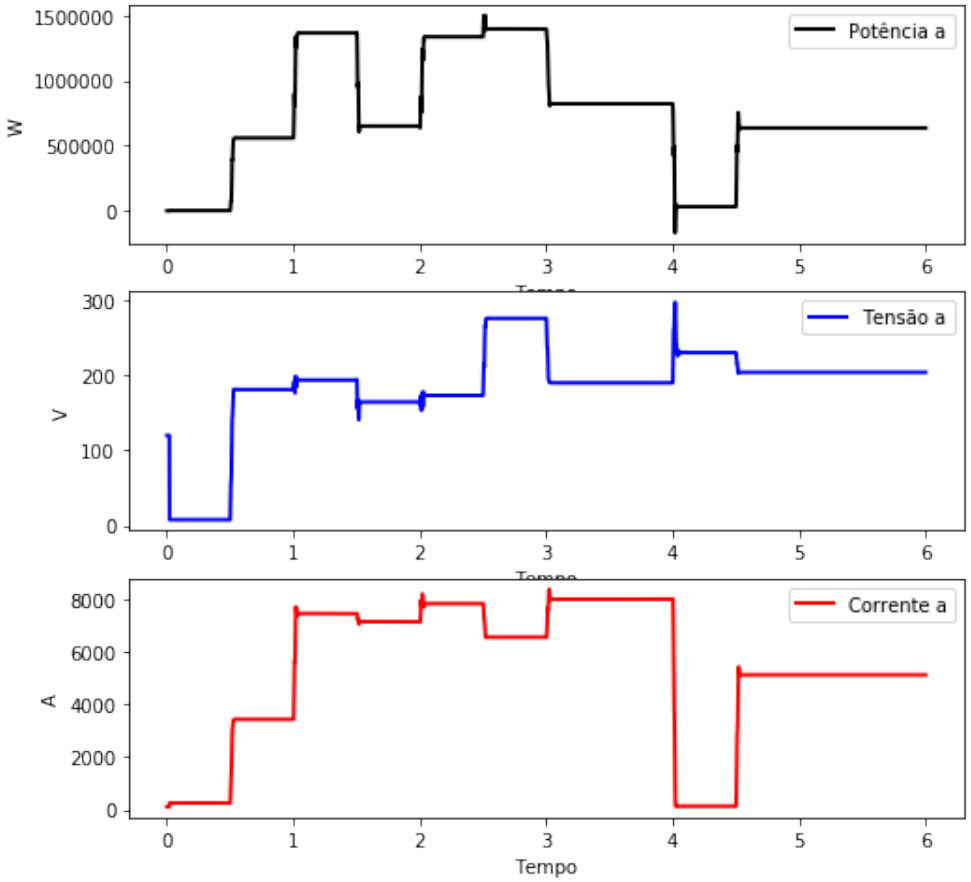


Tabela 19 – Resultados na identificação de todos os tipos de faltas em 4 classes. Fonte: Autoria Própria

Intervalo de impedância de falta	Acurácia	AUC
0,001	91,40%	94,35%
0,001 - 1	76,02%	83,60%
0,001-100	72,84%	83,07%
média	80,09%	87,01%

AUC altos levando em consideração que são 4 classes.

5.4.3 Identificar qual é o tipo e as fases da falta, 10 classes

Na tentativa de tentar identificar não somente o tipo de falta mas também as fases que esta falta envolve, o mesmo sinal da figura 49 foi utilizado só que ao invés de apenas 4 classes foram utilizadas 10 classes.

Apesar de terem sido tentadas diversas configurações de hiper-parâmetros a

Tabela 20 – Resultados na identificação de todos os tipos de faltas em 10 classes. Fonte: Autoria Própria

Intervalo de impedância de falta	Acurácia	AUC
0,001	28,69%	77,48%

melhor configuração foi aquela da tabela 17 e os resultados obtidos estão expressos na tabela 20. Uma acurácia de 28,69% pode parecer baixa porém no caso de 10 classes, se fosse um classificador totalmente aleatório a acurácia seria de 10% partindo do princípio que as classes são igualmente distribuídas. Porém, mesmo assim este resultado fica longe do necessário para um classificador confiável de fase e tipo de falta.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo desenvolver modelos de redes neurais artificiais capazes de detectar, localizar e identificar faltas em linhas de transmissão de energia elétrica criando um sistema inteligente para lidar com faltas. A primeira e segunda parte foram resolvidas com bastante êxito. Já a questão da classificação dos sinais ainda requer um pouco mais de estudo.

Na primeira parte foi mostrado que é possível classificar quando ocorre uma falta no sistema elétrico com bastante precisão e generalizando para diversas situações diferentes. Na segunda parte foi adotada uma estratégia simples porém bastante eficaz para localizar falta. Com base em dois sensores descobrir se a falta ocorre entre os sensores. Com esta abordagem conseguiu-se uma acurácia considerável e também boa generalização. Finalmente, na terceira parte, abordou-se o problema de identificar qual era o tipo da falta que estava ocorrendo. Um resultado razoável foi obtido para a classificação apenas do tipo de falta porém os resultados para identificar o tipo e a fase da falta não atingiram uma acurácia alta.

Um ponto importante que este trabalho tentou levar em consideração neste documento foi a replicabilidade do modelo e dos resultados. Durante a pesquisa para este trabalho algumas referências não deixavam claro como foram obtidos os resultados e levaram em consideração poucas métricas.É possível encontrar trabalhos semelhantes a este com métricas com 100% de acurácia ou muito próximo sem entrar em detalhes das condições dos experimentos e também não usando outras métricas que dariam mais informações sobre os resultados atingidos.

Seguindo os procedimentos aqui desenvolvidos trabalhos futuros semelhantes ou mas avançados conseguem resultados parecidos com os aqui obtidos, e com isso possuem um ponto de partida para alcançar objetivos mais complexos.

6.1 TRABALHOS FUTUROS

Evidentemente o fato de este trabalho ter conseguido bons resultados de classificação não quer dizer que os modelos de *machine learning* desenvolvidos aqui possam ser usados para classificar faltas reais. É muito provável que mesmo ainda em termos de simulações muitas características de faltas e do sistema elétrico não tenham sido modeladas

Levando como base o tema deste trabalho e os resultados obtidos são sugeridos como possíveis melhorias e trabalhos futuros os seguintes pontos:

- Tentar fazer um modelo que identifique a fase e a falta ao mesmo tempo uma vez que este trabalho não conseguiu resultados suficientemente bons para este tipo de problema.
- Fazer um estudo aprofundado de outras técnicas de redes neurais recorrentes e utiliza-las, técnicas como LSTMs, GRU, camadas de atenção e redes hibridas de convolução e recorrentes.
- Utilizar os modelos desenvolvidos aqui em um sistema mais complexo, com mais linhas de transmissão e outros elementos de sistemas de potência. A partir de um sistema grande avaliar se a técnica utilizada para a localização de falta funciona e criar um programa que devolve a localização exata de qual linha que ocorreu a falta.
- Utilizar outros sinais para simulações como os sinais de tensão e a corrente senoidais, fazer um processamento nos dados a fim de testar se os resultados melhoram.
- Simular com ruído e harmônicas os sinais da rede para testar se os modelos implementados continuam generalizando bem.

REFERÊNCIAS

ABADI, Martın *et al.* Tensorflow: A system for large-scale machine learning. *In:* 12TH {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). [*S.l.*: *s.n.*], 2016. p. 265–283.

BAQUI, Ibrahem *et al.* High impedance fault detection methodology using wavelet transform and artificial neural networks. **Electric Power Systems Research**, Elsevier, v. 81, n. 7, p. 1325–1333, 2011.

BENGIO, Yoshua. Practical recommendations for gradient-based training of deep architectures. **CoRR**, abs/1206.5533, 2012. arXiv: 1206.5533. Disponível em: http://arxiv.org/abs/1206.5533.

CARDOSO JR, Ghendy; ROLIM, Jacqueline G; ZÜRN, Hans Helmut. Diagnóstico de faltas em sistemas de potência: definição do problema e abordagens via inteligência artificial. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, SciELO Brasil, v. 15, n. 2, p. 215–229, 2004.

CHANG, Hyeong Soo *et al.* Google Deep Mind's AlphaGo. **OR/MS Today**, v. 43, n. 5, p. 24–29, 2016.

CHOLLET, François *et al.* **Keras**. [*S.l.*: *s.n.*], 2015.

CYBENKO, George. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems (MCSS)**, Springer, v. 5, n. 4, p. 455–455, 1992.

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems**. [*S.l.*]: "O'Reilly Media, Inc.", 2017.

GHADERI, Amin; GINN III, Herbert L; MOHAMMADPOUR, Hossein Ali. High impedance fault detection: A review. **Electric Power Systems Research**, Elsevier, v. 143, p. 376–388, 2017.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. [*S.l.*]: MIT press, 2016.

HAYKIN, Simon. **Neural Networks and Learning Machines, 3/E**. [*S.l.*]: Pearson Education India, 2010.

HUNTER, John D. Matplotlib: A 2D graphics environment. **Computing in science & engineering**, IEEE Computer Society, v. 9, n. 3, p. 90, 2007.

IDC. **Worldwide Spending on Artificial Intelligence Systems Will Grow to Nearly $35.8 Billion in 2019 According to New IDC Spending Guide**. [*S.l.*: *s.n.*], mar. 2019. https://www.idc.com/getdoc.jsp?containerId=prUS44911419. Acessado 05/11/2019.

JAMES, Gareth *et al.* **An introduction to statistical learning**. [*S.l.*]: Springer, 2013. v. 112.

JUSZCZAK, Piotr; TAX, D; DUIN, Robert PW. Feature scaling in support vector data description. *In:* CITESEER. PROC. ASCI. [*S.l.*: *s.n.*], 2002. p. 95–102.

KARLIK, Bekir; OLGAC, A Vehbi. Performance analysis of various activation functions in generalized MLP architectures of neural networks. **International Journal of Artificial Intelligence and Expert Systems**, v. 1, n. 4, p. 111–122, 2011.

KINDERMANN, Geraldo. **Curto-circuito, 2ª**. [*S.l.*: *s.n.*], 2010.

KLUYVER, Thomas *et al.* Jupyter Notebooks-a publishing format for reproducible computational workflows. *In:* ELPUB. [*S.l.*: *s.n.*], 2016. p. 87–90.

LECUN, Yann A *et al.* Efficient backprop. *In:* NEURAL networks: Tricks of the trade. [*S.l.*]: Springer, 2012. p. 9–48.

LIMA, Diomar Adonis Copetti. Localização de faltas em sistemas de transmissão de energia elétrica baseada na impedância aparente: algoritmo utilizando dados de um terminal, 2013.

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

MCKINNEY, Wes. pandas: a foundational Python library for data analysis and statistics. **Python for High Performance and Scientific Computing**, v. 14, 2011.

MITCHELL, Tom M. Does machine learning really work? **AI magazine**, v. 18, n. 3, p. 11–11, 1997.

MORETO, Miguel. Localizaçao de faltas de alta impedância em sistemas de distribuiçao de energia: Uma metodologia baseada em redes neurais artificiais, 2005.

NIELSEN, Michael A. **Neural networks and deep learning**. [*S.l.*]: Determination press San Francisco, CA, USA: 2015. v. 25.

OLESKOVICZ, Mário. **Aplicação de redes neurais artificiais na proteção de distância**. 2001. Tese (Doutorado) – Universidade de São Paulo.

OLIPHANT, Travis E. **A guide to NumPy**. [*S.l.*]: Trelgol Publishing USA, 2006. v. 1.

OLIVEIRA, AR de. Redes Neurais Artificiais aplicadas na detecção, classificação e localização de defeitos em linhas de transmissão. **UFJF. Juiz de Fora**, 2005.

PEDREGOSA, Fabian *et al.* Scikit-learn: Machine learning in Python. **Journal of machine learning research**, v. 12, Oct, p. 2825–2830, 2011.

PETITE, Fernanda Soares Vıtor; SANTOS, Ricardo Caneloi dos; ASANO, Patricia Teixeira Leite. APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS NA DETECÇÃO E LOCALIZAÇÃO DE FALTAS EM SISTEMAS DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA. **Revista Interdisciplinar De Pesquisa Em Engenharia**, v. 2, n. 10, p. 1–12, 2017.

PINTO, Milton Oliveira. Energia elétrica: geração, transmissão e sistemas interligados. **Rio de Janeiro: LTC**, 2014.

ROSENBLATT, Frank. 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, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. **Learning internal representations by error propagation**. [*S.l.*], 1985.

SALGADO, Roberto. **Apostila Introdução aos Sistemas de Energia Elétrica**. [*S.l.*: *s.n.*], 2018.

TRINDADE, F. **Desenvolvimento de metodologias para localização de defeitos em sistemas de distribuição com medidores inteligentes**. 2013. Tese (Doutorado) – Tese de doutorado, Unicamp/Campinas.

VAN ROSSUM, Guido *et al.* Python Programming Language. *In:* USENIX annual technical conference. [*S.l.*: *s.n.*], 2007. p. 36.

VASWANI, Ashish *et al.* Attention is all you need. *In:* ADVANCES in neural information processing systems. [*S.l.*: *s.n.*], 2017. p. 5998–6008.

WANG, Peter. **Popular Data Science Platform**. [*S.l.*: *s.n.*], 2012. Disponível em: https://anaconda.com/.

WU, Leon Li *et al.* Evaluating machine learning for improving power grid reliability, 2011.

ZANETTA JR, Luiz Cera. **Fundamentos de sistemas elétricos de potência**. [*S.l.*]: Editora Livraria da Fısica, 2006.

ZHANG, Yang; HUANG, Tao; BOMPARD, Ettore Francesco. Big data analytics in smart grids: a review. **Energy informatics**, Springer, v. 1, n. 1, p. 8, 2018.