UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TRANSPORTES

CURSO DE ENGENHARIA CIVIL

LUCIANA DA SILVA NASCIMENTO

APLICAÇÃO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO AUTOMÁTICA DE TRINCAS EM PAVIMENTOS

FORTALEZA

2022

LUCIANA DA SILVA NASCIMENTO

APLICAÇÃO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO AUTOMÁTICA DE TRINCAS EM PAVIMENTOS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Civil do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Engenheira Civil.

Orientador pedagógico: Prof. Dr. Ernesto Ferreira Nobre Júnior

FORTALEZA-CE

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

N196a Nascimento, Luciana da Silva.
Aplicação de aprendizagem profunda para detecção automática de trincas em pavimentos / Luciana da Silva Nascimento. – 2022.
96 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Civil, Fortaleza, 2022.
Orientação: Prof. Dr. Ernesto Ferreira Nobre Júnior.

1. Redes Neurais Convolucionais. 2. Defeitos no pavimento. 3. Detecção de trincas. 4. YOLO. 5. Gerenciamento de Pavimentos. I. Título.

CDD 620

LUCIANA DA SILVA NASCIMENTO

APLICAÇÃO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO AUTOMÁTICA DE TRINCAS EM PAVIMENTOS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Civil do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Engenheira Civil.

Aprovada em: 11/02/2022.

BANCA EXAMINADORA

_____

Prof. Dr. Ernesto Ferreira Nobre Júnior
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Aline Calheiros Espindola
Universidade Federal de Alagoas (UFAL)

_____

Prof. Dr. Mário Ângelo Nunes de Azevedo Filho
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Aos meus pais, Ana Célia da Silva Nascimento e Wilson Gomes do Nascimento, por todo carinho e apoio endereçados a mim e a meus irmãos ao longo de todos esses anos, e por nos encorajarem na realização de nossos sonhos, independente da dificuldade que iríamos enfrentar.

Ao meu orientador, Prof. Dr. Ernesto Ferreira Nobre Júnior, que mesmo em tempos difíceis de uma pandemia, se dispôs a me auxiliar, com muita paciência e muitos ensinamentos.

À professora Me. Aline Calheiros Espíndola, pela paciência de escutar minhas dificuldades, pelos incentivos, pelas ideias, por ler meus trabalhos incontáveis vezes até eu me sentir segura de enviar, por me tranquilizar quando eu achava que nada estava dando certo, por me ajudar a trilhar o caminho do método científico.

Aos membros da banca avaliadora, Prof. Dr. Mário Ângelo Nunes de Azevedo Filho e Prof. Me. Aline Calheiros Espíndola, por terem aceitado participar de minha defesa, apesar de suas obrigações pessoais e profissionais.

À empresa A, por ceder as imagens utilizadas na presente pesquisa, e à empresa B, por todo o aprendizado adquirido ao longo do estágio, incluindo cursos e conversas, pois sem isso não teria sido possível a realização deste trabalho.

Ao corpo docente da Universidade Federal do Ceará por todos os ensinamentos ao longo dos anos de graduação, principalmente ao Prof. Dr. Silvrano Adonias Dantas Neto, que, mesmo sem saber, foi o motivo de eu não ter desistido da faculdade no quarto semestre, com sua paixão e entusiasmo pela profissão que exerce e seu carinho pelos estudantes.

Aos amigos que a vida me proporcionou, Renato Gadelha, Liana Barbosa, Gabriela Gabriel, Antônio Nickson, Letícia Essabbá, Livia Maria, Ivan Acioli, Jonatas Medeiros, Igor Lira, Victor Abreu, Everton Jhons e todos os demais que não pude citar, mas que fizeram parte da minha história, que me deram forças nos momentos mais difíceis, sempre dispostos a me ajudar, a oferecer um ombro amigo, a enxugar minhas lágrimas, a me fazer sorrir.

Ao meu melhor amigo e grande amor, Felipe Alison Costa Alves, por encarar todas as aventuras da vida comigo, por escutar pacientemente todos os meus anseios e entender todas as minhas crises de ansiedade e choro, por tentar me fazer uma pessoa melhor e mais saudável e por auxiliar em algumas etapas deste trabalho.

“Eu não vim até aqui para desistir agora”.

(Humberto Gessinger)

RESUMO

A inventariação dos defeitos do pavimento é de extrema importância pois permite aos órgãos responsáveis planejar ações de manutenção e reparação no pavimento de modo a permitir o nível adequado de eficiência e qualidade necessários para o pleno desenvolvimento das atividades econômicas e sociais no país. No entanto, esse processo de coleta de dados pode se mostrar bastante moroso e oneroso. Sendo assim, algumas técnicas de visão computacional podem auxiliar na automatização desse processo de coleta, possibilitando ainda uma redução no grau de subjetividade de algumas avaliações. Sendo assim, a presente pesquisa tem por objetivo apresentar uma proposta de detecção automatizada das trincas no pavimento, tendo em vista que este defeito superficial evolui rapidamente para buracos, que é o defeito mais crítico para a segurança viária. Utilizou-se para essa atividade o algoritmo You Only Look Once (YOLO), em suas duas versões atuais mais estáveis, que são YOLOv3 e YOLOv4, sendo que este último foi o que apresentou melhores resultados, com dimensão da imagem de 416x416 pixels e 8000 iterações, cujas métricas gerais obtidas foram: F1-score de 0,49, IoU de 41,08% e mAP@0.50 de 47,71%.

Palavras-Chave: Redes Neurais Convolucionais; Defeitos no pavimento; Detecção de trincas; YOLO; Gerenciamento de Pavimentos

ABSTRACT

The inventory of pavement defects is extremely important because it allows the responsible organizations to plan maintenance and repair actions on the pavement in order to allow the adequate level of efficiency and quality necessary for the full development of economic and social activities in the country. However, this data collection process can prove to be quite time consuming and costly. Therefore, some computer vision techniques can help to automate this collection process, also allowing a reduction in the degree of subjectivity of some evaluations. Therefore, the present research aims to present a proposal for automated detection of cracks in the pavement, considering that this surface defect rapidly evolves into holes, which is the most critical defect for road safety. The You Only Look Once (YOLO) algorithm was used for this activity, in its two most stable current versions, which are YOLOv3 and YOLOv4, the latter being the one that presented the best results, with an image dimension of 416x416 pixels and 8000 iterations, whose general metrics were: F1-score of 0.49, IoU of 41.08% and mAP@0.50 of 47.71%.

Keywords: Convolutional Neural Network; Pavement Defect; Crack Detections; YOLO; Pavement Management

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AASHTO	*American Association of State Highway and Transportation Officials*
AI	*Artificial Intelligence* – Inteligência artificial
AP	*Average Precision* – Precisão média
CNT	Confederação Nacional dos Transportes
CNN	*Convolucional Neural Network* – Redes Neurais Convolucionais
DCNN	*Deep Convolutional Neural Network* – Redes Neurais Convolucionais Profundas
DL	*Deep Learning* – Aprendizado Profundo
DNIT	Departamento Nacional de Infraestrutura de Transportes
FPN	*Feature Pyramid Network* – Rede de Pirâmide de Recursos
FPS	*Frames per second* – Quadros por segundo
GPU	*Graphics Processing Units*
ICM	Índice de Condição de Manutenção
ICPF	Índice de Condição de Pavimentos Flexíveis
IGG	Índice de Gravidade Geral
IRI	*International Roughness Index* – Índice de Irregularidade Internacional
IoU	*Intersect over Union* – Interseção sobre a união
LVC	Levantamento Visual Contínuo
LSTM	*Long Short-Term Memory*
mAP	*Mean Average Precision* – Média das precisões de todas as métricas
ML	*Machine Learning* – Aprendizado de Máquina
NN	*Neural Networks* – Redes Neurais
PCI	*Pavement Condition Index* – Índice de Condição do Pavimento
R-CNN	*Region-Based Convolutional Neural Networks*
RNA	Redes Neurais Artificiais
ROI	*Region Of Interest* – Região de Interesse
SGP	Sistema de Gerência de Pavimentos
SSD	*Single Shot MultiBox Detector*
SVM	*Support Vector Machines* – Máquina de Vetor de Suporte
UFC	Universidade Federal do Ceará
VSA	Valor de Serventia Atual
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

A disponibilidade de sistemas de transporte com elevados níveis de eficiência e qualidade auxilia, juntamente com outros fatores, no pleno desenvolvimento das atividades econômicas e sociais que acontecem no país, seja referente aos serviços ofertados ou à infraestrutura da região. Segundo a pesquisa da Confederação Nacional de Transporte (CNT, 2021), o modal rodoviário é de grande importância pois possui a maior participação na matriz de transportes, concentrando aproximadamente 65% da movimentação de mercadorias e 95% da de passageiros, além de estar presente nas etapas iniciais e/ou finais das cadeias de transporte, considerando-se um contexto multimodal.

Devido à relevância do modal rodoviário no transporte brasileiro, espera-se que a malha viária existente seja suficiente para atender às necessidades do país e que apresente condições de trafegabilidade adequadas para o desenvolvimento das atividades. No entanto, as pesquisas CNT de rodovias efetuadas ao longo dos anos têm evidenciado a reduzida disponibilidade de rodovias pavimentadas (quando comparado com a extensão total do país), além de não apresentar adequado estado de conservação nas rodovias existentes.

O sistema de manutenção nas vias existentes não acompanhou o incremento das solicitações proveniente do crescimento da frota, fazendo com que houvesse um processo de desgaste e surgimento de defeitos de forma intensificada. A avaliação qualitativa dos pavimentos segundo a CNT (2021) mostra que 52,2% da malha apresenta algum tipo de defeito, sendo 30,6% classificando-se como regular, 15,8% como ruim e 5,8% como péssimo.

O estado de conservação do pavimento também está diretamente associado aos custos operacionais e ao aumento do risco da ocorrência de acidentes (CNT, 2021). A má condição da superfície de rolamento das rodovias, com a presença de afundamentos, ondulações e/ou buracos, contribui para a instabilidade do veículo e, consequentemente, para a dificuldade em mantê-lo na trajetória desejada, podendo, desse modo, gerar colisões devido à mudança brusca de direção e à perda do controle do veículo.

A superfície do pavimento deve ter a qualidade física que facilite o movimento dos veículos, oferecendo conforto e segurança (FERREIRA, 2010) e esta qualidade pode ser avaliada, segundo Bernucci *et al.* (2010), tanto no quesito estrutural, a partir da medição da

capacidade de carga da via, quanto no quesito funcional, cujo objetivo é analisar o nível de serventia da via.

O Departamento Nacional de Infraestrutura de Transportes (DNIT) se utiliza de algumas metodologias para avaliar o pavimento de forma funcional, por exemplo: (i) Índice de Gravidade Geral (IGG); (ii) Índice de Condição de Pavimentos Flexíveis (ICPF); (iii) Índice de irregularidade internacional (*International Roughness Index* – IRI); e (iv) Índice de Condição de Manutenção (ICM).

Para Ferreira (2010), o levantamento dos defeitos superficiais ainda é bastante dependente da intervenção humana, seja pela presença do técnico na pista ou pela interpretação visual de uma filmagem da pista. Tais processos podem ser executados por metodologias automáticas, substituindo procedimentos manuais. Os procedimentos manuais estão caindo em desuso devido à sua subjetividade, altos custos financeiros, tempo e recursos humanos, e está se tornando impraticável devido à escala e frequência exigidas (GOPALAKRISHNAN *et al*., 2017).

Atualmente, existem estudos com proposições de metodologia para detecção automatizada de buracos, remendos e trincas. O trabalho desenvolvido por PAZ *et al.*, (2020) se propõe a detectar automaticamente buracos com o auxílio do treinamento das redes VGG16 e VGG19; os trabalhos desenvolvidos por FREITAS; NOBRE JÚNIOR(2020) e ESPÍNDOLA *et al*., (2021), se propõem a detectar automaticamente buracos e remendos com a utilização do algoritmo You Only Look Once – YOLO e, referente às trincas, tem-se a pesquisa elaborada por DESTRI JUNIOR *et al., (*2019), que combinou técnicas de pré-processamento da imagem com a arquitetura das redes neurais convolucionais (*Convolutional Neural Networks* – CNN).

As trincas, segundo o DNIT (2005), são defeitos na superfície que enfraquecem o revestimento e sua evolução leva à formação de panelas, defeito este considerado o mais crítico para a segurança viária. Sendo assim, considerando a importância da presença de trincas para os estudos de manutenção do pavimento, este trabalho se propõe a desenvolver um método de automatização da detecção de trincas no pavimento asfáltico, de modo que essas possam ser utilizadas, futuramente, nos processos do cálculo dos índices que auxiliam na avaliação da condição do pavimento.

1.2 Questões de pesquisa

- O método de detecção automatizada utilizando redes neurais convolucionais e o algoritmo YOLO podem ser aplicados de forma satisfatória para identificação dos defeitos do pavimento, principalmente no que se refere às trincas isoladas?
- É viável tecnicamente migrar totalmente para avaliação do pavimento de forma automática?

1.3 Justificativa e relevância da pesquisa

O presente trabalho apresenta a possibilidade de substituir procedimentos manuais de levantamento de defeitos superficiais em pavimentos por um procedimento automático, o qual irá permitir conhecer o real estado do pavimento brasileiro de uma forma mais rápida, economizando recursos, além de reduzir a subjetividade do processo. Um levantamento realizado de forma tradicional pode demorar meses a depender da extensão analisada, procedimento esse que leva poucas horas com a utilização de métodos similares ao proposto no presente trabalho. Além disso, por se utilizar de um banco de dados construído em sua maior parte a partir de imagens de pavimentos do Brasil, mais especificamente do estado do Ceará, essa tecnologia irá melhor se adequar à nossa realidade. O uso da ciência de dados, sobretudo para a previsão de comportamento e tendências em diversas áreas da ciência ganhou relevância nos últimos anos. Assim, o uso e aprimoramento dessas técnicas especificamente na pavimentação mostra-se relevante tanto para o meio técnico quanto científico.

1.4 Objetivos

1.4.1 Objetivo geral

Este projeto tem como objetivo geral contribuir com a compreensão e aplicação de técnicas de *Deep Learning* através de desenvolvimento de um modelo de automação de levantamento de trincas no pavimento em revestimento asfáltico, com o intuito de corroborar o conceito de qualidade do pavimento.

1.4.2 Objetivo específico

A pesquisa apresenta como objetivos específicos:

- Obter algoritmo de classificação e detecção de objetos que possa ser utilizado no sistema de avaliação dos pavimentos.
- Analisar diferentes cenários de modelagem a fim de identificar a melhor performance para detecção e classificação de trincas em revestimento asfáltico.

1.5 Estrutura do trabalho

A presente pesquisa encontra-se organizada em cinco capítulos, a contar com a presente introdução. Os demais capítulos deste trabalho abordarão os seguintes tópicos:

- No Capítulo 2 são abordados alguns aspectos sobre o pavimento asfáltico, incluindo detalhes sobre a sua estrutura, os tipos de defeitos superficiais que podem acometê-lo, as diferentes avaliações que podem ser realizadas para classificar o pavimento qualitativa e funcionalmente, além de apresentar importantes características sobre o Sistema de Gerência de Pavimento (SGP).
- No Capítulo 3 aborda-se alguns aspectos da modelagem computacional, como visão computacional, aprendizagem profunda, métodos de detecção de objetos, algoritmos de redes neurais que serão utilizados no trabalho, dentre outros.
- O Capítulo 4 trata sobre a metodologia utilizada na presente pesquisa, contendo as estratégias e procedimentos utilizados para construir um detector automático de trincas. Cita detalhes sobre a construção do *dataset* utilizado, bem como a forma como as métricas foram avaliadas.
- O Capítulo 5 mostra os resultados obtidos provenientes da análise dos diferentes cenários estudados. Além disso, apresentará o estudo realizado em um trecho experimental, a fim de comparar os resultados obtidos no modelo com os resultados junto à inventariação manual.
- O Capítulo 6 discorre sobre as conclusões obtidas a partir dos resultados, além de discutir sobre sugestões para trabalhos futuros com base nas dificuldades e limitações encontradas durante o desenvolvimento da pesquisa.
- O Apêndice A contém o código desenvolvido em python que possibilita uma seleção aleatória das imagens e seus respectivos rótulos.

- O Apêndice B contém o código desenvolvido em python que permite cortar e redimensionar as imagens.
- O Apêndice C contém o código desenvolvido em python que permite uma alteração do contraste da imagem.
- O Apêndice D apresenta um código em python que permite cortar as imagens, juntamente com seus respectivos rótulos obtidos junto ao LabelImg.

2 LEVANTAMENTO DE DEFEITOS SUPERFICIAIS EM PAVIMENTOS ASFÁLTICOS

O presente capítulo tem por finalidade discorrer sobre o pavimento, incluindo sua estrutura, seu processo de degradação, os tipos de avaliações existentes que atestam a qualidade do pavimento, os tipos de defeitos superficiais existentes, além de discorrer um pouco sobre a definição do Sistema de Gerência de pavimento (SGP) e suas partes constituintes.

2.1. Introdução

Pavimento é uma estrutura de múltiplas camadas de espessuras finitas, construída sobre a superfície final de terraplenagem, destinada técnica e economicamente a resistir aos esforços oriundos do tráfego de veículos e do clima, e a propiciar aos usuários melhoria nas condições de rolamento, com conforto, economia e segurança (BERNUCCI *et al*., 2010).

Para atender a esses requisitos, a estrutura do pavimento é particularmente importante, devendo ser constituída por camadas que distribuam as solicitações de carga, limitando as tensões e as deformações de maneira a garantir um desempenho adequado da via por um longo período de tempo (CNT, 2021). O pavimento deve apresentar pelo menos duas camadas, sendo elas o revestimento e a base, devendo as demais camadas serem construídas a depender dos requisitos de projeto, quando necessário (CNT, 2021). As camadas complementares à base são: a sub-base, a de reforço do subleito e/ou a camada de regularização, representadas esquematicamente na Figura 1.

Figura 1 - Disposição das camadas de pavimento tipo flexível

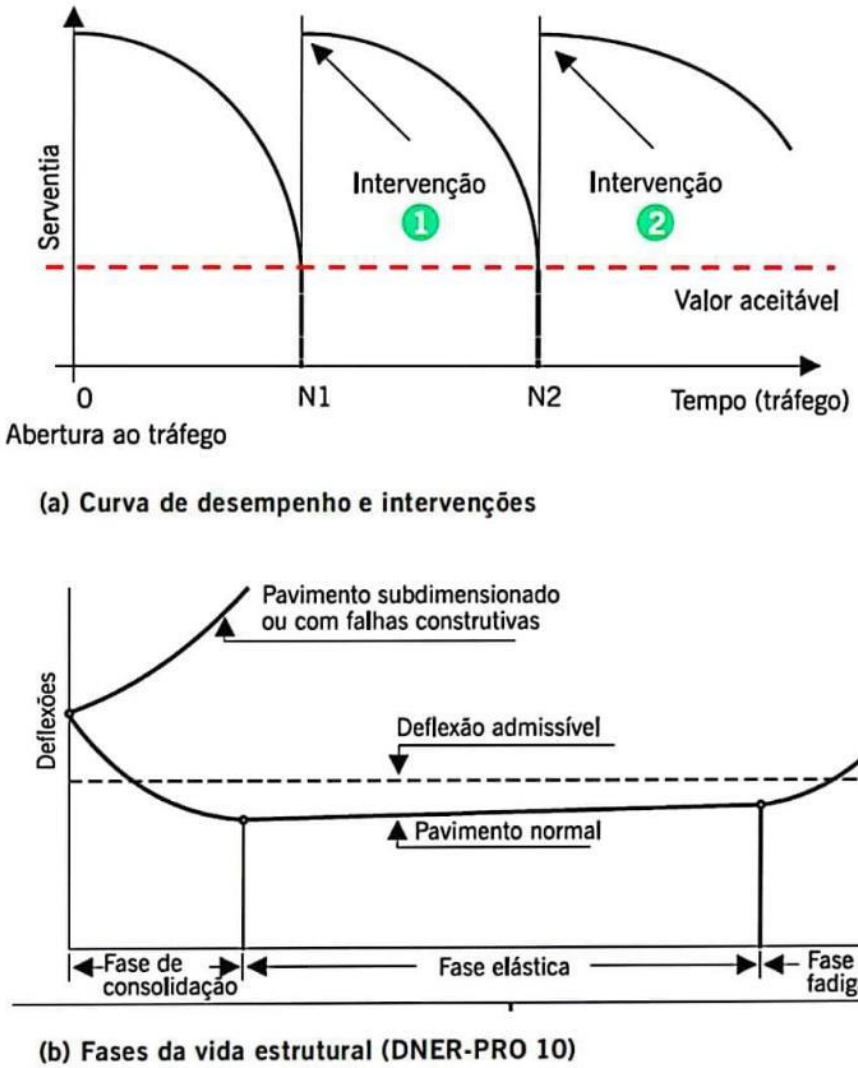
Fonte CNT (2021)

No Brasil, estima-se que 99% das rodovias são construídas em pavimento flexível, segundo a pesquisa CNT (2021), fato este que justifica o interesse em ter este tipo de

pavimento como objeto de estudo. Apesar de possuir uma vida útil de 8 a 12 anos, considerando corretas manutenções, é comum encontrarmos rodovias brasileiras deterioradas antes deste prazo de utilização, podendo ser decorrente de falhas no método de dimensionamento, no processo construtivo ou na falta de manutenção preventiva e de fiscalização (CNT, 2021).

De acordo com Bernucci *et al.* (2010), os pavimentos são estruturas que em geral não apresentam ruptura súbita, mas sim deterioração funcional e estrutural acumuladas a partir de sua abertura ao tráfego. Na Figura 2 é apresentado a curva de desempenho de um pavimento ao longo de vários ciclos de restauração.

Figura 2 - Representações esquemáticas de a) curvas de desempenho de um pavimento e b) fases da vida estrutural



Fonte: Bernucci *et al.* (2010)

Na Figura 2, é possível observar que o pavimento flexível apresenta, inicialmente, uma fase de consolidação, em que as deflexões vão diminuindo progressivamente e, depois

disso, passa por uma fase elástica, em que as deflexões apresentadas no pavimento deixam de existir cessada a carga aplicada. À medida que o tempo vai passando e os ciclos de carga e descarga vão se repetindo, o pavimento vai perdendo sua capacidade elástica e começa a entrar na fase de fadiga, em que as deflexões vão aumentando gradativamente até que o limite de deflexão admissível é ultrapassado. A execução de manutenções ou eventuais reparações são essenciais para manter o pavimento em uma condição trafegável, renovando seu ciclo de vida.

No que se refere aos tipos de deterioração, a parcela estrutural é associada aos danos ligados à capacidade de carga do pavimento, cuja avaliação é feita a partir de ensaios deflectométricos. Já a parcela funcional está atrelada aos conceitos de conforto ao rolamento, condições de superfície, interação pneu-pavimento, defeitos e irregularidades. O presente trabalho apresenta como enfoque os defeitos superficiais do pavimento, mais especificamente as trincas, cujas definições serão expostas na seção seguinte.

2.2 Defeitos superficiais

Os defeitos de superfície são os danos ou deteriorações na superfície dos pavimentos asfálticos que podem ser identificados a olho nu e classificados segundo uma terminologia normatizada (DNIT 005/2003-TER, 2003). O levantamento desses defeitos subsidia a definição de uma solução tecnicamente adequada e, em caso de necessidade, indica a melhor ou melhores alternativas de restauração do pavimento (BERNUCCI *et al.*, 2010).

Seguindo a norma brasileira DNIT 005/2003-TER, os tipos de defeitos são classificados em: fenda (F); afundamento (A); ondulação ou corrugação (O); escorregamento (E); exudação (EX); desgaste (D); panela ou buraco (P); remendo (R), cujas definições encontram-se a seguir, estando de acordo com a norma DNIT 005/2003-TER.

- Fenda: qualquer descontinuidade na superfície do pavimento, que conduza a aberturas de menor ou maior porte, apresentando-se sob as formas de fissuras ou trincas. As fendas representam um dos defeitos mais significativos dos pavimentos asfálticos e são subdivididas dependendo da tipologia e da gravidade.
    - Fissuras: fenda de largura capilar existente no revestimento, posicionada longitudinalmente, transversal ou obliquamente ao eixo da via, somente perceptível a vista desarmada de uma distância inferior a 1,5m. Não causam problemas funcionais no revestimento.

- Trincas: fenda existente no revestimento, facilmente visível a vista desarmada, com abertura superior à da fissura, podendo apresentar-se sob a forma de trinca isolada ou trinca interligada.
  - Trinca isolada
    - ❖ Trinca transversal: trinca isolada que apresenta direção predominantemente ortogonal ao eixo da via, podendo ser classificada em curta ou longa, com a extensão de 100cm sendo o limite que as define. Na Figura 3 observa-se exemplos desse tipo de trinca.

Figura 3 - Exemplo de trincas transversais

Fonte: Confederação Nacional de Transporte CNT. 2018.

- ❖ Trinca longitudinal: Trinca isolada que apresenta direção predominantemente paralela ao eixo da via, podendo ser classificada em curta ou longa, com a extensão de 100cm sendo o limite que as define. Na Figura 4 observa-se exemplos desse tipo de trinca.

Figura 4 - Exemplo de trincas a) Trincas isoladas curtas longitudinais (TLC) b) Trincas longitudinais longas (TLL)

a) b)

Fonte: Bernucci *et al.*, 2010

- ❖ Trinca de retração: trinca isolada não atribuída aos fenômenos de fadiga e sim aos fenômenos de retração térmica ou do material do revestimento de base rígida ou semi-rígida subjacentes ao revestimento trincado. Na Figura 5 observa-se exemplos desse tipo de trinca.

Figura 5 - Exemplo de trincas de retração

Fonte: Bernucci *et al.*, 2010

- o Trinca interligada
  - ❖ Trinca tipo "Couro de Jacaré": Conjunto de trincas interligadas sem direções preferenciais, assemelhando-se ao aspecto de couro de jacaré. Essas trincas podem apresentar, ou não, erosão acentuadas nas bordas. Na Figura 6 observa-se exemplos desse tipo de trinca.

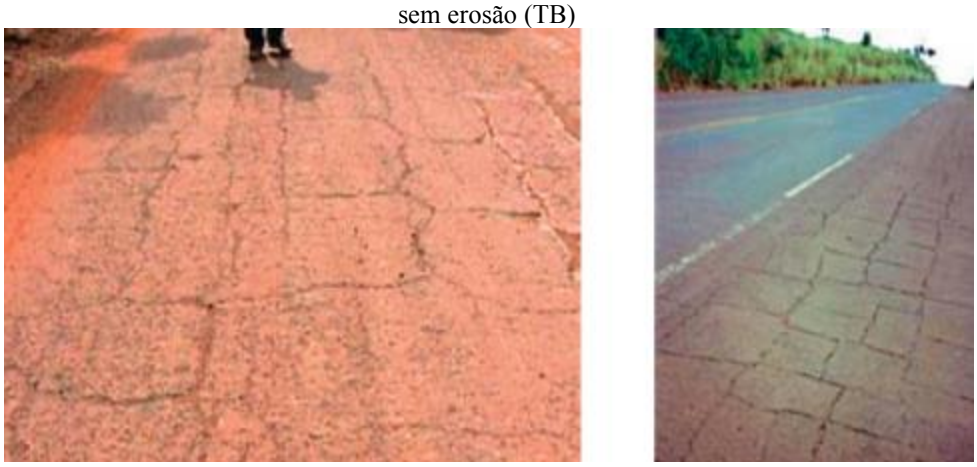
Figura 6 - Exemplo de trincas do tipo couro de jacaré

Fonte: Bernucci *et al.*, 2010

- ❖ Trinca tipo "Bloco": Conjunto de trincas interligadas caracterizadas pela configuração de blocos formados por lados bem definidos, podendo, ou

não, apresentar erosão acentuada nas bordas. Na Figura 7 observa-se exemplos desse tipo de trinca.

Figura 7 - Exemplo de trincas a) Trinca de bloco com erosão (TBE) b) Trinca de bloco sem erosão (TB)



a) b)

Fonte: Bernucci *et al.*, 2010

- Afundamento: deformação permanente caracterizada por depressão da superfície do pavimento, acompanhada, ou não, de solevamento, podendo apresentar-se sob a forma de afundamento plástico ou de consolidação.
  - Afundamento plástico: afundamento causado pela fluência plástica de uma ou mais camadas do pavimento ou do subleito, acompanhado de solevamento. Quando ocorre em extensão de até 6m é denominada afundamento plástico local; quando a extensão for superior a 6m e estiver localizado ao longo da trilha de roda é denominado afundamento plástico da trilha de roda. Na Figura 8 observa-se exemplo de afundamento plástico.

Figura 8 - Afundamento plástico nas trilhas de roda

Fonte: Bernucci *et al.*, 2010

- Afundamento de consolidação: é causado pela consolidação diferencial de uma ou mais camadas do pavimento ou subleito sem estar acompanhado de solevamento. Quando ocorre em extensão de até 6m é denominada afundamento de consolidação local; quando a extensão for superior a 6m e estiver localizado ao longo da trilha de roda é denominado afundamento de consolidação da trilha de roda. Na Figura 9 observa-se exemplos do afundamento de consolidação.

Figura 9 - Exemplos de a) Afundamento por consolidação em trilha de roda (ATC) b) Afundamento por consolidação localizado (ALC)

a) b)

Fonte: Bernucci *et al.*, 2010

- Ondulação ou corrugação: deformação caracterizada por ondulações ou corrugações transversais na superfície do pavimento. Na Figura 10 observa-se exemplo de corrugação.

Figura 10 - Exemplo de corrugação

Fonte: Bernucci *et al.*, 2010

- Escorregamento: Deslocamento do revestimento em relação à camada subjacente do pavimento, com aparecimento de fendas em forma de meia-lua. Na Figura 11 são apresentados exemplos de escorregamento.

Figura 11 - Exemplos de escorregamento

Fonte: Bernucci *et al.*, 2010

- Exsudação: excesso de ligante betuminoso na superfície do pavimento, causado pela migração do ligante através do revestimento. Nas Figura 12 é apresentado exemplo de exsudação.

Figura 12 - Exemplo de exsudação

Fonte: Bernucci *et al.*, 2010

- Desgaste: efeito do arrancamento progressivo do agregado do pavimento, caracterizado por aspereza superficial do revestimento e provocado por esforços tangenciais causados pelo tráfego. Na Figura 13 é apresentado exemplo de desgaste.

Figura 13 - Exemplo de desgaste

Fonte: Bernucci *et al.*, 2010

- Panela ou Buraco: cavidade que se forma no revestimento por diversas causas, podendo alcançar as camadas inferiores do pavimento, provocando a desagregação dessas camadas. Na Figura 14 são apresentados exemplos de buracos no pavimento.

Figura 14 - Exemplo de buracos no pavimento

Fonte: Confederação Nacional de Transporte CNT. 2018.

- Remendo: panela preenchida com uma ou mais camadas de pavimento na operação denominada “tapa-buraco”. Na Figura 15 são apresentados exemplos de remendos no pavimento.

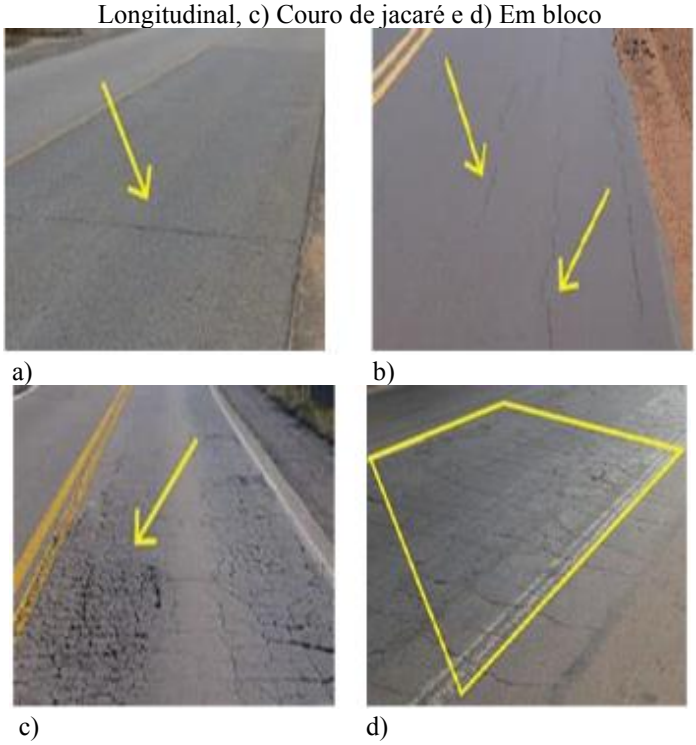
Figura 15 - Exemplo de a) Remendo mal executado b) Remendo bem executado

a) b)

Fonte: Bernucci *et al.*, 2010

As patologias em foco no presente trabalho são as trincas e, conforme visto nas definições, existem vários tipos de trincas. No entanto, as definidas para serem pesquisadas no presente trabalho são as trincas longitudinais, transversais, couro de jacaré e bloco, conforme exemplos exibidos na Figura 16.

Figura 16 – Exemplos de trincas a) Transversal, b) Longitudinal, c) Couro de jacaré e d) Em bloco



a) b)

c) d)

Fonte: Destri Junior *et al*., 2019

De acordo com o DNIT (2005), as trincas são defeitos na superfície que enfraquecem o revestimento, já que permitem a entrada de água pelas fendas, desintegrando o revestimento. O trincamento tende a aumentar sua extensão e sua severidade, levando à formação de panelas, defeito este mais crítico para a segurança viária. Por isso, ao longo do

tempo, a presença de trincas tem sido o critério importante para a deflagração de intervenções de restauração de pavimentos (DESTRI JUNIOR *et al*., 2019).

2.3 Sistema de Gerência de Pavimento (SGP)

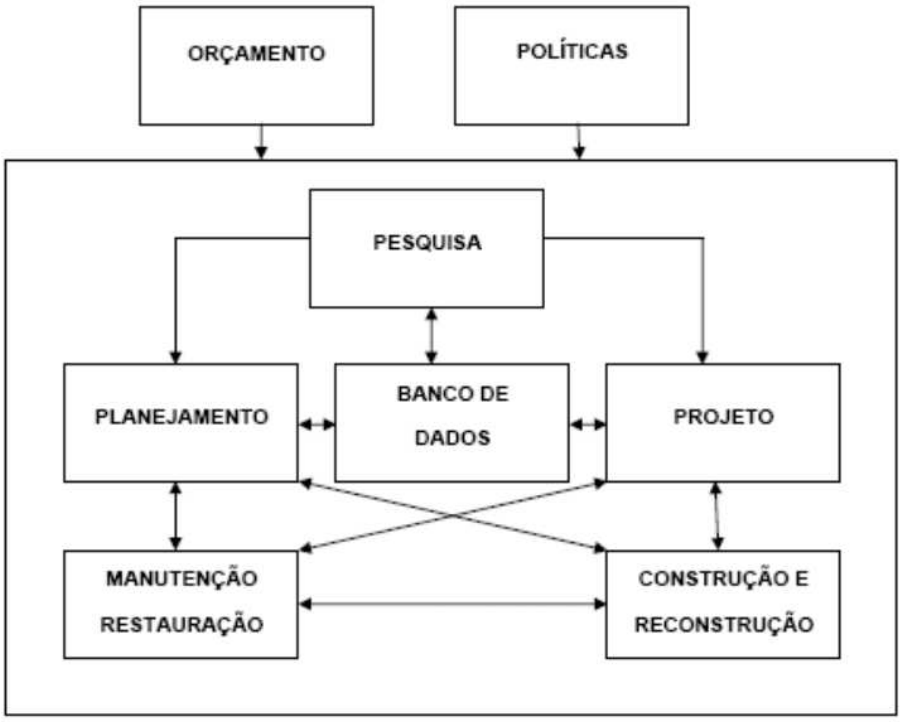
Os pavimentos rodoviários representam um patrimônio valioso, sendo essenciais para sua preservação a conservação e as restaurações oportunas. Segundo o DNIT (2011), qualquer interrupção ou redução na intensidade ou na frequência dos serviços necessários à manutenção desse patrimônio implica em aumentos substanciais nos custos de operação dos veículos e na necessidade de investimentos cada vez mais vultosos para sua recuperação. Sendo assim, o Sistema de Gerência de Pavimento (SGP) objetiva determinar a forma mais eficaz da aplicação dos recursos públicos disponíveis, em diversos níveis de intervenção, de sorte a responder às necessidades dos usuários dentro de um plano estratégico que garanta a melhor relação Custo x Benefício (DNIT, 2011).

Os fatores que despertaram um interesse crescente nos órgãos rodoviários em relação ao desenvolvimento e aplicação do SGP a partir de 1980 foram, segundo o DNIT (2011):

- Maior evidência da necessidade de manutenção oportuna e adequada da rede rodoviária, em virtude do envelhecimento dos pavimentos;
- As exigências dos órgãos financiadores, mais especificamente do Banco Mundial (BIRD), que passaram a estimular o emprego de técnicas racionais, visando melhores resultados na aplicação dos programas utilizando os empréstimos financeiros;
- A exiguidade dos recursos a serem aplicados no setor rodoviário, face às crescentes necessidades motivadas pela deterioração progressiva da rede;
- O reconhecimento do efeito direto da condição do pavimento nos custos operacionais dos veículos, principalmente no consumo de pneus e combustíveis, nos custos de manutenção e tempo de viagem;
- A utilização em nosso país de avançada tecnologia, envolvendo métodos e equipamentos para avaliação de pavimentos, com o emprego de processos informatizados.

Observando-se a Figura 17, percebe-se que o SGP apresenta o planejamento, o projeto, a manutenção e a construção como seus componentes, devendo tais componentes interagirem mutuamente, além de apresentar como fatores externos os recursos orçamentários, os dados necessários para alimentar o sistema, e as diretrizes políticas e administrativas.

Figura 17 – Sistema de Gerência de Pavimentos



Fonte: DNIT (2011)

O processo decisório de um SGP pode ser considerado em nível de rede e em nível de projeto. Segundo o DNIT (2011), a gerência em nível de rede indica os trechos prioritários da malha rodoviária que devem ser objeto de investimentos em manutenção, de forma que os recursos públicos alocados para um determinado período tenham o melhor retorno econômico. Já a gerência em nível de projeto envolve atividades detalhadas do próprio projeto e da execução de obras em um trecho específico da malha, atividades essas que deverão subsidiar orçamentos e programas de curto prazo.

2.4 Avaliação dos pavimentos

A avaliação da condição do pavimento é uma das etapas mais importantes na implementação do SGP, por ser o ponto de partida para as futuras decisões neste sistema, possibilitando que sejam definidas as condições funcionais, estruturais e operacionais dos pavimentos dos segmentos constituintes de uma malha viária em um determinado momento, mediante a obtenção dos dados fundamentais que alimentam periodicamente o SGP (DNIT, 2011).

2.4.1 Condições funcionais

Como dito na seção 2.1, a avaliação funcional de um pavimento relaciona-se à apreciação do estado de sua superfície e de como este estado influencia no conforto ao rolamento. Segundo o DNIT (2011), o Valor da Serventia Atual (VSA) e o Índice de Irregularidade Internacional (IRI) são parâmetros de avaliação bastante consistentes.

a) Valor de Serventia Atual (VSA)

O VSA foi concebido por *Carey e Iric* em 1960 para as pistas experimentais da *AASHTO* (*American Association of State Highway and Transportation Officials)*, consistindo em uma medida subjetiva baseada em notas de 0 a 5 dadas por técnicos avaliadores acerca do conforto ao rolamento de um veículo trafegando em determinado trecho (DNIT, 2011). Os cinco níveis de serventia também são adotados pelo Brasil, seguindo o procedimento proposto pelo DNIT 009/2003-PRO, estando disposta na Tabela 1 abaixo.

Tabela 1 - Níveis de serventia

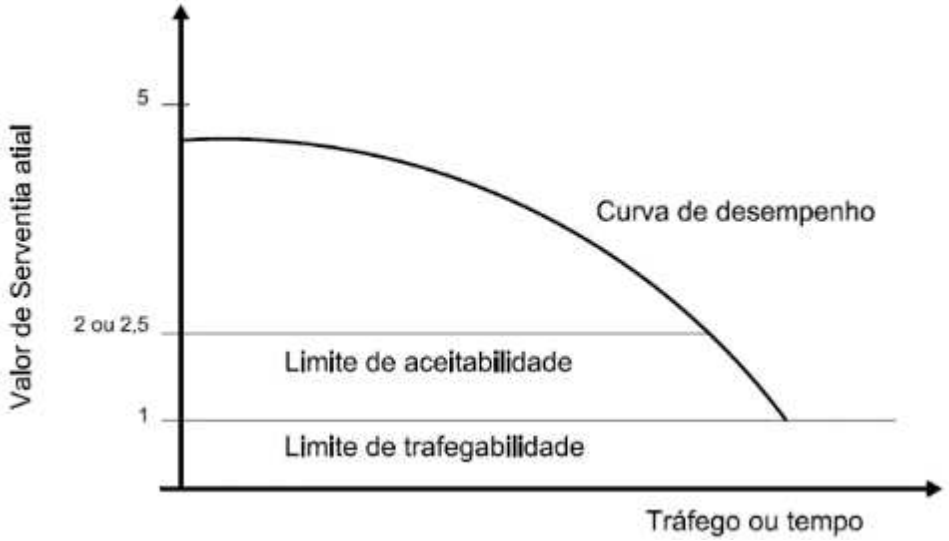
Padrão de conforto ao rolamento	**Avaliação (faixa de notas)**
Excelente	4 a 5
Bom	3 a 4
Regular	2 a 3
Ruim	1 a 2
Péssimo	0 a 1

Fonte: DNIT (2011)

O valor do VSA, logo após o término da construção do pavimento, em geral é próximo de 5, dependendo muito da qualidade executiva e das alternativas de pavimentação selecionadas. Com o passar do tempo, este valor vai reduzindo em consequência do tráfego que atua no pavimento e das intempéries que o acometem (DNIT, 2011).

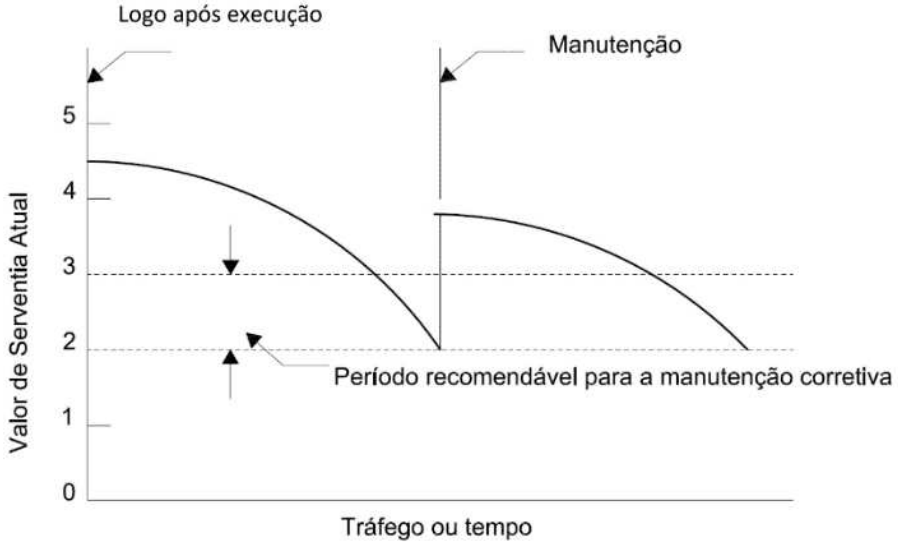
Observando-se a Figura 18, pode-se verificar os limites de aceitabilidade e de trafegabilidade, definidos pelo guia de dimensionamento de pavimentos norte-americano da AASHTO (1990). O limite de aceitabilidade define o limite a partir do qual o nível de conforto do usuário do pavimento passa a ser inaceitável, necessitando a realização de uma manutenção para reestabelecer os níveis do pavimento (como é possível observar na Figura 19). Quando não é feita a manutenção necessária ao se chegar no limite de aceitabilidade, o pavimento se degrada a ponto de chegar no limite de trafegabilidade, em que se faz necessário uma reconstrução do pavimento.

Figura 18 - Variação da serventia com o tráfego ou com o tempo decorrido de utilização da via



Fonte: DNIT (2011)

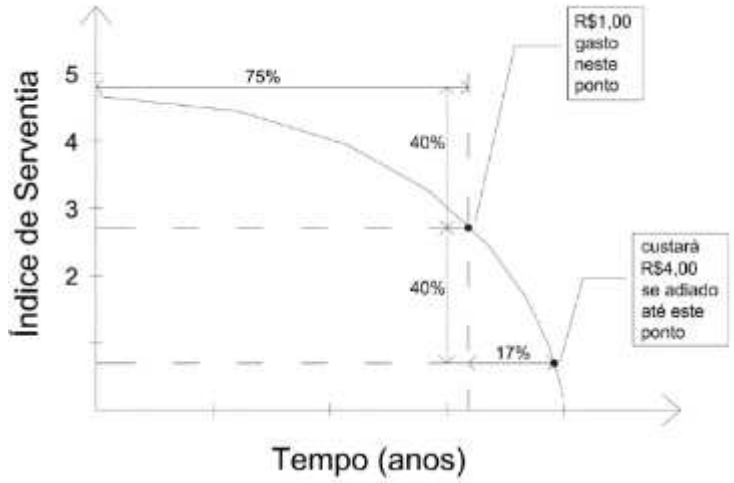
Figura 19 - Período recomendável para a manutenção dos pavimentos



Fonte: DNIT (2011)

Vale ressaltar que a aplicação de recursos na época devida para a execução dos serviços de manutenção e restauração pode proporcionar uma grande economia (DNIT, 2021). Observando-se a Figura 20, vê-se que ao deixar de aplicar R$1,00 na época certa, com Índice de Serventia de aproximadamente 2,5, esse valor passa a ser R$ 4,00 se aplicado após grande perda de serventia.

Figura 20 - Índice de serventia X Vida útil do pavimento



Fonte: DNIT (2011)

Segundo o DNIT (2011), atualmente a determinação do VSA vem sendo substituída pelo LVC – "Levantamento Visual Contínuo da Superfície de Pavimento Flexível e Semirrígido", sendo regulamentado pela norma DNIT 006/2003 PRO. Segundo orientações normatizadas, a equipe de levantamento deve ser composta por no mínimo dois técnicos, além do motorista do veículo, o veículo deve apresentar velocidade média de 40 km/h percorrendo a via em um único sentido e os avaliadores devem estabelecer os segmentos, preferencialmente, com 1 km de extensão. As condições dos pavimentos asfálticos são avaliadas pelas seguintes normas: DNIT 005/2003-TER, DNIT 006/2003–PRO, DNIT 007/2003–PRO, DNIT 008,2003–PRO, DNIT 009/2003–PRO.

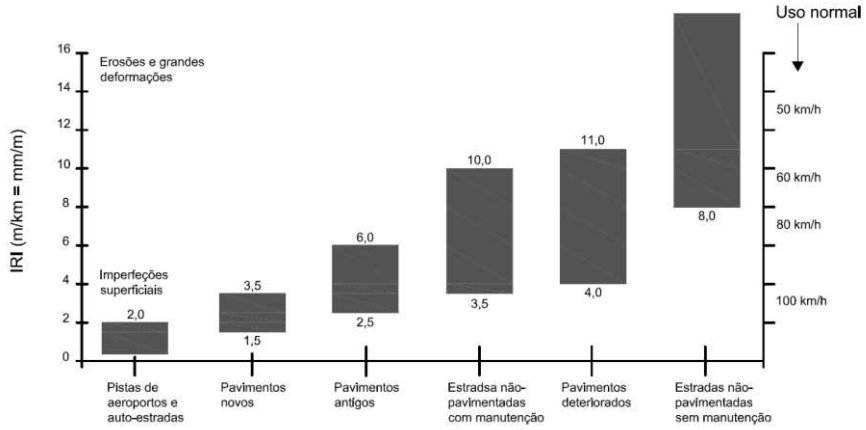
Com o técnico dentro do veículo ou percorrendo o trecho a pé, trata-se de uma operação com características essencialmente manual, perigosa, lenta, cansativa, de custo elevado e com interferência no trânsito. Devido a algumas das características citadas, há de se questionar a consistência dos resultados do trabalho de campo (FERREIRA, 2010).

Segundo Hollerweger (2019), o procedimento do LVC é realizado apenas para obtenção dos índices, não sendo suficiente para realizar as correções necessárias, fazendo com que as equipes percorram o mesmo trecho posteriormente, parando em cada ocorrência encontrada para obtenção de mais detalhes. Segundo ele, tais atividades podem ser otimizadas com o auxílio de ferramentas capazes de detectar automaticamente os defeitos, seccionando os segmentos em 1km e calculando as respectivas frequências, a fim de diminuir o custo do procedimento e dispensar a presença de um dos técnicos (que pode ser alocado para outra atividade).

b) Índice de Irregularidade Internacional (IRI)

A irregularidade longitudinal é o somatório dos desvios da superfície de um pavimento em relação a um plano de referência ideal de projeto, que afeta a dinâmica dos veículos, o efeito dinâmico das cargas, a qualidade ao rolamento e a drenagem superficial da via (DNIT, 2011). A Figura 21 mostra as faixas de variação do IRI em diversas situações.

Figura 21 - Faixas de variação do IRI



Fonte: DNIT (2011)

2.4.2 Condições estruturais

A avaliação estrutural, de acordo com o DNIT (2011), está associada ao conceito de capacidade de carga, manifestando-se em forma de deformações elásticas (ou recuperáveis) e plásticas (ou permanentes). As deformações elásticas são responsáveis pela maioria dos trincamentos ao longo do tempo de serviço do pavimento, enquanto as deformações plásticas estão associadas aos afundamentos localizados ou nas trilhas de roda.

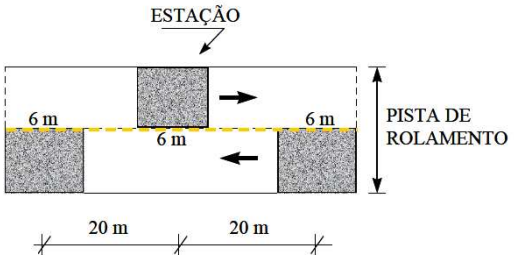
2.5 Índices do pavimento

Vários são os índices capazes de caracterizar os defeitos na superfície do pavimento quanto ao seu tipo, severidade e densidade, podendo-se citar, por exemplo, o Índice de Gravidade Global (IGG), descrito na norma DNIT 006/2003-PRO, e o *Pavement Condition Index* (PCI), definido por Shahin (1979), traduzido para o português seria Índice de Condição do Pavimento. Esses índices são bastante utilizados na análise da condição do pavimento e serão melhores descritos a seguir.

2.5.1 Índice de Gravidade Global

O Índice de Gravidade Global (IGG), segundo Bernucci *et al.* (2010), é um tipo de avaliação objetiva da condição do pavimento. Para obter este índice faz-se necessário seccionar a via em trechos de 20 metros a fim de selecionar as estações que serão inventariadas. Nas rodovias de pista simples, considera-se uma estação a cada 20 metros, alternando entre faixas, sendo em cada faixa a cada 40 metros. Nas rodovias de pista dupla, considera-se uma estação a cada 20 metros, na faixa mais solicitada pelo tráfego, em cada uma das pistas. A superfície de avaliação corresponde a 3m antes e 3m após cada uma das estacas, totalizando 6m de extensão pela largura da faixa a ser avaliada, como é possível observar na Figura 22.

Figura 22 - Esquema posicionamento estações



Fonte: Bernucci *et al.*, 2010

As informações dos defeitos observados são incluídas em um formulário apropriado, conforme norma DNIT 005/2003 – TER, similar ao apresentado na Figura 23.

Figura 23 - Exemplo de inventário de superfície

Inventário de Superfície

Rodovia: Via X — Trecho: — Operador:
data: — folha:1 — sub-trecho: — revestimento tipo: concreto asfáltico
estaca inicial:1 — estaca final: 18

Estação			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Faixa			D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E
Configuração da Terraplenagem			A	A	A	SMA	C	C	SMC	C	C	C	A	A	SMC	C	C	A	A	A
Tipo	OK	Sem Defeito																		
1	FI	Fissuras																		
(FCI)	TTC	Trincas Transversais Curtas					X													
	TTL	Trincas Transversais Longas				X						X	X			X		X		
	TLC	Trincas Longitudinais Curtas					X		X											
	TLL	Trincas Longitudinais Longas	X														X			
	TRR	Trincas Isoladas Retração																		
2	J	Couro de Jacaré				X				X								X		X
(FCII)	TB	Trincas em Bloco																		
3	JE	Couro de Jacaré c/ erosão		X					X		X		X	X	X	X	X		X	
(FCIII)	TBE	Trincas Bloco c/ erosão																		
4	ALP	Afundamento Plástico Local									X		X							
	ATP	Afundamento Plástico Trilha		X					X				X	X	X	X	X	X	X	X
5	O	Ondulação																		
	P	Panela											X				X			
6	EX	Exsudação																		
7	D	Desgaste	X		X	X	X		X				X	X	X	X	X		X	X
8	R	Remendo												X				X		
	ALC	Afundamento Consolidação Local																		
	ATC	Afundamento Consolidação Trilha																		
	E	Escorregamento													X					
	TRI	Afundamento Trilha Interna (mm)	0	0	1	0	0	0	0	1	0	0	4	6	8	3	2	5	5	4
	TRE	Afundamento Trilha Externa (mm)	0	3	1	0	1	0	3	1	0	1	7	4	9	6	7	6	4	8

Fonte: Bernucci *et al.*, 2010

Para o cálculo das frequências absolutas e relativas, faz-se necessário subdividir os segmentos de modo que eles possuam as mesmas características ou defeitos e, a partir disso, contar o número de estações em cada segmento. Os defeitos são agrupados segundo os 8 tipos presente na Figura 24, e o cálculo da frequência relativa segue a equação 1 abaixo.

$$fr = fa \times 100/n \quad (1)$$

Em que:

fr = frequência relativa do defeito ou do tipo;

fa = frequência absoluta (número de vezes que o defeito ou o tipo de defeito ocorreu em um mesmo segmento;

n = número de estações inventariadas em um segmento.

Vale ressaltar que, para efeito de ponderação, quando em uma mesma estação forem constatadas ocorrências tipo 1, 2 e 3 (relativas às fendas), considera-se sempre a trinca de maior severidade.

Para o cálculo do IGI utiliza-se a equação 2 e para o cálculo do IGG utiliza-se a equação 3.

$$IGI = fr \times fp \quad (2)$$

$$IGG = \sum IGI \quad (3)$$

Em que:

fr = frequência relativa

fp= fator de ponderação (pré-fixado pela norma DNIT 006/2003-PRO, presente na Tabela 2).

Tabela 2 - Valor do Fator de Ponderação

Ocorrência Tipo	Codificação de ocorrências de acordo com a Norma DNIT 005/2002-TER "Defeitos nos pavimentos flexíveis e semi-rígidos – Terminologia"	Fator de ponderação fp
1	Fissuras e Trincas Isoladas (FI, TTC, TTL, TLC, TLL e TRR	0,2
2	FC-2 (J e TB)	0,5
3	FC-3 (JE e TBE) Nota: Para efeito de ponderação, quando em uma mesma estação forem constatadas ocorrências tipo 1, 2 e 3, só considerar as do tipo 3 para o cálculo da frequência relativa em porcentagem (fr) e Índice de Gravidade Individual (IGI); do mesmo modo, quando forem verificadas ocorrências tipos 1 e 2 em uma mesma estação, só considerar as do tipo 2.	0,8
4	ALP, ATP E ALC, ATC	0,9
5	O, P, E	1,0
6	EX	0,5
7	D	0,3
8	R	0,6

Fonte: DNIT 006/2003-PRO

Para avaliar a condição do pavimento com base nesse índice, usa-se os limites presentes na Tabela 3.

Tabela 3 - Conceitos de degradação do pavimento em função do IGG

Conceitos	**Limites**
Ótimo	$0 < IGG \leq 20$
Bom	$20 < IGG \leq 40$
Regular	$40 < IGG \leq 80$
Ruim	$80 < IGG \leq 160$
Péssimo	$IGG > 60$

Fonte: DNIT 006/2003-PRO

2.5.2 Índice de Condição do Pavimento

Segundo a Usace (1982), o Índice de Condição do pavimento (PCI) é afetado por vários fatores, podendo-se destacar a integridade e capacidade estrutural, a rugosidade, o potencial de aquaplanagem e a taxa de deterioração do pavimento. O PCI é um indicador numérico baseado em uma escala de 0 a 100, medindo a integridade estrutural e condição operacional da superfície. A Tabela 4 apresenta a escala PCI e a classificação da condição do pavimento.

Tabela 4 - Escala PCI e classificação da condição do pavimento

PCI	Escala
100	Excelente
85	Muito bom
70	Bom
55	Regular
40	Ruim
25	Muito ruim
10	Péssimo
0	

Fonte: Adaptado de Usace (1984)

3 INTELIGÊNCIA ARTIFICIAL

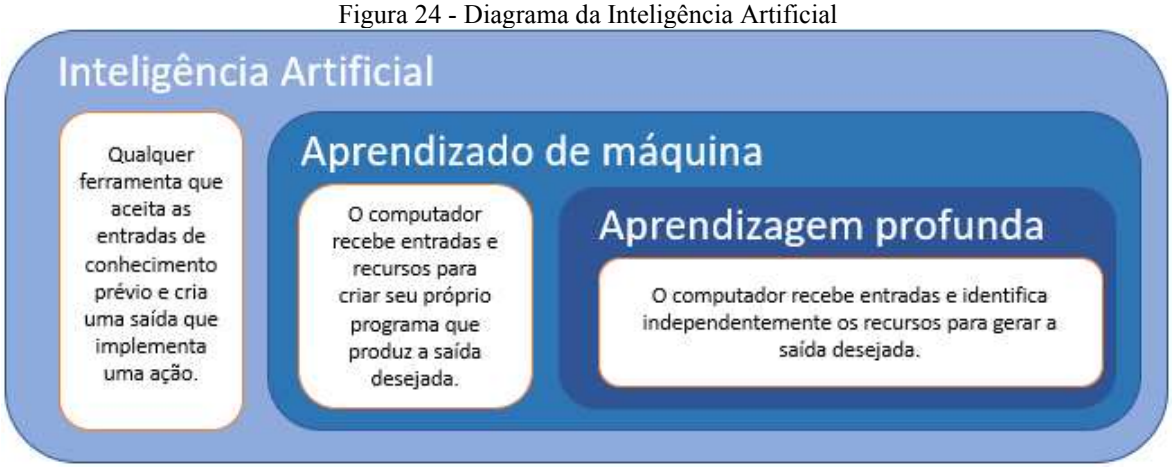
O Capítulo 3 busca fazer uma breve introdução sobre os conceitos teóricos que compõem a inteligência artificial, como aprendizagem profunda, métodos de detecção de objetos, algoritmos de redes neurais que serão utilizados no trabalho, dentre outros.

3.1. Introdução

A Inteligência Artificial (*Artificial Intelligence* - AI) é um ramo da ciência da computação que estuda e desenvolve teorias e métodos, técnicas e aplicação de sistemas para simular e estender a inteligência humana, funcionando interdisciplinarmente, envolvendo campos da psicologia, biologia, lógica matemática, linguística, engenharia, filosofia, entre outras áreas científicas (DA SILVA; BARONE, 2020).

Da Inteligência Artificial derivaram outros conceitos, a exemplo do Aprendizado de Máquina (*Machine Learning* – ML) e da Aprendizagem Profunda (*Deep Learning* – DL), representados na Figura 24, e essa evolução do aprendizado de máquina só pode ser efetivada devido ao aumento da capacidade de processamento dos computadores.

Figura 24 - Diagrama da Inteligência Artificial



Fonte: Adaptado de BARDIS et al. (2020)

Definindo melhor cada um, tem-se que a AI é uma ciência técnica que foca na pesquisa e no desenvolvimento de teorias, métodos, técnicas e aplicações de sistemas para simulação e extensão da inteligência humana. Segundo Copeland (2016), o Aprendizado de Máquina é um campo de pesquisa da AI que usa algoritmos para analisar dados, aprender com eles e, em seguida, fazer uma determinação ou previsão sobre algo no mundo. A aprendizagem profunda (*deep learning*), de acordo com LeCun, Bengio e Hinton (2015),

permite que modelos computacionais compostos de várias camadas de processamento aprendam representações de dados com vários níveis de abstração. Além disso, ainda segundo LeCun, Bengio e Hinton (2015), a aprendizagem profunda descobre estruturas complexas em grandes conjuntos de dados usando o algoritmo de retropropagação, para indicar como uma máquina deve alterar seus parâmetros internos que são usados para calcular a representação em cada camada a partir da representação na camada anterior. A utilização do termo "profunda" está relacionada à presença de um maior número de camadas internas ocultas se comparado às redes neurais tradicionais, camadas estas que são responsáveis por extrair/representar a informação a partir de diferentes graus de abstração.

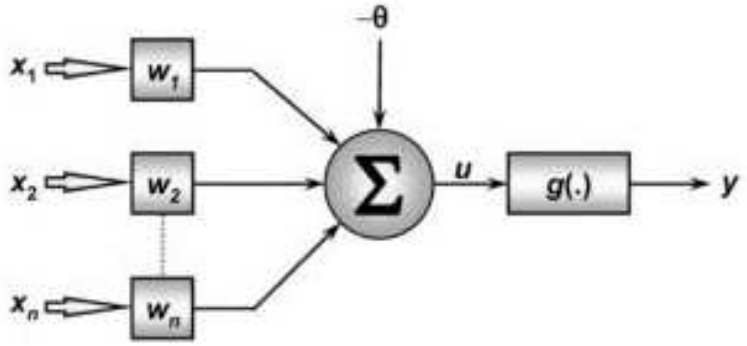
Várias tarefas de aprendizagem profunda, como classificação de imagens, detecção de objetos e segmentação de imagens, foram bem sucedidas com a utilização de tecnologias de aprendizagem profunda, tendo em vista que seus níveis de abstração permitem que um sistema aprenda funções complexas mapeando a entrada para a saída diretamente dos dados, sem depender de recursos criados por humanos (IKRAM, 2019).

3.2 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são técnicas computacionais que apresentam modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência, fornecendo meios para que o algoritmo seja capaz de extrair características dos dados a partir de um conjunto de treinamento e, assim, possa armazenar o conhecimento adquirido e aplicá-lo em situações reais (FERREIRA, 2010).

A principal unidade de processamento no cérebro são os neurônios, cujos componentes são os dentritos, que recebem os estímulos; o corpo, que coleta e combina informações; e o axônio, que transmite os estímulos. Em 1943, McCulloch e Pitts apresentaram um modelo de neurônio artificial que possuía uma estrutura similar ao neurônio biológico, cuja representação simplificada encontra-se na Figura 25.

Figura 25 - Representação simplificada do neurônio matemático



Fonte: Data Science Academy. *Deep Learning Book.* Disponível em: https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/

Segundo o Data Science Academy, os componentes de um neurônio matemático são:

- Sinais de entrada {X1, X2, …, Xn}: São os sinais externos normalmente normalizados para incrementar a eficiência computacional dos algoritmos de aprendizagem. São os dados que alimentam seu modelo preditivo.
- Pesos sinápticos {W1, W2, …, Wn}: São valores para ponderar os sinais de cada entrada da rede. Esses valores são aprendidos durante o treinamento.
- Combinador linear {Σ}: Agrega todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos e somados a coeficientes de bias (b), a fim de produzir um potencial de ativação.
- Limiar de ativação {Θ}: Especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo de ativação.
- Potencial de ativação {u}: É o resultado obtido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Se o valor for positivo, ou seja, se $u \geq 0$ então o neurônio produz um potencial excitatório; caso contrário, o potencial será inibitório.
- Função de ativação {g}: Seu objetivo é limitar a saída de um neurônio em um intervalo de valores.
- Sinal de saída {y}: É o valor final de saída podendo ser usado como entrada de outros neurônios que estão sequencialmente interligados.

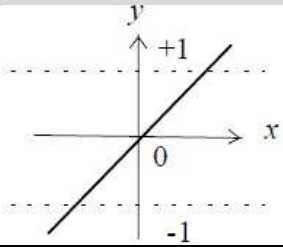
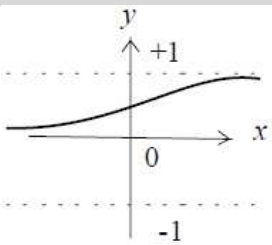
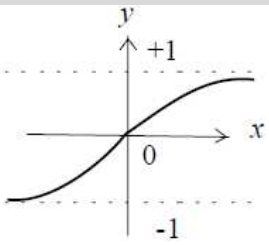
Essas operações podem ser resumidas por meio da equação 4.

$$\mathbf{y} = g(\mathbf{Xw} + \mathbf{b}) \quad (4)$$

Em que ‘X’ representa as entradas, ‘w’ os pesos, ‘b’ os coeficientes de bias, ‘y’ a saída e ‘g’ a função não-linear.

De acordo com Santos (2006), as funções de ativação mais utilizadas são a linear, a sigmoidal e a tangente hiperbólica, cujas equações, domínios e gráficos representativos encontram-se na Tabela 5.

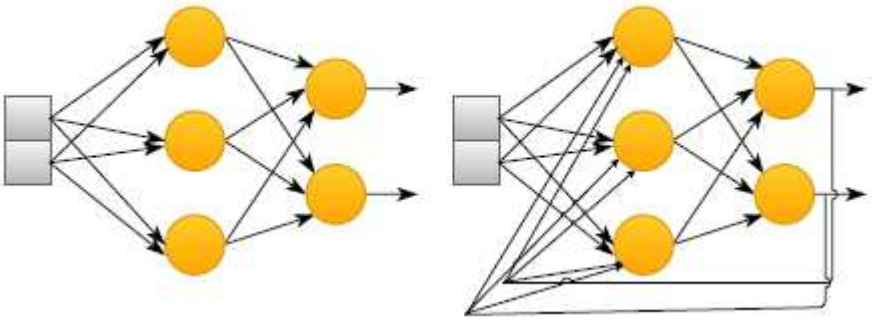
Tabela 5 - Representação das principais funções de transferência usadas atualmente

Linear	Sigmoidal	Tangente Hiperbólica
$y = a \cdot x$	$y = \frac{1}{1 + e^{-x}}$	$y = \frac{1 - e^{-x}}{1 + e^{-x}}$
[-:, +:]	[0,1]	[-1,1]
		

Fonte: Santos (2006)

As redes neurais artificiais apresentam diferentes arquiteturas (ou topologias, como é comumente chamado), podendo conter um número variado de camadas, com diferentes *designs* das camadas ocultas. Sendo assim, segundo Russell e Norvig (2014), as arquiteturas de redes neurais podem ser colocadas em duas categorias: redes neurais *feedforward* (que apresenta duas ou mais camadas de alimentação para frente), e as redes recorrentes. A Figura 26 apresenta uma representação gráfica dos tipos de redes.

Figura 26 - Exemplo de RNAs a) feedforward e b) recorrente



a) b)

Fonte: Reis (2018)

Para Russell e Norvig (2014), nas redes de propagação para frente (*feedforward*), o fluxo de informação é unidirecional, formando um grafo acíclico dirigido. Já as redes recorrentes, em contrapartida, alimentam sua saída de volta às suas próprias entradas, formando um sistema dinâmico que pode atingir um estado estável ou apresentar um

comportamento caótico. Segundo o *Data Science Academy*, as dez principais arquiteturas de redes neurais são: Redes *Multilayer Perceptron*, Redes Neurais Convolucionais, Redes Neurais Recorrentes, *Long Short-Term Memory* (LSTM), Redes de Hopfield, Máquinas de Boltzmann, *Deep Belief Network*, *Deep Auto-Encoders*, *Generative Adversarial Network*, *Deep Neural Network Capsules*. Será utilizado no presente trabalho a arquitetura das Redes Neurais Convolucionais, que será descrito detalhadamente a seguir.

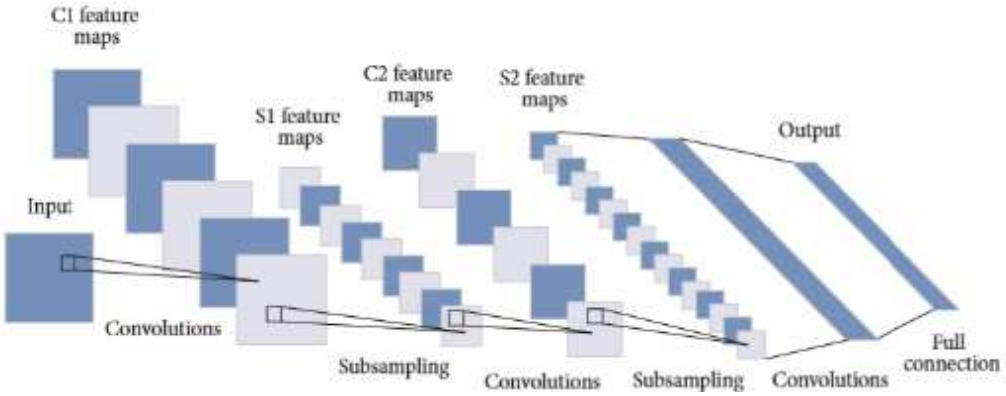
3.3 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (ConvNet / *Convolutional Neural Network* - CNN), segundo Data Sciency Academy (2021), são algoritmos de aprendizado profundo que podem captar uma imagem de entrada, atribuir importância (pesos e vieses) a vários aspectos de objetos da imagem e ser capaz de diferenciar um do outro. Além disso, são capazes de capturar com sucesso as dependências espaciais e temporais em uma imagem através da aplicação de filtros relevantes, possibilitando um melhor ajuste ao conjunto de dados da imagem devido à redução no número de parâmetros envolvidos e à capacidade de reutilização dos pesos.

De acordo com Gopalakrishnan *et al.* (2017), as *deep convolucional neural networks* (DCNNs) têm se mostrado altamente eficazes no processamento de imagens visuais, como imagens e vídeos. Os DCNNs pegam os dados brutos de entrada em um nível mais baixo e os transforma, processando-os por meio de uma sequência de unidades computacionais básicas para representações que possuem valores intrínsecos para classificação nas camadas superiores.

Segundo Xie, Zhang e Bai (2017), a rede neural convolucional (CNN) contém três tipos de camadas, sendo elas as camadas de convolução, as camadas de subamostragem e as camadas completamente conectadas, como é possível observar na Figura 27.

Figura 27 - Arquitetura de uma Rede Neural Convolucional (CNN)



Fonte: Xie, Zhang e Bai (2017)

3.3.1 Camada de convolução

Para Gopalakrishnan *et al.* (2017), a camada de convolução é parametrizada pelo número de canais, tamanho do *kernel*, fator de passo, modo de borda e tabela de conexão. Ainda utilizando a mesma referência, a camada de convolução pega a imagem de entrada e aplica a convolução nela para produzir a imagem de saída ou a resposta do filtro. Múltiplas camadas convolucionais são usadas para levar em consideração as dependências espaciais entre os pixels da imagem.

Observando a Figura 28, segundo Xie, Zhang e Bai (2017), na camada de convolução a matriz da esquerda é a entrada, que é uma imagem digital, e a matriz da direita é uma camada de convolução. A camada de convolução leva a matriz de convolução à imagem de entrada, gerando a imagem de saída, cujo exemplo de cálculo de convolução é demonstrado na Figura 29. A cada bloco de pixels convolucionado com filtro é gerado um pixel em uma nova imagem.

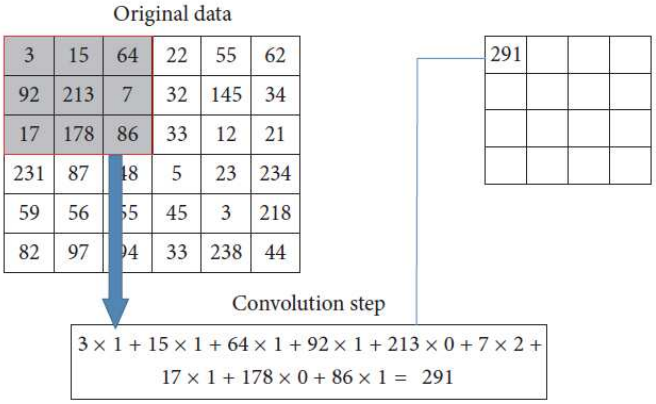
Figura 28 - Representação da imagem digital e da matriz de convolução

3	15	64	22	55	62
92	213	7	32	145	34
17	178	86	33	12	21
231	87	48	5	23	234
59	56	55	45	3	218
82	97	94	33	238	44

1	1	1
1	0	2
1	0	1

Fonte: Xie, Zhang e Bai (2017)

Figura 29 - Exemplo de cálculo de convolução

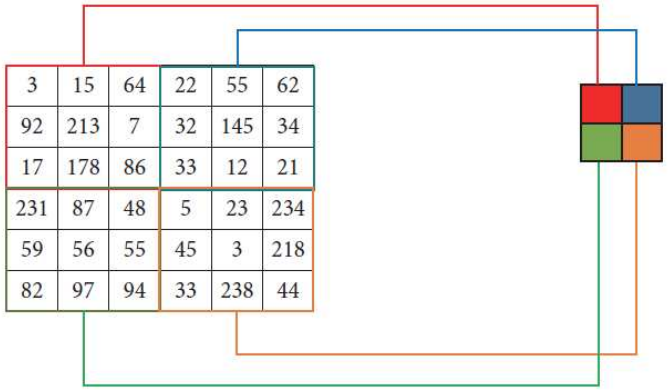


Fonte: Xie, Zhang e Bai (2017)

3.3.2 Camada de subamostragem

Segundo Xie, Zhang e Bai (2017), a camada de subamostragem é de grande importância para a rede neural convolucional, pois reduz o tamanho da imagem de entrada para dar mais invariância e robustez. O método *max-pooling*, que pode ser visualizado na Figura 30, é comumente usado para a subamostragem de camada, pois mostrou levar a uma convergência mais rápida, além de apresentar melhor capacidade de generalização (GOPALAKRISHNAN *et al*., 2017). Neste método, de acordo com Xie, Zhang e Bai (2017), a imagem é dividida em blocos e o máximo valor de cada bloco é o valor do pixel correspondente da imagem de saída. Para com Xie, Zhang e Bai (2017), a camada de subamostragem é vantajosa, pois apresenta menos parâmetros, sendo, portanto, mais fácil de treinar, além de tolerar translações e rotações entre os padrões de entrada.

Figura 30 - Exemplo de camada de subasmostragem



Fonte: Xie, Zhang e Bai (2017)

3.3.3 Camada totalmente conectada

Para Xie, Zhang e Bai (2017), as camadas completamente conectadas são similares às camadas *feedforward* tradicionais. Eles transformam a rede neural *feedforward* em vetores, com um comprimento pré-definido. Elas podem ajustar o vetor a determinadas categorias e torná-lo um vetor de apresentação para o processamento posterior.

3.4 Informações adicionais sobre as DCNNS

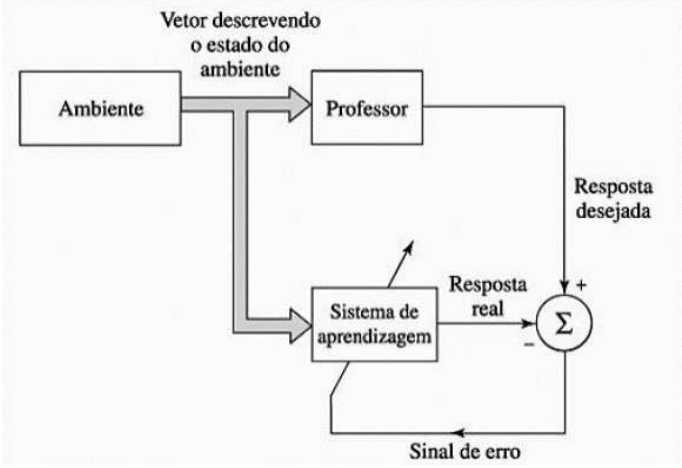
Segundo Gopalakrishnan *et al*. (2017), os DCNNs normalmente exigem grandes conjuntos de dados de imagens anotadas para alcançar alta precisão preditiva, sendo, no entanto, difícil adquirir tal quantidade de dados e de rótulos. A utilização de recursos DCNNs pré-treinados em conjunto de dados de grande escala têm se mostrado úteis para resolver problemas de classificação de imagens de domínio cruzado por meio do conceito de transferência de aprendizagem (*transfer learning*) e ajuste fino (SHIN et al, 2016). Pode-se citar, como exemplo desses recursos pré-treinados, o VGG-16, AlexNet e o GoogleLeNet. Para Gopalakrishnan *et al*. (2017) usar modelos de aprendizagem profunda pré-treinados e transferir sua capacidade de aprendizagem para um novo cenário de classificação é mais eficiente do que treinar um classificador DCNN do zero.

3.5 Aprendizagem em Redes Neurais

De acordo com Brega (1996), o objetivo da aprendizagem é fazer com que a aplicação de um conjunto de entradas da rede neural produza um conjunto de saídas consistente. Para isso, existem métodos de aprendizagem, podendo ser subdividido em duas categorias: aprendizagem supervisionada e aprendizagem não-supervisionada.

A aprendizagem supervisionada, para Brega (1996), é um processo que apresenta um instrutor externo que fornece ao sistema vetores de entrada e saída conhecidos. Quando um vetor de entrada é aplicado à rede, a saída é processada e o resultado é comparado com o vetor de saída desejado, possibilitando que o erro seja propagado pela rede e os pesos sejam alterados por um algoritmo que minimiza este erro. A representação esquemática deste tipo de aprendizagem pode ser vista na Figura 31.

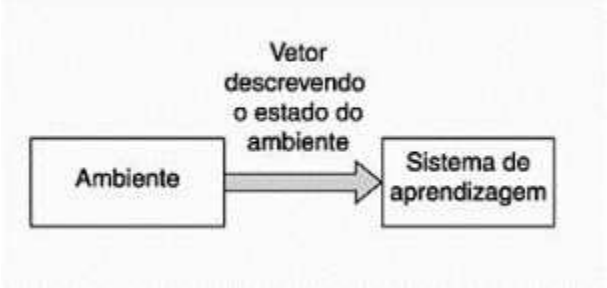
Figura 31 - Diagrama em blocos da aprendizagem supervisionada



Fonte: Haykin (2007)

Já a aprendizagem não-supervisionada, ainda segundo Brega (1996), é um processo que se baseia apenas em informações locais e controle interno, cujo conjunto de treinamento consiste apenas em vetores de entrada. O processo de treinamento extrai propriedades estatísticas do conjunto de treinamento e agrupa vetores similares em classes. A representação esquemática deste tipo de aprendizagem pode ser vista na Figura 32.

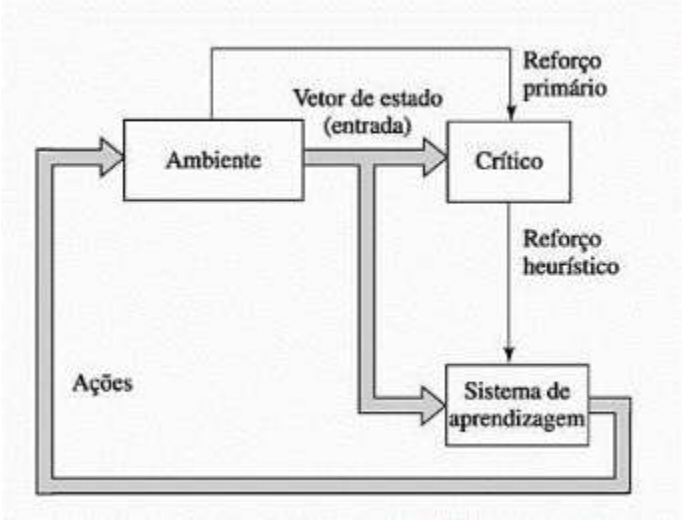
Figura 32 - Diagrama em bloco da aprendizagem não-supervisionada



Fonte: Haykin (2007)

Segundo Haykin (2007), existe também a aprendizagem por reforço, em que o algoritmo interage continuamento com o ambiente, ajustando os pesos e limiares baseando-se em informações procedentes da interação com o sistema mapeado, visando reforçar as respostas satisfatórias. A representação esquemática deste tipo de aprendizagem pode ser vista na Figura 33.

Figura 33 - Diagrama em bloco da aprendizagem por reforço



Fonte: Haykin (2007)

3.6 Detecção de objetos em Pavimentos

Existem vários métodos de detecção de objetos baseados na arquitetura das redes neurais profundas, sendo os principais o R-CNN (*Region-Based Convolutional Neural Networks*), Fast R-CNN, Faster R-CNN, SSD (*Single Shot MultiBox Detector*) e YOLO (*You Only Look Once*).

Segundo Girshick *et al.* (2016), os R-CNNs usam algoritmos de busca seletiva para extrair 2000 caixas delimitadoras da imagem de entrada na primeira etapa para ficar processando apenas os recursos mais importantes em uma entrada baseada na cor, padrão, forma e tamanho. Os R-CNNs usam um mecanismo de três estágios: extração de recursos via pesquisa seletiva, classificação SVM (Support Vector Machine) e modelagem de regressão para caixas delimitadoras apertadas. Segundo o Acervo Lima (2021), o SVM é um classificador discriminativo formalmente definido por um hiperplano de separação, ou seja, a partir dos dados de treinamento rotulados, o algoritmo produz um hiperplano ótimo que categoriza novos exemplos.

Para Girshick *et al.* (2015), o modelo Fast R-CNN usa um único mecanismo de estágio onde ele passa diretamente a entrada para uma CNN e a saída desta CNN são as Regiões de Interesse (RoI). Em seguida, uma camada de *pooling* RoI é aplicada à saída do CNN para deformar as imagens para o tamanho da rede acostumado a trabalhar. Esses RoIs são dados a um *Neural Network* (NN) conectada que os segrega e retorna caixas delimitadoras nas RoIs usando regressão linear e redes *softmax* trabalhando de forma paralela. Isso fornece ganhos drásticos de velocidade, bem como economias em termos de tamanho do modelo. Embora o

Fast R-CNN use um mecanismo de um estágio, ele depende do método de busca seletiva inicialmente e isso consome tempo.

De acordo com Ren *et al.* (2015), o Faster R-CNN passa a imagem de entrada para um CNN que retorna o mapa de características da entrada e estes mapas de recursos são passados para Redes de Propostas Regionais que fornecem propostas de objetos junto com uma pontuação de quanto de um objeto que a predição particular se assemelha. Isto é passado para uma camada de *pooling* RoI que distorce todos as regiões do mesmo tamanho para torná-las compatíveis com outras redes ocultas que serão usadas no processo. A camada final é semelhante à do R-CNN que é uma fusão de ambos o *softmax* e a camada de regressão linear que classifica e desenha caixas delimitadoras para os vários objetos disponíveis na imagem. Embora o Faster R-CNN tenha melhorado a velocidade, ele tem uma desvantagem fundamental como os outros dois R-CNNs, não olha para a imagem completa de uma só vez, em vez disso, eles focam em partes da imagem sequencialmente e isso requer muitas etapas complexas. Para contornar esse problema, o SSD e o Métodos YOLO foram criados.

A estrutura do YOLO produz ganhos drásticos de velocidade, processando até 45 fps, o que é uma grande melhoria em comparação com CNNs baseadas na região, favorecendo a detecção de objetos em tempo real (DHARNEESHKAR *et al*., 2020).

Para Liu *et al.* (2016), o SSD é semelhante ao YOLO, mas é mais brando em termos de compatibilidade de proporção. Funciona com mais 6 proporções do que YOLO. Isso permite um encapsulamento de objetos mais rígido do que YOLO. Ele também tem maior precisão do que YOLO, pois tem um maior número de camadas ocultas. Esta vantagem adicional de melhor precisão vem com um *trade-off*, que é a velocidade. Desde carros se movendo em velocidades muito altas, detecção de buracos em altas velocidades exige detecção de alta velocidade e uma das mais altas velocidades podem ser obtidas usando YOLO conforme as explicações disponível e ilustradas anteriormente.

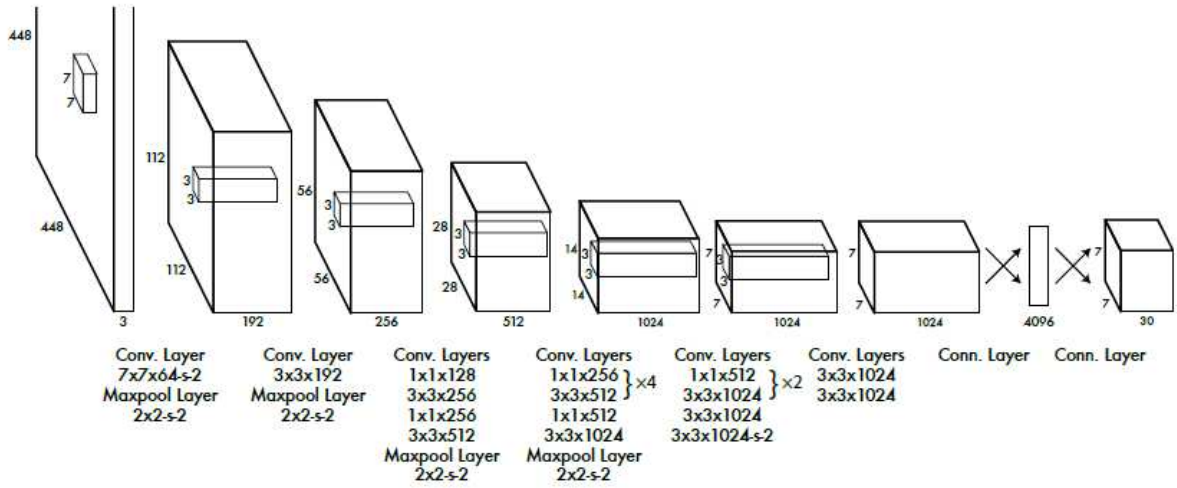
No presente trabalho, será utilizado o YOLO para a detecção de objetos, cujas características serão melhores descritas a seguir.

3.6.1 YOLO

YOLO é um algoritmo para detecção de objetos voltado para processamento em tempo real, que se baseia na arquitetura das redes neurais convolucionais, possuindo grande capacidade de generalização (REDMON *et al.* 2016). A arquitetura da CNN que o YOLO contempla foi inspirada no GoogLeNet para classificação de imagens e possui 24 camadas

convolucionais seguidas por 2 camadas totalmente conectadas. Diferente dos módulos *Inception* usados pelo GoogLeNet, o YOLO simplesmente utiliza camadas de redução 1 × 1 seguidas por camadas convolucionais 3 × 3 (REDMON *et al*., 2016). A rede completa pode ser visualizada na Figura 34.

Figura 34 - Arquitetura YOLO com 24 camadas convolucionais



Fonte: Redmon *et al*. (2016)

O procedimento realizado pelo YOLO para detecção de objetos segue os seguintes passos, de acordo com Hollerweger (2019):

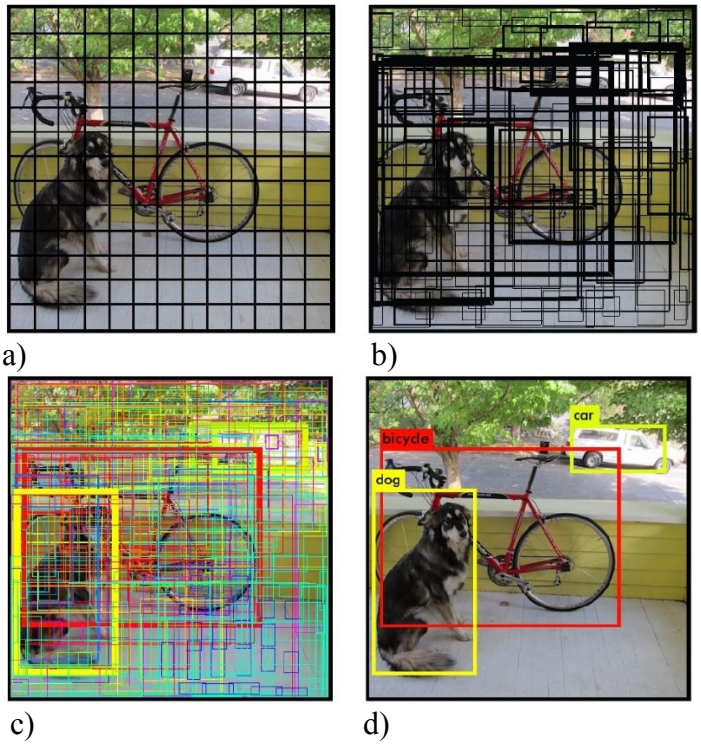
i. A imagem é dividida em *grid*, como demonstrado na Figura 35a.

ii. Cada uma das células é responsável pela predição de caixas de seleção. Tais caixas podem ter qualquer tamanho ou formato, e correspondem à área onde a rede acredita que exista um objeto. Neste passo, são obtidos *scores* de confiança para cada uma das caixas. Essas caixas são ilustradas na Figura 35b, em que caixas com a moldura mais espessa representam caixas com maior *score* de confiança.

iii. Para cada uma das caixas de seleção, a rede faz a predição de uma classe, o *score* de confiança de que a caixa contenha um objeto é combinado com um novo *score* que representa a probabilidade de acerto no objeto classificado. O resultado deste passo pode ser visualizado na Figura 35c, em que caixas de diferentes *scores* representam diferentes classes de objetos.

iv. Por fim, a grande maioria das caixas terá um *score* muito baixo, e são eliminadas as caixas com *score* baixo de um certo limiar (que costuma ser em torno de 30%). Após tal operação, são obtidas as caixas com maior chance de conter um objeto, e que esse objeto tenha sido reconhecido pela rede. O resultado final pode ser visualizado na Figura 35d.

Figura 35 - Estratégias YOLO a) Passo 1 b) Passo2 c) Passo 3 d) Passo 4



Fonte: Hollerweger (2019)

O YOLO apresenta três grandes vantagens, segundo Zhang *et al.* (2018): rompe o limite de velocidade máxima da CNN e atinge um excelente equilíbrio entre velocidade e precisão; raciocina globalmente e codifica informações contextuais sobre a imagem, sendo menos provável que preveja falsos positivos em segundo plano; e aprende representações gerais de objetos, ultrapassando outros métodos de detecção, como R-CNN .

No entanto, apesar das vantagens, o YOLO apresenta algumas limitações, também abordadas por Zhang *et al. (*2018), como imposição de fortes restrições espaciais nas previsões da caixa delimitadora como reconhecer pequenos objetos em grupo; dificuldade para generalizar objetos novos ou incomuns em relações de aspecto ou configurações; trata a função de perda do YOLO de forma semelhante tanto em pequenas caixas delimitadoras como em caixas grandes.

O YOLOv3 e o YOLOv4 são as duas últimas versões mais estáveis do YOLO, segundo Mendes e González (2020), e a diferença entre essas versões é que YOLOv3 usa redes de pirâmide características (*Feature Pyramids Networks* – FPN) para detecção de objetos, enquanto YOLOv4 usa PANet como um método de agregação de parâmetros para diferentes níveis de detecção junto com um aumento na precisão média (AP) e quadros por

segundo (FPS), um recurso que torna no geral a precisão do YOLOv4 maior do que a do YOLOv3.

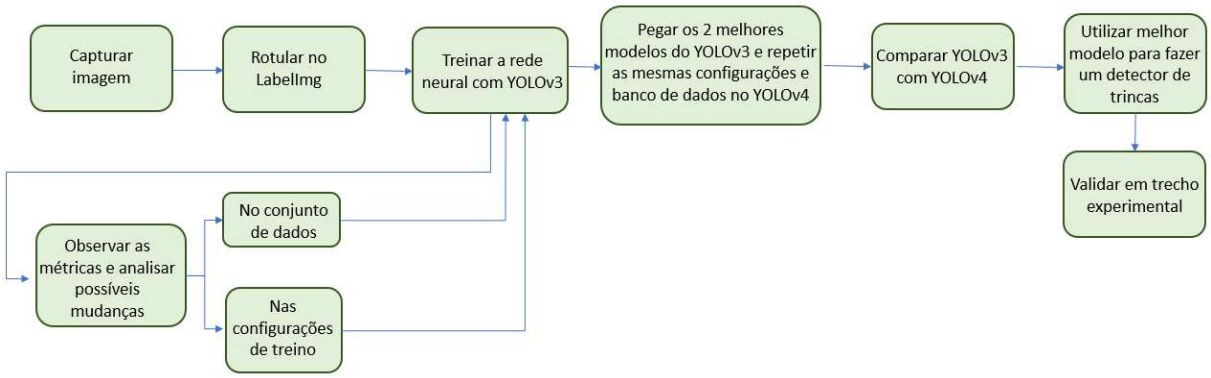
Explicando melhor alguns conceitos, segundo Oliveira (2020), o FPN é uma rede em que cria-se uma pirâmide, indo do topo para a base, utilizando conexões laterias, que possibilitam a construção de uma pirâmide de características ricas e multi-escala baseada em uma única imagem. O PanNet, segundo Erazo (2021), é um método de agregação que permite que os mapas de características vizinhas do fluxo descendente do *backbone* e o fluxo de características de cima para baixo sejam conectados antes de entrar nos blocos de predição.

Segundo Nascimento *et al.* (2021), o YOLOv5 não foi elaborado pelos mesmos desenvolvedores das versões anteriores, não possuindo, portanto, nenhuma ligação com as versões iniciais do YOLO. Além disso, até o momento, os autores não apresentaram artigo científico explicando os detalhes da estrutura e método utilizado na detecção, assim como também não foram publicados testes que atestem a real melhoria de desempenho desta versão. Sendo assim, optou-se por não utilizar o YOLOv5 na presente pesquisa.

4 MÉTODO

Este capítulo aborda o método utilizado na presente pesquisa, contendo as estratégias e procedimentos utilizados para construir um detector automático de trincas. O trabalho foi dividido em três etapas: criação do *dataset*, treinamento das redes e avaliação dos resultados, subdivididos em outras seções, cujos detalhes de cada etapa serão melhores descritos. A Figura 36 apresenta um fluxograma geral contendo as etapas da pesquisa.

Figura 36 - Etapas desenvolvidas na pesquisa



Fonte: autoria própria

4.1 Captura das imagens

O conjunto de dados utilizados nesta pesquisa consistiu em imagens do pavimento de diferentes localidades, capturadas com diferentes câmeras em diferentes posicionamentos. A maioria das imagens foram obtidas no estado do Ceará, embora o *dataset* também seja composto por imagens do estado de São Paulo, Alagoas, Rio Grande do Norte e algumas poucas imagens obtidas na Inglaterra. Todas as imagens foram obtidas em parceria com uma empresa privada, cuja sede se localiza em Fortaleza-CE, com exceção das imagens de Alagoas e Inglaterra. O banco de dados de imagens foi composto por quatro variações de defeitos de trincas do pavimento asfáltico, sendo elas, trincas longitudinais, trincas transversais, trincas couro de jacaré e trincas em bloco.

Parte das imagens do Ceará foram filmadas em rodovias estaduais localizadas no estado, mas especificamente as CEs 010, 025, 040, 060, 540, 455 e 352, sendo elas capturadas pela câmera Garmin VIRB Ultra30 Action, acoplada no capô e na traseira de um veículo, como ilustrado na Figura 37, produzindo imagens com dimensões de 3840 x 2160. As imagens desse conjunto não focam tanto no pavimento, apresentando uma boa faixa de céu,

como é possível observar na Figura 38. Esse conjunto apresenta 2.329 imagens que compõem o *dataset*.

Figura 37 - Posicionamento das câmeras para coleta de imagens da empresa A

Fonte: Empresa privada

Figura 38 - Exemplo de angulação da câmera das imagens obtidas junto à empresa A

Fonte: Empresa privada

O segundo conjunto de imagens do estado do Ceará foi filmado no trecho entre Fortaleza e Iguatu, com a mesma câmera, produzindo imagens com as mesmas dimensões citadas anteriormente, apresentando, no entanto, um diferente posicionamento de câmera, que permitiu que as imagens ficassem um pouco mais voltadas ao pavimento, como é possível observar na Figura 39. Esse conjunto apresenta 2.993 imagens que compõem o *dataset*.

Figura 39 - Exemplo de imagens obtidas junto à empresa privada, com a câmera voltada ao pavimento

Fonte: Empresa privada

O terceiro conjunto de imagens do estado do Ceará apresenta as dimensões 1920 x 1080, capturadas pela câmera Garmin VIRB Ultra30 Action. Essas imagens foram obtidas próximas à termoelétrica do Pecém. A imagem capturada não apresenta tanto céu, permitindo uma boa visualização do pavimento e suas patologias. A Figura 40 apresenta um exemplo das imagens desse conjunto. Esse conjunto apresenta 15 imagens que compõem o *dataset*.

Figura 40 - Exemplo de imagens obtidas próximo à termoelétrica do Pecém

Fonte: Empresa privada

As imagens obtidas no estado de Alagoas, mais especificamente na cidade de Maceió, foram filmadas com uma câmera GoPro, produzindo imagens de dimensão 4000 x 3000. Como é possível observar na Figura 41, a câmera também estava mais voltada para o pavimento, permitindo uma melhor visualização das trincas estudadas. Esse conjunto apresenta 314 imagens que compõem o *dataset*.

Figura 41 - Exemplo de imagens obtidas no estado de Alagoas

Fonte: Obtida junto ao representante da empresa privada em Alagoas

As imagens de São Paulo também apresentam dimensão de 3840 x 2160, e o poiscionamento da câmera não está muito voltado para o pavimento, como é possível observar na Figura 42. Esse conjunto apresenta 39 imagens que compõem o *dataset*.

Figura 42 - Exemplo de imagens obtidas no estado de São Paulo

Fonte: Empresa privada

As imagens do Rio Grande do Norte foram capturadas pela câmera Garmin, apresentando dimensões de 1920 x 1440. Pela Figura 43, é possível observar que a câmera estava localizada aparentemente na lateral do carro e com uma angulação que permitiu capturar uma parte considerável do carro, bem como do céu, fato este que dificulta um pouco a localização das trincas. Esse conjunto apresenta 1683 imagens que compõem o *dataset*.

Figura 43 - Exemplo de imagens obtidas no estado do Rio Grande do Norte

Fonte: Empresa privada

As imagens da Inglaterra foram obtidas junto a um colaborador, não sendo conhecido o tipo de câmera utilizada para obtenção das imagens. No entanto, o veículo utilizado para realização do levantamento de campo é similar ao apresentado na Figura 44, sendo bastante alto se comparado ao utilizado pela empresa privada. As imagens apresentam dimensões de 1600 x 1200, e o posicionamento da câmera permite pegar grande parte do entorno da via filmada, bem como do céu, como é possível observar na Figura 45. No entanto, a característica do pavimento permite uma boa visualização das trincas estudadas. Esse conjunto apresenta 202 imagens que compõem o *dataset*.

Figura 44 - Exemplo de veículo utilizado na obtenção das imagens da Inglaterra

Fonte: LCRIG. Disponível em: https://lcrig.org.uk/news/pts-announce-latest-multi-functional-vehicle-mfv-is-fully-ukpms-accredited-to-deliver-scanner-surveys. Acesso em 16 de dezembro de 2020

Figura 45 - Exemplo de imagens da Inglaterra

Fonte: Colaborador

4.2 Rotulação das imagens

Todas as imagens acima citadas foram rotuladas com o auxílio do software *LabelImg*, sendo consideradas quatro classes: trinca transversal, trinca longitudinal, trinca em bloco, e trinca couro de jacaré, nessa ordem. Cada imagem pode possuir mais de uma trinca, e essas trincas podem ser de diferentes tipos, conforme pode ser constatado na Figura 46, fato este que produziu uma quantidade de rótulos bem superior à quantidade de imagens.

Figura 46 - Exemplo de rotulação de trincas nas rodovias

Fonte: autoria própria

Tabela 6 mostra mais detalhes sobre os conjuntos de imagens utilizados na presente pesquisa, incluindo o quantitativo de rótulos de cada um.

Tabela 6 - Detalhes dos conjuntos de imagens

Origem	ID	Quantidade de imagens	Quantidade de rótulos			
			Transversal	Longitudinal	Em bloco	Couro de Jacaré
Ceará	CE010	45	0	1	8	67
	CE025	39	26	59	3	8
	CE040	512	305	718	47	156
	CE060	1.268	430	2.648	21	588
	CE269	125	219	329	2	5
	CE352	31	24	32	16	4
	CE455	289	82	176	12	492
	CE540	20	16	32	1	2
Ceará	VIRB0234	123	13	107	22	72
	VIRB0234-2	271	67	218	29	192
	VIRB0235	2.599	2.602	4.049	563	1.504
Ceará	-	15	1	22	2	4
Alagoas	-	314	226	410	22	68
São Paulo	-	48	25	49	5	6
Rio Grande do Norte	-	1.683	1.065	2.994	94	72
Inglaterra	-	202	60	280	3	38
Total		**7584**	**5161**	**12124**	**850**	**3278**
% de cada tipo de rótulo em relação ao total de rótulos		**-**	**24,10%**	**56,62%**	**3,97%**	**15,31%**

Fonte: autoria própria

Observando a Tabela 6, é possível notar que a quantidade de rótulos obtidos para os diferentes tipos de patologia não está na mesma proporção. As trincas isoladas, além de serem mais comuns nas regiões em que foram retiradas as imagens, são rotuladas individualmente, enquanto que nas trincas interligadas é realizado um único retângulo, fato este que gerou essa discrepância nos quantitativos. Para o processo de treinamento, o ideal era que a quantidade de rótulos para cada classe fosse mais ou menos parecida, para que não tivesse o favorecimento de uma classe em relação à outra. No entanto, apesar das tentativas de se obter mais imagens com defeitos no pavimento, infelizmente os dados obtidos não tinham a característica de apresentar trincas interligadas, principalmente trincas em bloco. Sendo assim, ao tentar fazer o balanceamento considerando os quantitativos da Tabela 6, a quantidade de imagens utilizadas seria insuficiente para fazer um treino adequado. Logo, optou-se por não levar em consideração esse fator, mesmo suspeitando que isso poderia impactar negativamente no resultado do treino.

Para o processo de rotulação, foram utilizadas as seguintes considerações:

- Longitudinais muito próximas umas às outras foram incluídas em um único retângulo com o rótulo ‘longitudinal’;
- Se as longitudinais estavam minimamente espaçadas, foram desenhados diferentes retângulos;
- Se houvesse trincas em um cruzamento, para definir o que seria ‘longitudinal’ e o que seria ‘transversal’, observou-se qual a via de maior fluxo (para a via mais larga, a trinca no sentido da via era rotulada como ‘longitudinal’;
- As trincas interligadas, em algumas situações, não estão muito bem definidas como apresentado na seção 2.2. Sendo assim, as trincas interligadas que formavam geometrias muito próximas umas às outras foram consideradas ‘couro de jacaré’. Se elas fossem um pouco mais espaçadas e com um formato mais bem definido, como um retângulo, eram consideradas como ‘bloco’;
- As trincas do acostamento também foram rotuladas, tendo em vista que a maioria das trincas em bloco se localizava nessa região, e a amostragem desse tipo de trinca não é muito comum.
- Só foram rotuladas como trincas transversais as trincas longas, ou curtas bem isoladas. As trincas muito curtas perpendiculares ao eixo da via que estavam coladas com trincas longitudinais foram consideradas ramificações da longitudinal, fazendo parte, portanto, da trinca longitudinal.
- O retângulo era desenhado de forma que uma de suas diagonais traçasse o caminho da trinca.

4.3 Procedimento de treinamento do modelo

O treinamento é um procedimento que envolve otimização matemática, envolvendo uma função de custo, chamada função de perda, para que os pesos e vieses da rede neural possam ser ajustados automaticamente. Neste processo, os dados de treinamento com rótulos esperados são apresentados ao procedimento de otimização para encontrar uma função de perda mínima global.

O treinamento da CNN requer um grande poder computacional, sendo necessário usar *Graphics Processing Units* (GPUs) que possam processar dados em paralelo. Para isso, os recursos de computação do *Google Colaboratory* (Google Colab) foram usados para

viabilizar o treinamento da CNN. Devido à natureza do Google Colab, cada treinamento é realizado em uma GPU alocada aleatoriamente pela Google.

Para o treinamento, são utilizados três arquivos: o arquivo com a extensão '.names', que contém os nomes dos rótulos considerados, sendo cada nome alocado em uma linha específica do arquivo; o arquivo com a extensão '.data', em que são gravados em variáveis específicas os caminhos das fotos de treino e teste, do arquivo '.names', da pasta backup em que são salvos os arquivos de pesos, bem como do número de classes estudadas; e o arquivo com a extensão '.cfg', que é o arquivo de configurações em que são definidos os valores de todos os parâmetros a serem considerados no treino.

Inicialmente foram realizados vários testes utilizando o YOLOv3, com verificação das métricas e possibilidades de melhorias no conjunto de dados ou nos parâmetros de configuração e, posteriormente, foi definido os dois melhores modelos treinados no YOLOv3 e o banco de imagens e as configurações foram repetidas para treino desses conjuntos no YOLOv4, a fim de verificar qual das duas versões do YOLO traz melhores resultados para a patologia estudada.

Os modelos são baseados em aprendizado de transferência, e os backbones usados para treinamento foram CSPDarknet53 para YOLOv4 e Darknet-53 para YOLOv3. Os tamanhos das imagens testados nos treinos foram de 416x416 pixels e 608x608 pixels (esses valores são padrões e são setados nos parâmetros *width* e *height*). Para todos os modelos desenvolvidos, foi definido um *batch* de 64 e *subdivisions* de 16, bem como foram utilizados os mesmos parâmetros de otimização, como *momentum* de 0,949, *decay* de 0,0005 e *learning rate* (LR) de 0,001. Em geral, esses parâmetros de otimização não são modificados, deixando os valores *default*. Os parâmetros que devem ser modificados, pois dependem da quantidade de classes a serem treinadas, são o *max_batches*, *steps*, *classes* e *filters*, cujas equações encontram-se representadas nas equações 5, 6, 7 e 8, em que 'n' representa o número de classes. O parâmetro *random* (r) foi definido inicialmente como sendo 0, sendo modificado posteriormente para 1 para verificação de melhora na acurácia.

Segundo Wang e Liao (2020), o número de interações recomendado pelo desenvolvedor YOLO é de 2.000 iterações para cada classe, com um número mínimo de 6.000 iterações. Durante o desenvolvimento desta pesquisa, foi analisada a expansão progressiva do número de iterações de treinamento, 1.000 de intervalo, com um valor mínimo de 1.000 a um valor máximo de 8.000.

$$\max\ \_batches = n \cdot 2.000 \geq 6.000 \quad (5)$$

$$steps\_\min = 0{,}8 \cdot \max\ \_batches \quad (6)$$

$$steps\_\max = 0{,}9 \cdot \max\ \_batches \quad (7)$$

$$filters = (n + 5) \cdot 3 \quad (8)$$

A Tabela 7 apresenta um resumo dos valores dos parâmetros utilizados no treino da rede neural.

Tabela 7 - Valores utilizados no treinamento das redes YOLOv3 e YOLOv4

Parâmetro	Valor
batch	64
subdivisions	16
n	4
max_batches	8.000
Steps	6.400, 7.200
filters	27

Fonte: autoria própria

Os parâmetros presentes no arquivo de extensão '.cfg' , de acordo com Nayak (2019), podem ser explicados como:

- *Batch*: este parâmetro indica o tamanho do lote usado durante o treinamento. O processo de treinamento envolve a atualização iterativa dos pesos da rede neural com base em quantos erros ela está cometendo no conjunto de dados de treinamento. É impraticável, e desnecessário, usar todas as imagens do conjunto de treinamento de uma só vez para atualizar os pesos. Assim, um pequeno subconjunto de imagens é usado em uma iteração, e esse subconjunto é chamado de lote (ou *batch*).
- *Subdivisions*: em alguns casos, não se tem GPU com memória suficiente para usar esse tamanho de lote definido, de 64. Para esses casos, o *darknet* permite especificar uma variável chamada *subdivisions* (ou subdivisões, em português) que permite processar uma fração do lote de uma só vez em uma GPU. A GPU processará o número 'lote/subdivisões' de imagens a qualquer momento, mas o lote completo ou a iteração só serão concluídos depois que todas as 64 (tamanho do lote definido) forem processadas. Durante o teste, tanto o lote quanto as subdivisões são definidos como 1.
- *Width, height, channels*: esses parâmetros de configuração especificam o tamanho da imagem de entrada e o número de canais. *Width* em português é largura, e *height* é altura e, para esta pesquisa, foi definido inicialmente os valores de 416x416, aumentando posteriormente para os tamanhos 608x608 e verificando se os resultados melhoram para treinos em imagens maiores (que levam mais tempo de treinamento).

Channels significa canais em português, e ao colocarmos o valor igual a 3, indica que estamos processando imagens de entrada RGB de 3 canais.

- *Momentum* e *decay*: esses parâmetros controlam como o peso é atualizado. Como foi dito anteriormente, os pesos são atualizados com base em um pequeno lote de imagens, o que acarreta em bastante flutuações nos valores. O parâmetro *momentum* (ou momento, em português) é usado para penalizar grandes mudanças de peso entre as iterações. E o parâmetro *decay* (ou decaimento, em português) controla valores de pesos muito grandes, pois tais valores podem indicar que ocorreu um *overfitting* no treinamento, ou seja, que a rede neural "memorizou" as respostas para todo o conjunto de treino, sem na verdade aprender o conceito por traz. Em geral usa-se os valores *default* para esses parâmetros.
- *Learning rate*, *steps* e *scales*: o *learning rate* (ou taxa de aprendizado, português) é um número entre 0,01 e 0,0001, e controla a agressividade com que a rede neural aprende com base no lote de dados. No início do processo de treinamento, a taxa de aprendizado precisa ser alta, tendo em vista que não tem muita informação e, com o tempo, a taxa de aprendizado vai diminuindo, visto que a rede já viu muitos dados. Esse decaimento na taxa de aprendizado é efetuado em etapas. Ao colocar a taxa de aprendizado em 0,001, ela vai iniciar com este valor e vai permanecer constante até o número de iterações definido no *step* e, em seguida, será multiplicada pelo fator *scale* (ou escala, em português) para obter a nova taxa de aprendizado.
- *Random:* As explicações sobre o parâmetro *random* não são tão claras, mas ele pode assumir valor igual a 0 ou igual a 1. Ao definir o valor igual a zero, em teoria, o treino ocorre de forma mais rápido, reduzindo, no entanto, a acurácia do modelo. Além disso, definir como 0 auxilia em casos em que não se tem memória suficiente na GPU. No *github* do *Alexeyab* é dito que, para um *random* igual a 1, haverá um aumento na precisão pois o YOLO será treinado para diferentes resoluções.

Definidos esses parâmetros, foram conduzidos vários treinos no YOLOv3 e no YOLOv4, cuja descrição do conjunto de treino e de teste e as configurações utilizadas encontram-se a seguir. Vale ressaltar que todos os treinos foram executados em uma placa gráfica NVIDIA GPU Tesla P100-PCIE com 16GB de memória RAM.

4.3.1 Treino 1

O primeiro treino foi realizado com todas as imagens que tinham sido rotuladas à época, consistindo em 6.269 imagens. Como o processo de rotulação é demorado, não se sabia ao certo se daria tempo rotular as 7.584 imagens que estavam disponíveis. Optou-se por realizar o primeiro treino e observar como o aprendizado da rede neural estava se comportando e, enquanto o treino era realizado (que, dependendo das configurações estabelecidas, pode demorar entre 6 a 12h para concluir), continuou-se o processo de rotulação das imagens.

Todas as pastas tinham sido rotuladas, com exceção do conjunto cujo ID é VIRB0235, presente na Tabela 6, que faltava rotular 1.315 imagens das 2.599. Sendo assim, as 6.269 imagens disponíveis e seus respectivos rótulos foram separados de forma aleatória em três conjuntos, sendo eles treino, teste e validação. Essa aleatoriedade foi possível com o auxílio de um código em *python* elaborado pela autora da presente pesquisa, presente no Apêndice A. Optou-se por deixar 1.000 imagens no conjunto de validação para que fosse possível identificar posteriormente quais os maiores erros que a rede neural estava cometendo. O restante das imagens foi divido em aproximadamente 80% para treino e 20% para teste. Para as configurações, foi definido que o *width* e o *height* iria ser 416, e o parâmetro *random* seria igual a 0. Os demais parâmetros permanecerem iguais aos valores presentes na Tabela 7. A Tabela 8 apresenta o quantitativo de rótulos de cada tipo de trinca existente nos conjuntos de treino e teste.

Tabela 8 - Quantitativos rótulos Treino 1

Conjunto	Quantidade de rótulos			
	Transversal	Longitudinal	Em bloco	Couro de Jacaré
Treino	2.448	7.015	311	1.731
Teste	552	1626	64	419

Fonte: autoria própria

4.3.2 Treino 2

Observando os resultados do treino 1, que serão expostos no próximo capítulo, observou-se que as métricas estavam muito aquém do esperado. Uma das hipóteses levantadas foi que as imagens cuja câmera não estava focando no pavimento talvez tivessem dificultando a aprendizagem. Sendo assim, optou-se por fazer um treino que considerasse apenas as imagens cuja câmera estava voltada para o pavimento. As configurações de treino

permaneceram as mesmas do treino 1, sendo modificado apenas o conjunto de imagens. Para esse treino, foram utilizadas as imagens de Alagoas, o segundo e terceiro conjuntos das imagens do Ceará, e as imagens da Inglaterra. O segundo conjunto das imagens do Ceará ainda estava com a rotulação incompleta, sendo assim, para este treino, foram utilizadas 2.216 imagens, que foram divididas em 80% para treino e 20% para teste. A Tabela 9 apresenta o quantitativo de rótulos de cada tipo de trinca existente nos conjuntos de treino e teste.

Tabela 9 - Quantitativos rótulos Treino 2

Conjunto	Quantidade de rótulos			
	Transversal	Longitudinal	Em bloco	Couro de Jacaré
Treino	1.123	2.616	192	946
Teste	305	629	46	242

Fonte: autoria própria

4.3.3 Treino 3

O terceiro treino foi realizado com as mesmas 2.216 imagens e com a mesma divisão de treino e teste realizada para o treino 2. A modificação foi nos parâmetros de tamanho da imagem, que passaram a ser 608 (width e height).

4.3.4 Treino 4

O treino 4 foi realizado com as mesmas imagens e as mesmas configurações do treino 2, modificando apenas o parâmetro random, que de 0 passou a ser 1. O intuito era verificar se essa configuração permite de fato uma melhora na acurácia do modelo.

4.3.5 Treino 5

Observando os erros mais comuns utilizando os pesos do treino 4, foi realizadouma revisão dos rótulos das imagens utilizadas no treino 2, a fim de verificar se não foi colocado algum rótulo errado, se deixou de rotular alguma trinca, se os retângulos estavam muito grandes, etc. Além disso, o restante das imagens que faltavam ser rotuladas foram incluídas nesse treino também. As configurações de treino ficaram iguais às do treino 2. O total de imagens foi de 3.530, divididas em 3 conjuntos, sendo 60% para treino, 20% para teste e 20% validação. No conjunto de validação realiza-se uma verificação visual do desempenho da

aprendizagem da rede, observando os erros mais comuns cometidos. A Tabela 10 apresenta o quantitativo de rótulos de cada tipo de trinca existente nos conjuntos de treino e teste.

Tabela 10 - Quantitativos rótulos Treino 5

Conjunto	Quantidade de rótulos			
	Transversal	Longitudinal	Em bloco	Couro de Jacaré
Treino	1.670	3.110	384	1.026
Teste	519	1094	122	371

Fonte: autoria própria

4.3.6 Treino 6

Para o treino 6, foi realizados um pre-processamento nas imagens cuja câmera permitia a captura de muito céu. Para que houvesse um aumento da quantidade de imagens disponível para treinamento, e considerando que, dentre as imagens disponibilizadas pela empresa privada e pelos colaboradores, não tinha mais nenhuma com a patologia de interesse dessa pesquisa, optou-se por usar as imagens disponíveis, fazendo um leve tratamento nessas imagens. De início, as imagens passaram por um processo de corte, em que foi descartado o céu, e em seguida elas foram redimensionadas para o tamanho original, fato este que distorceu as imagens. Essa distorção possibilitou um zoom no pavimento, deixando as patologias mais visíveis. Essa operação foi realizada com o auxílio de um código em python, , e se encontra no Apêndice B. Em seguida, foi aplicado um contraste automático nessas imagens (Apêndice C), que destacou levemente as trincas que estavam muito claras nos conjuntos do Ceará e São Paulo. Optou-se por deixar as imagens do Rio Grande do Norte de fora pois, mesmo após essas operações nas imagens, ainda não era possível visualizar a trinca de forma explicita. A Figura 47 mostra o antes e depois das imagens modificadas.

Figura 47 – Pré-Processamento na imagem a) original b) corte c) contraste

a) b) c)

Fonte: autoria própria

Foram testadas outras ações de pré-processamento de imagem, a exemplo de aplicações de filtros e tentativas de binarização (cujas imagens só apresentam as cores preto e branco), com o objetivo de destacar as trincas do pavimento e do entorno. No entanto, essas

tentativas não se mostraram bem sucedidas, e o pré-processamento das imagens ficaram limitados nas operações de corte, redimensionamento e contraste. Vale ressaltar que, com essas operações, os rótulos iniciais perderam a referência de localização, já que as imagens foram cortadas e redimensionadas. Sendo assim, elas tiveram que passar por um processo de revisão para que o treino 6 pudesse ser realizado. A Figura 48 mostra algumas tentativas de binarização da imagem que não renderam resultados positivos.

Figura 48 - Tentativa binarização a) original b) binarizada

a) b)

Fonte: autoria própria

Sendo assim, para compor o conjunto de imagens a ser utilizado no treino 6, foram incluídas as imagens do treino 5 com aplicação de contraste, as imagens de São Paulo cortadas com aplicação de contraste, e algumas imagens do primeiro conjunto do Ceará cortadas e com contraste, mais especificamente as CEs 010, 025, 040, 269, 352, 455 e 540. As imagens da CE060 não foram incluídas pois a revisão dos rótulos iria demorar, não sendo concluídas em tempo hábil. O total de imagens utilizadas foi de 4.072, divididas em três conjuntos, sendo 75% para treino, 15% para teste e 5% para validação. As configurações de treino seriam iguais às do treino 5, no entanto, por limitação de memória da GPU, não foi possível deixar o *random* = 1. Sendo assim, a imagem ficou com a dimensão de 416x416 e *random* = 0. A Tabela 11 apresenta o quantitativo de rótulos de cada tipo de trinca existente nos conjuntos de treino e teste.

Tabela 11 - Quantitativos rótulos Treino 6

Conjunto	Quantidade de rótulos			
	Transversal	Longitudinal	Em bloco	Couro de Jacaré
Treino	2.153	4.135	451	1.724
Teste	606	1143	113	428

Fonte: autoria própria

4.3.7 Treino 7 e 8

Os treinos 7 e 8 foram realizados utilizando o YOLOv4, para comparar qual das versões do YOLO apresenta melhores resultados na aprendizagem das patologias em questão. Para isso, observou-se quais eram os dois modelos do YOLOv3 que apresentaram melhores métricas, que foram nos treinos 5 e 6, como veremos melhor no próximo capítulo. Sendo assim, o treino 7 foi realizado com as mesmas imagens do treino 5, bem como as mesmas configurações, e o treino 8 foi realizado com as mesmas imagens e mesmos parâmetros de configuração do treino 6. A Tabela 12 apresenta o quantitativo de rótulos de cada tipo de trinca existente nos conjuntos de treino e teste.

Tabela 12 - Resumo das características dos conjuntos de treino

Treino	Algoritmo	Quantidade de imagens (treino e teste)	Dimensão imagem	random
1	YOLOv3	5.269	416 x 416	0
2	YOLOv3	2.216	416 x 416	0
3	YOLOv3	2.216	608 x 608	0
4	YOLOv3	2.216	416 x 416	1
5	YOLOv3	2.824	416 x 416	1
6	YOLOv3	3.868	416 x 416	0
7	YOLOv4	2.824	416 x 416	1
8	YOLOv4	3.868	416 x 416	0

Fonte: autoria própria

4.4 Métricas da avaliação de desempenho/peformance

Foram utilizados cinco métricas diferentes para avaliar o desempenho da aprendizagem da rede neural, a saber: *intersect over union* (IoU, em português é interseção sobre a união), *precision* (precisão), *recall* (sensibilidade), *F1-score*, *average precision* (AP, em português é precisão média) e *mean average precision* (mAP, em português é medias das precisões de todas as métricas).

A IoU pode ser calculada como a área de intersecção dividida sobre a área de união de duas caixas, portanto, a IoU deve estar entre 0 e 1. Ao prever caixas delimitadoras, é preciso encontrar o IoU entre a caixa delimitadora prevista e a caixa de verdade fundamental para ser ~ 1. A equação 9 apresenta o cálculo do IoU.

$$IoU = \frac{Área\ de\ interseção}{Área\ da\ União} \tag{9}$$

O parâmetro *precision* pode definir a precisão como a proporção de verdadeiros positivos - (previsões verdadeiras) e o número total de positivos previstos (previsões totais – verdadeiros positivos VP mais falsos positivos FP). A equação 10 apresenta o cálculo de *precision*.

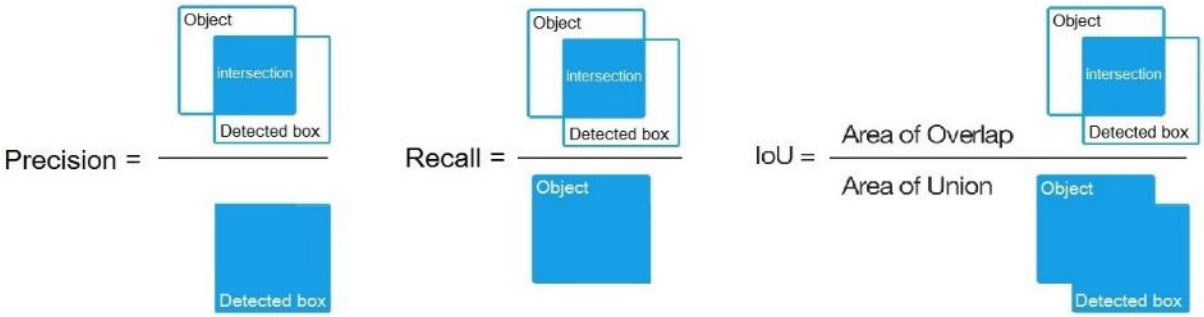
$$Precision = \frac{VP}{VP + FP} \tag{10}$$

Recall é a proporção de verdadeiros positivos (previsões verdadeiras) e o total de verdadeiros positivos (verdadeiros positivos VP mais falsos negativos FN). Na equação 11 é apresentado o cálculo de *recall.*

$$Recall = \frac{VP}{VP + FN} \tag{11}$$

A Figura 49 apresenta representações esquemáticas das equações 9, 10 e 11, para melhor visualização.

Figura 49 - Figura esquemática das fórmulas das métricas



Fonte: Github Alexeyab. Disponível em: https://github.com/alexeyab/darknet. Acesso em: 10/01/2021

O parâmetro F1-score é uma média harmônica dos parâmetros *precision e recall*, sendo representada pela equação 12. A média harmônica dá mais peso aos valores baixos, sendo assim, só será obtida uma pontuação alta de F1-score se tanto o *recall* quando as precisões forem altas.

$$F_1 = \frac{VP}{VP + \frac{FN+FP}{2}} \tag{12}$$

Average precision (AP) e *mean average precision* (mAP): uma breve definição para a precisão média é a área sob a curva de recuperação de precisão. AP combina precisão e recall. Tem um valor entre 0 e 1 (quanto maior, melhor). Para obter AP = 1, é necessário que a precisão e a recuperação sejam iguais a 1. O mAP é a média do AP calculado para todas as classes.

Segundo o desenvolvedor do YOLO, o treino deve ser parado quando o *average loss* não diminuir mais ao longo das iterações. Além disso, enquanto o mAP continuar aumentando, o treino deve continuar, ao contrário deve-se interromper o treino. Para definição

de qual peso apresenta as melhores métricas, ainda segundo *AlexeyAB*, os pesos que apresentarem maior mAP ou maior IoU são os melhores.

Embora a acurácia seja a métrica geralmente mais usada, dependendo do contexto estudado, cada métrica apresenta uma importância diferente. Quando se trata de Gerência dos Pavimentos, deixar de indicar uma trinca, principalmente se for interligada, é pior do que identificar uma trinca onde não existe, pois a evolução de trinca interligada para buraco em geral não é muito demorada, sendo o buraco o pior defeito do pavimento, que pode causar graves acidentes. Considerar a pior situação permite que os órgãos responsáveis tenham tempo de planejar a intervenção necessária em um período otimizado de forma a evitar o surgimento de grandes patologias que afetam a segurança do usuário.

4.5 Análise trecho experimental

O trecho escolhido para ser estudado foi a CE060, tendo em vista que suas imagens não foram consideradas no treino por não conseguirem ser revisadas a tempo. Como a base de dados já está toda rotulada manualmente, é possível pegar o peso que apresenta as melhores métricas e com ele fazer um detector que vai passar por todas as imagens da pasta e indicar a posição e o tipo de trinca existente no pavimento, a fim de fazer uma comparação entre os rótulos obtidos manualmente e os rótulos provenientes do detector.

Essa análise vai levar em consideração a quantidade de imagens que ele conseguiu identificar algum tipo de trinca se comparado à quantidade total de imagens com essas patologias, além de considerar se os rótulos foram indicados corretamente e se foram rotuladas de fato todas as trincas da imagem. A questão do posicionamento em si não vai ser levada em consideração, a menos que esteja representada em um local muito destoante de onde o rótulo era para estar de fato.

5 RESULTADOS E DISCUSSÕES

Este capítulo é dedicado à apresentação dos resultados dos treinos considerando diferentes cenários realizados no contexto do presente trabalho. Além disso, apresenta o estudo realizado em um trecho experimental a fim de comparar os resultados obtidos no modelo com os resultados junto à inventariação manual.

5.1 Análise dos treinos

A Tabela 13 apresenta os resultados dos treinamentos da rede neural realizados no presente trabalho. Observando-se as métricas de avaliação e levando-se em consideração as características de cada treino, como conjunto de imagens componentes e configurações estabelecidas, pode-se chegar a algumas conclusões sobre melhor posicionamento de câmera e melhor versão do YOLO para fazer a detecção das patologias estudadas de forma automática, dentre outras.

Tabela 13 - Resultado das métricas dos oito treinamentos realizados

Iterações	AP (%)				Precision (%)	Recall (%)	F1-Score (%)	Average IoU (%)	mAP@ 0.50 (%)
	Transversal	Longitudinal	Em bloco	Couro de jacaré					
Treino 1									
1000	7,68	9,79	8,79	20,29	36	5	9	23,06	11,64
2000	17,33	19,54	34,26	30,25	53	14	23	35,65	25,34
3000	16,42	19,58	25,89	34,08	55	15	23	38,39	23,99
4000	18,53	18,00	28,99	37,80	45	21	29	30,83	25,83
5000	18,21	19,67	30,04	34,99	48	21	29	33,10	25,73
6000	**16,69**	**20,53**	**29,45**	**37,49**	**44**	**25**	**32**	**30,20**	**26,04**
7000	14,86	20,47	21,23	35,66	48	23	31	33,01	23,05
8000	14,82	19,93	21,94	35,67	47	23	30	32,66	23,09
best	**18,15**	**21,39**	**33,70**	**40,27**	**46**	**23**	**31**	**31,55**	**28,38**
Treino 2									
‘1000	13,03	21,93	16,76	44,34	50	17	25	33,14	24,01
2000	17,82	27,92	48,47	59,02	59	26	36	41,03	38,31
3000	18,03	25,77	35,17	57,56	56	27	37	39,26	34,13
4000	17,20	25,42	45,93	60,55	54	29	38	37,57	37,27
5000	11,68	23,67	33,08	52,94	57	25	35	40,97	30,34
6000	16,89	23,75	45,50	56,96	56	29	38	38,99	35,77
7000	18,94	28,07	53,70	60,93	60	32	42	42,85	40,41
8000	**19,47**	**28,00**	**52,52**	**61,87**	**60**	**32**	**41**	**43,08**	**40,46**
best	**20,26**	**27,62**	**53,52**	**63,11**	**60**	**32**	**42**	**43,38**	**41,13**

Continuação Tabela 13 – Resultado das métricas dos oito treinamentos realizados

Iterações	AP (%)				Precision (%)	Recall (%)	F1-Score (%)	Average IoU (%)	mAP@ 0.50 (%)
	Transversal	Longitudinal	Em bloco	Couro de jacaré					
Treino 3									
1000	10,53	16,51	13,67	35,48	49	11	18	31,57	19,05
2000	20,61	31,36	42,83	62,84	56	30	40	39,12	39,41
3000	19,80	28,35	34,80	58,18	49	34	40	33,99	35,28
4000	15,01	19,60	38,42	50,52	57	22	32	40,83	30,89
5000	16,51	25	41,08	58,40	52	30	38	37,10	35,25
6000	18,00	22,20	38,10	60,16	54	31	39	37,69	34,62
7000	18,82	27,38	50,26	60,80	54	33	41	39,01	39,32
8000	**19,57**	**28,23**	**50,68**	**59,27**	**56**	**34**	**42**	**40,52**	**39,44**
best	**19,45**	**28,40**	**51,51**	**59,87**	**56**	**33**	**42**	**40,55**	**39,81**
Treino 4									
1000	5,34	14,48	17,43	43,57	55	5	10	35,92	20,21
2000	18,52	24,69	40,30	61,98	61	22	32	43,76	36,37
3000	17,45	24,53	40,42	60,11	50	29	36	35,25	35,63
4000	17,91	23,01	30,36	60,57	51	29	37	36,39	32,96
5000	17,92	24,59	40,16	60,89	58	28	38	40,58	35,89
6000	17,08	21,64	32,61	55,95	56	25	35	40,01	31,82
7000	22,05	25,46	55,83	60,18	59	31	41	42,15	40,88
8000	**24,11**	**25,15**	**51,73**	**63,50**	**58**	**33**	**42**	**41,46**	**41,12**
best	**22,97**	**27,23**	**51,40**	**65,50**	**57**	**32**	**41**	**40,03**	**41,78**
Treino 5									
1000	14,33	22,41	36,19	41,01	55	15	23	37,59	28,49
2000	27,72	31,76	43,68	56,04	50	33	40	35,07	39,80
3000	27,50	32,98	52,81	61,83	61	30	40	43,08	43,78
4000	27,67	34,50	59,63	64,01	58	38	46	41,11	46,45
5000	30,72	28,78	41,02	55,53	52	34	42	36,22	39,01
6000	28,33	32	56,94	60,30	57	37	45	40,51	44,39
7000	27,19	31,19	55,54	63,86	61	36	45	43,99	44,45
8000	**28,23**	**31,99**	**58,84**	**64,52**	**59**	**38**	**47**	**42,28**	**45,90**
best	**29,69**	**32,50**	**60,16**	**65,83**	**59**	**39**	**47**	**42,67**	**47,04**
Treino 6									
1000	22,08	19,40	47,80	36,93	53	15	24	35	31,55
2000	28,80	29,46	41,82	49,57	56	26	36	38,53	37,41
3000	28,11	31,06	52,27	51,56	58	30	40	40,44	40,75
4000	22,38	26,7	48,41	49,02	51	30	38	35,64	36,63
5000	22,39	28,43	50,85	49,13	56	29	39	39,24	37,7
6000	22,43	23,57	60,89	46,82	48	31	38	33,77	38,43
7000	25,81	28,14	63,39	53,48	58	33	42	41,98	42,71
8000	**25,45**	**29,35**	**64,29**	**53,36**	**59**	**34**	**43**	**42,15**	**43,11**
best	**26,64**	**29,24**	**63,56**	**53,11**	**56**	**34**	**42**	**40,41**	**43,14**
Treino 7									
1000	16,03	20,64	21,94	32,64	45	9	15	29,43	22,81
2000	21,27	29,70	38,62	59,80	56	25	35	37,7	37,35
3000	29,38	38,99	58,62	59,70	47	44	46	33,41	46,67
4000	31,11	39,34	55,35	57,15	58	40	48	42,03	45,74
5000	29,91	33,15	54,32	61,54	52	41	46	37,43	44,73
6000	31,72	35,39	57,70	60,82	53	44	48	38,18	46,41
7000	31,56	32,48	59,08	60,98	56	41	47	40,39	46,02
8000	**32,21**	**33,99**	**60,61**	**64,03**	**57**	**43**	**49**	**41,08**	**47,71**
best	**33,35**	**39,33**	**66,39**	**65,55**	**60**	**40**	**48**	**43,17**	**51,15**
Treino 8									
1000	18,96	18,15	45,22	37,65	51	12	19	33,83	30
2000	33,50	35,67	57,74	53,48	56	33	41	39,19	45,10
3000	28,37	36,92	62,53	55,58	58	37	45	40,58	45,85
4000	**32,63**	**34,79**	**62,82**	**54,97**	**55**	**40**	**46**	**39,06**	**46,30**
5000	31,89	30,49	55,11	53,47	54	39	45	38,34	42,74
6000	28,52	28,90	62,58	50,32	54	36	43	38,32	42,58
7000	31,35	30,87	68,02	51,58	56	39	46	40,75	45,45
8000	**30,88**	**30,38**	**68,79**	**51,96**	**56**	**39**	**46**	**40,83**	**45,50**

Fonte: autoria própria

Iniciando pelos primeiros dois treinos realizados, pode-se concluir que o treino 2 apresentou melhores métricas combinadas se comparado ao treino 1, indicando que a limitação da utilização apenas de imagens em que a câmera estivesse voltada para o pavimento surtiu um efeito positivo. Embora a precisão média das trincas isoladas não tenha apresentado grande melhoria, as trincas interligadas apresentaram um aumento em torno de 20% nessa mesma métrica. Além disso, as demais métricas apresentaram um aumento de mais de 10% de um treino para outro.

Contrariando a literatura, em que se afirma que o aumento das dimensões da imagem proporciona um aumento na acurácia, o treino 2 apresentou melhores métricas combinadas se comparado ao treino 3, cuja dimensão das imagens foi de 608 x 608. Embora os valores individuais das métricas dos dois treinos não estejam muito distantes, e em algumas situações o treino 3 supere o 2, no geral o 2 se saiu melhor, significando que, para este conjunto de dados e de defeitos, o aumento da imagem não impactou de forma significativa os resultados.

No que se refere à comparação entre os treinos 2 e 4, em que a diferença entre eles é o parâmetro *random*, não se chegou a uma conclusão exata quanto ao que apresenta melhor desempenho. Embora o treino 4 apresente mais métricas com valores levemente superiores, o treino 2 apresenta uma precisão e um IoU 2% superiores ao treino 4. Um fato curioso ao analisar estes dois treinos é que a disposição dos melhores APs das quatro classes é a mesma. Para as trincas transversais e couro de jacaré, o melhor AP é o da iteração 7.000, e para as trincas longitudinais e bloco é o da iteração 8.000. Diferentemente do que a literatura sugeriu, a alteração do valor de *random* para 1 não trouxe melhoras significativas nas métricas, fato este que talvez indique que estre parâmetro não foi tão relevante para este conjunto de dados.

O treino 5 apresentou o melhor resultado entre os treinamentos realizados no YOLOv3. Ao compararmos ele ao treino 2, os melhores valores de AP do treino 2 ocorreram por volta da iteração 7.000 em diante, enquanto esses valores foram superados já por volta da iteração 3.000 do treino 5. O *recall* e o F1-score aumentaram 6% e o mAP@0.50 aumentou em 4%. As demais métricas tiveram uma leve redução, que talvez não seja tão relevante se comparado aos demais resultados. No entanto, não é possível afirmar com certeza a causa da melhoria dos resultados, tendo em vista que foram efetuadas várias modificações no conjunto de imagens: as imagens utilizadas no treino 2 tiveram seus rótulos revisados e, além disso, houve um incremento de mais imagens cuja câmera estava voltada para o pavimento. Além disso, o parâmetro *random* estava igual a 1. Uma das hipóteses possíveis de se levantar é que

este parâmetro afete melhor, positivamente, em conjunto de dados maiores (desde que não haja problema de memória na GPU alocada).

O treino 6, embora tenha incorporado as imagens que passaram por um certo nível de pré-processamento, como corte e operação de contraste, não conseguiu apresentar um resultado superior ao treino 5, indicando que não basta a imagem estar com um certo nível de zoom, a angulação da câmera voltada mais para baixo é de grande valia para a obtenção de resultados melhores. Com exceção do valor de AP para o tipo de trinca 'bloco', todos os demais valores foram superiores para o treino 5.

Analisando agora o efeito da mudança da versão do YOLO nos resultados, comparando inicialmente o treino 5 com o treino 7, é possível observar que o treino 7 apresenta melhores métricas, aumentando em quase 3% na maioria dos casos. Apenas o IoU que deu uma leve reduzida, mas que pode ser considerada irrelevante se comparado à melhora das demais métricas. Um fato curioso de se citar é que no treino 5, os melhores valores de AP ocorreram por volta da iteração 4.000 e, a partir disso os valores foram diminuindo, acontecendo o mesmo para o parâmetro mAP@0.50, enquanto o treino 7 apresenta um padrão crescente da maioria das métricas em relação ao aumento das iterações. Este comportamento do treino 5 indica que pode ter tido um *overfitting*, situação esta em que o treino fica muito ajustado aos dados de treinamento, não apresentando grande capacidade de generalização.

Comparando agora os treinos 8 e 6, o treino 8 apresenta melhores métricas combinadas se comparado ao treino 6. O AP das trincas isoladas apresentou um aumento considerável no treino 8, em torno de 4%, enquanto as demais métricas apresentaram uma melhora de 3%, ficando apenas o IoU fora desse padrão, apresentando uma leve piora. Pode-se perceber que tanto o treino 7 como o treino 8 apresentaram melhora na acurácia, em detrimento de uma leve piora no parâmetro IoU. Sendo assim, de modo geral, é possível afirmar que, para a detecção de patologias no pavimento do tipo trinca, o YOLOv4 apresenta melhores resultados.

Olhando para a Tabela 13, é possível perceber facilmente que as trincas interligadas (trinca em bloco e trinca couro de jacaré) apresenta valores de AP maiores se comparado ao valor das trincas isoladas (trinca transversal e trinca longitudinal), isso em todos os oito treinos que foram executados para a realização da pesquisa. Uma possível explicação para isso é que as trincas isoladas ocupam um espaço muito pequeno da imagem, pois são muito finas, fato este que dificulta sua visualização, enquanto as trincas interligadas ocupam um espaço considerável da imagem na maioria dos casos. Segundo a literatura, o YOLO

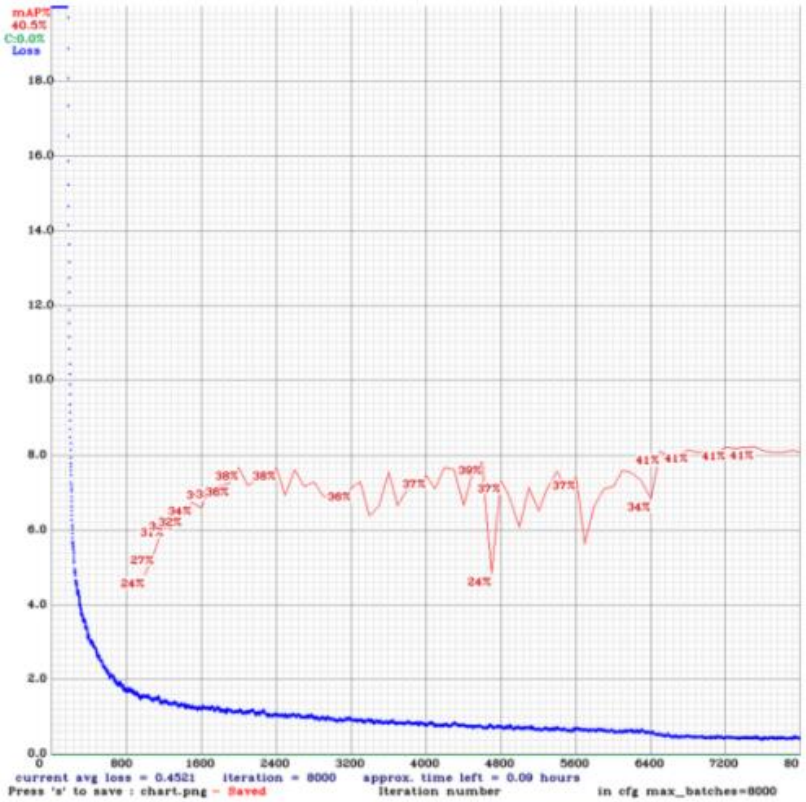
apresenta dificuldade em detectar objetos pequenos, sendo essa, portanto, uma possível justificativa para os resultados obtidos junto às trincas isoladas.

Embora as trincas interligadas apresentem melhores resultados, eles ainda estão muito aquém do esperado, que seria acima de 90% de acurácia (desejado). Uma possível justificativa para os baixos valores de acurácia pode recair sobre a necessidade de se ter um banco de imagens maior, contendo mais imagens cuja câmera esteja mais voltada para o pavimento, e de preferência que haja representatividade de todos os tipos de trinca de forma igualitária. Foi possível observar na Tabela 6 a discrepância entre os quatro tipos de trincas. Um fato curioso é que, mesmo havendo uma quantidade superior de trincas isoladas, ainda assim o resultado do AP foi bastante inferior ao das trincas interligadas, fato este que reforça a suposição da dificuldade de se enxergar esse tipo de trinca, principalmente em imagens cujo enfoque não é o pavimento.

No que se refere às iterações, no geral observa-se que as métricas apresentam melhores resultados com o aumento das iterações, sendo poucos os casos em que esse padrão não se repete. Isso só não acontece muito quando se observa a métrica AP, que em alguns casos oscila um pouco. Além disso, é possível notar que, para grande parte dos treinos executados, as melhores métricas combinadas ocorreram na iteração 8.000. Seguindo a literatura, é recomendado parar o treino quando o erro não diminui à medida que as iterações vão ocorrendo, ou quando o valor de mAP@0.50 não apresenta mais incrementos significativos. Observando os gráficos dos treinos 2 e 5 presentes na Figura 50 e na Figura 51, e considerando que os melhores resultados em geral estão na iteração 8.000, é possível afirmar que, pela tendência dos gráficos, espera-se que o mAP@0.50 aumente ainda mais com o acréscimo de iterações superiores a 8.000 e que a perda diminua ainda mais. Esta tendência também pode ser observada nos gráficos dos demais treinos. Sendo assim, sugere-se que, para os próximos treinos, seja colocado um valor de *max_batches* de 10.000 para observar a tendência do gráfico.

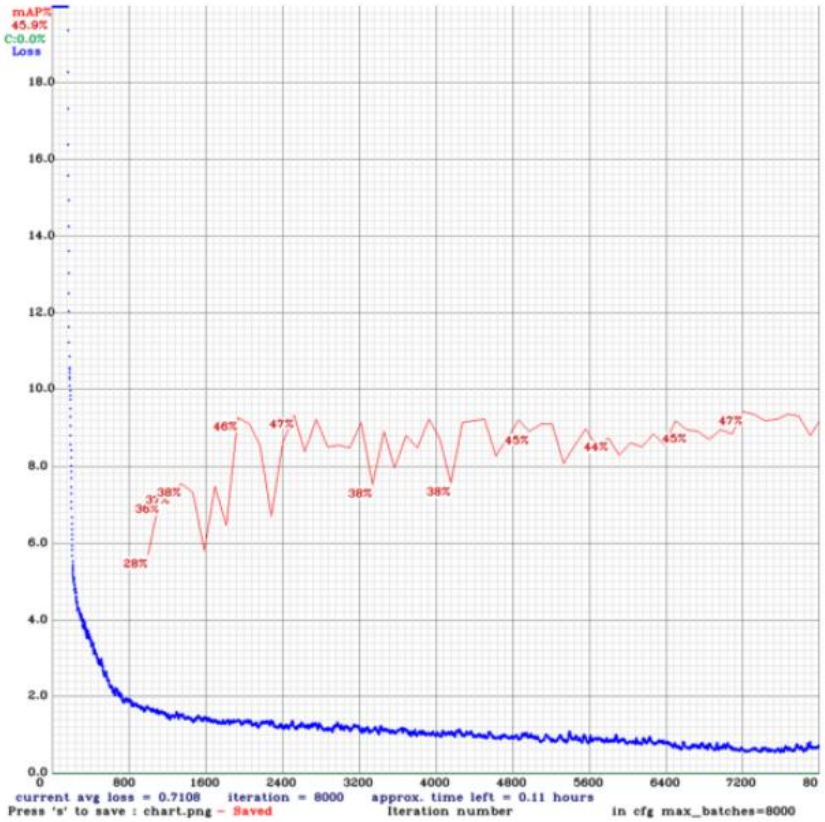
Na Tabela 13, as linhas em laranja representam os melhores pesos obtidos para cada treino executado (excetuando-se os pesos nomeados com *best*), e as células em verde representam os maiores valores de AP para cada treino considerando cada classe isoladamente.

Figura 50 - Gráfico da perda e do mAP pelas iterações do treino 2



Fonte: autoria própria

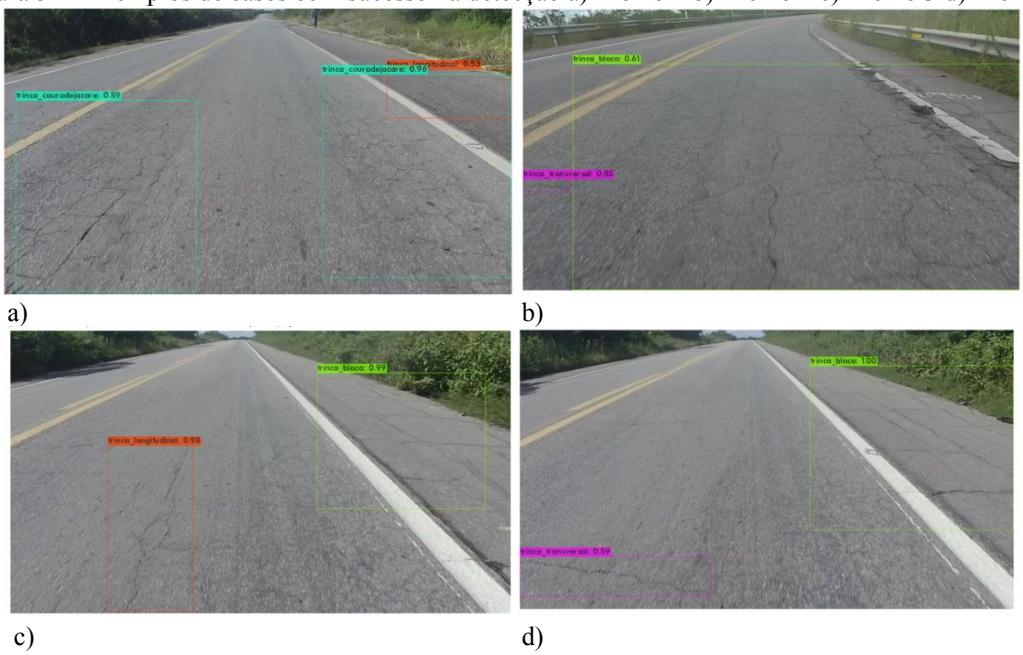
Figura 51 - Gráfico da perda e do mAP pelas iterações do treino 5



Fonte: autoria própria

As imagens presentes nas figuras 52 e 53 apresentam os casos de sucesso e insucesso durante o processo de detecção utilizando-se pesos de diferentes treinos. Na Figura 52 é possível observar que o algoritmo conseguiu acertar a classe das trincas com precisões relativamente altas, apresentando o retângulo relativamente ajustado ao tamanho das trincas, fato este muito positivo, pois vai auxiliar futuramente em pesquisas cujo objetivo é mensurar o tamanho das trincas, a área trincada do pavimento, dentre outras coisas.

Figura 52 - Exemplos de casos com sucesso na detecção a) Treino 1 b) Treino 2 c) Treino 3 d) Treino 4



a) b)

c) d)

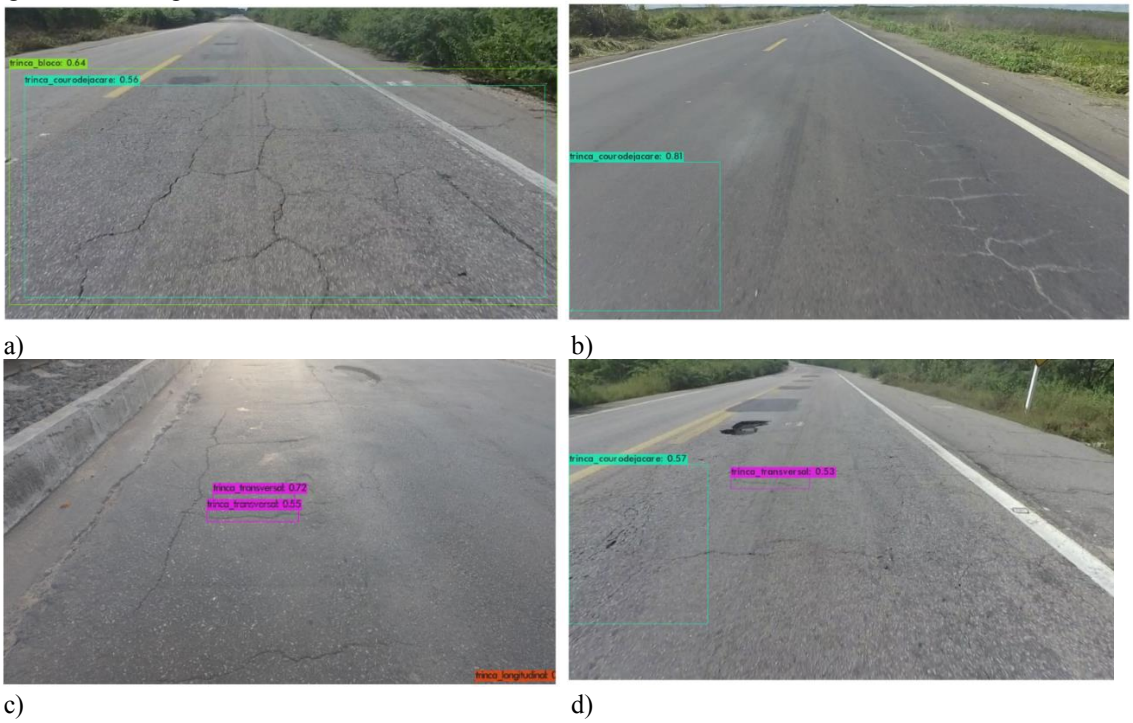
Fonte: autoria própria

A Figura 53 apresenta casos em que a detecção não foi efetuado de forma assertiva. Na Figura 53 a), é possível ver que o algoritmo detectou dois tipos diferentes de trinca para o mesmo conjunto de trincas, e com uma precisão de mais de 50%. Isso mostra que, em alguns casos, ele não está conseguindo diferenciar muito bem o que é uma trinca em bloco e o que é uma trinca couro de jacaré. Seguindo o critério de rotulação estabelecido, em que se considera trinca em bloco se o espaçamento entre as interligações for relativamente grande, em uma rotulação manual a classe que seria estabelecida para esse conjunto de trincas seria em bloco. No entanto, faz sentido a confusão cometida, tendo em vista que em alguns casos também houve dúvidas durante a rotulação manual. Para estudos futuros, talvez fosse interessante unir as duas classes em uma só para aumentar o nível de acerto. Tal estratégia não apresentaria grandes problemas tendo em vista que, em casos em que estuda-se a severidade da patologia no pavimento, o peso dado aos índices é o mesmo valor tanto para trincas em bloco quanto

para trincas interligadas, fato este que permite, portanto, a unificação dos rótulos, já que não traria grandes prejuízos.

Na Figura 53 b), embora o algoritmo tenha acertado a classe da trinca, errou completamente a posição. O conjunto de trincas que era para ter sido identificado encontra-se à direita, sem nenhum retângulo indicando sua classificação, enquanto há um retângulo delimitando uma área sem nenhuma patologia, e ainda com um nível de precisão elevado.

Figura 53 - Exemplos de casos com insucesso na detecção a) Treino 3 b) Treino 6 c) Treino 6 d) Treino 4



Fonte: autoria própria

Nas Figuras 53 c) e d), observa-se a dificuldade que o algoritmo apresenta em detectar trincas transversais, mesmo elas apresentando certo nível de nitidez na imagem. Além disso, na Figura 53 c), o algoritmo também não foi capaz de identificar a trinca longitudinal presente, fato este que reforça a dificuldade em encontrar as trincas isoladas.

5.2 Análise de trecho experimental

Como foi dito na seção 4.3.6, as imagens da CE060 não foram utilizadas no treino 6 pois não tinham passado pelo processo de revisão dos rótulos depois da operação de corte da imagem. Sendo assim, optou-se por utilizá-las para fazer uma análise do desempenho dos pesos frente à rotulação manual. Para fazermos este teste, foi utilizado os pesos provenientes do treino 8. Embora o treino 7 tenha sido declarado o melhor modelo testado na presente

pesquisa, muito em função da característica das imagens que o compuseram, ele não foi treinado para reconhecer trincas com imagens no padrão da CE060, já que não tem nenhuma nesse estilo no seu banco de imagens de treino, diferentemente do treino 8, que teve imagens similares como as das demais CEs disponíveis.

Sendo assim, foi escolhido o peso da iteração 8.000 do treino 8 para fazer um detector que passe pelas imagens da CE060 e gere txts com as coordenadas e classe dos rótulos para que seja possível visualizá-los no *labelImg*. As imagens originais da CE060 passaram por um processamento, semelhante ao executado nas imagens utilizadas no treino 6, porém com leves diferenças: a aplicação de contraste se manteve, porém, as imagens foram recortadas em posição diferente e não passaram pelo processo de redimensionamento. A Figura 54 mostra o antes e depois do processamento realizado.

Figura 54 – Antes e depois do processamento das imagens da CE060

Fonte: autoria própria

Com o auxílio de um código em *python* os rótulos gerados pelo *labelImg* foram recortados juntamente com a imagem, possibilitando o reaproveitamento do trabalho original. Este código encontra-se no Apêndice D e foi adaptado de Neskorozhenyi (2021). Devido ao corte da região de interesse, muito rótulos se perderam, assim como algumas imagens também. Além disso, as imagens passaram por um processo de revisão, para garantir que os rótulos estivessem de acordo com a convenção exposta na Seção 4.2. Sendo assim, há um novo quantitativo de imagens e um novo quantitativo de rótulos, como é possível observar na Tabela 14.

O objetivo é comparar a quantidade de cada tipo de trinca que fora obtida na rotulação manual e com o auxílio do detector e, além disso, discorrer sobre os erros mais comuns. A

quantidade de trincas foi contada com o auxílio de um código em *python* (encontra-se no Apêndice E), estando ambos os conjuntos expostos na Tabela 14.

Tabela 14 - Quantitativo rótulo manual x detector

Conjunto	Quantidade de imagens rotuladas	Quantidade de rótulos			
		Transversal	Longitudinal	Em bloco	Couro de Jacaré
CE060_manual	1.265	394	2.277	9	718
CE060_detector	725	37	381	15	506

Fonte: autoria própria

O primeiro fato observado foi o quantitativo de imagens rotuladas. Das 1.265 imagens o detector conseguiu identificar algum tipo de patologia em apenas 725 imagens, que representa 57% do total. Além disso, se compararmos os quantitativos das trincas isoladas com as trincas interligadas, a diferença de rótulos nas trincas isoladas é gritante, fato este que era esperado com base na análise da Seção 5.1. Das trincas transversais, o detector foi capaz de identificar apenas 9% dos rótulos, das trincas longitudinais, foram detectados 16% dos rótulos, das trincas couro de jacaré foram detectados 90%, sobre as trincas em bloco, ele detectou mais do que realmente tinha, indicando que possivelmente ele confundiu trinca em bloco com trinca couro de jacaré. Tal análise foi realizada olhando apenas para os números.

Observando as imagens rotuladas automaticamente, foi possível notar que a maioria das trincas que estavam no acostamento não foram detectadas (independente dos rótulos), embora o conjunto de treino contivesse imagens desse tipo. Muito provavelmente a amostra de trincas no acostamento não foi grande o suficiente para permitir que ele considerasse essa região. A Figura 55 mostra um exemplo de imagens com trincas no acostamento. Além disso, as trincas que não estavam na faixa que o carro estava trafegando não foram identificadas facilmente, como é possível observar na Figura 56, fato este que nos permite tirar conclusões acerca da melhor definição da área de interesse para realização de pesquisas futuras.

Figura 55 - Exemplo de imagens com trinca no acostamento que não foram detectadas

Fonte: autoria própria

Figura 56 - Exemplo de imagem com trinca na outra faixa de rolamento

Fonte: autoria própria

Um equívoco que aconteceu em algumas imagens foi o detector identificar como trinca couro de jacaré o pavimento do tipo intertravado, como é possível observar na Figura 58. Ainda na Figura 57, foi possível notar que, em alguns casos, o algoritmo confunde trinca couro de jacaré com trinca em bloco.

Figura 57 – Exemplo de equívoco na rotulação automática

Fonte: autoria própria

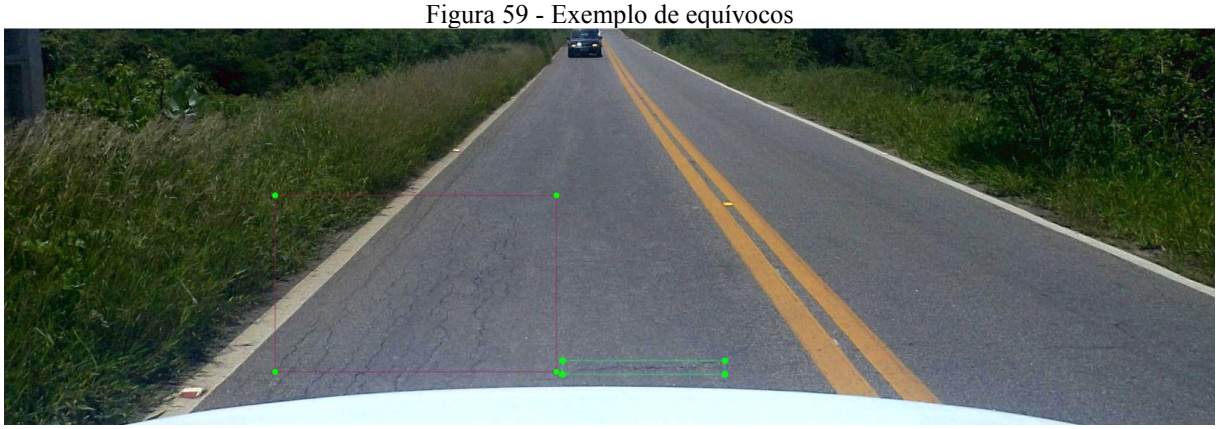
Apresenta dificuldades em identificar trincas isoladas mesmo que elas sejam bastante evidentes e de grande comprimento, como é possível observar na Figura 58. Ainda nessa figura, é possível notar novamente o equívoco em classificar o pavimento intertravado como sendo couro de jacaré, bem como a não detecção do couro de jacaré que se encontrava na região direita inferior da imagem.

Figura 58 - Exemplo de equívocos e falta de detecção

Fonte: autoria própria

Além disso, o detector apresenta dificuldades em diferenciar trincas longitudinais muito próximas umas às outras com couro de jacaré, muito provavelmente porque a densidade de trincas em uma única área pode ser parecida em algumas situações. A Figura 59 apresenta um exemplo desse tipo de equívoco.

Figura 59 - Exemplo de equívocos



Fonte: autoria própria

Essas foram as falhas mais comuns cometidas pelo detector, indicando que ele ainda não está adequado para informar não só o tipo de trinca como também sua localização.

6 CONCLUSÃO

A automatização vem sendo amplamente utilizada para auxiliar na avaliação dos defeitos rodoviários, principalmente ajudando na detecção das patologias presentes nos pavimentos, em substituição aos métodos manuais e de interpretação visual usados no Brasil. As características analisadas do pavimento foram os defeitos superficiais do tipo trinca considerando-se quatro classes: trinca transversal, trinca longitudinal, trinca em bloco e trinca couro de jacaré.

Com a pesquisa realizada neste trabalho, foi possível identificar que o YOLO consegue identificar as trincas do pavimento, embora precise fazer algumas considerações para a obtenção das imagens que serão utilizadas para realização do treino da rede neural para obtenção de melhores resultados. Chegou-se à conclusão que o ideal para este tipo de pesquisa é que a câmera esteja voltada para o pavimento, pois permite uma melhor visualização das patologias a serem estudadas. Além disso, seria interessante definir melhor a região de interesse para detecção, denominada em inglês de ROI (*Region Of Interest*). Apesar de ter sido definido que só seriam rotuladas as trincas que estivessem da metade da foto para baixo, observou-se uma dificuldade na detecção de trincas na faixa de rolamento que o carro não está trafegando.

No que diz respeito à versão do YOLO, observou-se que a versão 4 apresentou melhores resultados para a detecção desse tipo de patologia se comparado à versão 3, devendo os testes futuros que viessem a melhorar os resultados apresentados nessa pesquisa serem executados com o YOLOv4. Em relação às dimensões da imagem, 416x416 pixels apresentou resultados melhores do que 608x608 pixels, indicando que as próximas pesquisas podem ser efetuadas com essa dimensão sem prejuízos nos resultados das métricas finais. O treino com a dimensão da imagem de 608x608 pixels não ficou com resultados tão inferiores assim se comparada à outra dimensão testada, no entanto, a quantidade de horas a mais necessárias para o treino com essas configurações não justifica seu uso.

Referentes às iterações, recomenda-se testar valores maiores que 8.000, a fim de verificar o comportamento da perda e do mAP. Sobre as trincas em si, observou-se que as trincas interligadas (em bloco e couro de jacaré), são mais facilmente identificadas se comparadas com as trincas isoladas (transversal e longitudinal), muito provavelmente por ocuparem uma maior porção da imagem.

Para recomendações de trabalhos futuros, indica-se tentar melhorar as métricas a partir do aumento de conjunto de dados, que possibilite um treino com pelo menos 8.000 imagens,

além de testar tentar fazer o treino com os rótulos balanceados para ver se realmente melhora o resultado. As questões de posicionamento de câmera e definição de um ROI menor também devem ser consideradas. A obtenção de maior acurácia e maior IoU irá permitir, futuramente, trabalhar com as características das trincas, para obtenção, por exemplo, de área trincada no pavimento, com o intuito de auxiliar futuramente na obtenção automática dos índices do pavimento mais comumente utilizados no Brasil.

REFERÊNCIAS

AASHTO, 1990, "**Guide for Design Management Systems**". American Association of State Highway and Transportation Officials, Washington DC

ACERVO LIMA (2021). Disponível em: https://acervolima.com/classificacao-de-dados-usando-support-vector-machines-svms-em-r/. Acesso em: 09 de jan. de 2022.

Ajuste automático de contraste e brilho de uma foto colorida de uma folha de papel com OpenCV. Disponível em: https://www.desenv-web-rp.com/pt/python/ajuste-automatico-de-contraste-e-brilho-de-uma-foto-colorida-de-uma-folha-de-papel-com-opencv/811144052/. Acesso em 05/01/2021

AUGUSTO, R.; OLIVEIRA, D. **Reconhecimento Facial com Super-Resolução : uma abordagem utilizando Redes Generativas**, 2020.

BARDIS, M. D. et al. **Applications of artificial intelligence to prostate multiparametric mri (Mpmri): Current and emerging trends. Cancers**, 2020.

BERNUCCI, L. B.; MOTTA, L. M.; CERATTI, J. A. P.; SOARES, J. B. (2010) **Pavimentação Asfáltica: Formação Básica para Engenheiros**. 3. ed. Rio de Janeiro, RJ.

BREGA, J. R. F. **A utilização de redes neurais artificiais em um sistema de gerência de pavimentos**, 1996. Escola de Engenharia de São Carlos – Universidade de São Paulo.

C. R. IKRAM. **A benchmark for evaluating Deep Learning based Image Analytics**. UNIVERSITY OF OSLO, 2019.

Copeland, M. (2016). What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning. Disponível em: < https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>. Acesso em: 26 out. 2021.

DA SILVA, A. M.; BARONE, D. A. C. **Inteligência Artificial e o Futuro do Trabalho: entre Fausto e Prometeu**, 2020.

Data Science Academy (2021). Deep Learning Book. Disponível em: < http://deeplearningbook.com.br>. Acesso em: 09 de jan. de 2022.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, "Norma 005/2003 - TER – Defeitos nos pavimentos flexíveis e semi-rígidos – Terminologia, 2003.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, "Norma 006/2003 - PRO – Avaliação objetiva da superfície de pavimentos flexíveis e semi-rígidos – Procedimento, 2003.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, "Norma 007/2003 - PRO – Levantamento para avaliação da condição de superfície de subtrecho homogêneo de rodovias de pavimentos flexíveis e semi-rígidos para gerência de pavimentos e estudos e projetos – Procedimento, 2003.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, “Norma 008/2003 - PRO - Levantamento visual contínuo para avaliação da superfície de pavimentos flexíveis e semi-rígidos, 2003.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, “Norma 009/2003 - PRO – Avaliação subjetiva da superfície de pavimentos flexíveis e semi-rígidos – Procedimentos, 2003.

DESTRI JUNIOR, J. et al. **Detecção E Quantificção Automatizadas De Trincas Em Pavimentos De Rodovias**. 33º Congresso de Pesquisa e Ensino em Transporte da ANPET, 2019.

DHARNEESHKAR, J. et al. **Deep Learning based Detection of potholes in Indian roads using YOLO**. Proceedings of the 5th International Conference on Inventive Computation Technologies, 2020.

DNIT - Departamento Nacional de Infra-estrutura de Transportes. Manual de restauração de pavimentos asfálticos. Diretoria de Planejamento e Pesquisa/IPR, R.J. 2 ed. 310 p. 2005.

DNIT. Manual de Gerência de Pavimentos. p. 189, 2011.

ERAZO, J. J. M. **Desenvolvimento de um sistema de contagem e classificação de veículos utilizando redes neurais convolucionais**, 2021. Universidade Federal de Santa Catarina.

ESPÍNDOLA, A. C. et al. **Pothole and patch detection on asphalt pavement using deep convolutional neural network**, 2021.

FREITAS, G. T. DE M.; NOBRE JÚNIOR, E. F. **Identificação De Patologias Em Pavimentos Rodoviários Utilizando Inteligência Artificial**. 34º Congresso de Pesquisa e Ensino em Transporte da ANPET, 2020.

FERREIRA, E. R.. **Procedimentos automático para apoio na avaliação de pavimentos com o uso de imagens digitais**, 2010. Universidade Federal de Viçosa.

GIRSHICK, R. et al. **Region-Based Convolutional Networks for Accurate Object Detection and Segmentation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016.

GIRSHICK, R. **Fast r-cnn**. Proceedings of the IEEE international conference on computer vision. 2015.

Github Alexeyab. Disponível em: https://github.com/alexeyab/darknet. Acesso em: 10 de jan. de 2022.

GOPALAKRISHNAN, K. et al. **Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection**. Construction and Building Materials, 2017.

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

HOLLERWEGER, M. M.. **Aplicação de Visão Computacional no Auxílio ao Levantamento de Defeitos em Pavimento Rodoviário**, 2019.

J. REDMON, S. DIVVALA, R. GIRSHICK, AND A. FARHADI. **You only look once: Unified, real-time object detection**, 2016.

JR, O. J. S. **Análise de dados de instrumentação de túneis do metrô de São Paulo – Uma abordagem por redes neurais**, 2006.

LIU, WEI, ET AL. **Ssd: Single shot multibox detector**. European conference on computer vision. Springer, Cham, 2016.

MENDES, A. S.; GONZÁLEZ, G. V. **An Architectural Multi-Agent System for a Pavement Monitoring System with Pothole Recognition in UAV Images,** 2020.

Murtaza's Workshop - Robotics and AI. How to Crop and Resize images - OpenCV Python Tutorials for Beginners 2020. Disponível em: < https://www.youtube.com/watch?v=GiVVCu7l34A>. Acesso em: 10 de jan. de 2021.

NASCIMENTO, L. S. et al. **Aplicação de aprendizagem profunda para detecção automática de sinalização vertical em rodovias.** 35º Congresso de Pesquisa e Ensino em Transporte da ANPET, 2021.

NAYAK. S. **Training YOLOv3: Deep Learning based Costum Object Detector**, 2019. Disponível em: <https://learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/>. Acesso em: Acesso em 10 de jan. de 2021.

NESKOROZHENYI, R. Tile (Slice) YOLO Dataset for Small Objects Detection. **Towards Data Science**, 2021. Disponível em: < https://towardsdatascience.com/tile-slice-yolo-dataset-for-small-objects-detection-a75bf26f7fa2>. Acesso em 10 de jan. de 2021.

PAZ et al. **Identificação de defeitos do tipo "panela" em pavimento asfáltico por meio de redes neurais convolucionais.** 34º Congresso de Pesquisa e Ensino em Transporte da ANPET, 2020.

Pesquisa CNT de rodovias 2021. – Brasília : CNT : SEST SENAT, 2021.

REIS, C. H. **Otimização de Hiperparâmetros em Redes Neurais Profundas**, 2018.

REN, SHAOQING, et al. **Faster r-cnn: Towards real-time object detection with region proposal networks**. Advances in neural information processing systems. 2015.

RUSSELL, S.; NORVIG, P. **Inteligência artificial: uma abordagem moderna**. ed. Prentice-Hall, 3ª Edição. São Paulo, Brazil, 2014.

Shahin, M.Y. e S.D. Kohn. (1979). **Development of a Pavement Condition Rating Procedure for Roads, Streets and Parking Lots.** U.S. Army corps of engineers, Technical Report M-268.

SHIN, H. C. et al. **Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning**. IEEE Transactions on Medical Imaging, 2016.

USACE (1982). **Technical Manual TM 5-623: Pavement Maintenance Management** – 1982. U.S. Army Corps of Engineers, Headquarters, Department of the Army, Washington, DC.

WANG, C.; LIAO, H. M. YOLOv4: **Optimal Speed and Accuracy of Object Detection,** 2020.

XIE, D.; ZHANG, L.; BAI, L. Review Article: **Deep Learning in Visual Computing and Signal Processing**. Hindawi: Applied Computational Intelligence and Soft Computing. Volume 2017, Article ID 1320780. https://doi.org/10.1155/2017/1320780

Y. LECUN, Y. BENGIO, AND G. HINTON. **Deep learning.** Nature, vol. 521, no. 7553, 2015.

ZHANG, Y. et al. **Understanding of Object Detection Based on CNN Family and YOLO**, 2018.

APÊNDICE A – CÓDIGO PYTHON SELEÇÃO ALEATÓRIA DE IMAGENS E RÓTULOS

```
import os
import random
from random import choice
import shutil

oldAdress = 'C:/treino/'
newAdress = 'C:/teste/'

files = os.listdir (r'C:/treino/')

l= []
for file in files:
    nome = file.split('.')[0]
    l.append(nome)

l= set(l)
print (l)
print(len(l))

#faz a conta de quantos % quer e insere o valor em tamanho
tamanho=1000
teste = random.sample(l, tamanho)
print(teste)
lista_len=len(teste)
print(lista_len)

x=0
while x < lista_len:
    caminhoCompleto_old1 = oldAdress + teste[x] + '.txt'
    caminhoCompleto_old2 = oldAdress + teste[x] + '.jpg'
    caminhoCompleto_new1 = newAdress + teste[x] + '.txt'
    caminhoCompleto_new2 = newAdress + teste[x] + '.jpg'
    shutil.move(caminhoCompleto_old1, caminhoCompleto_new1)
    shutil.move(caminhoCompleto_old2, caminhoCompleto_new2)
    print(x, '-', teste[x])
    x+=1
```

APÊNDICE B – CÓDIGO PYTHON PARA CORTE E REDIMENSIONAMENTO DAS IMAGENS

```
#Codigo adaptado de <https://www.youtube.com/watch?v=GiVVCu7l34A>
import cv2
import os
from os import path

input_path = "D:\imagens_cortadas\CE025"
output_path = "D:\imagens_cortadas\CE025_cortadas"

files = [path.join(input_path, f) for f in os.listdir(input_path) if f.endswith('JPG')]

for file in files:
    img = cv2.imread(file)
    #primeiro pixel em y, depois pixel em x
    imgCropped = img[1080:1953, 413:3577]
    imgCropResize = cv2.resize(imgCropped,(img.shape[1],img.shape[0]))
    file_name = path.join(output_path, path.basename(file))
    #cv2.imwrite(file_name,imgCropResize)
    cv2.imwrite(file_name,imgCropped)


cv2.waitKey(0)
```

APÊNDICE C – CÓDIGO PYTHON PARA ALTERAÇÃO CONTRASTE DA IMAGEM

```
#Codigo adaptado de <https://www.desenv-web-rp.com/pt/python/ajuste-automatico-de-contraste-e-brilho-de-uma-foto-colorida-de-uma-folha-de-papel-com-opencv/811144052/>

import cv2
import os
from os import path

# Automatic brightness and contrast optimization with optional histogram clipping
def automatic_brightness_and_contrast(image, clip_hist_percent=1):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Calculate grayscale histogram
    hist = cv2.calcHist([gray],[0],None,[256],[0,256])
    hist_size = len(hist)

    # Calculate cumulative distribution from the histogram
    accumulator = []
    accumulator.append(float(hist[0]))
    for index in range(1, hist_size):
        accumulator.append(accumulator[index -1] + float(hist[index]))

    # Locate points to clip
    maximum = accumulator[-1]
    clip_hist_percent *= (maximum/100.0)
    clip_hist_percent /= 2.0

    # Locate left cut
    minimum_gray = 0
    while accumulator[minimum_gray] < clip_hist_percent:
        minimum_gray += 1

    # Locate right cut
    maximum_gray = hist_size -1
    while accumulator[maximum_gray] >= (maximum – clip_hist_percent):
        maximum_gray -= 1

    # Calculate alpha and beta values
    alpha = 255 / (maximum_gray – minimum_gray)
    beta = -minimum_gray * alpha

    auto_result = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    return (auto_result, alpha, beta)

input_path = “D:\PG\TRINCAS\Detran 2021\imagens_cortadas\CE010_cortadas”
output_path = “D:\PG\TRINCAS\Detran 2021\imagens_cortadas\CE010_ cortada_contraste”
```

```
files = [path.join(input_path, f) for f in os.listdir(input_path)]

for file in files:
    img = cv2.imread(file)
    auto_result, alpha, beta = automatic_brightness_and_contrast(img)
    #print('alpha', alpha)
    #print('beta', beta)
    #cv2.imshow('auto_result', auto_result)
    file_name = path.join(output_path, path.basename(file))
    cv2.imwrite(file_name, auto_result)

cv2.waitKey()
```

APÊNDICE D – CÓDIGO PYTHON PARA CORTE DE IMAGENS E SEUS RESPECTIVOS RÓTULOS

```
#Codigo adaptado de < https://towardsdatascience.com/tile-slice-yolo-dataset-for-small-objects-detection-a75bf26f7fa2>
import pandas as pd
import numpy as np
from PIL import Image
from shapely.geometry import Polygon
import glob
import argparse
import os
import random
from shutil import copyfile


def tiler(imnames, newpath, falsepath, slice_size_x,slice_size_y, ext):
    for imname in imnames:
        im = Image.open(imname)
        imr = np.array(im, dtype=np.uint8)
        height = imr.shape[0]
        width = imr.shape[1]
        labname = imname.replace(ext, '.txt')
        labels = pd.read_csv(labname, sep=' ', names=['class', 'x1', 'y1', 'w', 'h'])

        # we need to rescale coordinates from 0-1 to real image height and width
        labels[['x1', 'w']] = labels[['x1', 'w']] * width
        labels[['y1', 'h']] = labels[['y1', 'h']] * height

        boxes = []

        # convert bounding boxes to shapely polygons. We need to invert Y and find polygon vertices from center points
        for row in labels.iterrows():
            x1 = row[1]['x1'] - row[1]['w']/2
            y1 = (height - row[1]['y1']) - row[1]['h']/2
            x2 = row[1]['x1'] + row[1]['w']/2
            y2 = (height - row[1]['y1']) + row[1]['h']/2

            boxes.append((int(row[1]['class']), Polygon([(x1, y1), (x2, y1), (x2, y2), (x1, y2)])))

        counter = 0
        print('Image:', imname)
        # create tiles and find intersection with bounding boxes for each tile
        for i in range((height // slice_size_y)):
            for j in range((width // slice_size_x)):
                x1 = j*slice_size_x
                y1 = height - (i*slice_size_y)
                x2 = ((j+1)*slice_size_x) - 1
```

```
            y2 = (height - (i+1)*slice_size_y) + 1

            pol = Polygon([(x1, y1), (x2, y1), (x2, y2), (x1, y2)])
            imsaved = False
            slice_labels = []

            for box in boxes:
               if pol.intersects(box[1]):
                  inter = pol.intersection(box[1])

                  if not imsaved:
                     sliced = imr[i*slice_size_y:(i+1)*slice_size_y,
j*slice_size_x:(j+1)*slice_size_x]
                     sliced_im = Image.fromarray(sliced)
                     #filename = imname.split('/')[-1]
                     filename = os.path.basename(imname)
                     #slice_path = newpath + "/" + filename.replace(ext, f'_{i}_{j}{ext}')
                     slice_path = os.path.join(newpath, filename.replace(ext,
f'_{i}_{j}{ext}'))
                     #slice_labels_path = newpath + "/" + filename.replace(ext, f'_{i}_{j}.txt')
                     slice_labels_path = os.path.join(newpath, filename.replace(ext,
f'_{i}_{j}.txt'))
                     print(slice_path)
                     sliced_im.save(slice_path)
                     imsaved = True

                  # get smallest rectangular polygon (with sides parallel to the coordinate axes)
that contains the intersection
                  new_box = inter.envelope

                  # get central point for the new bounding box
                  centre = new_box.centroid

                  # get coordinates of polygon vertices
                  x, y = new_box.exterior.coords.xy

                  # get bounding box width and height normalized to slice size
                  new_width = (max(x) - min(x)) / slice_size_x
                  new_height = (max(y) - min(y)) / slice_size_y

                  # we have to normalize central x and invert y for yolo format
                  new_x = (centre.coords.xy[0][0] - x1) / slice_size_x
                  new_y = (y1 - centre.coords.xy[1][0]) / slice_size_y

                  counter += 1

                  slice_labels.append([box[0], new_x, new_y, new_width, new_height])

            if len(slice_labels) > 0:
               slice_df = pd.DataFrame(slice_labels, columns=['class', 'x1', 'y1', 'w', 'h'])
```

```
                print(slice_df)
                slice_df.to_csv(slice_labels_path, sep=' ', index=False, header=False,
float_format='%.6f')

            if not imsaved and falsepath:
                sliced = imr[i*slice_size_x:(i+1)*slice_size_x, j*slice_size_y:(j+1)*slice_size_y]
                sliced_im = Image.fromarray(sliced)
                #filename = imname.split('/')[-1]
                filename = os.path.basename(imname)
                #slice_path = falsepath + "/" + filename.replace(ext, f'_{i}_{j}{ext}')
                slice_path = os.path.join(falsepath, filename.replace(ext,
f'_{i}_{j}{ext}'))

                sliced_im.save(slice_path)
                print('Slice without boxes saved')
                imsaved = True

def splitter(target, target_upfolder, ext, ratio):
    imnames = glob.glob(f'{target}/*{ext}')
    names = [name.split('/')[-1] for name in imnames]

    # split dataset for train and test

    train = []
    test = []
    for name in names:
        if random.random() > ratio:
            test.append(os.path.join(target, name))
        else:
            train.append(os.path.join(target, name))
    print('train:', len(train))
    print('test:', len(test))

    # we will put test.txt, train.txt in a folder one level higher than images

    # save train part
    with open(f'{target_upfolder}/train.txt', 'w') as f:
        for item in train:
            f.write("%s\n" % item)

    # save test part
    with open(f'{target_upfolder}/test.txt', 'w') as f:
        for item in test:
            f.write("%s\n" % item)

if __name__ == "__main__":
    # Initialize parser
    parser = argparse.ArgumentParser()
```

```
    parser.add_argument("-source",
default="D:\PG\TRINCAS\Tentativa2_imagens_cortadas\Detran
2021\CE025\Rotuladas_cortadas", help = "Source folder with images and labels needed to be
tiled")
    parser.add_argument("-target",
default="D:\PG\TRINCAS\Tentativa2_imagens_cortadas\Detran
2021\CE025\cortadas_subdivididas", help = "Target folder for a new sliced dataset")
    parser.add_argument("-ext", default=".JPG", help = "Image extension in a dataset. Default:
.JPG")
    parser.add_argument("-falsefolder", default=None, help = "Folder for tiles without
bounding boxes")
    parser.add_argument("-size_x", type=int, default=700, help = "Size of a tile. Dafault: 700")
    parser.add_argument("-size_y", type=int, default=700, help = "Size of a tile. Dafault: 700")
    parser.add_argument("-ratio", type=float, default=0.8, help = "Train/test split ratio. Dafault:
0.8")

    args = parser.parse_args()

    imnames = glob.glob(f'{args.source}/*{args.ext}')
    labnames = glob.glob(f'{args.source}/*.txt')

    if len(imnames) == 0:
        raise Exception("Source folder should contain some images")
    elif len(imnames) != len(labnames):
        raise Exception("Dataset should contain equal number of images and txt files with
labels")

    if not os.path.exists(args.target):
        os.makedirs(args.target)
    elif len(os.listdir(args.target)) > 0:
        raise Exception("Target folder should be empty")

    # classes.names should be located one level higher than images
    # this file is not changing, so we will just copy it to a target folder
    upfolder = os.path.join(args.source, '..' )
    target_upfolder = os.path.join(args.target, '..' )
    if not os.path.exists(os.path.join(upfolder, 'classes.names')):
        print('classes.names not found. It should be located one level higher than images')
    else:
        copyfile(os.path.join(upfolder, 'classes.names'), os.path.join(target_upfolder,
'classes.names'))

    if args.falsefolder:
        if not os.path.exists(args.falsefolder):
            os.makedirs(args.falsefolder)
        elif len(os.listdir(args.falsefolder)) > 0:
            raise Exception("Folder for tiles without boxes should be empty")

    tiler(imnames, args.target, args.falsefolder, args.size_x, args.size_y, args.ext)
    splitter(args.target, target_upfolder, args.ext, args.ratio)
```

APÊNDICE E – CÓDIGO PYTHON PARA CONTAGEM DE RÓTULOS NO CONJUNTO DE DADOS

```
# pega os arquivos da pasta teste e conta a quantidade de
# ocorrencias dos numeros 1, 2, 3, 4 e 5 na primeira coluna

from os import listdir
from os.path import isfile, join

# pasta chamada teste no mesmo diretorio que este arquivo de codigo
path = 'txt'

files = [f for f in listdir(path) if isfile(join(path, f))]

# a posicao i da lista rotulos guarda a quantidade de ocorrencias do numero i
#          0  1  2  3
rotulos = [0, 0, 0, 0]

for file in files:
    ref_arquivo = open(join(path, file), "r")

    for linha in ref_arquivo:
        valores = linha.split()
        rotulos[int(valores[0])] += 1

    ref_arquivo.close()

print(rotulos)
```