CENTRO UNIVERSITÁRIO DO SUL DE MINAS
ENGENHARIA MECÂNICA
CLÁUDIO ESTEVAM LEITE DA SILVA

RECONHECIMENTO DE FALHAS POR IMAGENS: um estudo sobre o incremento de inteligência artificial no processo de detecção de não conformidades

Varginha
2021

CLÁUDIO ESTEVAM LEITE DA SILVA

RECONHECIMENTO DE FALHAS POR IMAGENS: um estudo sobre o incremento de inteligência artificial no processo de detecção de não conformidades

Projeto de pesquisa apresentado ao curso de Engenharia Mecânica do Centro Universitário do Sul de Minas UNIS MG, sob a orientação do Prof. Me. Alessandro Ferreira Alves.

Varginha

2021

CLÁUDIO ESTEVAM LEITE DA SILVA

RECONHECIMENTO DE FALHAS POR IMAGENS: um estudo sobre o incremento de inteligência artificial no processo de detecção de não conformidades

Trabalho de conclusao de curso apresntado ao curso de Engenharia Mecânica do Centro Universitário do Sul de Minas como pré-requisito para obtenção do grau de bacharel pela Banca Examinadora composta pelos membro:

Aprovado em / /

Profº

Profº

Profº

Obs:

RESUMO

Este trabalho teve como objetivo construir uma solução para ser implantada no processo de controle de qualidade com a finalidade de tornar os procedimentos de localização e classificação de falhas mais rápido e preciso. Sendo, para isso utilizado as redes neurais, que é um algoritmo capaz de inferir sobre um tipo de problema que os dados de entrada mais se aproximaram, além disso, é uma ferramenta que trabalha bem com um grande volume de dados, sendo que o trabalho com imagens o volume de informações é muito superior que um simples problema com entradas e saídas com poucas dezenas de entradas. Por fim o modelo construído se mostrou possível, sendo que atingiu uma precisão relativamente satisfatória para o objetivo inicial que era verificar se realmente tal aplicação seria possível, com isso se torna viável novos trabalhos sobre a possibilidade de melhoramento do algoritmo a fim de aumentar seus resultados e aplicar o algoritmo em situações reais.

Palavras-chave: Classificação de imagens, redes neurais no controle de qualidade, inteligência artificial na localização de falhas mecânicas.

ABSTRACT

This work aimed to build a solution to be implemented in the quality control process in order to make the procedures for locating and classifying faults faster and more accurate. For this, neural networks are used, which is an algorithm capable of inferring about a type of problem that the input data came closest to, in addition, it is a tool that works well with a large volume of data, and the work with images the volume of information is much greater than a simple problem with inputs and outputs with a few dozen inputs. Finally, the constructed model proved to be possible, and it reached a relatively satisfactory precision for the initial objective, which was to verify if such an application would really be possible. apply the algorithm in real situations.

Key words: *Image classification, neural networks in quality control, artificial intelligence in mechanical fault location*

SUMÁRIO

1. INTRODUÇÃO

2. REFERENCIAL TEÓRICO

2.1. Redes neurais artificiais

Redes neurais artificiais consistem em maneiras de encontrar soluções para inteligência artificial e resolução de problemas de modo a alcançar melhores resultados que simplesmente a criação de rotinas para determinado problema específico. A exemplo disso são os problemas que não possuem um comportamento linear, ou seja, não há uma maneira de esperar que uma determinada variável se comporte do mesmo modo crescente ou descente sempre a depender de correlações entre outras variáveis, pois a imprecisão desses modelos torna, em alguns casos, inviável a sua aplicação.

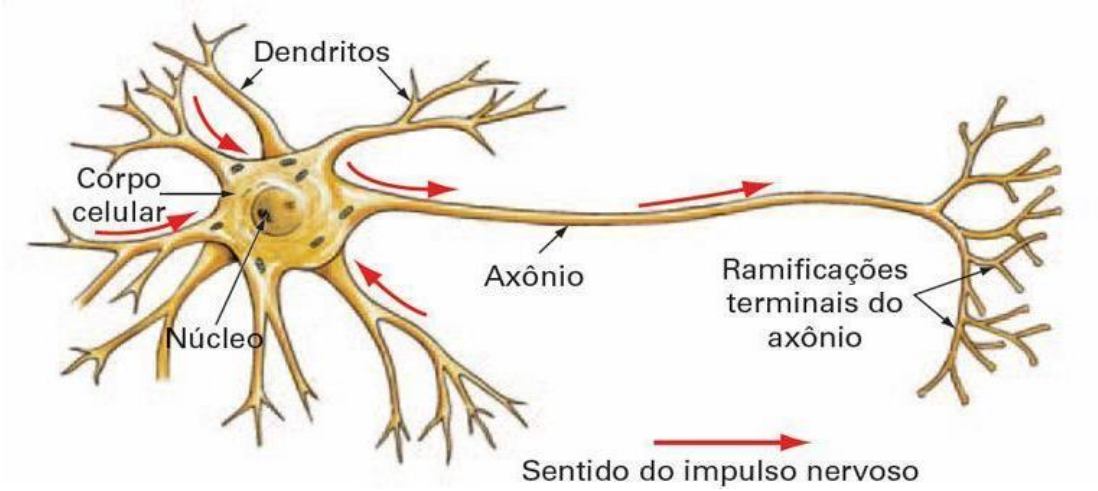
Para resolver problemas que possuem um comportamento incerto e não linear foram construindo algoritmos que sejam capazes de treinar e a partir desse treinamento se auto ajustarem e serem novamente treinados até que alcance um grau de precisão suficientemente satisfatório para cada problema (BARRETO, 2002).

Vale lembrar que em cada situação deve-se anteriormente analisar quais variáveis são importantes para a construção de uma rede e qual o grau de precisão desejado, sendo que 100% de acertos podem não significar algo perfeito, pois tem a possibilidade de "*overfitting*", ou sobreajuste, que significa que o modelo está perfeito para o conjunto de treinamento, porém não quer dizer que vá se performar tão bem para os novos dados que forem mensurados ao colocar o modelo na prática.

Dentro do desenvolvimento de uma rede neural artificial tem-se processos que são replicados de um processo de aprendizagem de uma rede neural biológica, que acontece dentro do cérebro humano. Para exemplificar isso tome como exemplo o processo de aprender que toda pessoa passa, quando criança, nunca tendo visto um avião ao vê-lo pode ter uma dúvida e perguntar a alguém o que seria aquele objeto que voa, ao ser obtida a informação que se trata de um avião essa informação é armazenada em seu cérebro, se em outra ocasião for avistado um avião, porém de pequeno porte talvez ainda seja a mesma dúvida do que seria aquilo, e novamente ao ser apresentada, essa nova informação é armazenada em seu cérebro, até que seu cérebro percebe alguns atributos que torne possível identificar que se trata de um avião, como exemplo é um objeto que voa, de duas asas e de grande porte, pronto a partir dessas informações sempre que for visto algo que se encaixe nas características ela irá classificá-lo como um avião.

Com isso, observa-se que o aprendizado humano passa por uma série de processos até que efetivamente aconteça. Na Figura 1 tem-se um neurônio biológico que é a unidade mais básica do cérebro humano que contém bilhões destes, e é responsável pela transmissão de informações. Inicialmente as informações vindas dos receptores, pode ser de qualquer um dos sentidos, é recebida pelos dendritos e transmitida ao corpo celular, no qual a informação é processada e são gerados saídas, que são novos impulsos, e passam para o axônio e dele para novos dendritos de neurônios seguintes. O ponto que conecta uma terminação axônica no dendrito de outro neurônio é chamado de sinapse, e por ela são unidos os neurônios e assim formam as redes neurais.

Figura 1: Neurônio biológico



Fonte: Data Science Academy

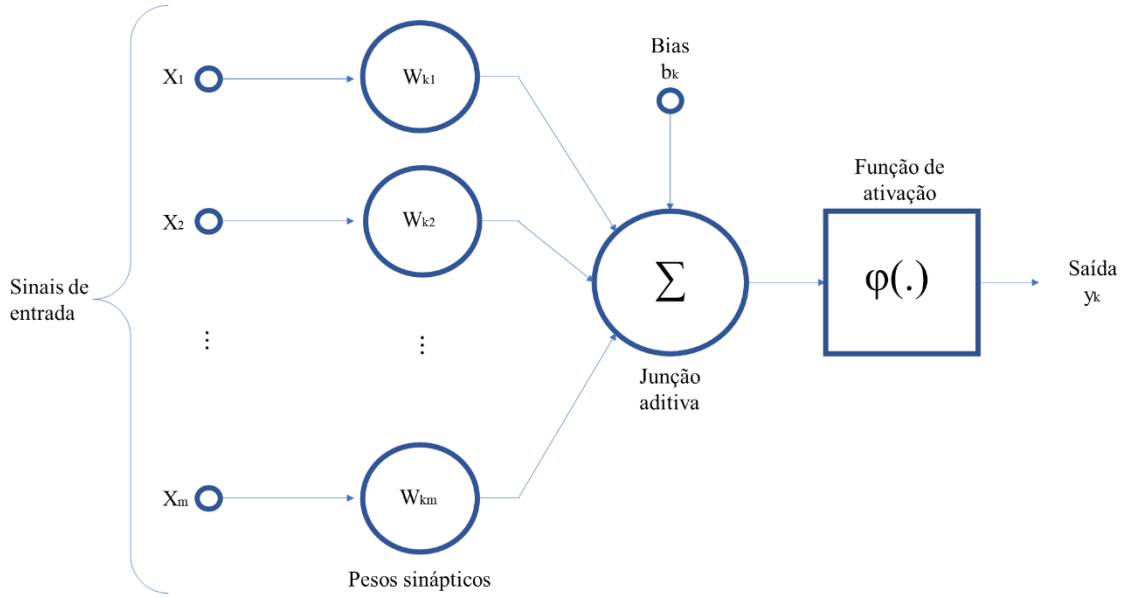
Semelhante ao exemplo dado, o algoritmo genético busca tornar possível que um programa de computador replique esse processo de aprendizagem para que o computador possa processar os dados que entram e a partir delas tirar conclusões e aprendem características e atributos dessas informações que possam ser usadas para que sejam usadas em novas observações que forem inseridas (HAYKIN, 2007).

2.1.1. Perceptron

O Perceptron é a unidade básica de uma rede neural artificial e tenta simular o processo realizado em um único neurônio como visto anteriormente. Para isso ele é constituído por quatro partes como vistas na Figura 2. A primeira etapa está relacionada ao que é apresentado ao modelo que são as entradas, a segunda contém os pesos que serão responsáveis por dar um direcionamento ao modelo, sendo estes multiplicados pela entrada, a terceira camada são os bias que serão somados ao resultado da etapa anterior sendo feito o somatório de todos os

resultados obtidos após a soma do bias e passado adiante para a quarta etapa em que o valor obtido é inserido em uma função de ativação e como resultado encontra-se a saída da rede que será o valor inferido sobre os dados que foram inserido (HAYKIN, 2007).

Figura 2: Representação do Perceptron



Fonte: Adaptado de HAYKIN, 2007

Ressalvando que para a camada de entrada pode conter um vetor com inúmeras entradas e saídas, contudo quanto maior for a quantidade desses parâmetros, maior será o consumo de processamento pelo computador e em alguns casos pode se tornar inviável dado a demora para treinar e testar os resultados.

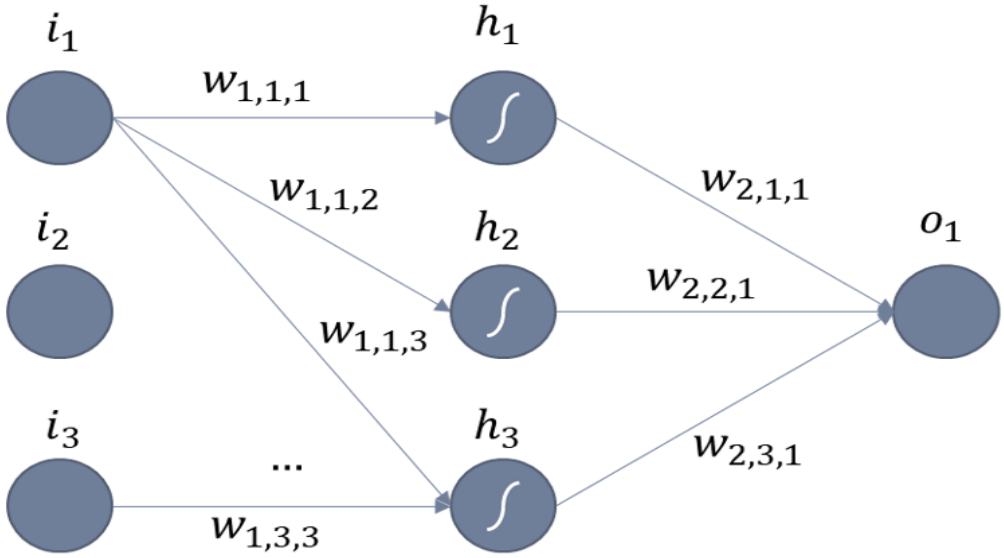
2.1.2. Rede multicamadas

Uma rede multicamadas seria o conglomerado de vários Perceptron juntos formando assim uma rede neural em si, de tal modo que as sinapses realizadas no cérebro serão aqui representadas por uma saída de um neurônio que é levada a outro, sendo um Perceptron direcionado ao próximo. Esse passo, repetido varias vezer, torna o processo de aprendizagem melhor, uma vez que os dados podem ser mais bem trabalhados e os parâmetros equilibrados.

Como pode ser observado na Figura 3, cada entrada é direcionada a cada função de ativação passando pelos pesos que corresponde à combinação feita unicamente entre uma entrada e uma camada escondida (h), com isso W1,1,1 representa o primeiro peso que liga a

primeira entrada na primeira camada oculta, W1,1,2 liga a primeira entrada na segunda camada oculta, e assim sucessivamente até que seja feito para todas as entradas (HAYKIN, 2007).

Figura 3: Representação rede neural



Fonte: Adaptado de Data Science Academy

Além disso, novas camadas ocultas podem ser adicionadas ao modelo, tornando ele mais preparado para lidar com situações adversas como dados muito fora do padrão, pois mais de uma camada torna a rede multicamadas faz com que o cada parâmetro inserido passe por uma série de pré-processamentos antes de chegar a uma resposta final, isso torna possível um melhor equilíbrio dos pesos, pois estes são iniciados aleatoriamente logo quanto maior a sua quantidade melhor iriam estar se ajustar.

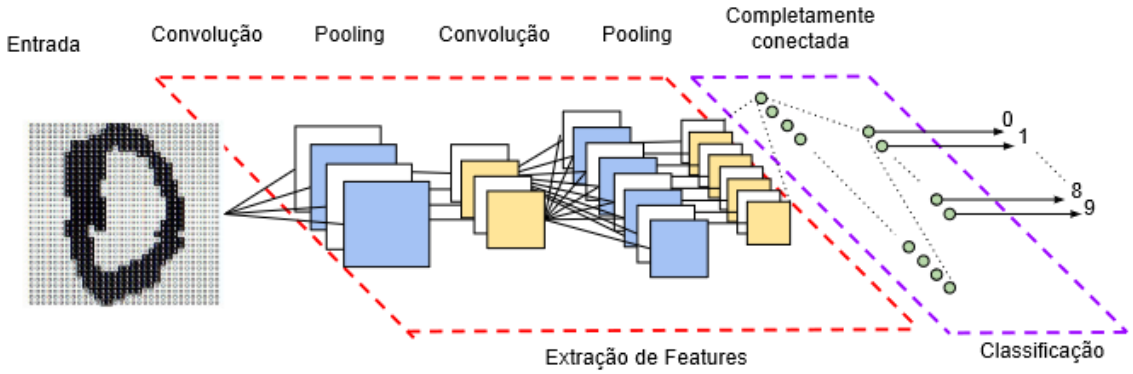
2.1.3. Rede Neural Convolucional

Uma *Convolutional Neural Network* ou rede neural convolucional (RNC) é uma evolução das redes Perceptron multicamadas, também chamadas de multi-Perceptrons, sendo a RNC adaptada para o processo de processamento de imagem, semelhante ao que acontece ao cérebro humano ao ver alguma coisa.

A Figura 4 ilustra como é o processamento de uma imagem ao ser inserido em uma rede e o que teoricamente acontece nas camadas ocultas, lembrando que uma rede convolucional é uma rede multicamadas seguida de outra multicamadas, essa organização de redes tem para

cada uma, um fim determinado, como exemplo uma rede pode ser responsável por filtrar a cor, outra por encontrar determinado padrão de pixels, e outra responsável por distinguir diferentes padrões nas imagens. Com esse processamento em série é possível destrinchar a imagem em várias partes até que se encontre um padrão que possa ser apreendido pelo sistema e com isso gerar uma resposta.

Figura 4: Representação de uma Rede Neural Convolucional



Fonte: VARGAS, 2016

Na Figura 4 a rede tinha o objetivo de identificar o número escrito a mão e posteriormente inserido no modelo, com isso a imagem foi tratada nas camadas ocultas e eram possíveis 10 saídas distintas para a qual rede estava sendo treinada (0, 1, 2, 3, ..., 9).

2.1.4. Aplicabilidade das redes neurais

Atualmente com o incremento tecnológico do avanço dos processadores é possível aplicar conceitos de redes neurais que antes existiam apenas de forma teórica por não se ter tal capacidade para testar suas aplicações. Agora, vê-se diferentes ramos dando saltos exponenciais na construção de ferramentas que auxiliam para realização de suas tarefas, sendo aplicados seus conceitos desde simular uma pintura como se tivesse sido feita por grandes pintores como Van Gogh, até diagnósticos de doenças a partir de imagens de radiografias.

Alguns estudos estão sendo aplicados de forma eficiente como o feito por Rosa, 2018 que conseguiu um classificar de 92,8% determinar pragas em plantações, o que estava causando prejuízos econômicos até o momento, porém com a rede neural foi possível tornar o processo de detecção de pragas não apenas automático como também mais rápido e eficaz.

Estudo como o de VOGADO, 2019 trouxe uma aplicação totalmente relevante, onde alcançou uma acurácia de 98,84% de precisão no diagnóstico de leucemia, um estudo que contou com uma base de imagem onde era apresentada algumas com a doença outras sem para treinar o modelo a classificação caso haja ou não essa confirmação.

Ainda dentro da área da saúde um relevante estudo foi feito por SILVA, 2018 que trouxe uma análise sobre a doença de Alzheimer, e treinou uma rede neural convolucional para que fosse possível realizar seu diagnóstico dados imagens do cérebro de possíveis pacientes, o modelo alcançou 84% de precisão o que para os autores na situação analisada foi considerado suficientemente satisfatório.

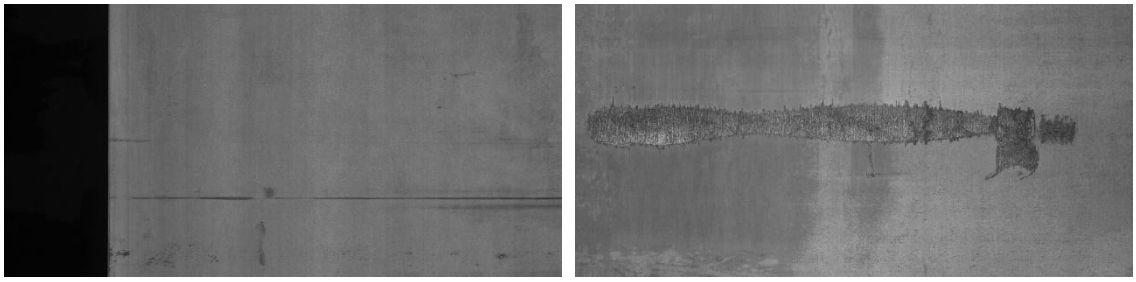
2.2. Defeitos considerados

Os dados que serão utilizados neste estudo são imagens de deformações encontradas em superfície metálica, sendo classificada cada imagem com seu respectivo problema, aqui será considerado dez tipos de não conformidade, que são vinco, lacuna crescente, inclusão, mancha de óleo, furo de perfuração, poço rolado, mancha de seda, cintura dobrada, mancha de água e linha de soldagem, cada um destes defeitos pode ser melhor visualizados nas imagens abaixo.

Este banco de dados conta com um total de 2306 imagens devidamente catalogadas. O conjunto de dados foi retirado do repositório Kaggle[1] o qual é uma subsidiária da Google voltada para compartilhamento de datasets com o propósito de aplicações para o aprendizado de máquina.

Figura 5: Apontamento de cada uma das falhas consideradas.

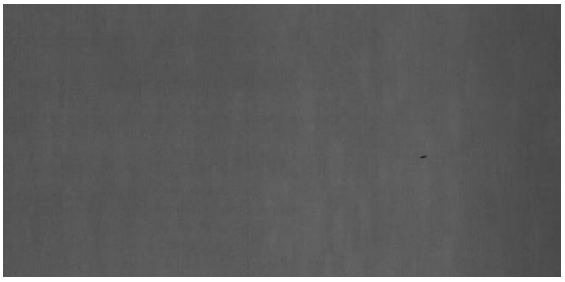
(a) Vinco (b) Poço rolado



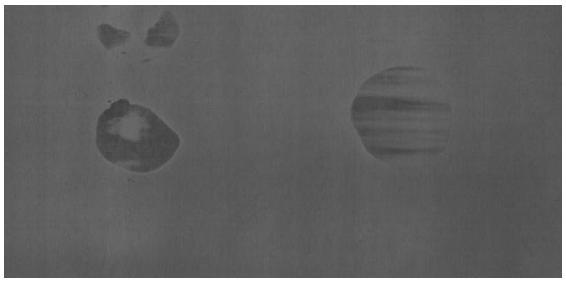
[1] Conjunto de dados disponível em: https://www.kaggle.com/zhangyunsheng/defects-class-and-location

(c) Lacuna crescente

(d) Inclusão

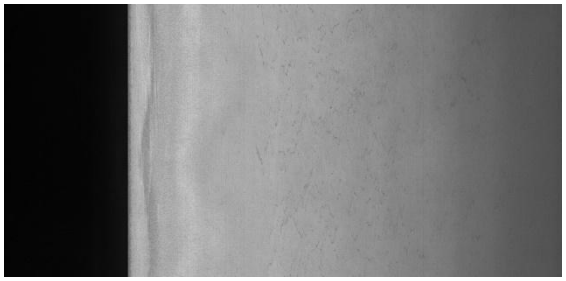


(e) Mancha de óleo



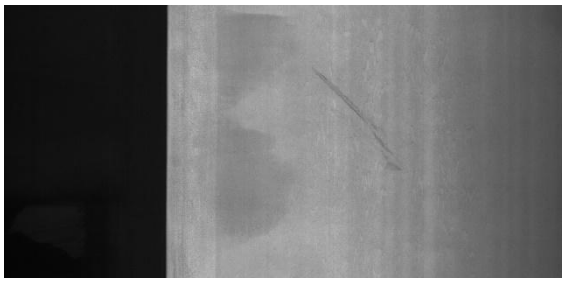
(f) Furo de perfuração

(g) Mancha de seda

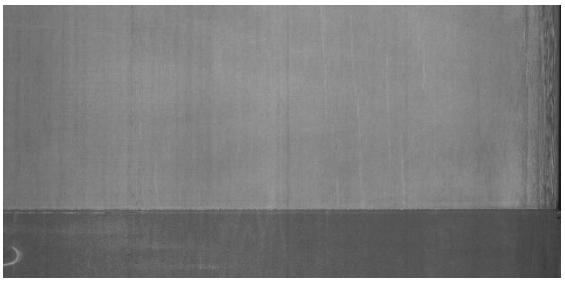


(h) Cintura dobrada

(i) Mancha de água



(j) Linha de soldagem



Fonte: Conjunto de dados Kaggle

3. METODOLOGIA

O projeto para a construção da inteligência artificial para detecção de falhas em superfícies metálicas será desenvolvido utilizando a linguagem de programação Python, e o conjunto de dados colhidos no repositório Kaggle. Tendo em vista que, por se tratar de imagem e com isso os dados de entradas serão os pixels das figuras logo será uma quantidade

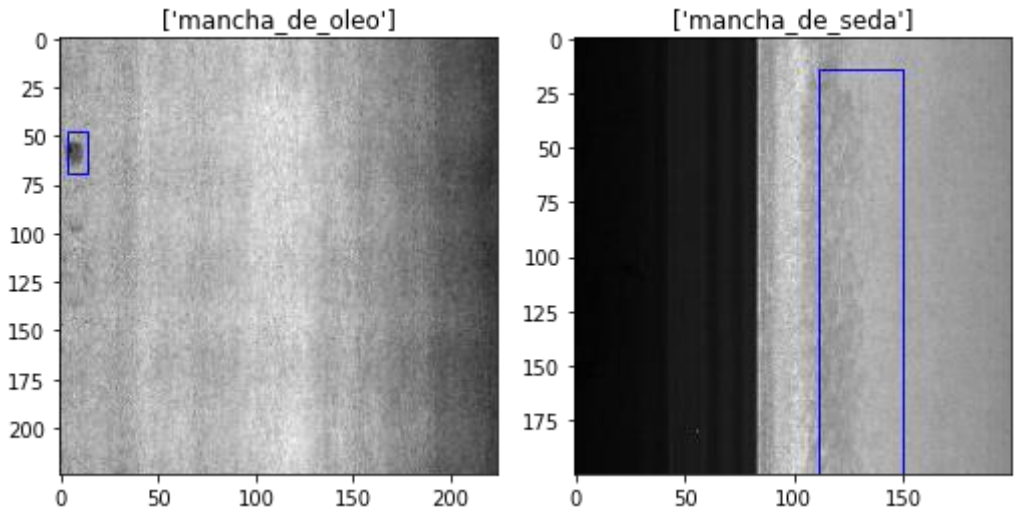
considerável de dados de entrada, tomando como estimativa a rede neural contará com pouco mais de 40 milhões de nós, o processamento demandado para executar o treinamento da rede será alto e não é viável sua realização em um hardware que não seja capaz de realizá-lo de forma eficiente

3.1. Construção do banco de dados

Para a construção do banco que será usado como base no modelo, inicialmente, foi preciso definir uma forma de determinar como se identificaria uma imagem que apresenta falha e uma que não, para isso foi definido que a melhor maneira em um primeiro momento seria dar um destaque para o trecho na imagem que traz tal problema, para isso foi utilizado um software de edição de imagem que foi possível desenhar um retângulo em torno da falha identificada pelo operador.

Como pode ser apresentado na Figura 6, o programa de edição exportou um arquivo com as coordenadas dos pontos x e y que representam a localização de cada vértice do quadrilátero que foi alocado destacando determinada falha. Este processo foi feito para todas as imagens e por fim teve duas pastas, uma com todas as imagens originais e outra com os arquivos no formato .xls que traz as coordenadas das falhas.

Figura 6: destaque para os defeitos apresentados



Fonte: o autor.

Para iniciar a construção do modelo, foi preciso dividir o conjunto de dados em dois grupos, o primeiro com 80% dos dados representa o conjunto de treino, responsável por servir como aprendizado ao modelo, o segundo conjunto foi o de teste que representa o restante dos dados, e foi utilizado no final para mensurar como o modelo se comporta quando é apresentado

dados que até então eram desconhecidos, então este modelo foi deixado imutável até a última etapa do desenvolvimento. Ressaltando que a divisão das amostras foi feita de forma aleatória, para que não seja favorecido nenhum tipo de viés.

Por outros experimentos anteriores a este em que foram atribuídos números para definir cada uma das características e estes modelos se mostraram eficiente, visto que quando se tem variáveis qualitativas, como é o caso da variável tipo de falha que foi usada aqui e se substitui por numerais a fim de podem mensurar sua precisão, a estratégia se mostra contra intuitiva, pois se colocar um valor para uma característica e/ou valor maior, o algoritmo começa a entender que há uma hierarquia entre os valores apresentados, onde isso não deveria ocorrer.

Um exemplo do conceito anterior é a variável sexo, em que tem duas possibilidades masculino e feminino, logo é uma variavel qualitativa nominal, pois é uma caracteristica e não pode ser colocada em ordem, agora se atribuir valor 0 (zero) para o atributo masculino e 1 (um) para o feminino e inserir essa variável como uma possível entrada em um modelo, pode ser que seja dado mais relevância aos pesos quando o indivíduo de entrada for mulher, pois neste exemplo seu valor foi maior que a outra opção.

Tendo o conceito anterior em mente optou-se pela sugestão apresentada por Géron (2019) que é codificar as variáveis pelo padrão "*OneHotEncoder"*, de forma simplificada o que essa codificação faz é definir uma matriz em que as colunas representam as variáveis, até então qualitativas, nas linhas são as peças, e no meio da matriz tem-se a definição de 0 (zero) para as peças que não possui a variável e 1 (um) para as que possui, pode-se observar como ficou neste caso na Tabela 1.

Essa nova codificação determina o mesmo valor a todas as características, todas valem um, o que muda é qual característica cada imagem apresentou. Assim, retira-se logo de início a possibilidade de enviesamento do modelo por uma característica ou outra.

Tabela 1 : banco de dados de entrada, abreviado, após realizar-se o "*OneHotEncoder"*.

	Cintura dobrada	Furo de perfuração	Inclusão	Lacuna crescente	Linha de soldagem	Mancha de água	Mancha de óleo
0	**1**	0	0	0	0	0	0
1	**1**	0	0	0	0	0	0
2	0	**1**	0	0	0	0	0

3	0	0	**1**	0	0	0	0
4	0	0	**1**	0	0	0	0
...	...	...	...	...	...	...	...
2275	0	0	**1**	0	0	0	0
2276	0	0	**1**	0	0	0	0
2277	0	0	**1**	0	0	0	0
2278	0	0	**1**	0	0	0	0
2279	**1**	0	0	0	0	0	0

Fonte: O autor

3.2. Definições no modelo

Feita as correções necessárias no banco de dados, ele está pronto para ser a entrada de um modelo, lembrando que, por se tratar de imagens o que é levado em consideração são os valores de cada pixel, sendo que cada pixel possui um intervalo de existência de 0 ate 255, com isso é possível saber, por meio de um conjunto de pixels específicos onde está uma possível falha.

Logo as imagens possuem inicialmente a resolução de 2048 x 100, porém esta é uma resolução considerada alta, e um modelo assim seria bem preciso, porém seu gasto computacional demandado seria relativamente maior, portanto foi reduzido a resolução das imagens para 200 x 200 pixels, a fim se se verificar a eficiência do modelo, e deixando em aberto a possibilidade de retorno para alterações neste parâmetro.

Com isso, construiu-se o modelo com os parâmetros apresentados no Quadro **X**, que de forma simples pode ser entendido como sendo a entrada os valores captados dos pixels das imagens e na saída cinco respostas, que são a localização e a classificação da falha. Entre a entrada e a saída do modelo que é feito todo o processamento com diversas camadas cada uma com sua função específica a fim de trazer as melhores respostas, sendo a configuração interior do modelo obtida de forma empírica.

O interior de uma rede neural também é conhecido como uma caixa preta, por se tratar de um local de onde não se pode tirar muitas estatísticas visto que consiste em matrizes sobre matrizes que até mesmo uma interpretação mais razoável do tema se torna complexo, sendo o mais importante aqui saber o que acontece lá dentro, como apresentado ao longo do referencial

teórico, contudo os respectivos valores de cada peso ou bias não tem grande relevância, pois como pode-se observar ao todo este modelo consiste em um total de 26.114.614 parâmetros, sendo que caso fosse considerado uma imagem com melhor resolução este valor aumentaria proporcionalmente à qualidade da imagem.

Quadro 1: Construção do modelo

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_5 (InputLayer)            [(None, 200, 200, 3) 0
__________________________________________________________________________________________________
xception (Functional)           (None, 2048)         20861480    input_5[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 1024)         2098176     xception[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 1024)         2098176     xception[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 1024)         0           dense_6[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 512)          524800      dense_4[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 512)          524800      dropout_1[0][0]
__________________________________________________________________________________________________
out1 (Dense)                    (None, 1)            513         dense_5[0][0]
__________________________________________________________________________________________________
out2 (Dense)                    (None, 1)            513         dense_5[0][0]
__________________________________________________________________________________________________
out3 (Dense)                    (None, 1)            513         dense_5[0][0]
__________________________________________________________________________________________________
out4 (Dense)                    (None, 1)            513         dense_5[0][0]
__________________________________________________________________________________________________
out_item (Dense)                (None, 10)           5130        dense_7[0][0]
==================================================================================================
Total params: 26,114,614
Trainable params: 26,060,086
Non-trainable params: 54,528
__________________________________________________________________________________________________
```

Fonte: O autor

A saída do modelo é feita por uma lista contendo 5 valores, cada um com a sua utilidade para o objetivo inicialmente definido. Os 4 primeiros representam a localização da falha, trazendo as coordenadas, x e y, formando assim os quatro vértices de um quadrilátero que deverá circundar a falha encontrada. O último elemento desta lista representa uma outra lista encontrada para definir qual é o tipo de defeito, sendo que é possível localizar 10 possibilidades de não conformidade, o 5 elemento da lista trará um valor que deve ser interpretado como sendo a característica que mais se aproxima.

3.3. Evolução

Existem diferentes formas de se treinar um modelo de redes neurais, pode-se citar aqui duas formas em especial, o aprendizado supervisionado e o não supervisionado, o primeiro é realizado quando se conhece as saídas esperadas e se determina a partir delas, uma forma esperada para o modelo se comporte, já a segunda é quando não se tem dados suficientes para serem passados como parâmetro ao modelo, este tipo é utilizado para modelar situações novas e processos que têm uma alta aleatoriedade.

Portanto, como para este estudo foi construído as respostas que se esperava do modelo, foi utilizado o aprendizado supervisionado, que funciona por meio do ajuste dos pesos por meio de repetidas vezes que são passados as entradas que ao serem confrontadas com as saídas esperadas se alteram a fim de que na próxima vez a resposta apresentada se aproxime da esperada, este processo é feito até que atinja um grau de erro que seja suportado para o processo.

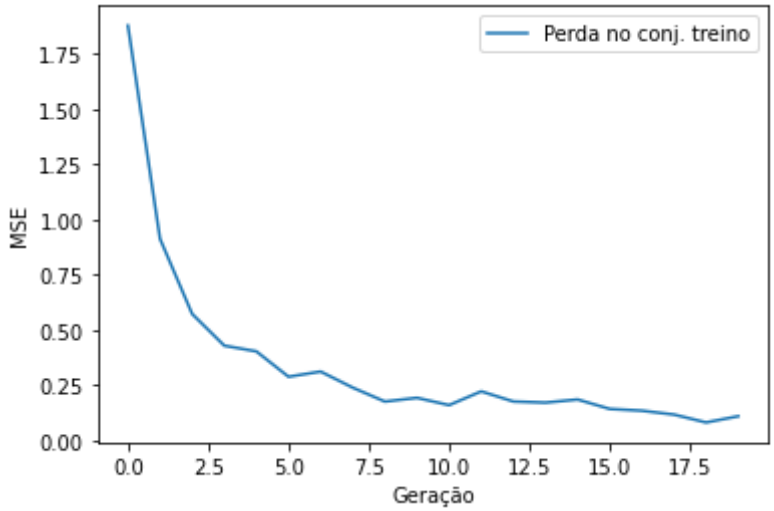
Tendo em vista que rodar um modelo por um longo período de tempo até se atinja um erro muito próximo de zero pode ser pior do que aceitar um modelo que tenha uma precisão um pouco menos, isso ocorre pelo problema do sobre ajuste, que consiste no modelo se tornar tão bom em prever as respostas para aqueles dados de treino que pode acabar sendo ruim na previsão de dados inéditos.

Neste trabalho o modelo foi exposto ao conjunto de treino e suas respectivas respostas esperadas por 20 gerações, que para o objetivo deste estudo foi o suficiente, visto que mais gerações tornaria o modelo mais preciso, porém seu custo de tempo seria desproporcionalmente maior. Por isso dado como suficiente às 20 gerações demandaram um total de 7 horas até que o modelo tivesse evoluído.

4. Resultados e discussão

O modelo finalizado, ou seja, após o treinamento que foi exposto ao conjunto de dados do treinamento, por vinte gerações apresentou os resultados demonstrados nos gráficos seguintes, em que pode-se observar no gráfico **X** que o MSE (da sigla em inglês Mean Squared Error) que significa o erro quadrado médio foi diminuindo conforme se passaram as gerações, esse fato se apresenta de forma satisfatório para o modelo, visto que na última geração o MSE se aproxima de zero, com isso sabe-se que em média de todas as respostas que o modelo deu seus respectivos erros têm diminuído conforme a rede neural aprende.

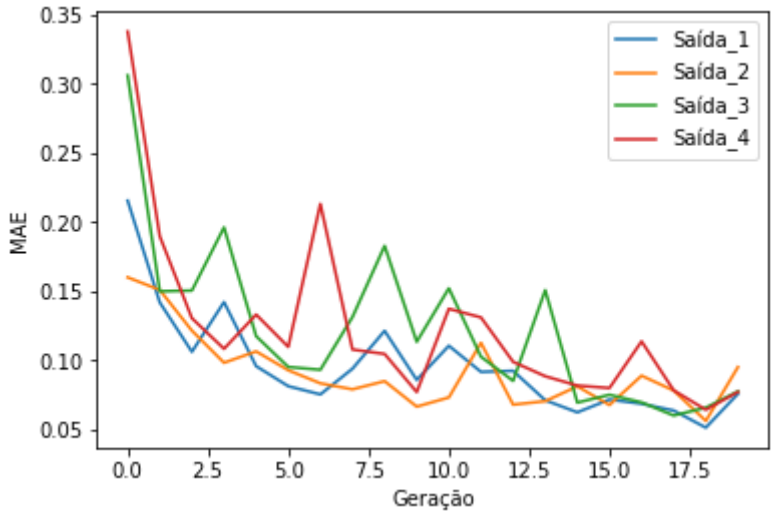
Gráfico 1: Relação do decrescimento do erro quadrado médio e a evolução do modelo.



Fonte: O autor.

Como segundo resultado foi plotado o gráfico com a MAE (sigla do inglês Mean Absolute *Error*) que significa o erro médio absoluto, ele pode ser observado abaixo. Nele foram inseridos os resultados de 4 das 5 saídas da rede neural, sendo elas as 4 coordenadas correspondentes por destacar a falha encontrada. Assim como no gráfico anterior é possível notar que com o passar das gerações o erro está diminuindo, porém, aqui não está de forma consecutiva, ou seja, não é uma redução após a outra até atingir seu mínimo, o que acontece é que em determinadas gerações o erro abaixa e outras sobe um pouco, porém no geral ele está sempre de forma a reduzir. Esse fato seria melhor definido caso houvesse mais gerações, nas quais seria possível observar como resultado uma maior definição para a curva.

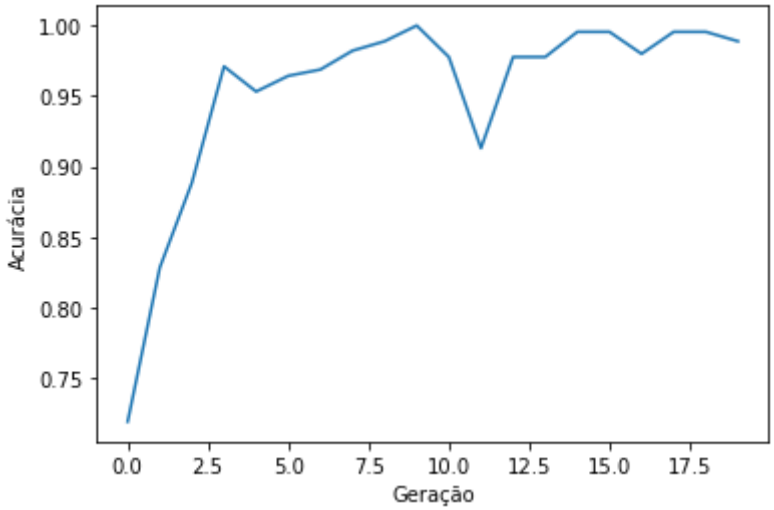
Gráfico 2: relação entre o erro médio absoluto e a evolução do modelo.



Fonte: O autor.

Por fim, o último resultado colhido foi a acurácia, que é uma importante medida para verificar o quão preciso o modelo é e o se há algum viés que interfira nos resultados. O Gráfico x traz a acurácia da última resposta do modelo que é a classificação, e nota-se que quanto mais gerações se passam melhor é o desempenho da rede, sendo assim possível dizer que os resultados para o conjunto de treino ultrapassam os 90% para a determinação de qual foi a falha encontrada na imagem (GÉRON, 2019; HAYKIN,2007).

Gráfico 3: relação entre a acurácia e a evolução do modelo.



Fonte: O autor.

Tendo sido realizadas as etapas de treinamento da rede neural foi possível prosseguir para a próxima etapa, que é o teste do modelo no conjunto de dados que foi inicialmente separado justamente para este propósito. O objetivo aqui é tentar como será o seu comportamento quando apresentado a imagens que até o momento eram desconhecidas, para tentar simular como aconteceria em uma possível aplicação, pois os padrões de falhas seguem o mesmo padrão, porém com algumas variações, por exemplo, um mesmo problema pode se apresentar com diferentes configurações, mais alongado, mais profundo, mais fino, essas pequenas variações que o modelo deve ser capaz de captar.

Como resultados, apresentados no Quadro **X**, obteve-se um erro quadrado médio próximo a 80% para as 4 saídas responsáveis por indicar a localização da falha, e a quinta saída apresentou um resultado de 90% de acurácia, então em cada 10 novas imagens apresentadas o modelo se mostrou capaz de classificar corretamente 9 destas estes resultados se apresentaram satisfatórios para o objetivo deste trabalho, provando ser possível a construção e aplicação do modelo, porém ainda há o que ser melhorado e aperfeiçoado para deixá-lo mais eficiente.

Quadro 2: Resultados para o conjunto de teste

Erro quadrado médio	
Saída 1	0,0835
Saída 2	0,0855
Saída 3	0,08
Saída 4	0,0796
Acurácia	
Classificação	0,9013

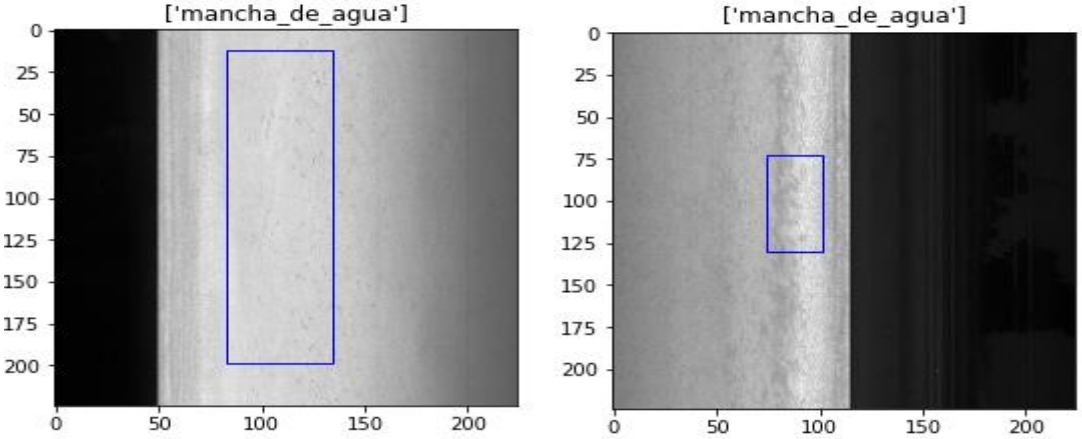
Fonte: O autor

Contudo, um problema percebido no teste do modelo foi que ele não estava sendo capaz de prever corretamente os quatro parâmetros, necessários para a identificação da falha, de forma simultânea, ou seja, uma vez acertava o 1 dos 4 errava os demais, e isso caracteriza um erro considerando a finalidade desses valores, pois um erro torna a localização do quadrilátero distorcida, levando assim a uma falha do sistema.

O problema apresentado serve para concluir o porquê o modelo foi melhor em classificar a falha do que apresentá-la de forma visual, isso aconteceu, pois, para classificar só era necessária uma saída que ao passar por todo o treinamento foi se aperfeiçoando, enquanto para destacar a falha são necessários 4 valores, que apesar de terem sido treinados ainda não foram suficientemente assertivos em conjunto.

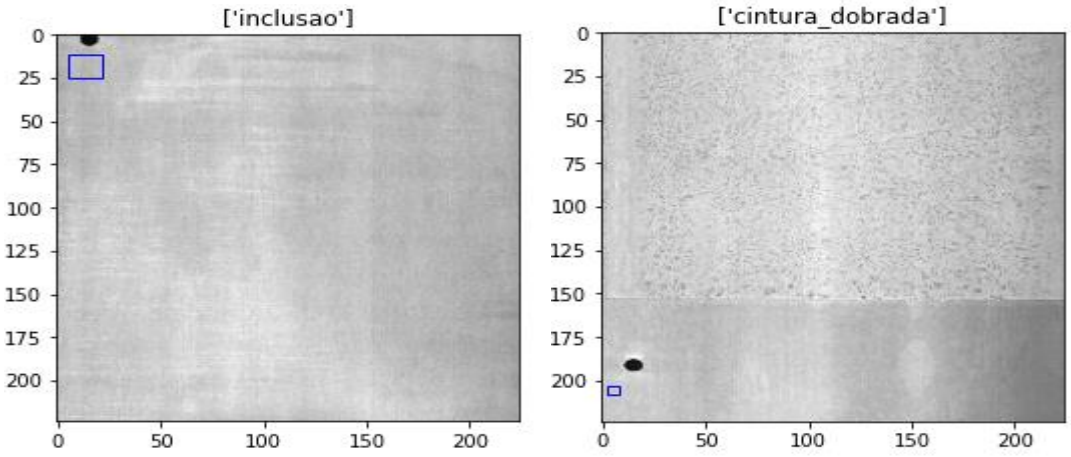
Alguns resultados do teste realizado com as imagens do conjunto de teste podem ser observadas nas figuras abaixo, em que são apresentadas tanto situações em que o modelo se mostrou assertivo em ambos propósitos, quanto em apenas um deles, assim como quando ele se mostrou falho nos dois.

Figura 7: Modelo se mostrou correto em ambos critérios



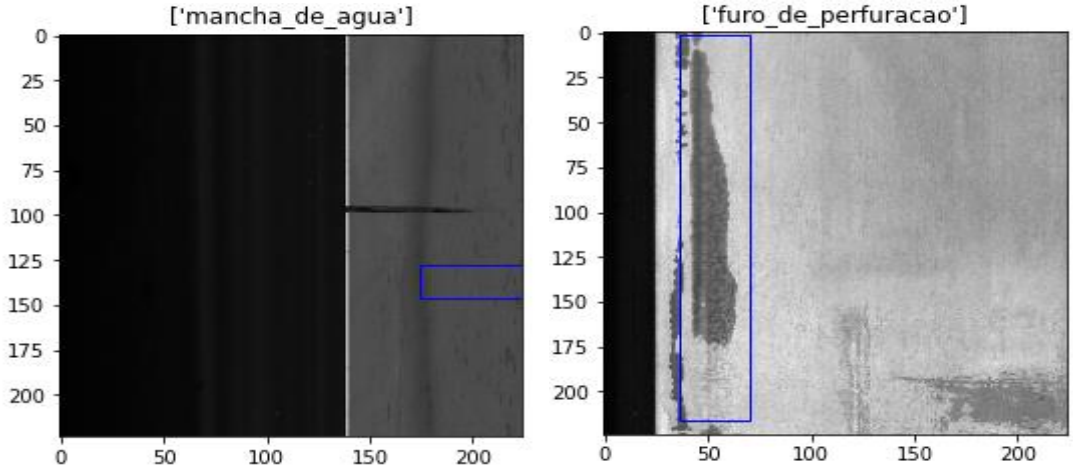
Fonte: O autor.

Figura 8: Modelo se mostrou correto apenas na classificação



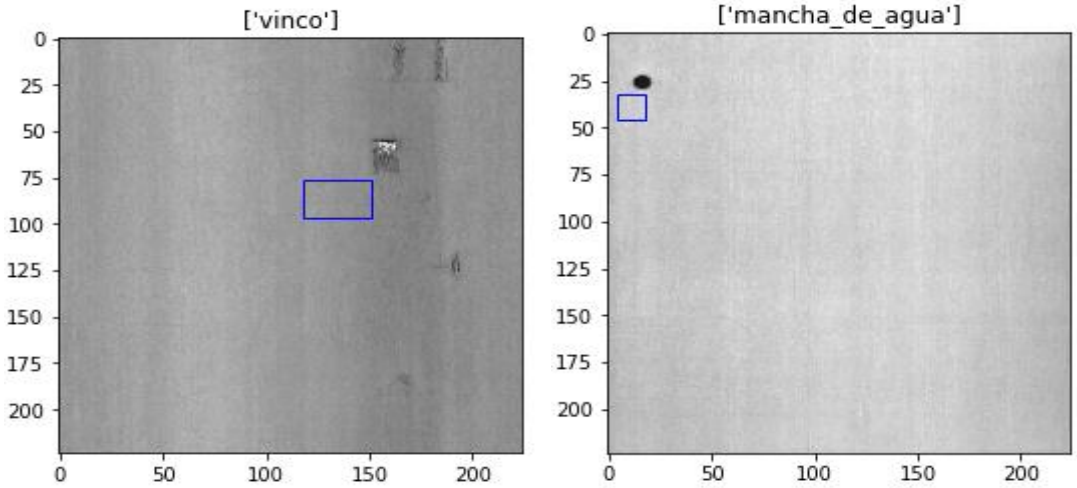
Fonte: O autor.

Figura 9: Modelo se mostrou correto apenas no apontamento da falha



Fonte: O autor.

Figura 10: Modelo se mostrou incorreto em ambos critérios.



Fonte: O autor.

5. Considerações finais

Considerando tudo que foi feito e o objetivo inicialmente traçado, pode-se afirmar que este trabalho se mostrou suficientemente satisfatório e relevante para o seu propósito, visto que o modelo construindo esta com uma boa precisão, levando em conta a capacidade dos hardwares e o tempo dedicado, sabe-se que tem grande potencial para ser otimizado visando uma maior precisão, investindo, para isso em um maior poder computacional, para posteriormente implantar o algoritmo em um software aplicativo capaz de trabalhar de forma autônoma, não só tendo a capacidade de encontrar falhas conhecidas mas também a se adaptar a novos problemas encontrados de forma autônoma, se utilizando para isso de uma parcela de seu aprendizado sendo não supervisionado.

Para trabalhos futuros há duas linhas que podem ser seguidas, a acadêmica e a industrial, sendo que na primeira este trabalho serve de demonstração de uma das possíveis aplicações do aprendizado de máquina para melhor desempenho e automação de uma tarefa que a um tempo poderia ser classificada como totalmente dependente de um ser humano. A segunda linha de continuidade poderia ser explorada por empresas e indústrias que tenham interesse em tornar um processo automático e de maior controle e padronização.

Assim finalizando esta pesquisa, porém há muito a ser explorado neste campo, podendo trazer grande reconhecimento tanto para o profissional como para o Brasil, que por se tratar de um assunto ainda considerado novo no mundo, pode ter um bom incentivo e ter destaque frente a outros países.

REFERÊNCIAS

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. O'Reilly Media, 2019.

BARRETO, Jorge M. Introdução às redes neurais artificiais. **V Escola Regional de Informática. Sociedade Brasileira de Computação, Regional Sul, Santa Maria, Florianópolis, Maringá**, p. 5-10, 2002.

Data Science Academy. **Deep Learning Book**, 2021. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 10 Abril. 2021.

HAYKIN, Simon. **Redes neurais: princípios e prática**. Bookman Editora, 2007.

ROSA, Renan de Paula et al. **Método de classificação de pragas por meio de rede neural convolucional profunda**. 2018.

SILVA, Iago RR et al. Utilização de redes convolucionais para classificação e diagnóstico da doença de alzheimer. **II Simpósio de Inovação em Engenharia Biomédica**, p. 73-76, 2018.

VARGAS, Ana Caroline Gomes; PAES, Aline; VASCONCELOS, Cristina Nader. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: **Proceedings of the xxix conference on graphics, patterns and images**. sn, 2016.

VOGADO, Luis HS et al. Rede Neural Convolucional para o Diagnóstico de Leucemia. In: **Anais do XIX Simpósio Brasileiro de Computação Aplicada à Saúde**. SBC, 2019. p. 46-57.

MORETTIN, Pedro Alberto; BUSSAB, Wilton Oliveira. **Estatística básica**. Saraiva Educação SA, 2017.