Aplicação de Redes Neurais Convolucionais Baseadas em Região na Detecção de Componentes Hidrossanitários Industrializados

Nelson Dutra Junior
Universidade Federal de Pelotas
ndjunior@inf.ufpel.edu.br

Fabrício Barbosa Viegas
Universidade Federal de Pelotas
fbviegas@inf.ufpel.edu.br

João Carrett
Universidade Federal de Pelotas
jcacarrett@inf.ufpel.edu.br

Anderson Priebe Ferrugem
Universidade Federal de Pelotas
ferrugem@inf.ufpel.edu.br

Fábio Kellermann Schramm
Universidade Federal de Pelotas
fabioks@ufpel.edu.br

Abstract

The increasing industrialization of construction processes has intensified the demand for automated, scalable, and reliable inspection methods capable of supporting sustainable and resilient urban development. In particular, the validation of hydrosanitary installations in construction sites remains largely manual, leading to inefficiencies, increased costs, and limited traceability. This work presents an artificial intelligence–based approach for the automated detection, localization, and classification of components of industrialized hydrosanitary kits using region-based convolutional neural networks. The results indicate that deep learning–based object detection can effectively support automated visual validation in construction workflows, contributing to digital transformation, improved quality control, and increased operational efficiency. By enabling scalable and data-driven inspection processes, the proposed solution aligns with the United Nations Sustainable Development Goal 11, fostering more inclusive, resilient, and sustainable urban infrastructure.

Keywords

Visão Computacional, Aprendizado Profundo, Detecção de Objetos, Inspeção Automatizada, Canteiros de Obras.

1 Introdução

Em resposta à crescente necessidade de agilidade e redução de custos, estão sendo desenvolvidos produtos e soluções que buscam aumentar a produtividade nas diversas atividades realizadas nos canteiros de obras. Em particular, a utilização de kits industrializados para a execução de instalações hidrossanitárias surge como uma alternativa que não somente diminui a dependência de mão de obra, mas também otimiza os processos de produção e estabelece padrões no sistema. Essa abordagem acelera significativamente o processo de instalação hidrossanitária ao empregar elementos pré-montados e fabricados fora do canteiro, garantindo, assim, a qualidade do produto final [Vieira 2019][1].

As instalações hidráulicas são parte importante de uma obra e precisam ser conferidas com exatidão, exigindo o deslocamento de um responsável até o canteiro de obras para validação, isto por sua vez implica em custos de tempo e recursos para o deslocamento e em ineficiência ou tempo de obra parada para o cliente.

Ao mesmo tempo, a digitalização do ambiente da construção civil e de sua mão de obra facilita o uso da tecnologia para resolução de problemas e desafios do dia a dia. Aplicar técnicas de visão computacional e de detecção de objetos para automatizar o processo de validação com acurácia contribui para a otimização do tempo, cumprimento de prazos e redução de custos, e é especialmente importante para profissionais ou empresas que gerenciam diversos canteiros de obras simultaneamente, criando um diferencial competitivo.

Na primeira etapa do processo de validação, foco dessa pesquisa, é necessário identificar e localizar os diferentes componentes que compõem os kits hidrossanitários em uma imagem capturada no canteiro de obras.

2 Metodologia

Esta pesquisa adota uma metodologia experimental e aplicada focada no desenvolvimento e avaliação de modelos de detecção de objetos baseados em aprendizado profundo para validação visual automatizada de kits hidrossanitários industrializados em ambientes de construção. A metodologia proposta é estruturada como um fluxo de trabalho composto por aquisição de dados, pré-processamento, construção do conjunto de dados, aumento de dados, treinamento do modelo e avaliação de desempenho.

2.1 Coleta de Dados

O conjunto de dados utilizado neste estudo foi construído a partir de imagens de componentes de kits hidrossanitários industrializados para sistemas de água quente. Inicialmente, foram coletadas 63 imagens em formato RGB diretamente em canteiros de obras, contemplando diferentes condições de iluminação, ângulos de captura e cenários de fundo.

Visando ampliar a diversidade visual e reduzir o desbalanceamento entre classes, foram incorporadas 197 imagens adicionais provenientes de catálogos técnicos de fabricantes. Foram consideradas quatro classes de componentes, selecionadas por sua relevância prática nas instalações hidrossanitárias: joelho, misturador, registro e tê. A combinação de imagens reais e de catálogo visa melhorar a capacidade de generalização dos modelos treinados.

Apesar do número relativamente reduzido de imagens disponíveis, o *dataset* foi considerado adequado para uma avaliação inicial da abordagem proposta. Dessa forma, os experimentos realizados têm caráter exploratório, buscando verificar a viabilidade do uso de modelos de detecção de objetos nesse contexto.

2.2 Pré-processamento dos Dados

O pré-processamento das imagens foi realizado para padronizar os dados de entrada e reduzir a complexidade computacional durante o treinamento dos modelos. As imagens originais, com resolução

de *2592×1728px*, foram redimensionadas para *800×800px*, utilizando corte central quando necessário, para evitar distorções geométricas.

Adicionalmente, as imagens foram convertidas do espaço de cores RGB para tons de cinza. Essa estratégia reduz o número de canais de entrada e o custo computacional, preservando informações estruturais e de forma relevantes para a tarefa de detecção de objetos. Considerando que os componentes analisados apresentam contorno e geometria bem definidos, e que podem ser encontrados em diferentes cores dependendo do fabricante, optou-se por remover a informação de cor e priorizar características estruturais e geométricas para a detecção dos objetos. Ainda assim, investigações futuras podem avaliar o impacto da utilização de imagens RGB na distinção entre diferentes componentes.

2.3 Construção e Anotação dos *Datasets*

Os datasets foram estruturados no formato COCO (*Common Objects in Context*), amplamente adotado em tarefas de detecção de objetos por oferecer suporte padronizado à rotulação e à delimitação de regiões de interesse por meio de *bounding boxes*. A anotação das imagens foi realizada manualmente, garantindo a correta localização e classificação de cada componente presente nas imagens.

Foram construídos dois conjuntos de dados distintos: um dataset de treinamento, contendo 50 imagens por classe, e um dataset de validação, composto por aproximadamente 15 imagens por classe. A separação entre os conjuntos foi realizada para assegurar uma avaliação imparcial do desempenho dos modelos.

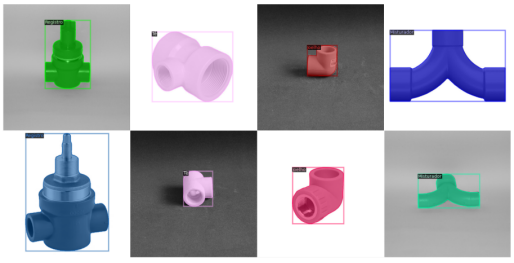

Figura 1: Exemplos de imagens classificadas

2.4 Aumento de Dados *(Data Augmentation)*

Considerando o número limitado de imagens disponíveis, técnicas de aumento de dados foram aplicadas para ampliar a variabilidade do conjunto de treinamento e reduzir o risco de *overfitting*. O processo de *data augmentation* foi realizado utilizando o *pipeline* de carregamento de dados do *framework* Detectron2.

Para cada imagem do conjunto de treinamento, foram geradas quatro variações, elevando o número de exemplos de 50 para 200 por classe, totalizando 800 amostras. As técnicas aplicadas incluem espelhamento horizontal e vertical com probabilidade de 50%, rotações fixas de 90°, 180° e 270°, além de rotações aleatórias entre 0,5° e 30°. Essas transformações simulam variações comuns encontradas em ambientes reais de obra.

2.5 Arquitetura dos Modelos e Treinamento

Os modelos de detecção de objetos foram implementados utilizando o *framework* Detectron2, baseado em PyTorch, escolhido por sua flexibilidade e suporte a arquiteturas modernas de aprendizado profundo. Neste trabalho, foi adotada a arquitetura Faster R-CNN, uma abordagem baseada em regiões amplamente utilizada para detecção de múltiplos objetos em imagens complexas.

Foram avaliadas diferentes variações de modelos Faster R-CNN pré-treinados disponíveis no cronograma 3× do Detectron2, combinando *backbones* das famílias ResNet e ResNeXt com *Feature Pyramid Networks* (FPN). Como abordagem de estágio único, foram avaliados modelos da arquitetura RetinaNet, com *backbone* da família ResNet, que em seu *neck* utiliza uma FPN para tratar as características, e emprega duas sub-redes para a classificação e a localização dos objetos. Durante os experimentos, parâmetros de treinamento como número de iterações, tamanho do lote (*batch size*) e taxa de aprendizado (*learning rate*) foram sistematicamente ajustados para analisar seu impacto no desempenho dos modelos.

2.6 Avaliação de Desempenho

A avaliação dos modelos foi realizada por meio de métricas consagradas em tarefas de detecção de objetos e classificação, incluindo *Average Precision* (AP), acurácia, precisão e F1-score. Essas métricas foram obtidas a partir das matrizes de confusão geradas com base nas predições dos modelos aplicados ao conjunto de validação.

A análise comparativa dos resultados permitiu identificar a configuração mais eficiente em termos de desempenho e estabilidade. A métrica de *Average Precision* foi adotada como critério principal para a seleção do modelo final, por considerar simultaneamente a qualidade da localização e da classificação dos objetos detectados.

3 Fundamentação teórica

As Redes Neurais Artificiais (RNA) são uma área do aprendizado de máquina, subcampo da inteligência artificial (AI), inspirada no funcionamento dos sistemas nervosos biológicos, simulando como o cérebro humano realiza determinadas tarefas, como reconhecimento de padrões e tomada de decisão. Uma RNA é composta por neurônios que executam o processamento e implementam uma função não linear simples, os neurônios se comunicam entre si recebendo entradas de neurônios próximos e propagando sua saída, um valor numérico escalar, para outros neurônios. Os neurônios são organizados por camadas e a cada conexão entre os neurônios possui um peso que determina sua relevância [Dongare et al. 2012][2].

Outro componente essencial de uma RNA é a função de ativação, também chamada de função de transferência, as funções de ativação contribuem para determinar a saída de um neurônio, a partir de suas entradas e pesos. As funções de ativação incorporam a não-linearidade nos modelos, fazendo com que a RNA consiga aprender padrões complexos e se molde aos tipos de dados. Alguns exemplos comuns de funções de ativação são a função sigmoide, a função arco tangente e a função tangente hiperbólica [Wang 2003][3].

São ditas de aprendizado profundo (*Deep Learning*) as RNAs com muitas camadas e elevado número de parâmetros, se diferem pela habilidade de modelar padrões complexos e extrair automaticamente características dos dados brutos através de sua hierarquia de múltiplas camadas. As camadas iniciais aprendem características e padrões mais simples, enquanto as camadas mais altas aprendem

representações mais complexas, essas extraídas dos padrões encontrados pelas camadas mais baixas, essa organização cria RNAs eficientes em extrair conhecimento útil [Shinde and Shah 2018][4].

3.1 Redes Neurais Convolucionais

As redes neurais convolucionais (RNC) são uma arquitetura de RNA de aprendizado profundo pensadas para aplicação em dados estruturados como matrizes, como imagens. As RNCs utilizam operações de convolução com o uso de filtros que varrem os dados de entrada, permitindo o reconhecimento de padrões como bordas e texturas. Uma particularidade das RNCs é a redução dimensional dos dados de entrada mantendo características importantes, as características e padrões extraídos são levados para camadas totalmente conectadas que são usadas para fazer as previsões finais [Gu et al. 2018][5].

Em sua arquitetura profunda, os filtros treinados na primeira camada capturam bordas ou manchas, os filtros de segunda camada reconhecem formas, os filtros das camadas seguintes detectam partes de objetos e, por fim, nas camadas finais os filtros reconhecem objetos inteiros [Aloysius and Geetha 2017][6]. A Figura 1 mostra a arquitetura básica de uma RNC, exemplificando o seu funcionamento e as conexões entre as camadas de convolução, camadas de *pooling* e camadas totalmente conectadas.

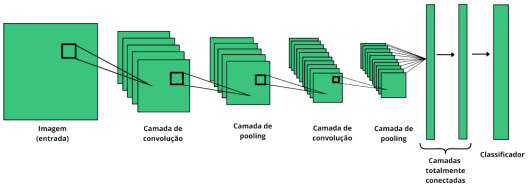

Figura 2: Arquitetura básica de uma RNC[6]

A camada de convolução é o núcleo de uma RNC. Ela aplica filtros sobre a imagem de entrada para extrair padrões e gerar mapas de características, permitindo que o modelo identifique elementos relevantes [Lopez Pinaya et al. 2020][7]. As camadas de *pooling* reduzem a dimensão dos mapas de características após a convolução, simplificando o modelo. A forma mais comum é o *max-pooling*, que mantém somente os valores mais relevantes, mas também existem variações como o *average pooling* [O'Shea and Nash 2015][8]. Já as camadas totalmente conectadas, nas quais os neurônios são conectados a todos os neurônios da camada anterior, usam as características encontradas nas camadas anteriores para realizar as predições, são encontradas ao fim da arquitetura e geram o resultado [Krichen 2023][9].

3.2 Redes Neurais Convolucionais Baseadas Em Região

As redes neurais convolucionais baseadas em região, do inglês *region-based convolutional neural networks* (R-CNN), apresentam bons resultados para imagens com vários objetos, para isso elas identificam regiões de interesse e utilizam RNCs para categorizá-las como fundo ou primeiro plano [Bharati and Pramanik 2020][10].

Como apresentado em [Girshick et al. 2014][11] as R-CNNs são compostas por 3 componentes principais, o primeiro é responsável por gerar propostas de regiões, o segundo é uma RNC que para cada região proposta extrai um vetor de características de tamanho fixo, ao fim um conjunto de classificadores lineares que classifica cada região através de seu vetor de características e a regressão linear é utilizada para delimitar o objeto, essa estrutura é mostrada na Figura 2.

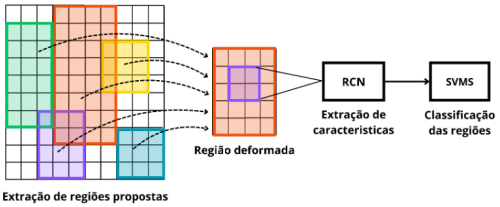

Figura 3: Componentes e funcionamento da R-CNN[11]

As R-CNNs originalmente utilizavam como classificador linear na etapa final as *support vector machines* (SVMs). No entanto, nas variantes posteriores da R-CNN, como Fast R-CNN e Faster R-CNN, essa abordagem foi modificada. Em vez de utilizar uma SVM na camada final, essas redes passaram a adotar uma camada totalmente conectada, seguida por uma *softmax* para realizar a classificação diretamente no modelo de rede neural [Ren et al. 2016][12].

4 Experimentos

Como ponto de partida para os experimentos foram utilizados os parâmetros pré-definidos na documentação do Detectron2, onde o número de iterações é definido como 300, o tamanho de lote em 2 imagens e a taxa de aprendizado em 0,00025.

4.1 Experimento 1

Neste primeiro experimento, foram testadas todas as variações de modelos pré-treinados no cronograma 3x disponíveis para a arquitetura Faster R-CNN, mantendo os parâmetros de treinamento em seus valores padrões, a fim de identificar a resposta dos modelos ao *dataset* criado. Na Figura 4, são apresentados os resultados obtidos bem como os parâmetros utilizados. O modelo X101-FPN foi o que apresentou os melhores resultados para AP, acurácia e precisão.

Modelo	Iterações	Lote	Taxa de aprendizado	AP	Acurácia	Precisão	F1-Score
R50-C4	300	2	0,00025	43,82	0,80	0,82	0,81
R50-DC5	300	2	0,00025	45,56	0,75	0,62	0,68
R50-FPN	300	2	0,00025	47,25	0,87	0,84	0,86
R101-C4	300	2	0,00025	47,25	0,87	0,84	0,86
R101-DC5	300	2	0,00025	51,78	0,76	0,88	0,82
R101-FPN	300	2	0,00025	33,83	0,57	0,74	0,59
X101-FPN	300	2	0,00025	58,18	0,87	0,88	0,87

Figura 4: Avaliação dos modelos da arquitetura Faster R-CNN para os parâmetros padrões

4.2 Experimento 2

No segundo experimento, o objetivo foi avaliar o impacto do tamanho do lote, ou seja, o número de imagens utilizadas em cada iteração do treinamento, nas métricas utilizadas. O parâmetro variado foi o tamanho do lote, testado com os valores: 1, 2, 4 e 8. Verificou-se que o aumento do número de imagens por iteração traz

um impacto positivo nos resultados avaliados como mostrado na Figura 5.

Modelo	Iterações	Lote	Taxa de aprendizado	AP	Acurácia	Precisão	F1-Score
X101-FPN	300	1	0,00025	33,35	0,72	0,40	0,44
X101-FPN	300	2	0,00025	58,18	0,87	0,88	0,87
X101-FPN	300	4	0,00025	64,04	0,81	0,81	0,79
X101-FPN	300	8	0,00025	61,09	0,88	0,87	0,87

Figura 5: Avaliação do modelo Faster R-CNN X101-FPN para 300 iterações variando o tamanho do lote

4.3 Experimento 3

O número de iterações é um dos parâmetros mais importantes para o treinamento do modelo e para um *dataset* prático 300 iterações podem ser muito pouco, desta forma, neste experimento o número de iterações foi aumentado para 600, 1200, 2400 e 4800 para testar o efeito nos resultados. Foi possível validar uma evolução considerável nos resultados, essa evolução pode ser vista na Figura 6.

Modelo	Iterações	Lote	Taxa de aprendizado	AP	Acurácia	Precisão	F1-Score
X101-FPN	300	2	0,00025	58,18	0,87	0,88	0,87
X101-FPN	600	2	0,00025	70,98	0,86	0,86	0,86
X101-FPN	1200	2	0,00025	77,19	0,87	0,86	0,86
X101-FPN	2400	2	0,00025	81,32	0,88	0,89	0,88
X101-FPN	4800	2	0,00025	85,16	0,88	0,89	0,88

Figura 6: Comparação de resultados do modelo Faster R-CNN X101-FPN para 300, 600, 1200, 2400 e 4800 iterações

4.4 Experimento 4

O último parâmetro a ser testado foi a taxa de aprendizado, que representa o quanto os pesos da rede podem ser atualizados em uma iteração. Foram testadas as seguintes taxas de aprendizado: 0,00025, 0,00050, 00100 e 0,01000. Na Figura 7 é possível visualizar que somente o aumento da taxa de aprendizado para 0,00050 foi efetivo.

Modelo	Iterações	Lote	Taxa de aprendizado	AP	Acurácia	Precisão	F1-Score
X101-FPN	300	2	0,00025	58,18	0,87	0,88	0,87
X101-FPN	300	2	0,00050	67,03	0,86	0,91	0,89
X101-FPN	300	2	0,00100	53,76	0,75	0,62	0,68
X101-FPN	300	2	0,01000	41,40	0,72	0,61	0,66
X101-FPN	300	4	0,00025	64,04	0,81	0,81	0,79
X101-FPN	300	4	0,00050	72,65	0,91	0,75	0,73
X101-FPN	300	4	0,00100	77,64	0,85	0,85	0,85
X101-FPN	300	4	0,01000	75,58	0,87	0,88	0,87

Figura 7: Avaliação do impacto da taxa de aprendizado para o modelo Faster R-CNN X101-FPN

4.5 Experimento 5

Por fim, no último experimento foi treinado o melhor modelo para a arquitetura Faster R-CNN, o X101-FPN, encontrado através do experimento 1. Para os treinamentos foram utilizados os parâmetros que apresentaram os melhores resultados nos experimentos 2, 3 e 4. O modelo foi treinado em 4800 e 9600 iterações, para a taxa de aprendizado foram utilizados os valores de 0,00025 e 0,00050, e para o tamanho do lote foram testadas 2 e 4 imagens.

Na Figura 8 são apresentados os resultados desse experimento, o melhor resultado foi atingido com 9600 iterações. Os resultados da combinação de 2 imagens por lote e taxa de aprendizado de 0,00025, e a combinação de 4 imagens por lote e taxa de aprendizado de 0,00050 também apresentaram resultados muito próximos.

Modelo	Iterações	Lote	Taxa de aprendizado	AP	Acurácia	Precisão	F1-Score
Faster R-CNN X101-FPN	4800	2	0,00025	85,16	0,88	0,89	0,88
Faster R-CNN X101-FPN	4800	2	0,00050	84,89	0,85	0,85	0,85
Faster R-CNN X101-FPN	4800	4	0,00050	85,43	0,87	0,87	0,87
Faster R-CNN X101-FPN	9600	2	0,00025	84,59	0,87	0,87	0,87
Faster R-CNN X101-FPN	9600	4	0,00050	86,29	0,85	0,85	0,85

Figura 8: Comparação dos resultados da combinação de múltiplos parâmetros de treinamento

Utilizando a AP como fator de decisão consideramos o modelo Faster R-CNN X101-FPN como o que apresentou os melhores resultados, este foi treinado em 9600 iterações, 4 imagens por lote e taxa de aprendizado de 0,00050. A Figura 9 apresenta a matriz de confusão para esse caso, em que é possível verificar que o modelo Faster R-CNN X101-FPN acertou todas as predições para as classes misturador e registro, errando 3 para a classe joelho e 5 predições para a classe tê.

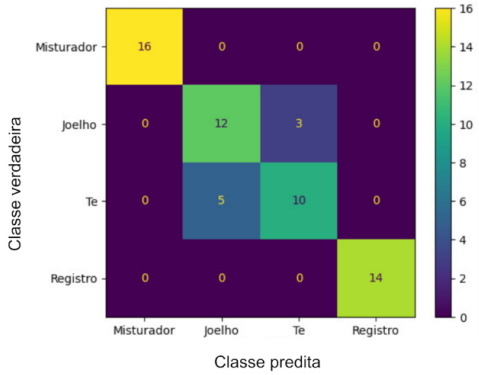

Figura 9: Matriz de confusão para o modelo Faster R-CNN X101-FPN com os melhores resultados

As curvas de *loss* durante o treinamento do modelo Faster R-CNN X101-FPN mostram um desempenho eficaz. Inicialmente, a perda diminui acentuadamente, indicando uma rápida adaptação dos modelos aos dados. Em seguida, a perda se estabiliza, mostrando que ambos os modelos alcançaram um ponto de aprendizado consistente. Esse comportamento, como apresentado na Figura 10, reflete um processo de treinamento bem-sucedido, com os modelos convergindo para uma solução adequada.

Na Figura 11, é apresentado o resultado da aplicação do modelo Faster R-CNN X101-FPN para a detecção dos componentes, utilizando uma imagem de cada classe do dataset de validação.

5 Conclusão

Este trabalho demonstrou que, por meio da arquitetura Faster R-CNN, foi possível desenvolver um modelo capaz de localizar e detectar os componentes dos kits hidrossanitários industrializados. O modelo final conseguiu detectar e classificar corretamente todos os exemplos avaliados para as classes de misturador e registro, errando a predição para 3 exemplos da classe joelho e 5 exemplos da classe tê, dentre os 15 exemplos avaliados para cada uma dessas classes.

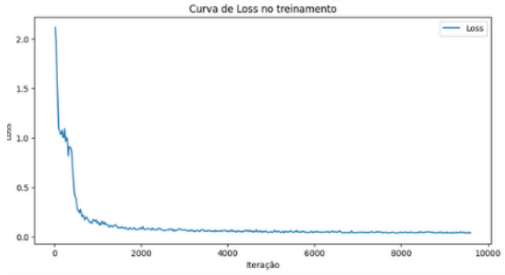

Figura 10: Curvas de *loss* para o modelo Faster R-CNN X101-FPN

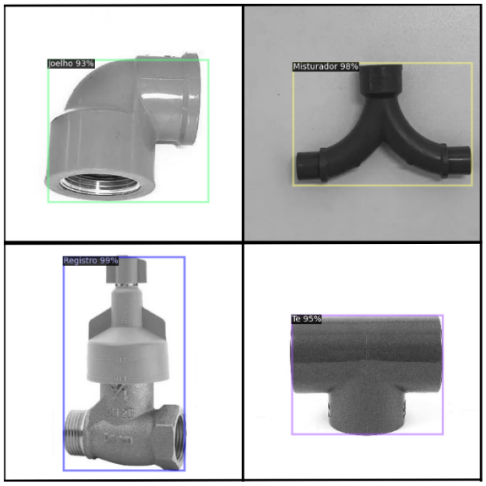

Figura 11: Resultado da detecção dos componentes pelo modelo Faster R-CNN X101-FPN

O melhor desempenho foi obtido com o modelo Faster R-CNN X101-FPN, que alcançou AP de 86,29, acurácia de 0,85 e precisão de 0,85. A otimização dos parâmetros foi fundamental para esse resultado, atingido com 9600 iterações, 4 imagens por lote e taxa de aprendizado de 0,00050. Os experimentos para otimização dos parâmetros foram especialmente importantes, visto que com os parâmetros padrões da biblioteca (300 iterações, 2 imagens por lote e taxa de aprendizado 0,00025) a AP foi de 58,18, a acurácia 0,87 e a precisão de 0,88.

Embora os resultados obtidos sejam promissores, este estudo representa uma etapa inicial da investigação. O conjunto de dados utilizado possui dimensão limitada e os experimentos foram conduzidos em um cenário controlado. Dessa forma, os resultados apresentados devem ser interpretados como uma validação preliminar da abordagem proposta.

Como trabalhos futuros, pretende-se ampliar o conjunto de dados com novas imagens coletadas em diferentes canteiros de obras, incorporando maior variabilidade de iluminação, enquadramento e dispositivos de captura. Também se pretende avaliar o impacto de diferentes representações de imagem, como o uso de dados RGB, bem como investigar o desempenho do modelo em cenários reais adicionais. Além disso, pretende-se avaliar e comparar os resultados obtidos com outras arquiteturas de detecção de objetos, a fim de analisar possíveis ganhos de desempenho e robustez para a tarefa proposta.

Referências

[1] André de Souza Vieira. Aplicação de kits pré-montados de instalações hidrossanitárias em obra de parede de concreto. Trabalho de conclusão de curso, Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Goiânia, 2019. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Civil).

[2] A. Dongare, R. Kharde, A. D. Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.

[3] Sun-Chong Wang. *Artificial Neural Network*, page 81–100. Springer US, 2003. ISBN 9781461503774. doi: 10.1007/978-1-4615-0377-4_5. URL http://dx.doi.org/10.1007/978-1-4615-0377-4_5.

[4] Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, page 1–6. IEEE, August 2018. doi: 10.1109/iccubea.2018.8697857. URL http://dx.doi.org/10.1109/ICCUBEA.2018.8697857.

[5] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77: 354–377, May 2018. ISSN 0031-3203. doi: 10.1016/j.patcog.2017.10.013. URL http://dx.doi.org/10.1016/j.patcog.2017.10.013.

[6] Neena Aloysius and M. Geetha. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, page 0588–0592. IEEE, April 2017. doi: 10.1109/iccsp.2017.8286426. URL http://dx.doi.org/10.1109/ICCSP.2017.8286426.

[7] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. *Convolutional neural networks*, page 173–191. Elsevier, 2020. ISBN 9780128157398. doi: 10.1016/b978-0-12-815739-8.00010-9. URL http://dx.doi.org/10.1016/B978-0-12-815739-8.00010-9.

[8] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL https://arxiv.org/abs/1511.08458.

[9] Moez Krichen. Convolutional neural networks: A survey. *Computers*, 12(8): 151, July 2023. ISSN 2073-431X. doi: 10.3390/computers12080151. URL http://dx.doi.org/10.3390/computers12080151.

[10] Puja Bharati and Ankita Pramanik. *Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey*, page 657–668. Springer Singapore, August 2019. ISBN 9789811390425. doi: 10.1007/978-981-13-9042-5_56. URL http://dx.doi.org/10.1007/978-981-13-9042-5_56.

[11] Ross Girshick. Fast r-cnn, 2015. URL https://arxiv.org/abs/1504.08083.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. URL https://arxiv.org/abs/1506.01497.