UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE ENGENHARIA

CÂMPUS DE ILHA SOLTEIRA

João Vitor Durso Ferraz

TREINAMENTO DE UMA REDE NEURAL PARA DETECTAR RACHADURAS

Ilha Solteira

2023

João Vitor Durco Ferraz

TREINAMENTO DE UMA REDE NEURAL PARA DETECTAR RACHADURAS

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de Bacharel em Engenharia Mecânica.

Orientador: **Profa. Dra. Christiane Marie Schweitzer**

Ilha Solteira

2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Ferraz, João Vitor Durso.
F381t Treinamento de uma rede neural para detectar rachaduras / João Vitor Durso Ferraz. -- Ilha Solteira: [s.n.], 2023
46 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Mecânica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Christiane Marie Schweitzer

Inclui bibliografia

1. Redes neurais. 2. Visão computacional. 3. Detecção.

Amanda Sertori dos Santos
Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos onze dias do mês de dezembro do ano de dois mil e vinte e três, o discente **JOÃO VITOR DURSO FERRAZ**, matriculado sob o RA: 191050083 , tendo como banca examinadora sua orientadora a Profa. Dra. Christiane Marie Schweitzer, o Prof. Dr. Douglas D. Bueno e a Profa. Dra. Mara Lúcia Martins Lopes, apresentou o Trabalho de Graduação do Curso de Engenharia Mecânica (FEIS/UNESP) intitulado "**TREINAMENTO DE UMA REDE NEURAL PARA DETECTAR RACHADURAS**", obtendo a nota 10,0 (Dez).

Aprovado (X) – Reprovado ( ) pela Comissão Examinadora

João Vitor Durso Ferraz
Discente (Engenharia Mecânica)

Profa. Dra. Christiane Marie Schweitzer
DMAT/FEIS/UNESP
Orientadora

Prof. Dr. Douglas D. Bueno
DMAT/FEIS/UNESP
Membro da banca

Profa. Dra. Mara Lúcia Martins Lopes
DMAT/FEIS/UNESP
Membro da banca

AGRADECIMENTOS

Ao longo da vida conhecemos muitas pessoas que são importantes para chegar aonde estamos, portanto é humanamente impossível elencar todas. Entretanto é necessário reconhecer e honrar àquelas que se destacam. Primeiramente gostaria de agradecer aos meus pais, Alexandra Durso e João Renato Ferraz, sem eles não só não teria me formado como também não seria a pessoa que sou hoje. Eles foram o meu porto seguro ao longo de toda a faculdade, mesmo nos momentos mais dolorosos em que pensei em desistir eu sabia que podia contar com eles; por isso, não tenho palavras para descrever minha gratidão a eles.

Outro agradecimento especial vai para dois professores, Christiane e Carlos. Christiane além de uma grande professora e orientadora, foi muito humana, quando tive crises e pensei em abandonar os projetos da faculdade, você me acalmou e me ofereceu total apoio, não existem palavras para agradecer. Quanto ao professor Carlos ele foi de fundamental importância para a realização desse TCC, ajudando e ensinando muito sobre redes neurais. Vocês dois foram grandes profissionais que me ensinaram muito, não somente conhecimento técnico, mas de ética.

Eu não posso esquecer também dos meus amigos. Tornaram a jornada por essa faculdade um caminho muito mais prazeroso, e se me permitem a intimidade muito mais divertido. Me ajudaram a dividir o peso dos problemas e a multiplicar os momentos de felicidade. Sem dúvida vocês são um dos presentes mais valiosos que ganhei na faculdade, mesmo que o tempo esfrie esse laço, ele não pode apagar o que vivemos nesses cinco anos. Vocês foram os irmãos que a Unesp me deu.

Por último, mas indiscutivelmente o mais importante, quero expressar minha profunda gratidão a Deus. Ele esteve ao meu lado, tanto nos momentos de alegria quanto nos de tristeza. Em particular, quando enfrentei desafios esmagadores e me senti à beira do abismo, foi o Senhor que me sustentou e renovou minhas forças. Não posso deixar de reconhecer que foi Sua presença e graça que me trouxeram através dos momentos mais difíceis. E, com imensa gratidão, também reconheço que Ele colocou as pessoas maravilhosas que mencionei em meu caminho para me apoiar. Compreendo que, mesmo nos momentos em que me senti mais sozinho, nunca estive verdadeiramente só, pois Deus sempre estava comigo, orientando-me e fortalecendo-me para continuar avançando.

“Tudo o que fizerem, façam de todo o coração, como para o Senhor, e não para os homens, sabendo que receberão do Senhor a recompensa da herança. É a Cristo, o Senhor, que vocês estão servindo.” Colossenses 3:23-24

RESUMO

A visão é um dos sentidos mais importantes para nós humanos, permitindo-nos identificar formas, cores, texturas e contornos. Todas essas informações são rapidamente processadas pelo nosso cérebro em questão de milissegundos. Esse processo, que para nós é automático, é incrivelmente preciso e veloz, possibilitando que qualquer pessoa reconheça imediatamente uma imagem. Embora profissionais altamente treinados possam extrair informações adicionais, mesmo uma pessoa com conhecimentos básicos seria capaz de identificar, por exemplo, uma rachadura. Por esse motivo, é comum que algumas pessoas se dediquem à inspeção de locais em busca de problemas estruturais. No entanto, essa abordagem envolve riscos, pois requer que indivíduos entrem em locais potencialmente perigosos. Além disso, é necessário alocar essas pessoas até o local, e elas não podem permanecer lá o tempo todo. Isso significa que pode haver um período considerável entre o surgimento da rachadura e sua detecção, o que pode agravar a situação ou colocar vidas em perigo. Uma solução para esse problema é usar o computador para identificar essas falhas. Os computadores podem realizar essa tarefa de maneira rápida e contínua, sem colocar em risco a segurança de um ser humano. Além disso, o uso de drones pode ser uma opção eficaz para inspeções em locais de difícil acesso. Portanto a maior dificuldade está em fazer com que o computador consiga reconhecer padrões em imagens para detectar objetos. Assim, o presente trabalho tem por objetivo utilizar os conhecimentos adquiridos em visão computacional para buscar uma solução, usando ferramentas de reconhecimento de imagens e padrões, como ferramentas de redes neurais, para detectar rachaduras, localizá-las e informar as pessoas sobre sua confiança. Os resultados gerados serão analisados em conformidade a precisão alcançada.

Palavras-chave: visão computacional, redes neurais, reconhecimento de imagens

ABSTRACT

The vision is one of the most important senses for us humans, allowing us to identify shapes, colors, textures, and contours. All this information is rapidly processed by our brain in a matter of milliseconds. This process, which is automatic for us, is incredibly accurate and fast, enabling anyone to immediately recognize an image. Although highly trained professionals may extract additional information, even a person with basic knowledge would be able to identify, for example, a crack. For this reason, it is common for some people to dedicate themselves to inspecting locations for structural problems. However, this approach involves risks, as it requires individuals to enter potentially dangerous places. Additionally, these people need to be allocated to the site, and they cannot remain there all the time. This means there may be a considerable period between the emergence of the crack and its detection, which can worsen the situation or put lives in danger. A solution to this problem is to use computers to identify these flaws. Computers can perform this task quickly and continuously without jeopardizing the safety of a human being. Furthermore, the use of drones can be an effective option for inspections in hard-to-reach locations. Therefore, the major challenge lies in enabling the computer to recognize patterns in images to detect objects. Thus, the present work aims to use the knowledge acquired in computer vision to seek a solution, employing image and pattern recognition tools such as neural network tools to detect cracks, locate them, and inform people about their confidence. The generated results will be analyzed in accordance with the achieved accuracy.

Keywords: *computer vision, neural networks, image regnition*

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O trabalho de graduação tem como foco um modelo de rede neural artificial para detecção de imagens, mais especificamente, reconhecimento de imagens que representam rachaduras em estruturas, abordando desde as técnicas e algoritmos de processamento de imagens até seu treinamento. Modelo esse que compreenderá o treinamento em um banco de imagens (*dataset*) customizado e a transferência de aprendizado para outros modelos. No presente trabalho o termo rachadura foi usado como forma generalista para trincas, fissuras e rachaduras, uma vez que o mesmo não tem como foco a diferenciação das mesmas, mas sim a detecção de possíveis falhas mecânicas.

Além disso, será realizado uma análise abrangente dos dados obtidos durante o processo de treinamento e validação, avaliando o sucesso do modelo em termos de métricas específicas de desempenho, identificando pontos de melhoria e refinamento que podem ser aplicados em trabalhos futuros.

Para testar o protótipo, serão utilizados conjuntos de dados representativos, adaptados às necessidades do projeto durante a fase de implementação. Este trabalho visa contribuir para a melhoria da detecção de imagens em contextos específicos, fornecendo um modelo que possa ser usado futuramente em aplicações práticas.

1.1 Justificativa

Existe um perigo real em rachaduras, dependendo do local onde elas se encontrem trazendo grande risco as pessoas a volta. Por exemplo, alguns pontos podem ser de vital importância estrutural e seu comprometimento pode levá-los ao colapso. Pontes, pilares ou barragens são exemplos de estruturas que comprometidas podem colocar em risco a vida de várias pessoas. Além disso alguns desses lugares podem ser de difícil acesso, o que dificulta uma fiscalização contínua ou coloca em risco a vida do inspetor.

Outro fator que faz necessário o uso de detecção de imagens é a necessidade de se lidar com grande volume de construções que devem ser analisadas. Segundo o Sistema Nacional de Informações sobre segurança de barragens (SNISB) o Brasil possui 22.654 barragens, além disso o Instituto Brasileiro de Concreto (Ibracon) diz que em solo nacional existem em torno de 137 mil pontes espalhadas. Ou seja, existe um grande volume que deve ser analisado e somente o poder de trabalho humano não será rápido o suficiente para agir de forma eficiente.

Nesse contexto a detecção deste tipo de imagens pode vir a desempenhar um papel fundamental na segurança. Permitindo realizar inspeções contínuas sem expor pessoas ao risco, garantindo assim a integridade das estruturas e prevenindo acidentes. Assim se tornando uma ferramenta valiosa para mitigar riscos desnecessários e proteger a vida.

1.2 Visão Computacional

Segundo Forsyth (2023), "*a Visão Computacional é a disciplina que ensina aos computadores como extrair informações de imagens"*. Essa área envolve a aquisição, processamento e análise de informações visuais, permitindo que os computadores analisem imagens e vídeos. Esse campo abrange uma ampla variedade de aplicações e metodologias, no caso em questão será utilizado o conceito de classe.

Nesse conceito um conjunto de “objetos” é caracterizado por uma classe. Assim o programa reconhece as características das classes e com isso determina de qual objeto se trata. Por exemplo, se houver a necessidade de se detectar um carro, é necessário um treinamento utilizando imagens de carros, isso significa que o algoritmo aprenderá os parâmetros comuns a todo carro. Se for necessário especificar a análise para um tipo de carro ou um modelo, o treinamento deve ser feito usando somente objetos dessa subclasse de carro. Portanto, quando se faz uma análise de classe geral como rachadura, obtém-se imagens de tipos diferentes de rachaduras e o mais distintas possíveis.

A visão computacional desempenha um papel fundamental na sociedade moderna, impulsionando avanços tecnológicos em diversas áreas como medicina, segurança e automação por exemplo. Estes ramos da tecnologia se beneficiam com esses avanços, uma vez que a interação entre homem e máquina acaba ficando mais próximo, o que permite a futuros programadores o uso dessas aplicações em produtos futuros.

1.3 Objetivo

Neste trabalho, o objetivo é usar ferramentas de reconhecimento de imagens e padrões, como redes neurais artificiais, para detectar rachaduras, localizá-las e informar sua confiança. Em suma treinar um modelo de rede neural capaz de identificar rachaduras. Os resultados gerados serão analisados em conformidade a precisão alcançada.

1.4 Estrutura do Trabalho

Além da Seção 1, o trabalho segue a seguinte estrutura: na Seção 2, encontra-se uma revisão da literatura, sendo essa fundamental para o entendimento do trabalho. A Seção 3 é

dedicada à implementação do programa e seu treinamento. A Seção 4 se concentra na apresentação dos dados gerados pelo programa e uma análise dos mesmos. Na Seção 5, as considerações finais, e, também, discussão das possibilidades de projetos futuros.

2 REDES NEURAIS ARTIFICIAIS

Nesta seção é realizada uma revisão da literatura, onde são reunidos os conceitos importantes e pertinentes para as discussões necessárias neste trabalho de graduação.

2.1 Funcionamento de uma Rede Neural

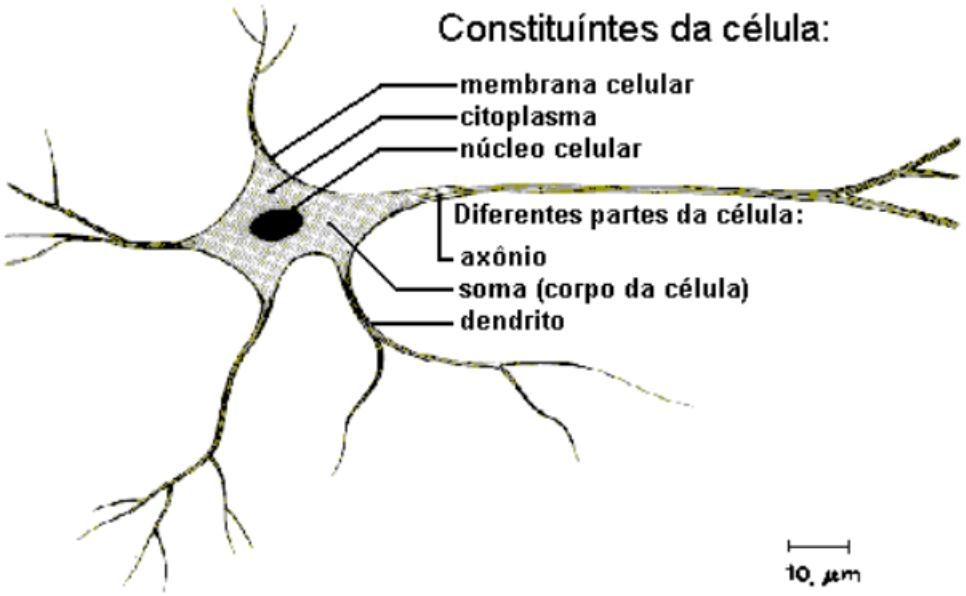
O primeiro passo para entender o processo de detecção de imagens proposto é entender o funcionamento de uma rede neural. Primeiramente o porquê desse nome, elas atuam com base em nós que trabalham conjuntamente para reconhecer padrões, sendo assim seus algoritmos conseguem analisar montantes de dados. E com o passar do tempo e treinamento ela é capaz de se adaptar e melhorar seu resultado, ou seja, ela tem a capacidade de se desenvolver com o tempo.

Em tese os primeiros conceitos surgem por volta de 1943 com Warren McCulloch e Walter Pitts, nessa época elas eram feitas usando circuitos elétricos, sendo bem distantes da forma que atualmente é abordado.

A rede mais próxima começa a aparecer em 1975 com Kunihiko Fukushima. Quanto aos tipos de redes neurais existem uma grande diversidade, e nesta aplicação será utilizado as redes neurais convolucionais, sendo essas bastante utilizadas na área de detecção de objetos e classificação de imagem (Carvalho, 2009).

Portanto o primeiro passo para o entendimento do que está sendo feito aqui é entender o que é um neurônio, eles são as unidades fundamentais desse processo. Obviamente como pode ser notado pelo nome eles se inspiram na estrutura neural de organismos inteligentes, nosso sistema nervoso é formado por células complexas os denominados neurônios. Eles são formados por dendritos, que são os responsáveis pelas entradas das informações, o corpo central e os axônios que são os terminais de saída, uma ilustração se encontra na Figura 1.

Figura 1 - Constituintes da célula neuronal.



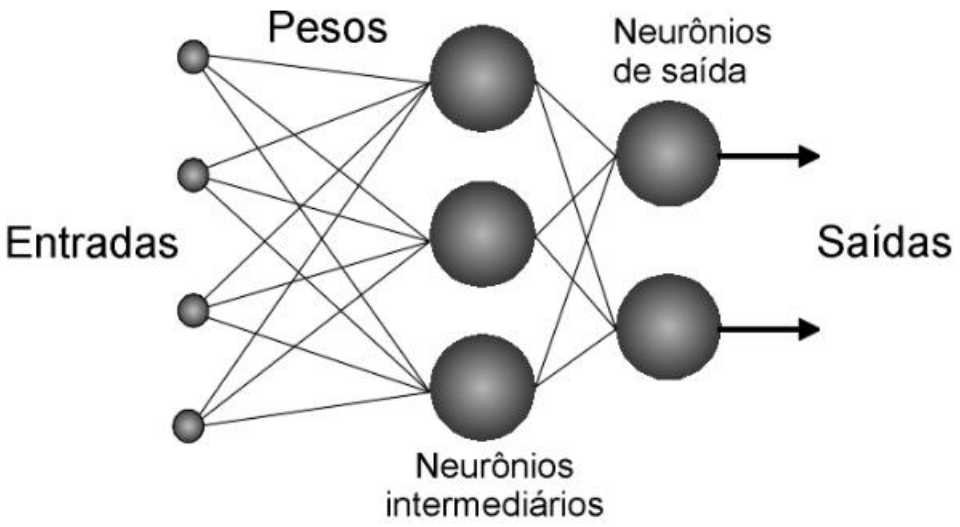
Fonte: (Carvalho, 2009).

Os neurônios em uma rede neural, são as unidades fundamentais que simulam os neurônios do cérebro humano em um contexto computacional. Cada neurônio recebe um ou mais sinais de entrada, realiza uma série de cálculos nesses sinais e produz uma saída. Essa saída, por sua vez, pode ser enviada para outros neurônios na rede ou ser a saída final da rede neural.

Cada neurônio é caracterizado por ter um conjunto de pesos associados a suas entradas. Esses pesos representam a importância relativa das diferentes entradas para a operação do neurônio. A função soma realiza a soma ponderada dessas entradas multiplicadas pelos pesos e aplica uma função de ativação a ela. A função de ativação determina se o neurônio deve ser ativado (ou seja, produzir uma saída) com base no resultado da soma ponderada.

No geral ao visualizar a estrutura de uma rede neural é importante ter a seguinte visão a entrada é uma camada que pode ser constituído por vários neurônios, a camada oculta são os neurônios intermediários responsáveis pelo processamento e a camada de saída são os resultados obtidos pela mesma como pode ser visto na Figura 2.

Figura 2 - Exemplo de uma Rede Neural Artificial de 2 camadas com 4 entradas e 2 saídas.



Fonte: (Tafner, 2023).

Vale ressaltar, que é possível se criar uma rede contendo apenas uma camada intermediaria, onde toda as entradas seriam analisadas diretamente por essa camada e já seria obtido um resultado, entretanto embora possível na teoria na prática esse processo é inviável, já que o seu treinamento seria muito extenso. Isso ocorre, pois, as alterações do treinamento atingirão todos os neurônios envolvidos, enquanto ao utilizar camadas pode-se alterar somente as que serão mais relevantes.

Portanto, para facilitar o processo, usa-se o artifício chamado de rede profunda onde mais camadas são criadas, assim a entrada é avaliada em vários níveis diferentes, onde cada nível seria uma camada. Assim o tempo de treinamento pode ser mais rápido pois pode se treinar somente as camadas relevantes. Esse artifício é muito comum denominado de redes de aprendizado profundo.

Um recurso muito importante para o treinamento e evolução de uma rede é o ***backpropagation*** onde os erros que forem cometidos pela rede retornam para ela como forma de aprendizado, assim ela aprende e melhora com os erros. Para isso ela compara o valor final de suas saídas com um conjunto de dados já validados, se o resultado estiver muito distante ela retorna isso mudando os pesos e o ciclo se repete, até que esse valor se torne mais próximo do ideal.

Elucidando um pouco mais a fundo o *backpropagtion* ele funciona em duas fases a propagação e a retropropagação. Quando um modelo novo é criado seus pesos surgem de forma aleatória, portanto é importante que esse processo seja feito para corrigir os mesmos ao

longo do treinamento. Para se dar início começa-se com a propagação, onde um sinal de entrada percorre a rede com os pesos atuais e obtém-se uma saída. Essa saída é comparada com um valor de resultado já obtido que é a resposta correta. Com base nisso pode-se encontrar o erro entre o que foi obtido ao final da rede. Entretanto a rede é um mecanismo matemático a sua saída pode ser visto como uma função formada por funções menores, cada função pode ser vista como a resposta de saída de um neurônio. Por exemplo uma rede de 3 camadas com 2 neurônios cada, vai possuir 3 funções que estarão uma dentro da outra onde em cada uma delas existe a soma dos neurônios. Esse processo parece um tanto confuso ao se descrever dessa forma, mas o importante é entender que o resultado será uma função com funções internas referentes a cada etapa do processo. O erro obtido no final passa então para a etapa de repropagação onde ele retorna ao indo da saída até a entrada, para isso ele faz alterações nos pesos. Cada peso vai ser alterado com base na derivação ao longo de de cada etapa já que a resposta é uma função. Dessa forma os pesos serão atualizados, e assim sucessivamente a cada treinamento, até o encerramento do treinamento.

Na camada de entrada o número de neurônios será igual ao número de entradas, e o mesmo é valido para a saída, se houver apenas uma saída, a mesma contém somente um neurônio na camada de saída. Se afirmar a hipótese de que, a rede foi treinada para avaliar a probabilidade de uma pessoa sofrer um acidente, para isso, a entrada fornecida é o perfil da pessoa e o carro que ela está usando, logo, a entrada possui dois neurônios. Para a saída, o mesmo raciocínio é valido, como se quer saber somente a probabilidade, tem-se um neurônio. No entanto, se quer saber a probabilidade e a data do acidente, por exemplo, há a necessidade de se ter dois neurônios na saída.

Entretanto, definir o número de camadas ou neurônios por camadas, nas camadas ocultas é mais complicado, não existindo uma regra isso variará conforme cada arquitetura.

Normalmente, a interação entre as camadas é feita do seguinte modo: cada neurônio de uma camada se liga a todos os outros neurônios da camada seguinte, tal ligação é chamada de sinapse, baseada no nome biológico da interação de neurônios, entretanto ele nunca faz essa ligação com neurônios da mesma camada que ele. Cada uma dessas sinapses vai ter um peso, elemento fundamental para o aprendizado, uma vez que ela afeta como as camadas interagirão uma com outra.

Na prática durante o treinamento os neurônios não são alterados, o que é alterado nesse processo são os pesos, por meio do treinamento eles são modificados a cada época do treinamento. A estrutura com mais camadas permite que esse treinamento seja feito com uma

quantidade menor de neurônios, e como poderá ser visto adiante, ela permite uma transferência de aprendizagem mais efetiva, pois é possível congelar algumas camadas visando realizar treinamento somente a partir de certo ponto.

O treinamento de uma rede pode ser realizado de três maneiras, sendo as principais: supervisionado, não supervisionado e reforço.

- **Supervisionado** é quando os resultados obtidos pela rede são comparados a outros dados previamente rotulados, de certa forma é como se o programa tivesse um gabarito prévio de alguns resultados otimizados para algumas entradas, assim ele consegue analisar se o que foi obtido faz sentido ou não.
- **Não supervisionado** é quando a rede compara, não a uma resposta, mas separa padrões dentro dos conjuntos de dados mostrados, de certa forma é como se esse processo contasse com a intuição da máquina reconhecendo padrões dos resultados obtidos.
- **Reforço** é quando um crítico externo avalia a resposta da rede.

Em resumo, uma entrada é apresentada a rede, cada uma delas é multiplicada por um peso e vai de um neurônio para outro. Em seguida é feita a soma ponderada desses valores no neurônio que recebeu os dados, isso gera uma atividade, que caso seja suficiente, produzirá uma saída.

2.2 Redes Neurais Convolucionais

O termo convolucionais que é apresentado se refere à operação de detecção de mesmo nome, que é usada nesse processo. As redes neurais convolucionais (CNN – Convolutional Neural Networks) são as mais utilizadas na análise de imagens, por isso a utilização desta no presente projeto. “Uma rede convolutiva é uma rede de múltiplas camadas projetada especificamente para reconhecer formas bidimensionais com muita invariância quanto a translação, escalamento, inclinação e outras formas de distorção. Está difícil tarefa é aprendida de uma forma supervisionada por meio de uma rede cuja estrutura inclui restrições” Haykin (2001).

O funcionamento de uma rede convulocional utiliza de matrizes para varrerem a imagem e aplicarem filtros, cada filtro auxilia a selecionar as características e os pixels mais importantes. Por conta disso uma rede convolucional se diferencia (destaca) das demais pela presença de camadas especializadas que identificarão padrões na imagem de forma

hierarquizada. Sendo assim algumas camadas ficam responsáveis por pontos, outras por linhas, outras por cantos e assim por diante. Dessa forma a camada posterior não muda os resultados da camada anterior, passando a informação a diante de forma rápida e efetiva.

Conforme a complexidade das entradas aumenta é necessário se ter um aumento da complexidade das células envolvidas. O mesmo ocorre com as redes neurais, quando há uma entrada mais complexa, como imagens, deve se ter uma estrutura mais complexa e hierarquizada. O reconhecimento dos padrões é feito de forma escalar, pixels se unem em arestas, formando padrões e esses são combinados, dando origem as partes que formam objetos e objetos formam cenas.

Quando se olha para uma imagem na camada inicial, cada neurônio é conectado a uma pequena região da imagem de entrada. Por exemplo, cada neurônio na camada teria um "campo receptivo" de tamanho 3, o que significa que ele recebe informações de 3 neurônios da camada anterior (ou seja, 3 pixels na imagem).

À medida em que se sobe na hierarquia das camadas, como para a camada posterior, o campo receptivo é maior em relação à camada anterior. Isso significa que cada neurônio na camada seguinte também recebe informações de uma área maior da imagem de entrada.

Cada "campo receptivo" atua como um filtro não linear. Os pesos associados a esse filtro precisam ser aprendidos durante o treinamento da rede neural. Esses pesos são ajustados para que o neurônio na camada seja ativado, somente quando um padrão específico, correspondente ao estímulo desejado, estiver presente na área onde o filtro foi aplicado na imagem. Isso permite que a CNN identifique características e padrões relevantes em uma imagem, como bordas, texturas e formas, à medida que passa pelas várias camadas da rede.

A convolução é um processo que ajuda a destacar padrões e características específicas na imagem à medida que ela passa pela rede neural convolucional. Isso é fundamental para a detecção de bordas, texturas e outros elementos visuais em tarefas como reconhecimento de objetos em imagens. Pois isso reduz a dimensão do mapa de características, aumenta a eficiência computacional, fazendo com que o modelo generalize melhor. Como é mostrado a seguir esse fator é muito útil para auxiliar o processo de transferência de aprendizagem.

2.3 Transferência de Aprendizagem

Nos diversos trabalhos e modelos de redes, encontrados na literatura, identificou-se que as redes são, na sua maioria, montadas a partir de um treinamento extenso, com grandes

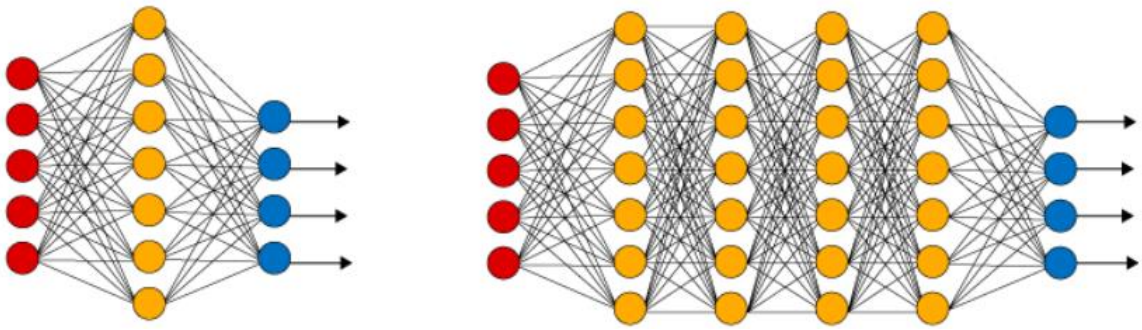
bancos de dados. Assim, não há a necessidade de começar uma rede do zero, aproveitando-se desse conhecimento, agilizando a formação de novos conhecimentos.

Este raciocínio, também, é aplicado as redes neurais, não é necessário iniciar do zero, ensinando-a a identificar um pixel, um ponto, uma linha ou um traço por exemplo. É possível partir de modelos que já treinaram essas identificações e ao adicionar mais algumas camadas a essa rede, obtém-se novos neurônios, os quais direcionarão a rede a sua nova detecção.

Neste trabalho, imagens com rachaduras serão analisadas, então acrescenta-se novas camadas que detectarão esse tipo de objeto, especificamente. Para isso, é necessário alterar o funcionamento padrão da rede que está sendo utilizada, pois ela já detecta pontos, traços e formas, e criará um novo conhecimento a partir deste.

Em suma, é como se essa rede fosse uma criança, ela já sabe o que é uma linha, aprendeu o que é uma cor e coisas do tipo, e agora é ensinado a reconhecer os padrões que definem uma rachadura. Por conta disso, o conhecimento que ela já tem é “congelado”, não repetindo o treinamento de conceitos anteriores e gastando esforços no treinamento de um novo conjunto de camadas especializadas voltadas para o objeto que selecionamos. A Figura 3 representa esta transferência de aprendizado.

Figura 3 - Transferência de aprendizado.



Fonte:(Academy, 2023).

2.4 YOLO v8

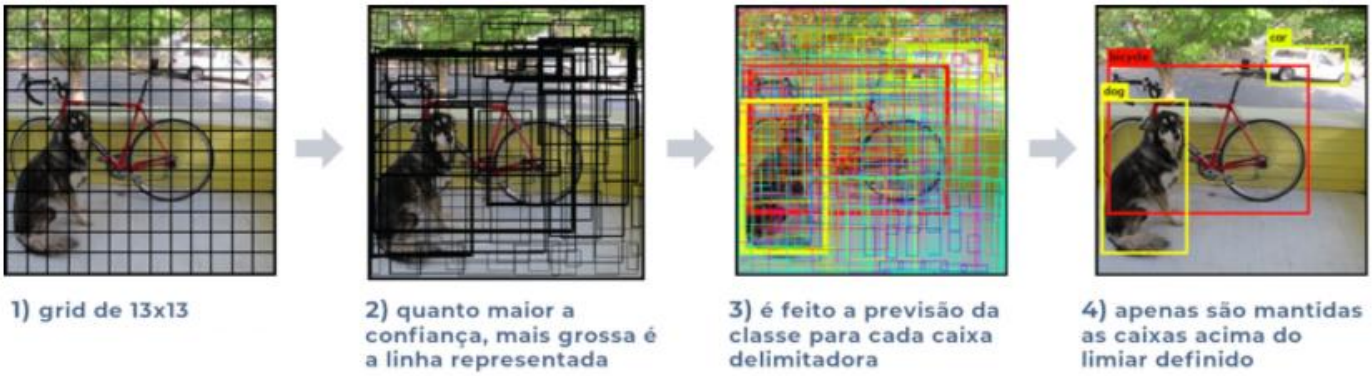
A ferramenta YOLOv8 é derivada da ferramenta original YOLO. O YOLO é um recurso utilizado para a detecção/reconhecimento de imagens, o qual é bastante utilizado pela

comunidade de Inteligência Artificial. YOLO é a sigla de "*You Only Look Once*", o que significa," você olha apenas uma vez".

O objetivo desta ferramenta enviar a imagem para uma rede neural, analisando a imagem inteira detectando caixas que contêm objetos. Essas caixas recebem o nome de *bounding box*, que significa caixa delimitadora.

YOLO consiste em um método de detecção de objetos, de passada única, que se utiliza de uma rede convolucional para extrair características formando *bounding boxes* que podem conter ou não objetos.

Figura 4 - Funcionamento Yolo.



Fonte: (Alves, 2023).

O funcionamento básico do YOLO consiste, então, no algoritmo dividir a imagem em um *grid* quadrado tendo n células em cada dimensão, como pode ser visualizado na Figura 4. Cada uma dessas células fará a predição de até 5 caixas delimitadoras, ou seja, cada uma pode detectar até 5 objetos diferentes. Retornando uma pontuação de confiança, que é basicamente, qual a chance de que naquela caixa, realmente, contém um objeto.

Dessa forma, ele consegue encontrar os locais mais prováveis de se ter um objeto. Assim, encontrando uma probabilidade para o tamanho das caixas e seus formatos, e na sequência, usando uma rede convolucional faz a previsão de qual é a classe desse objeto.

Em resumo, uma vez que a rede encontre o formato da caixa, ela saberá que ali tem um objeto, trabalhando na parte delimitada pela caixa, a fim de encontrar o que é esse objeto e qual a probabilidade dele, realmente, ser um objeto. Um parâmetro mínimo de confiança é definido para que o sistema não pegue qualquer coisa e torne em um objeto. Assim, esta é a forma que o YOLO reconhece a presença de um objeto e sua classe.

Normalmente, o YOLO é usado com a arquitetura de rede conhecida como *darknet*. Ambas são compatíveis, pois possuem o mesmo criador. Entretanto, é possível se utilizar da rede YOLO em parceria com o OpenCv (*Open Computer Vision Library*), a qual possui mais recursos visuais.

Como citado anteriormente, o Yolo/Darknet possuem o mesmo criador, Joseph Chet Redmon, que trabalhou até a versão 4 e as demais versões foram implementadas por terceiros, sobre o projeto original.

Neste trabalho será utilizado o YOLOv8, o qual pode ser operado de forma independente, sendo a versão mais atualizada. Sua escolha se dá pela simplicidade e operação amigável com o usuário, e, também, por ser uma versão que possui maior facilidade em reconhecer arquivos de formatos.

3 DESENVOLVIMENTO

Para o desenvolvimento deste trabalho, uma série de passos foi seguida, separando em 3 etapas, criação de *dataset*, treinamento da rede e validação do teste.

3.1 Conjunto de Dados (*Dataset*)

O treinamento é feito com base em um conjunto de dados, conjunto esse nomeado de *dataset*. Este é composto por imagens e por arquivos txt, arquivos principais para se poder realizar um treinamento na YOLO. As imagens são as mais intuitivas de entender, elas devem ser imagens dos objetos que se quer detectar, os quais podem estar em primeiro plano ou no meio da imagem junto com outros objetos. Na Figura 5 são apresentados exemplos de imagens presentes no *dataset*.

Figura 5 - Exemplos de imagem do dataset.

Fonte: (YoKun, 2023).

Cada imagem possuirá um nome, os arquivos .txt devem ter o nome correspondente a imagem a que eles se referem. Cada arquivo .txt conterá a seguinte estrutura:

- Em cada linha o primeiro número identificará a qual classe o objeto pertence;

- Os dois próximos números são informações da posição do *bounding* box no eixo x, e o comprimento do *bounding box*,
- Os dois últimos são a posição do *bounding box* no eixo y e sua largura
- Caso haja mais de um objeto por imagem haverá mais de uma linha.

Alguns exemplos de txt se encontram na Figura 6

Figura 6 - Exemplos de txt do dataset.

```
0 0.5 0.54140625 0.9984375 0.096875
0 0.46171875 0.41328125 0.73515625 0.17578125
```

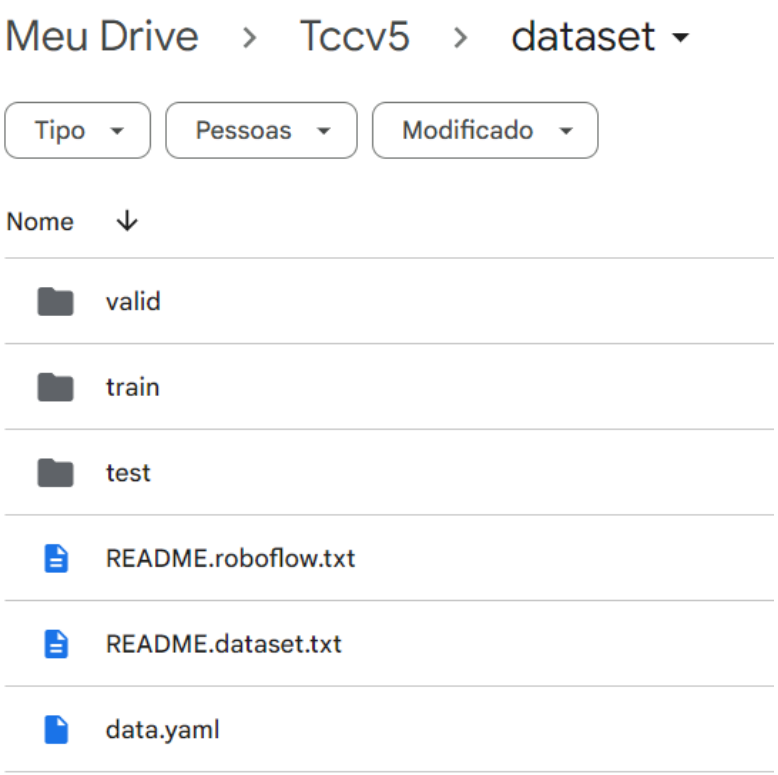
Fonte: (YoKun, 2023).

O *dataset* usado para o treinamento foi elaborado por YuKun, 2023 que disponibilizou o mesmo para utilização livre na plataforma *Roboflow*. Dentro do *dataset* a estrutura deve ser a seguinte:

- a pasta *dataset* conterá duas outras pastas **train** e **valid**;
- estas duas pastas possuem outras pastas com os nomes **images** e **labels**, na primeira ficarão as imagens e na segunda os txts respectivos as imagens;
- a pasta *dataset*, *também*, contém um arquivo com o nome **data.yaml**, que possui algumas informações importantes, como pastas de treinamento e validação da rede.

Essa organização pode ser vista na Figura 7 a seguir, e vale ressaltar que ao se replicar o projeto deve-se mudar o diretório para os nomes presentes na nova máquina a ser utilizada.

Figura 7 - Organização da pasta dataset.



Fonte: Próprio autor.

Na Figura 8 mostra-se o data.yaml, arquivo que contém as informações e diretórios usados durante o treinamento.

Figura 8 - Exemplo de data.yaml.

```yaml
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 1
names: ['rachadura']

roboflow:
  workspace: yukun
  project: crack-an2nx
  version: 3
  license: CC BY 4.0
  url: https://universe.roboflow.com/yukun/crack-an2nx/dataset/3
```

Fonte: Próprio autor.

3.2 Treinamento da Rede

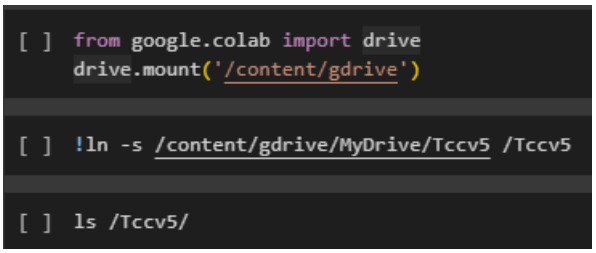
Para o treinamento da rede, algumas etapas devem ser seguidas. O primeiro passo é escolher em qual local a implementação será desenvolvida, nesse caso escolheu-se o *Google Colab*. Esta escolha dá-se pela facilidade de aplicação e de usar o poder computacional de outro equipamento, permitindo que o treinamento seja feito de forma mais efetiva do que ao usar o computador local.

Em seguida, é necessário fazer o *upload* do *dataset* para uma pasta no *drive*, organizando melhor os resultados. Essa pasta também armazenará os resultados e o modelo, facilitando e agilizando todo o processo.

3.2.1 Acesso ao Google Drive

Uma vez os arquivos armazenados no *Google Drive*, é necessário vincular o mesmo ao *Colab*, através da importação do *drive*. Através de um atalho para a pasta Tccv5 é possível gerenciar os arquivos. Este processo pode ser visto Figura 9.

Figura 9 – Comando de acesso ao google drive.

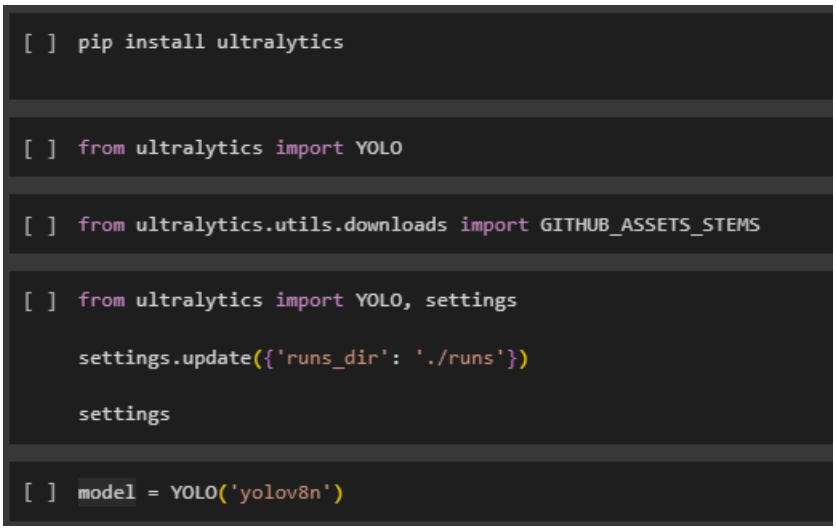


Fonte: Próprio autor.

3.2.2 Criação do Novo Modelo

O primeiro passo é instalar o *Yolov8*, importando através da ferramenta *ultralytics*. Tal processo pode ser visto na Figura 10.

Figura 10 – Comando para criar novo modelo.



Fonte: Próprio autor.

Usando os conhecimentos vistos de transferência de aprendizado, um modelo já previamente treinado foi selecionado (yolov8n), pois este modelo de rede já tem um treinamento de detecção mais generalizado, o que permite agilizar o treinamento.

3.2.3 Treinamento

A partir de um cenário de treinamento, devidamente montado, configurações iniciais de parâmetros deverão ser realizadas. O primeiro parâmetro (*data*) é fundamental para o treinamento, pois deve conter o caminho para o arquivo **data.yaml**. Para realizar o treinamento a função ***train*** deve ser executada., com os seguintes parâmetros configurados:

- ***Epochs*** (épocas): indica quantas vezes o programa executará o treinamento buscando novos pesos neste trabalho utilizou-se de 100 épocas.
- ***Patience***: indica quantas épocas delimitam a parada do treinamento, visando diminuir efeitos de *overfloating*, o qual foi configurado em 8 épocas, um valor considerado alto, pois a função ***train*** salva os melhores pesos, fazendo com que o treinamento continue a fim de buscar um melhor resultado;
- ***Batch***: indica o número de amostras de dados que passam pela rede antes que os pesos sejam atualizados, que significa o número de imagens analisadas em cada época;
- ***Imgsz***: indica o tamanho da imagem, cujo valor é dependente do seu ***dataset***, e deve ser um valor próximo a maioria das imagens, a fim de evitar distorções e divisível por

32; isto se dá, pois é o padrão da *Yolo* e o maior valor de análise na arquitetura. Neste projeto o valor 416 foi o melhor a ser configurado;

- ***worker***: indica a quantidade de *threads* que serão usados da *cpu*, como o processo é feito usando a máquina do *google colab* usou-se o valor recomendado de 8 unidades;
- ***pretrained***: um valor lógico verdadeiro (*True*) de vital importância, a fim de indicar um gatilho para início do treinamento com os pesos do modelo base. Em suma, esse comando permite a transferência de aprendizado, pois ele vai fazer com que o treinamento atual aproveite os dados já presentes no modelo;
- ***resume***: opção marcada para continuar o treinamento, caso ele não tenha chegado ao final das épocas. Isso é útil quando o modelo vai passar por muitas épocas ou possui *batchs* com muitas imagens, nesse caso acabou por não ser necessário deixar o mesmo ativo;
- ***single_*cls**: utilizado quando o objetivo é treinar somente uma classe, como nesse caso o treinamento será de somente uma classe não tem diferença ficar ativo ou não;
- ***box***: prioriza ao *bounding box* focando na detecção do contorno do objeto, ou seja, busca melhorar o enquadramento do objeto, entretanto como efeito colateral ele perde um pouco da precisão por isso não se pode usar um valor muito alto no caso o valor escolhido foi o recomendado pela biblioteca;
- **cls**: utilizado na classificação das classes no bounding box, dfl vai focar na borda dos bounidng boxes;
- **val**: fundamental para a análise, pois através dele é possível obter as informações ao longo do treinamento, ele irá realizar a validação ao final de cada época, quando ativo;
- ***argumentations:*** visam expandir o *dataset*, rotacionando, mudando a posição e a escala. Assim, o treinamento pode ser feito mesmo com um *dataset,* relativamente, pequeno. Pode assumir os valores arbitrários ou mais conservadores, neste trabalho a abordagem foi mais conservadora.

A seguir na Figura 11 é possível visualizar o código do treinamento.

Figura 11 - Código do treinamento.

```
model.train(data = '/content/gdrive/MyDrive/Tccv5/dataset/data.yaml',
            epochs=100,
            patience=8,
            batch=16,
            imgsz=416,
            workers=8,
            pretrained=True,
            resume=False,
            single_cls= True,
            box=7.5,
            cls=0.5,
            dfl=1.5,
            val=True,
            degrees=0.3,
            hsv_s=0.3,
            hsv_v=0.3,
            scale=0.5,
            fliplr=0.5)
```

Fonte: Próprio autor.

3.3 Validação e Teste

Uma vez o modelo treinado, os melhores pesos serão carregados e analisados. A fim de verificar se o modelo está operando de forma correta, o modelo deve analisar uma **imagem_controle** e marcar um *bounding box* na estrutura da rachadura.

A visualização no *Colab*, algumas vezes é falha, pois o mesmo não suporta o cv2.imshow(), que faz parte da estrutura do comando *predict* do *Yolov8*. Uma imagem da resposta será salva na pasta *predict*.

O modelo será configurado com novos pesos e será verificado se a imagem_controle está na pasta. Na sequência, a função *predict* será utilizada, o caminho da imagem deve ser informado e configurado o tamanho, a confiança mínima de detecção e a sobreposição mínima permitida entre caixas (***iou***).

Os resultados serão salvos, tanto o txt do *bounidng box* quanto a imagem com o contorno. A Figura 12 mostra o código para validação.

Figura 12 - Código da validação.

```
[ ]  model = YOLO('/content/gdrive/MyDrive/resultados/detect/train/weights/best.pt')

[ ]  import os

     file_path = '/content/gdrive/MyDrive/Tccv5/imagem_controle3.jpg'

     if os.path.exists(file_path):
         print("O arquivo de imagem existe.")
     else:
         print("O arquivo de imagem não foi encontrado no caminho especificado.")

[ ]  results_img = model.predict(source="/content/gdrive/MyDrive/Tccv5/imagem_controle6.jpg",
                                 conf=0.25,
                                 iou=0.7,
                                 imgsz=416,
                                 show=True,
                                 save=True,
                                 save_txt=True,
                                 save_conf=True,
                                 save_crop=True,
                                 stream=False)

[ ]  import shutil

     # Define os caminhos para a pasta de origem e a pasta de destino
     pasta_origem = '/content/runs'
     pasta_destino = '/content/gdrive/MyDrive/resultados4'

     # Copia a pasta de origem para a pasta de destino
     shutil.copytree(pasta_origem, pasta_destino)
```

Fonte: Próprio autor.

4 RESULTADOS E DISCUSSÕES

Uma vez realizado o treinamento, o programa retorna arquivos de dois pesos, sendo esses os últimos pesos e os melhores pesos, que não necessariamente serão os mesmos. Nesse caso os melhores pesos se encontraram na época 49, nele encontra-se um **MAP** (*Mean Average Precision*) de 0,65942, o que pode variar de treinamento para treinamento.

É importante destacar que, não necessariamente o último peso vai ser o melhor, pode ocorrer um *overfloating*. Um *overfloating* significa que o treinamento da rede chega a um determinado pico e depois disso, seu resultado começa a cair. Por conta disso, é importante sempre guardar os pesos que obtiveram o melhor resultado em cada treinamento. No caso apresentado a rede neural foi até a época 68, parando nesta, pois não evoluía há 8 épocas.

4.1 Matriz Confusão

O objetivo de uma matriz confusão é auxiliar a análise do aprendizado de máquina, ela fornece uma visão detalhada de como as classes estão sendo classificadas. Esta matriz indica se o que foi analisado se enquadra como um **verdadeiro positivo**, **falso negativo**, **falso positivo** ou **verdadeiro negativo**.

- **Verdadeiro Positivo**: o objeto foi detectado como uma rachadura e realmente é uma rachadura;
- **Falso Positivo:** o objeto não é uma rachadura, mas foi detectado como tal;
- **Verdadeiro Negativo:** o objeto não é uma rachadura, e não foi detectado como rachadura;
- **Falso Negativo:** o objeto é uma rachadura, e não foi detectado como rachadura.

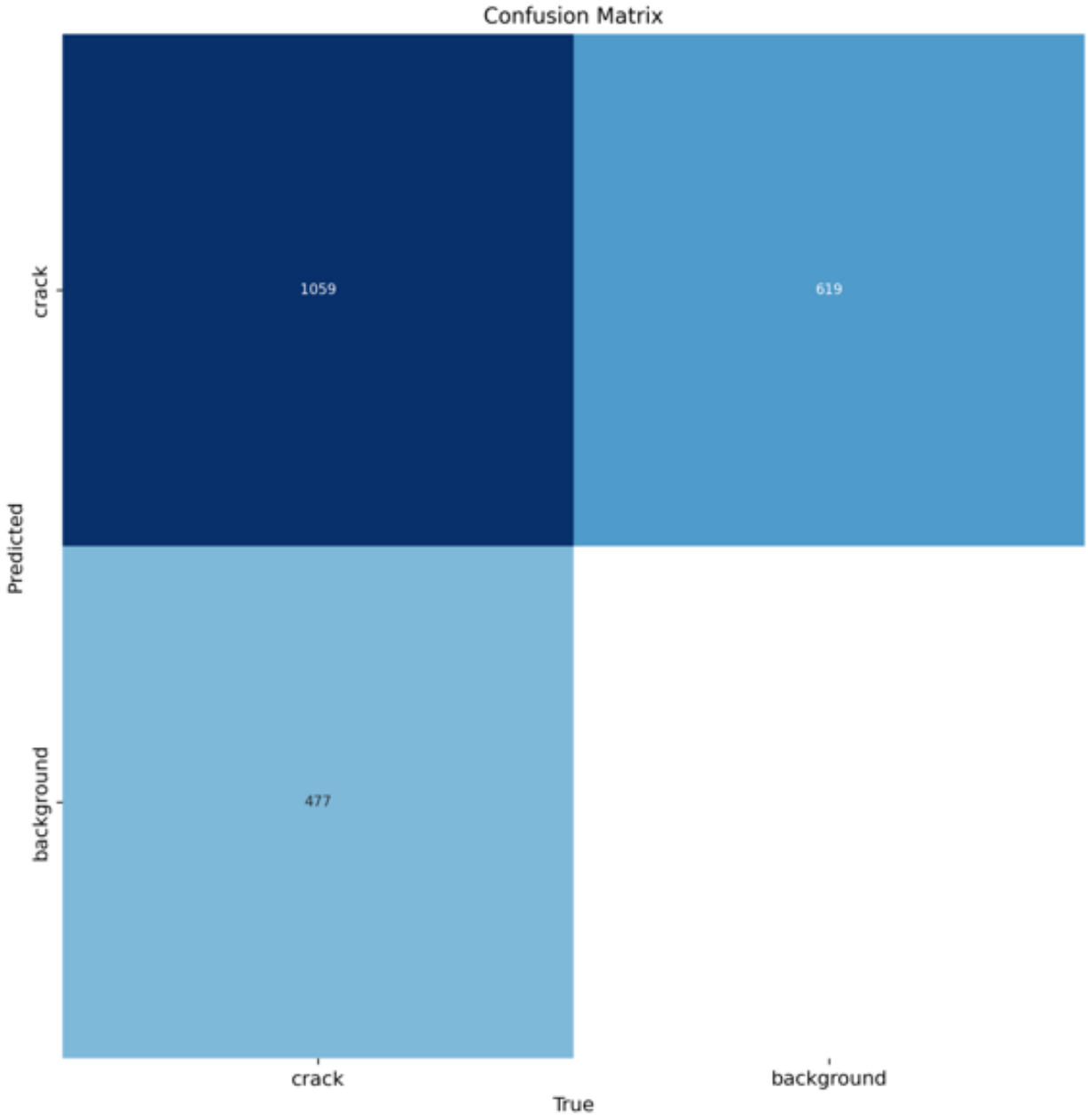
Por exemplo, uma imagem que possui uma rachadura e que foi analisada, resultando em uma rachadura é um **verdadeiro positivo**. No entanto, quando da análise ocorre um **falso positivo** ou **verdadeiro negativo**, significa que o programa está tendo dificuldade em reconhecer a imagem. Os resultados analisados se encontram na Figura 14.

Figura 13 - Matriz confusão genérica.

		Valor predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo	Falso Negativo
	Não	Falso Positivo	Verdadeiro Negativo

Fonte: Próprio autor.

Figura 14 - Matriz confusão com número de rachaduras.



Fonte: Próprio autor.

Analisando as Figuras 13 e 14, identificou-se que 0.69 (69%) das imagens que possuíam rachadura foram analisadas corretamente (**verdadeiro positivo**), entretanto no *dataset* utilizado, nenhuma imagem possuía ausência total de rachadura, mesmo que fossem muito pequenas ou finas. Isso explica o porquê dá ausência de **falsos negativos**, pois não existem imagens que não tenham rachadura. Isto acarreta a presença de **falsos positivos**, o que significa que o programa reconheceu alguma parte da imagem que não erá rachadura como sendo rachadura.

A partir destes resultados, pode-se concluir que os dados utilizados estavam desbalanceados, portanto, ele encontra dificuldade com a ausência de rachaduras. Isto é justificado pelo treinamento da rede, a qual foi treinada somente com a classe rachadura, sem uma classe não rachadura. Embora, entende-se que treinar com a classe rachadura criasse um critério de não rachadura, isso não justifica pois o programa está acostumado a procurar rachaduras e não a procurar não rachaduras, o que leva a equívocos de **falsos positivos**.

4.2 Curva F1

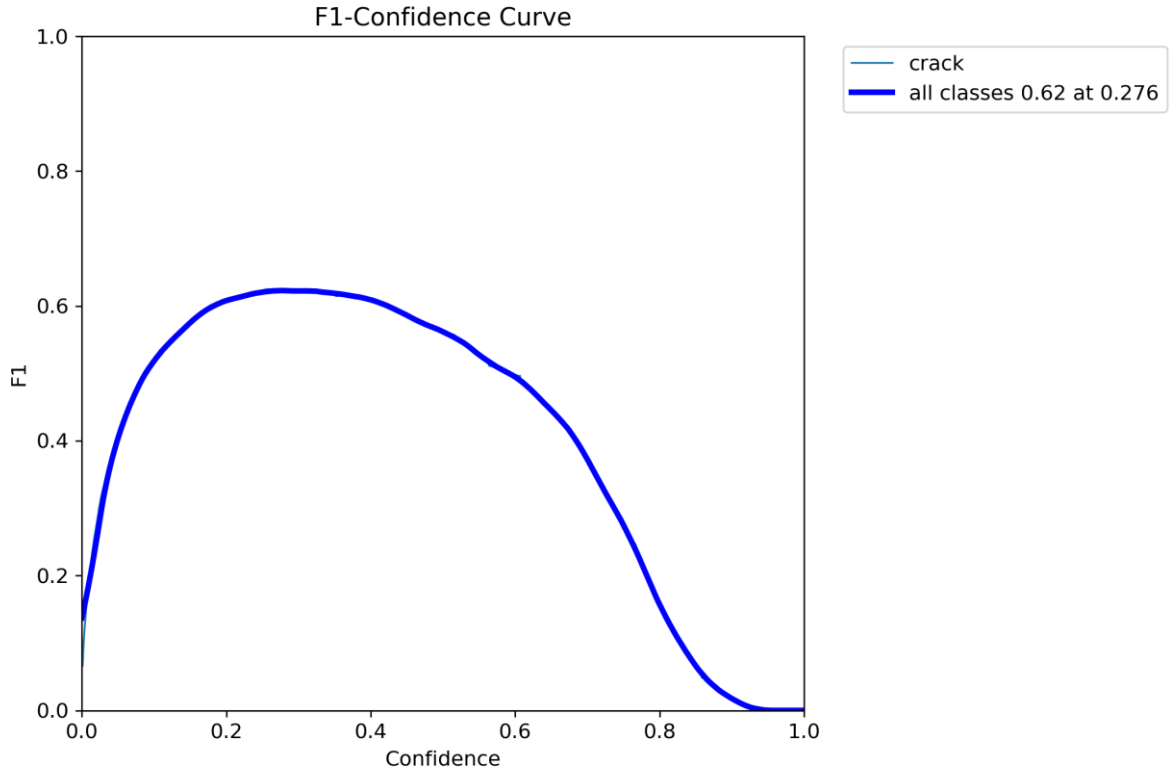
O objetivo da curva F1-Score é apresentar o desempenho de um modelo de classificação neste caso uma rede neural, considerando a precisão da detecção e revocação conforme se altera a confiança.

- **Precisão** é a capacidade da rede detectar o objeto e ser da classe correta. Podendo ser colocado como Verdadeiros positivos / (Verdadeiros Positivos + Falsos Positivos);
- **Revocação** é a capacidade de não deixar objetos sem serem detectados. Podendo ser colocado como Verdadeiros positivos / (Verdadeiros Positivos + Falsos Negativos);
- **F1-Score** média harmônica entre precisão e recall**.** Podendo ser calculado como 2*Precisão*Revocação/ (Precisão + Revocação)

Por exemplo, caso a confiança pedida seja alta, e a chance de o objeto realmente estar correto, quando ele for selecionado, existe uma chance maior de deixar objetos, que sejam da classe, de fora. Isto se dá, por ter um revocação e certeza baixas.

Por conta disso, um equilíbrio é importante, pois não se quer que a análise esteja errada, mas, também, não se quer a rede fique sem identificar objetos. A curva da Figura 15 representa a curva F1-Score do modelo montado.

Figura 15 - Curva F1-Score.



Fonte: Próprio autor.

No caso apresentado, o ponto ótimo se encontra em torno de 0,276 de confiança quando a curva F1 das classes se encontra em 0,62 de precisão o que se torna o **maior** valor, levando-se em consideração precisão e revocação. **Não significa** que esse é o **melhor** valor a ser usado, depende do objetivo da aplicação. Caso o objetivo seja uma ação pontual, somente quando se tem certeza, é melhor buscar confianças maiores. Caso se esteja buscando qualidade ao invés de quantidade, pode ser mais interessante, ter esse valor, pois terá uma precisão e revocação mais equilibradas.

4.3 Gráfico *Labels*

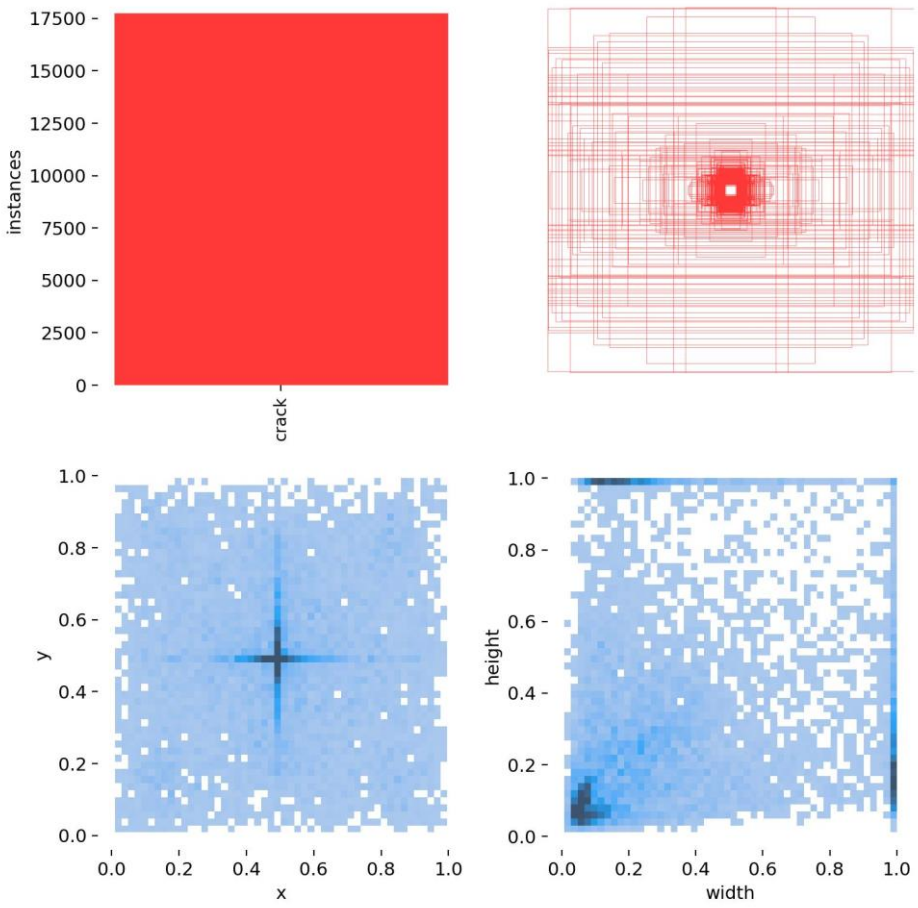
O gráfico Labels auxilia a visualizar a distribuição das classes ou rótulos de um *dataset*. No caso, como só existia uma classe sendo treinada, rachadura, não haverá outras classes sendo alocadas nesse treinamento. Portanto, não é necessário ter uma classe com rachadura e

uma sem para o treinamento, não significa que os resultados serão melhores ou piores com a presença de outra classe, significa que ele não necessita para poder ser treinado.

Por exemplo, em um caso em que se quer identificar quais pontos estão com rachadura e quais não estão, esse tipo de treinamento é essencial. Neste trabalho, o objetivo era treinar a detecção de rachaduras, não havendo necessidade de se preparar um novo *dataset* para realizar o treinamento sem rachadura. Ambas as formas têm seus prós e contras, pois ao se utilizar de duas classes, tem-se um *dataset* consideravelmente maior e que deve estar devidamente anotado em seus txts. Isto acarreta a um treinamento consideravelmente mais demorado, porém incrementa a qualidade da detecção.

Deve-se ter em vista que os Falsos Negativos são o maior problema, pois significa que uma rachadura não foi identificada. No tipo de análise adotada isso significa que um potencial risco passou despercebido, sendo esse atualmente o principal ponto de melhoria a ser corrigido com o tempo. Na Figura 16 pode-se visualizar o gráfico *Labels.*

Figura 16 - Gráfico Labels (A).

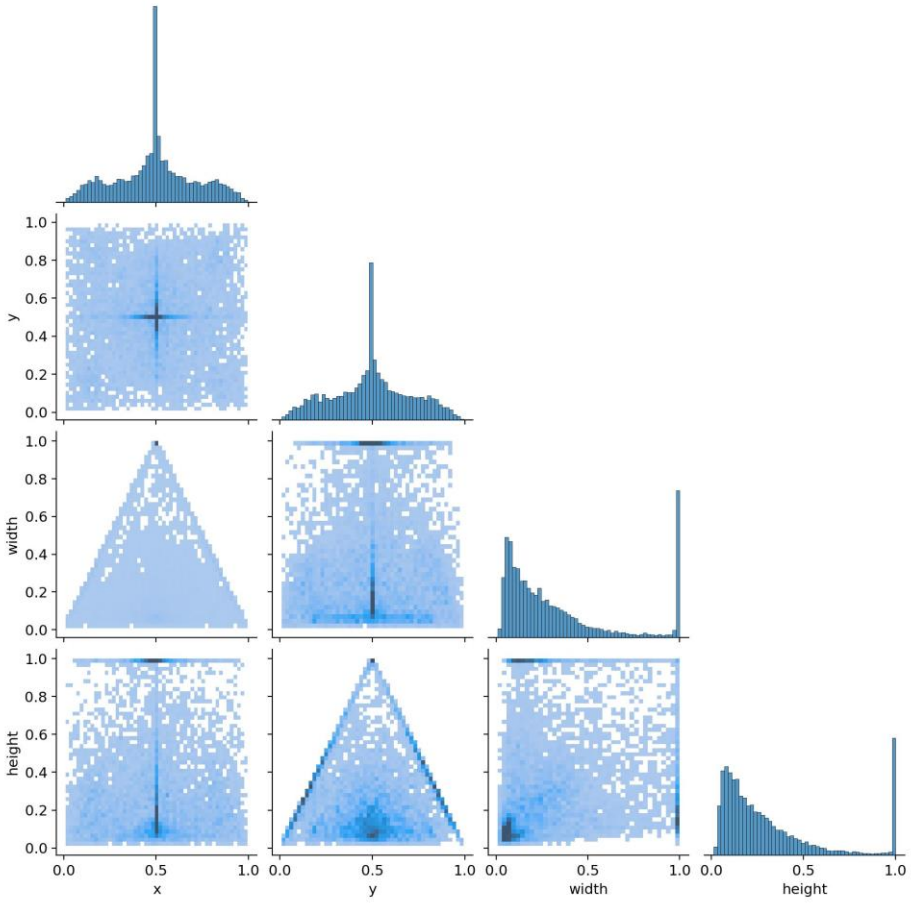


Fonte: Próprio autor.

Além de apresentar a presença, predominante, de uma classe, esse gráfico fornece mais algumas informações a respeito do nosso banco de dados. Se observar o gráfico no plano cartesiano XY, identifica-se a concentração de anotações que possuem posições como referência. É possível notar que a maior parte das rachaduras analisadas foram encontradas centralizadas na imagem, criando um padrão de detecção no programa. Suas melhores análises serão de imagens que, também, tenham suas rachaduras centralizadas. Tal circunstância, não impede a detecção em outros pontos, apenas cria uma melhor detecção nesse setor.

O mesmo vale para o gráfico de *Height* por *width*, o qual indica, em média, o tamanho relativo de uma rachadura na imagem. Uma alta concentração nas bordas pode criar um padrão do tamanho médio de uma rachadura, onde conjuntos pequenos de rachaduras podem ser englobados em uma mesma rachadura ou uma muito grande pode ser dividida em mais de uma rachadura.

Figura 17 - Gráfico Labels (B).



Fonte: Próprio autor.

A Figura 17 possui visão mais organizada, a partir das *Labels* apresentadas anteriormente. Nela há a separação do comportamento de cada uma das possíveis variáveis do *bounidng box*, x, y, *height*, *width*. Nessa analise o fenômeno da centralização se torna mais nítido, onde em todos os gráficos de x e y, os pontos que terão maior destaque se encontram no centro do gráfico.

4.4 Curva P

Curva de P indica qual foi a precisão do modelo com base em sua confiança.

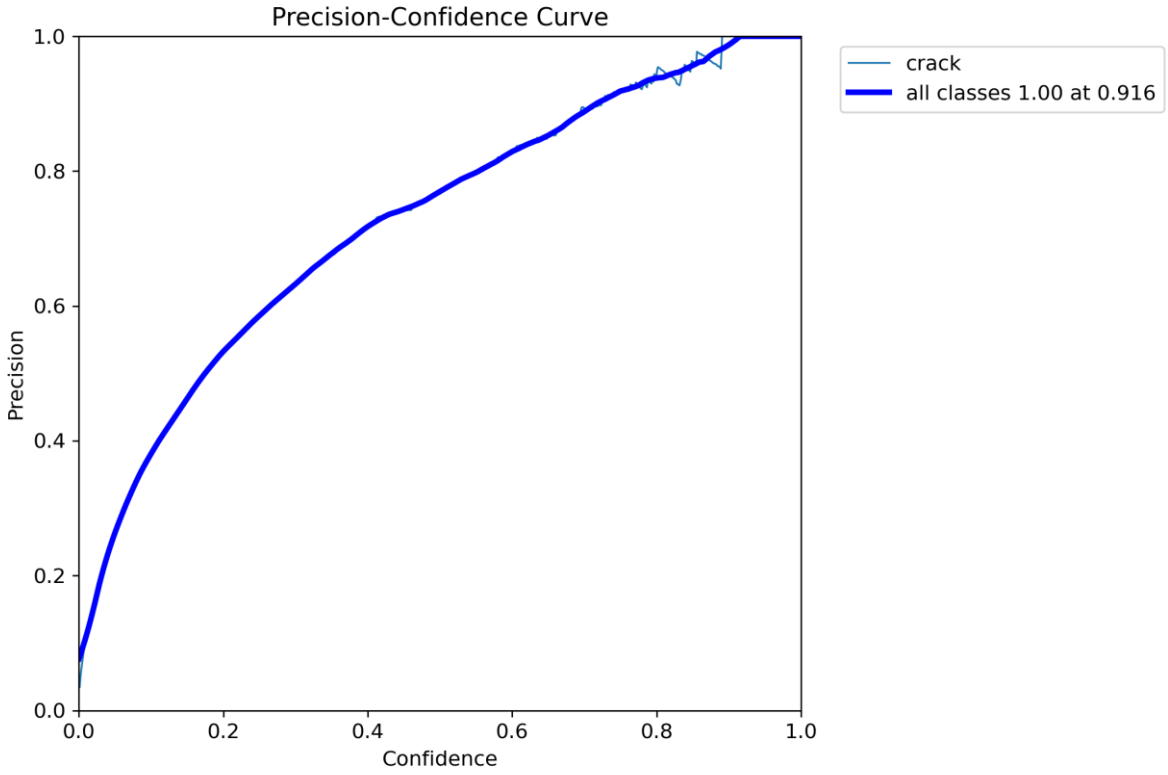
- **Precisão** indica a taxa de previsões corretas em relação ao total de previsões positivas, ou seja, Verdadeiros positivos/(Verdadeiros positivos + Falsos positivos).
- **Confiança** é o grau de certeza associado a uma previsão, ou seja, o quão seguro o modelo está de sua previsão.

Assim, a curva P indicará o quão, realmente, certo o programa estará baseado na confiança de sua detecção. Isso é útil para compreender se o valor previsto por ele se aproxima da realidade. Analogamente, se meu modelo tem confiança 0,5 e precisão de 0,3, significa que quando ele tem 50% de confiança, que a resposta dada por ele está certa, na realidade, essa probabilidade de estar certa poderia ser dita como 30%.

Portanto, no modelo apresentado, a análise cresce, muito rapidamente, com sua confiança no início, e no final esse fator desacelera. Isto pode acontecer, pois o aumento da precisão e da confiança, embora correlacionados, não são diretamente proporcionais.

Uma vez que a análise atinge um determinado nível de precisão, é necessário muito mais esforço para que o modelo possa reconhecer, incrementando esta precisão, já que quanto mais evoluído ele se torna é mais difícil refinar o processo. Analogamente, pode se imaginar uma camiseta branca, e ao jogar um pouco de terra nela, ela ficará um pouco suja, e a medida que se repete o processo de sujar, chegará a um momento em que está tão suja que novas repetições não serão mais perceptíveis. É por isso que a taxa de crescimento da precisão tende a aumentar conforme a confiança aumenta, ao longo do tempo baseado em seus treinamentos. Como pode ser visto na Figura 18 depois de 0,916 de confiabilidade a precisão será praticamente igual a 1 em todos os casos.

Figura 18 - Curva precisão.



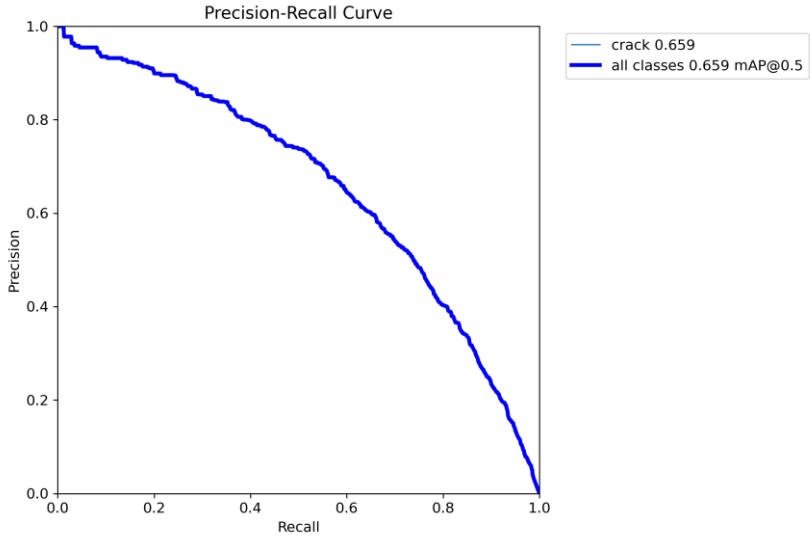
Fonte: Próprio autor.

4.5 Curva Pr

A curva Pr indica a **precisão** em relação a **sensibilidade**, também nomeada de ***recall***. A **precisão** possui o mesmo significado, explicado anteriormente. E a **sensibilidade** representa a proporção de exemplos positivos, do conjunto de dados, que foram corretamente identificados pelo modelo, ou seja, Verdadeiros positivos/(Verdadeiros positivos + Falsos negativos).

O comportamento visualizado na Figura 19 é considerado normal, conforme o *recall* aumenta a precisão diminui. O objetivo deste é encontrar um equilíbrio, onde nenhum caso positivo seja perdido, mas que a análise não seja incorreta, auxiliando, assim a escolher um limiar de confiança para o modelo trabalhar.

Figura 19 - Curva Pr.

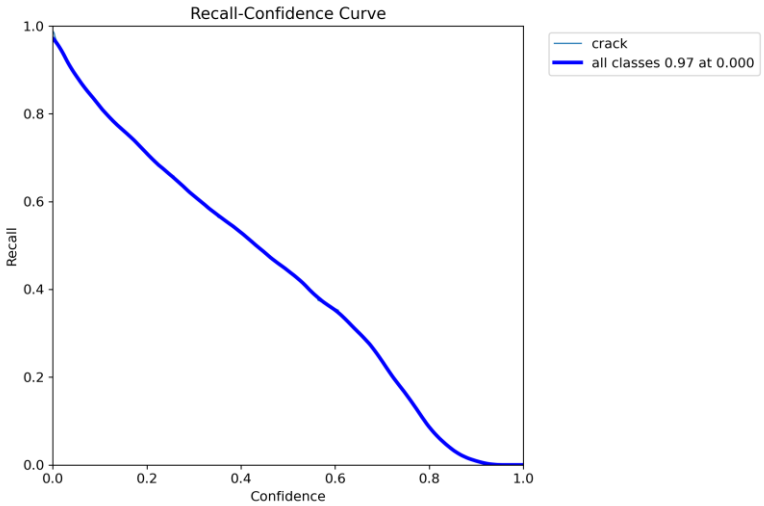


Fonte: Próprio autor.

4.6 Curva R

O objetivo dessa curva é apresentar o comportamento do ***recall*** em função da **confiança**, auxiliando no entendimento do *trade-off* entre a capacidade do modelo de capturar exemplos positivos e a confiança nas previsões. Por isso eles tendem a ser inversamente proporcionais, uma vez que quanto maior a confiança o modelo se torna mais seletivo e faz previsões quando tem certeza. Na Figura 20, o comportamento foi conforme esperado.

Figura 20 - Curva R.



Fonte: Próprio autor.

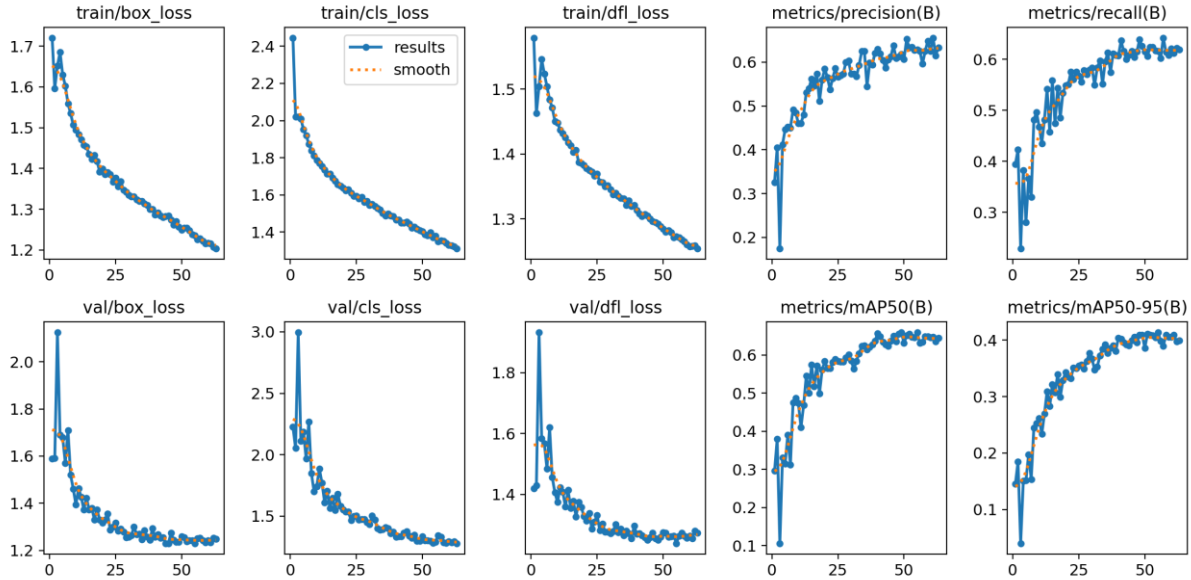
4.7 Resultados em Função de Épocas

Alguns gráficos foram gerados, a fim de visualizar o treinamento em cada época, permitindo obter uma grande série de informações. Essas informações são obtidas a cada época, pois na configuração do treinamento foi definido validação a cada época. Como apresentado anteriormente, embora tenha sido configurado 100 época, a execução não foi completa pelo efeito de *overfloating* que esteve presente nas últimas 8 épocas que acarretou no encerramento do processo.

Os gráficos contêm os resultados e podem ser visualizados através de uma curva "*smooth"*, a qual suaviza as variações bruscas. Essas variações são muito presentes nas primeiras épocas, por serem as primeiras fases do treinamento e os pesos mudam com muita intensidade. À medida que o treinamento vai se refinando, a curva se torna mais suave, já que as variações se tornam menores. Isso acontece, principalmente, pela decisão de se adotar uma abordagem conservadora.

Esta abordagem com uma taxa de variação mais baixa, visa a diminuição do efeito *overfloating*, o qual ocorre em treinamentos extensos. Uma rede mais conservadora evolui mais devagar, mas tem menos riscos de regredir sua qualidade de detecção.

Figura 21 - Gráficos dos resultados em função das épocas.



Fonte: Próprio autor.

Dentre os parâmetros apresentados, o de maior importância é o ***metrics/mAp50***, também, conhecido como **MAP** (*Mean Average Precision*), ou seja, precisão média do modelo. Neste trabalho o melhor resultado obtido foi 0,65942, o qual poderá ser analisado futuramente sobre o seu uso, pois até o presente momento, não existe uma legislação ou norma a respeito de um controle de qualidade de redes neurais na área de detecção de falhas.

Entretanto, há algumas suposições que podem ser feitas, pois os índices mais exigentes de MAP se encontram na área da saúde e de veículos autônomos, onde podem alcançar até 0,9 de exigência. Tais exigências estão relacionadas ao nível de criticidade do sistema, pois as consequências de não se garantir precisão e confiabilidade podem trazer prejuízo a estruturas e vidas, considerando resultados falsos negativos como os mais perigosos.

Analogicamente, um falso positivo na medicina coloca um paciente em um tratamento desnecessário e que pode colocar risco a sua saúde, um falso positivo na rachadura somente faria com que o responsável da manutenção tivesse que cobrir uma área onde não existe uma rachadura. No entanto, um falso negativo poderia comprometer tanto a vida de uma pessoa em estar doente e não receber o devido tratamento, ou ainda de muitas se uma estrutura de uma ponte, por exemplo, estiver comprometida.

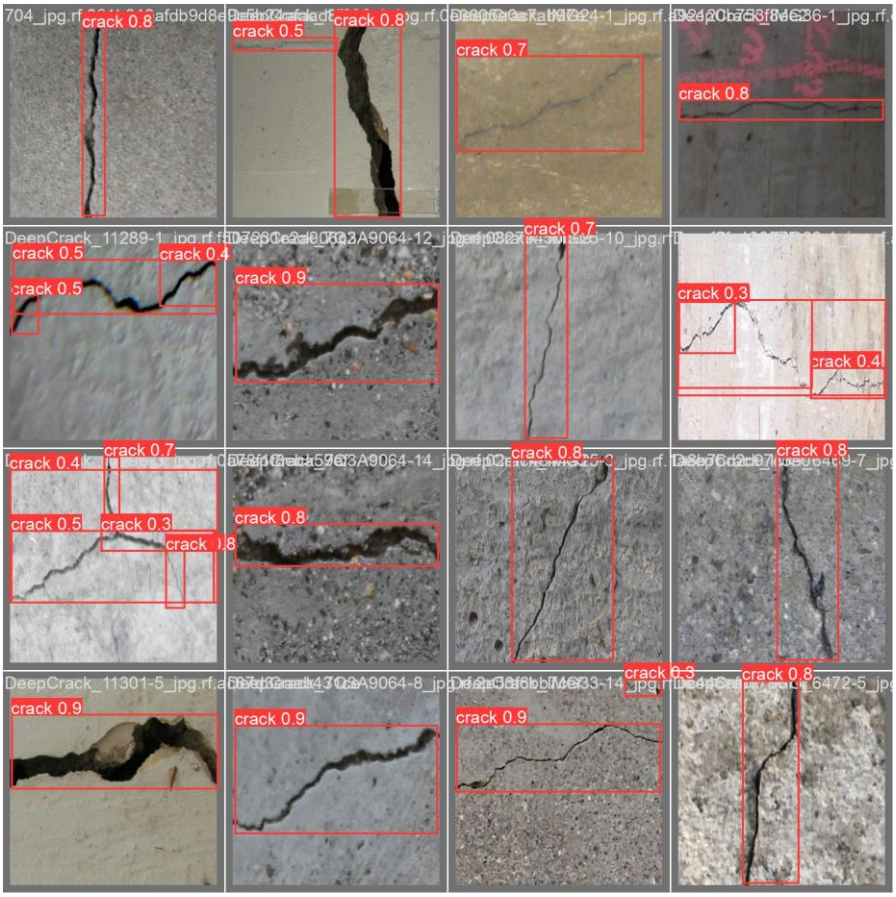
Este tipo de abordagem, com certeza, é dependente da criticidade, custo e benefícios. Para isso, normas ou legislações específicas são imprescindíveis.

Neste trabalho, não foram encontrados, na literatura, motivos para não se aceitar um valor de 0,65942 como um resultado aceitável para realizar pelo menos a triagem inicial de uma análise.

4.8 Visualização de Alguns Treinamentos

Alguns treinamentos foram realizados e podem ser visualizados nas Figuras 22, compostas de imagens e suas análises de detecção de objetos. Estas mostram o comportamento de uma rede e os resultados esperados ao final da análise de uma imagem. Imagens de controle foram adicionadas visando checar a qualidade do modelo em situações atípicas do treinamento, como a presença de uma rachadura com perfil mais largo, rachaduras com objetos em perspectiva, com sobras, com maiores distâncias e com a presença de outros objetos no cenário.

Figura 22 - Imagens marcando classes e sua confiança.



Fonte: Próprio autor.

Figura 23 - Imagem sem nenhuma rachadura.

Fonte: Próprio autor.

Figura 24 - Imagem com rachadura e um canto inclinado.

Fonte: Próprio autor.

Figura 25 - Imagem com rachadura, mas muito distante.

Fonte: Próprio autor.

Figura 26 - Imagem idealizada.

Fonte: Próprio autor.

Como pode se ver pelas imagens, o comportamento quando a rachadura está em destaque é o idealizado, no entanto, quando existem variações muito grandes, que fogem do *dataset* usado, a qualidade da detecção é muito comprometida. Isso não inviabiliza a utilização da classe, entretanto é necessário ter em mente que a imagem deve ser nítida e clara, além disso objetos distintos podem fazer a qualidade da detecção cair.

Um *dataset* com objetos variados tende a reduzir problemas de detecção. É interessante ressaltar que, isto não significa que a detecção será perfeita, uma vez que o campo da ciência visual ainda possui muitos pontos a serem melhorados, e um deles é a variação de cenário com uma boa estabilização de resultados. Imagens mais simples e com boa qualidade, a classe possui uma boa confiança tendo um MAP de 0,65942, podendo chegar a confianças durante as análises de até 0,9. Isso indica detecção no cenário apresentado, o que poderia ser usado como um processo de triagem, sua utilização seria ainda melhor em pontos estratégicos com pouca movimentação.

Por exemplo, se usar um drone para obter imagens de uma ponte, existiriam poucos objetos adversos que poderiam atrapalhar a detecção de rachaduras, o que tornaria essa rede, extremamente, útil nesses casos. No entanto, se analisar imagens provenientes de ambiente

movimentado, onde muitas pessoas estão passando, o processo seria mais difícil, uma vez que objetos adversos atrapalhariam a detecção.

5 CONSIDERAÇÕES FINAIS

A partir do treinamento da rede, o programa retornou alguns dados e predições importantes. Estes resultados foram analisados, passo a passo, chegando a algumas conclusões.

Em um cenário idealizado, a detecção por meio da rede neural modelada é de grande valia. No entanto, muitas dificuldades foram encontradas ao se variar o cenário, desde a não detecção para rachaduras muito distantes ou pequenas nas imagens, quanto a detecção de objetos estranhos ao treinamento que podem se passar por rachaduras.

Em geral, ao se utilizar a rede modelada, da forma como ela se encontra atualmente, seria possível em situações controladas, como locais sem muito movimento e com a imagem nítida das falhas. Portanto para tornar a mesma realmente eficiente e apta a uso pode-se melhorar a qualidade diversidade das imagens utilizadas no treinamento.

Em trabalhos futuros, pretende-se realizar um melhoramento da rede, com treinamentos mais elaborados e com *datasets* mais diversificados, a fim de melhorar em sua detecção. É valido ressaltar que, mesmo com esses novos treinamentos, a rede ainda não seria capaz de ultrapassar a precisão de detecção humana. No entanto, com a velocidade da análise poderia se cobrir uma maior quantidade de imagens para serem analisadas, e as que tiverem um resultado mais crítico seriam filtradas para a análise de um especialista.

Assim, é possível concluir que é possível que uma rede neural consiga analisar falhas mecânicas. Entretanto ainda é nítido que com um map de 0,65942 ela ainda deve possuir um treinamento muito mais diversificado e refinado para poder ser usada em uma situação real. Portanto ainda existe um caminho considerável para a automatização de análises, mas o objetivo mostra-se possível.

REFERÊNCIAS

CARVALHO, André Ponce de Leon F. de. Redes Neurais Artificiais. Disponível em: https://sites.icmc.usp.br/andre/research/neural/. Acesso em: 31 out. 2023.

TAFNER, Malcon Anderson. **O Que São as Redes Neurais Artificiais**. Disponível em: https://cerebromente.org.br/n05/tecnologia/rna_i.htm. Acesso em: 31 out. 2023.

ACADEMY, Data Science. Deep learning book. Disponível em: https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/. Acesso em: 31 dez. 2023.

ALVES, Gabriel. Detecção de Objetos com YOLO – Uma abordagem moderna. Disponível em: https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/. Acesso em: 31 out. 2023.

YUKUN. Crack Computer Vision Project. Disponível em: https://universe.roboflow.com/yukun/crack-an2nx. Acesso em: 1 nov. 2023.

(2011). Computer Vision: A Modern Approach. Prentice Hall. ISBN 978-0136085928.

REDMON, Joseph Chet. Home. Disponível em: https://pjreddie.com/. Acesso em: 1 nov. 2023.

HAYKIN, Simon. **Redes neurais princípios e prática**. Porto Alegre: Bookman, 1999.