Universidade Federal da Bahia - UFBA

Escola Politécnica

Programa de Pós-Graduação em Engenharia Elétrica - PPGEE

Métodos para Recomendação de Hiperparâmetros de Aprendizado de Máquina na Classificação de Imagens da Construção Civil

André Luiz Carvalho Ottoni

Orientadora: Profª. Drª. Marcela Silva Novo

Salvador (BA), dezembro de 2022

Universidade Federal da Bahia - UFBA

Escola Politécnica

Programa de Pós-Graduação em Engenharia Elétrica - PPGEE

Métodos para Recomendação de Hiperparâmetros de Aprendizado de Máquina na Classificação de Imagens da Construção Civil

André Luiz Carvalho Ottoni

Tese apresentada à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como parte dos requisitos necessários para aprovação no exame de defesa de tese no curso de doutorado.

Orientadora: Profª. Drª. Marcela Silva Novo

Salvador (BA), dezembro de 2022

O91 Ottoni, André Luiz Carvalho.
Métodos para recomendação de hiperparâmetros de aprendizado de máquina na classificação de imagens da construção civil /André Luiz Carvalho Ottoni. – Salvador, 2022.
121 f.: il. color.

Orientador: Profa. Dra. Marcela Silva Novo.

Tese (doutorado) – Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal da Bahia - Escola Politécnica, 2022.

1. Aprendizado de Máquina. 2. Classificação de Imagens. 3. *Data Augmentation*. 4. Construção Civil. 5. Inteligência Artificial. I. Ottoni, André Luiz Carvalho. II. Universidade Federal da Bahia. III. Título.

CDD: 621.3

Universidade Federal da Bahia

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA (PPGEE)

ATA Nº 41

Ata da sessão pública do Colegiado do PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA (PPGEE), realizada em 12/12/2022 para procedimento de defesa da Tese de DOUTORADO EM ENGENHARIA ELÉTRICA no. 41, área de concentração Processamento de Informação e Energia, do(a) candidato(a) ANDRÉ LUIZ CARVALHO OTTONI, de matrícula 2020127299, intitulada Métodos para Recomendação de Hiperparâmetros de Aprendizado de Máquina na Classificação de Imagens da Construção Civil. Às 14:00 do citado dia, https://meet.google.com/dkt-vinb-wcd, foi aberta a sessão pelo(a) presidente da banca examinadora Profª. Dra. MARCELA SILVA NOVO que apresentou os outros membros da banca: Prof. Dr. KARCIUS DAY ROSARIO ASSIS, Prof. Dr. JES DE JESUS FIAIS CERQUEIRA, Prof. Dr. ANDRÉ CARLOS PONCE DE LEON FERREIRA DE CARVALHO e Prof. Dr. ADRIÃO DUARTE DÓRIA NETO. Em seguida foram esclarecidos os procedimentos pelo(a) presidente que passou a palavra ao(à) examinado(a) para apresentação do trabalho de Doutorado. Ao final da apresentação, passou-se à arguição por parte da banca, a qual, em seguida, reuniu-se para a elaboração do parecer. No seu retorno, foi lido o parecer final a respeito do trabalho apresentado pelo candidato, tendo a banca examinadora aprovado o trabalho apresentado, sendo esta aprovação um requisito parcial para a obtenção do grau de Doutor. Em seguida, nada mais havendo a tratar, foi encerrada a sessão pelo(a) presidente da banca, tendo sido, logo a seguir, lavrada a presente ata, abaixo assinada por todos os membros da banca.

Dr. ANDRÉ CARLOS PONCE DE LEON FERREIRA DE CARVALHO, USP

Examinador Externo à Instituição

Dr. ADRIÃO DUARTE DÓRIA NETO, UFRN

Examinador Externo à Instituição

Dr. KARCIUS DAY ROSARIO ASSIS, UFBA

Examinador Interno

Dr. JES DE JESUS FIAIS CERQUEIRA, UFBA

Examinador Interno

Dra. MARCELA SILVA NOVO, UFBA

Presidente

André Luiz Carvalho Ottoni

ANDRÉ LUIZ CARVALHO OTTONI

Doutorando(a)

Dedico este trabalho ao meu pai (José Luiz - in memorian),

minha mãe (Arlinda) e minha esposa (Lara).

Agradecimentos

Agradeço a Deus por iluminar e abençoar meus caminhos. Agradeço ao meu pai e minha mãe por todos os ensinamentos, carinho e amor. Agradeço à minha esposa, por todo o amor e por caminharmos sempre juntos. Agradeço à Família Ottoni e Família Toledo Cordeiro, por estarem sempre ao meu lado. Agradeço aos meus irmãos, Gustavo e Marcelo, e suas famílias por me apoiarem e incentivarem a lutar pelos meus objetivos. Agradeço à Profª. Marcela, pela oportunidade de ser seu aluno e por todas as valiosas orientações. Agradeço aos membros da banca examinadora pelas importantes contribuições e direcionamentos. Agradeço também aos professores que participaram de toda a minha trajetória acadêmica, em especial, Prof. Erivelton e Prof. Marcos. Agradeço à Universidade Federal da Bahia, Escola Politécnica e PPGEE por essa experiência incrível em realizar o Doutorado em Engenharia Elétrica. Também agradeço à Universidade Federal do Recôncavo da Bahia e Centro de Ciências Exatas e Tecnológicas por todas as oportunidades em ensino, pesquisa e extensão. Agradeço ao CEFET-MG e Universidade Federal de São João del-Rei, instituições que possibilitaram meu crescimento até o momento da realização do doutorado. Um agradecimento especial aos meus alunos e alunas da UFRB. Também agradeço aos amigos e colegas, em especial, do Grupo de Processamento de Imagens, GETEC e UFRB. Finalmente, agradeço à todas as agências financiadoras que contribuíram com a minha formação desde o ensino médio técnico, FAPEMIG, FAPESB, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - código de financiamento 001 e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Chamada CNPq/MCTI/FNDCT Nº 18/2021 - Universal.

Muito obrigado!

Resumo

O objetivo desta tese é propor métodos criteriosos para recomendação de hiperparâmetros de Aprendizado de Máquina na classificação de imagens da construção civil. Para isso, os métodos computacionais denominados HyperTuningSK e AutoHyperTuningSK são propostos. Esses algoritmos utilizam técnicas estatísticas para recomendação de hiperparâmetros, como Análise de Variância e o algoritmo de agrupamentos de Scott-Knott. Os experimentos observaram duas classes de hiperparâmetros: treinamento (taxa de aprendizado e otimizador) e *data augmentation*. Além disso, os métodos desenvolvidos foram testados em quatro estudos de casos: reconhecimento de vegetação em fachadas, detecção de patologias em calhas, classificação de máquinas e classificação de rachaduras. Os resultados obtidos demonstraram que os hiperparâmetros analisados influenciaram diretamente no desempenho na classificação de imagens nas aplicações analisadas. Também vale ressaltar que as configurações de hiperparâmetros ajustadas pelo HyperTuningSK obtiveram recomendações distintas de acordo com a arquitetura neural utilizada. Nesse sentido, a combinação adagrad025 alcançou $HyperScore = AAA$ para a arquitetura Densenet121. Os resultados para análise de *data augmentation*, por sua vez, mostram que duas transformações foram as mais recomendadas pelo HyperTuningSK: deslocamento em largura e deslocamento em altura. Nos experimentos do quarto estudo de caso, o algoritmo AutoHyperTuningSK recomendou otimizador adagrad e taxa de aprendizado de $0{,}02220$. Essa combinação conseguiu acurácia máxima de $99{,}48\%$, ou seja, o equivalente a classificação correta de 3.979 imagens (total de 4.000) no conjunto de teste. Os resultados para recomendação de hiperparâmetros de *data augmentation* também reforçam a eficiência da abordagem proposta usando o algoritmo AutoHyperTuningSK-DA. Nesse aspecto, a combinação recomendada alcançou acurácia média de $99{,}2\%$ nos experimentos de teste.

Palavras-chave: Aprendizado de Máquina, Classificação de Imagens, *Data Augmentation*, Construção Civil, Recomendação de Hiperparâmetros.

Abstract

This work proposes rigorous methods for hyperparameter tuning of machine learning for classifying images in construction. For this, computational methods called HyperTuningSK and AutoHyperTuningSK are proposed. These algorithms use statistical techniques for recommending hyperparameters, such as Analysis of Variance and the Scott-Knott clustering algorithm. The approach uses statistical experimental design concepts, such as analysis of variance and the Scott-Knott clustering algorithm. In addition, four case studies were used: façade vegetation detection, gutter integrity detection, machinery classification, and crack classification. The results showed that the hyperparameters affect the performance of image classification. It is also worth noting that the hyperparameter configurations adjusted by HyperTuningSK resulted in different recommendations depending on the neural architecture used. In this sense, the adagrad025 combination achieved a $HyperScore = AAA$ for the Densenet121 architecture. The results of the data augmentation analysis show that two transformations were the most recommended by HyperTuningSK: width shift and height shift. Moreover, the AutoHyperTuningSK algorithm recommended an adagrad optimizer and a learning rate of 0.02220 in the experiments for the fourth case study. This combination achieved a maximum accuracy of 99.48%, that is, the correct classification of $3{,}979$ images ($4{,}000$ in total) in the test dataset. The results for tuning data augmentation hyperparameters also confirm the efficiency of the proposed approach using the AutoHyperTuningSK-DA algorithm. In this regard, the recommended combination achieved an average accuracy of 99.2% in the test experiments.

Keywords: Building Construction, Data Augmentation, Hyperparameter Tuning, Image Classification, Machine Learning.

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

O aprendizado de máquina, em inglês, *machine learning* (ML), é um importante campo da Inteligência Artificial (IA). Sistemas inteligentes de ML possibilitam explorar e aprender sobre dados e, em seguida, reconhecer padrões. Nesse contexto, recentemente modelos de *deep learning* apresentaram grandes avanços para a IA, como por exemplo, com a adoção das Redes Neurais Convolucionais, em inglês, *Convolutional Neural Networks* (CNN) [1, 2, 3]. CNNs possuem relevantes aplicações, especialmente no processamento digital de imagens (PDI) e visão computacional [4, 5, 6]. Nesse sentido, pode-se destacar estudos com CNNs em diferentes frentes, como na agricultura [7], artes [8], medicina [9], rastreamento de objetos [10], reconhecimento de ações [11], reconhecimento facial [12], robótica [13], sensoriamento remoto [14] e veículos aéreos não-tripulados [15].

Outra área de estudo refere-se a aplicação de CNNs na análise de imagens da construção civil, ou Construção 4.0, em referência ao termo de Indústria 4.0 [16, 17, 18]. Por exemplo, na literatura, técnicas de *deep learning* são frequentemente abordadas para o monitoramento de estruturas [19, 20, 21]. Nesse sentido, uma das principais aplicações é a detecção de patologias em edificações, como rachaduras (*crack detection*) [22, 23, 24]. Nessa mesma linha, modelos de *deep learning* também são utilizados na análise de imagens de pavimentações [25, 26, 27], pontes [28, 29, 30] e tubulações [31, 32, 33]. Além disso, o *machine learning* também pode ser adotado para auxiliar na segurança dos canteiros de obras [34, 35, 36], como na detecção de equipamentos de proteção individual [37] e coletiva [38]. Nesse sentido, a literatura também apresenta estudos da aplicação de *deep learning* no reconhecimento de máquinas da construção [39, 40, 41].

Por outro lado, a adoção de técnicas de *deep learning* contempla algumas dificuldades na aplicação na indústria na construção civil e também em outras áreas de pesquisa. O primeiro entrave é a seleção de arquiteturas que garantam bons níveis de métricas de desempenho, como acurácia e precisão [42, 43]. Nesse aspecto, a literatura reúne um conjunto de possíveis estruturas, como VGG-16 [44], ResNet [45] e Xception [46]. Também é necessário a definição de outros diversos hiperparâmetros para os experimentos. Ressalta-se que, Redes Neurais Convolucionais possuem diferentes configurações iniciais (taxa de aprendizado, otimizador, número de camadas, filtros) que podem influenciar diretamente no desempenho final do modelo [47, 48].

Seguindo essa vertente, os autores de [49] destacam que muitos métodos de aprendizado de máquina dependem criticamente de configurações de hiperparâmetros. Assim, um dos maiores desafios da adoção de métodos de aprendizado de máquina é a recomendação ou otimização de hiperparâmetros (*hyperparameter tuning* ou *hyperparameter optimization*) [49, 50, 51]. Alguns métodos frequentemente aplicados na otimização de hiperparâmetros são: *Grid Search* e *Random Search* [52]. Em outra abordagem, o estudo de [53] investiga o ajuste de hiperparâmetros

para tarefas de classificação usando *Support Vector Machines*. Para isso, os autores propõem um sistema de recomendação baseado em *meta-learning* para identificar quando adotar valores padrão ou realizar o ajuste de hiperparâmetros para novos *datasets*. Na literatura, também destaca-se o crescente aumento de pesquisas dedicadas ao ajuste de hiperparâmetros de Redes Neurais Convolucionais em diversas aplicações de classificação, como: análise de sentimentos [54], classificação histopatológica [55], detecção de retinopatia diabética [56], classificação de sinais de eletroencefalograma [57] e classificação de espécies de plantas [58].

Nessa linha, um recente campo de pesquisa é o Aprendizado de Máquina Automatizado, em inglês, *Automated Machine Learning* (AutoML) [59, 60, 61]. O objetivo dos sistemas de AutoML é automatizar o processo de aplicação dos modelos de *machine learning*, e assim, facilitar o uso desses métodos por usuários com pouco conhecimento prático em IA. Os estudos em AutoML abordam diferentes caminhos no processo de automatização do aprendizado de máquina, como seleção de algoritmos [62], transferência de aprendizado [63], *meta-learning* [64] e seleção da arquitetura da rede neural [65]. Outra relevante área do AutoML refere-se justamente à otimização e recomendação automática de hiperparâmetros para modelos de *Deep Learning* [66, 67]. Assim, essa recente área de pesquisa, denominada de Deep Learning 2.0[1], busca aprimorar a primeira geração dos métodos de aprendizado profundo, evitando os ajustes manuais e adaptando-se aos objetivos do usuário.

Outro possível problema da adoção de técnicas de *machine learning* na análise de imagens da construção civil é a qualidade das bases de dados adotadas para treinamento desses modelos. Em muitos casos, estão disponíveis apenas conjuntos com poucas imagens e é necessário a aplicação de técnicas para o aumento de dados *(data augmentation)* [68, 69, 70]. De fato, técnicas de *data augmentation* têm um papel importante na aplicação de ML em bases de dados pequenas [71, 72, 73]. Isso porque, a geração de dados artificiais contribui diretamente para aumentar a capacidade de generalização dos modelos de *deep learning*, e assim, diminuir as chances de *overfitting*. Nesse aspecto, um dos desafios de se trabalhar com *data augmentation* é a definição de quais transformações (ex.: *zoom*, rotação, *flip*) serão aplicadas na geração das novas imagens [74, 75, 76]. Em termos de aprendizado de máquina, esse problema também pode ser tratado como da área de recomendação de hiperparâmetros [77, 78].

No entanto, a literatura carece do desenvolvimento de métodos para recomendação de hiperparâmetros de CNNs e *data augmentation* para a classificação de imagens da construção civil. Nesse aspecto, no Capítulo 3 será apresentada uma revisão sistemática da literatura, na qual, observou-se que apenas $18{,}4\%$ dos 76 estudos avaliados, realizaram alguma análise de hiperparâmetros. Além disso, somente $25{,}0\%$ desse total artigos analisados aplicaram alguma técnica de geração artificial de imagens. Verifica-se também uma necessidade da proposta de abordagens mais criteriosas para o estudo de hiperparâmetros na construção civil, especialmente usando métodos estatísticos, como Análise de Variância (ANOVA) [79] e o método de Scott-Knott *clustering algorithm* [80, 81].

Assim, baseando-se na atual relevância da aplicação de modelos de aprendizado de máquina na Construção 4.0 e na importância da análise de hiperparâmetros, esta tese busca responder os seguintes desafios existentes na literatura:

1. Os hiperparâmetros influenciam no desempenho de CNNs em tarefas de classificação de imagens da construção civil?

2. Como selecionar as combinações de hiperparâmetros para obter melhores métricas de acurácia?

3. Quais transformações devem ser recomendadas para a geração de imagens artificiais de acordo com o estudo de caso da Construção 4.0 analisado?

[1] https://www.automl.org/deep-learning-2-0-extending-the-power-of-deep-learning-to-the-meta-level/

1.1 Objetivos

O significativo avanço de modelos de *deep learning* nos últimos anos possibilitou o emprego de IA em várias tarefas de visão computacional da construção civil, como em detecção de patologias, segurança de canteiro de obras e reconhecimento de máquinas. No entanto, essa área de pesquisa ainda apresenta muitas adversidades que necessitam ser discutidas e trabalhadas para a obtenção de resultados mais precisos. Um desses desafios refere-se à seleção de hiperparâmetros, tendo em vista que os métodos de aprendizado de máquina podem depender diretamente dessas configurações. Além disso, o ajuste manual dessas combinações de hiperparâmetros tende a ser trabalhoso e complexo para usuários como pouco conhecimento em IA.

Outro desafio é a qualidade das bases de dados, sendo que, em muitos casos estão disponíveis apenas conjuntos de imagens com poucas amostras. Sabe-se que, dispor de uma quantidade de dados adequada para treinamento tem papel relevante na capacidade de generalização das técnicas de aprendizado profundo e em evitar o problema de *overfitting*. Entretanto, a utilização de *data augmentation* também carece da definição de quais transformações serão adotadas na criação de novas imagens artificais.

Apesar da relevância desses desafios, essa área de pesquisa ainda requer a proposta de métodos criteriosos para a definição de hiperparâmetros em tarefas de análise de imagens da Construção 4.0. Nesse sentido, os objetivos desta tese alinham-se a explorar essas dificuldades da literatura, conforme descrito na sequência.

1.1.1 Objetivo Geral

O objetivo desta tese é desenvolver métodos para análise e recomendação de hiperparâmetros aplicando aprendizado de máquina na classificação de imagens da construção civil.

1.1.2 Objetivos Específicos

Esta tese possui os seguintes objetivos específicos:

- Propor métodos para a seleção de hiperparâmetros de aprendizado (otimizador e taxa de aprendizado) usando técnicas estatísticas.
- Propor métodos para analisar e recomendar diferentes técnicas de processamento digital imagens na aplicação de *data augmentation*.

1.2 Contribuições

A principal contribuição desta tese é o desenvolvimento de métodos criteriosos para recomendação de hiperparâmetros de aprendizado de máquina na classificação de imagens da construção civil usando métodos estatísticos. Assim, pode-se destacar três pontos dos métodos propostos: (i) planejamento de experimentos para análise de hiperparâmetros de treinamento e *data augmentation*; (ii) recomendação de hiperparâmetros usando técnicas estatísticas; e (iii) algoritmos de aprendizado de máquina automatizado para ajuste de hiperparâmetros.

O planejamento de experimentos proposto visa a análise de hiperparâmetros na simulação de arquiteturas de Redes Neurais Convolucionais em problemas de classificação da construção. Nesse sentido, são investigados os efeitos de dois hiperparâmetros de treinamento (otimizador e taxa de aprendizado) nos modelos de *deep learning*. No entanto, um dos possíveis problemas na análise de imagens da construção civil são os poucos exemplos de dados para treinamento. Nesse aspecto, foram propostos também experimentos para investigar combinações de transformações nas imagens aplicadas na geração artificial de dados (*data augmentation*). Destaca-se

também a adoção de quatro estudos de casos distintos da construção civil: reconhecimento de vegetação em fachadas, detecção de patologias em calhas, classificação de máquinas na construção e classificação de rachaduras em edificações.

O segundo ponto concentra-se na proposta do método HyperTuningSK, algoritmo que utiliza técnicas estatísticas para análise e recomendação dos hiperparâmetros, a partir dos resultados experimentais. Destaca-se as seguintes etapas: análise preliminar, análise de variância e *ranking* de hiperparâmetros. A análise preliminar utiliza da estatística descritiva (médias de acurácia) no pré-processamento dos resultados dos hiperparâmetros. Na sequência, é adotado o modelo linear estatístico de análise de variância (ANOVA) [79]. O objetivo é investigar se existe diferença significativa entre as médias de acurácia dos hiperparâmetros. Posteriormente, a abordagem propõe a adoção de *rankings* de recomendação de hiperparâmetros gerados a partir do método de agrupamento de Scott-Knott [80, 81]. Destaca-se que para as fases de análise de variância e *ranking* de hiperparâmetros foi proposto um algoritmo computacional em linguagem R.

Além disso, também foi proposto o método AutoHyperTuningSK para ajuste automático de hiperparâmetros para classificação de imagens da construção civil, usando conceitos de AutoML. Esse método apresenta avanços em relação ao algoritmo HyperTuningSK, considerando que o AutoHyperTuningSK realiza a integração automática dos experimentos de aprendizado profundo e a recomendação de hiperparâmetros. Para isso, três novos algoritmos são propostos: AutoHyperTuningSK, AutoHyperTuningSK-test e AutoHyperTuningSK-DA. Nesse sentido, três tipos de hiperparâmetros podem ser automaticamente selecionados: otimizador, taxa de aprendizado e *data augmentation*. Destaca-se também a adoção da arquitetura MobileNet, reconhecida por seu desempenho para aplicações de visão computacional embarcadas, e por consequência, baixo custo computacional.

É importante salientar que outros trabalhos da literatura já utilizaram métodos estatísticos para análise de hiperparâmetros [49, 82, 83, 84]. Nesse aspecto, resultados promissores sobre o uso de conceitos estatísticos para ajuste de hiperparâmetros já foram publicados [49, 82, 83]. O estudo de [49] adota a técnica de *functional* ANOVA (fANOVA) para quantificar a importância de hiperparâmetros para métodos de ML. Nessa mesma linha, em [82] é utilizada uma abordagem com fANOVA para explorar o impacto de hiperparâmetros em modelos de *deep learning*. Em outro estudo recente, os autores de [83] aplicam o método de Scott-Knott para análise de hiperparâmetros e algoritmos de Aprendizado por Reforço na aplicação de um problema de otimização combinatória. Nessa mesma linha, o trabalho de [84] adota o método de Scott-Knott para ranquear arquiteturas de CNN na detecção de diabetes em imagens de retina. Os autores [85] também utilizam o algoritmo de Scott-Knott para agrupar arquiteturas de *deep learning* em uma aplicação de classificação de imagens para diagnóstico de câncer de mama. No entanto, esta tese inova na proposta de métodos que incorporam as técnicas de ANOVA e Scott-Knott em algoritmos computacionais para recomendação de hiperparâmetros de dois tipos: treinamento (taxa de aprendizado de otimizador) e *data augmentation*. Além disso, os estudos anteriores desse campo de pesquisa não concentraram-se na recomendação de hiperparâmetros em aplicações de classificação de imagens da construção civil.

Vale ressaltar também as contribuições em termos de artigos desenvolvidos, frutos dos estudos ao longo do desenvolvimento desta tese. Dessa forma, destaca-se que quatro trabalhos encontram-se aceitos e já publicados:

- Ottoni, A. L. C., Novo, M. S., Costa, D. B. (2022). *Hyperparameter tuning of convolutional neural networks for building construction image classification*. ***The Visual Computer***. DOI: http://doi.org/10.1007/s00371-021-02350-9 [86]. Fator de Impacto (2021): 2,835.

- Ottoni, A. L. C., Amorim, R. M., Novo, M. S., Costa, D. B. (2023). *Tuning of data augmentation hyperparameters in deep learning to building construction image classifica-*

tion with small datasets. ***International Journal of Machine Learning and Cybernetics***, 14, 171–186. DOI: http://doi.org/10.1007/s13042-022-01555-1 [87]. Fator de Impacto (2021): 4,377.

- Ottoni, A. L. C., Novo, M. S., Costa, D. B. (2022) *Deep Learning for Vision Systems in Construction 4.0: A Systematic Review*. ***Signal, Image and Video Processing***. DOI: https://doi.org/10.1007/s11760-022-02393-y [88]. Fator de Impacto (2021): 1,583.
- Ottoni, A. L. C., Novo, M. S. (2021). *A Deep Learning Approach to Vegetation Images Recognition in Buildings: a Hyperparameter Tuning Case Study*. ***IEEE Latin America Transactions***, 19(12), 2062–2070. DOI: http://doi.org/10.1109/TLA.2021.9480148 [89]. Fator de Impacto (2021): 0,967.

Além disso, dois trabalhos estão em processo de revisão:

- Ottoni, A. L. C., Souza, A. M., Novo, M. S. *Automated hyperparameter tuning for crack image classification with deep learning*. Submetido para ao periódico ***Soft Computing***.
- Ottoni, A. L. C., Novo, M. S., Oliveira, M. S. *Hyperparameter tuning for deep learning in construction machines classification*. Submetido ao periódico ***SN Computer Science***.

1.3 Organização da Tese

Esta tese está organizada em sete capítulos. No Capítulo 2, são apresentados os principais fundamentos teóricos para o desenvolvimento da pesquisa. Em seguida, o Capítulo 3 apresenta uma revisão de literatura sistemática sobre a aplicação de *deep learning* na construção civil. Na sequência, os materiais e estudo de casos adotados são descritos no Capítulo 4. O Capítulo 5, por sua vez, apresenta o método HyperTuningSK para recomendação de hiperparâmetros. Posteriormente, no Capítulo 6 é proposto a abordagem do AutoHyperTuningSK para ajuste automático de hiperperpâmetros. Finalmente, no Capítulo 7 são descritas as conclusões desta tese e propostas de continuidade.

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo, são descritos os aspectos teóricos que fundamentam os estudos desta tese. O conteúdo está dividido em quatro seções: (i) Classificação de Imagens com Aprendizado de Máquina; (ii) Redes Neurais Artificiais; (iii) Redes Neurais Convolucionais; e (iv) Recomendação de Hiperparâmetros.

2.1 Classificação de Imagens com Aprendizado de Máquina

Nesta seção, são apresentados os aspectos introdutórios sobre a classificação de imagens com algoritmos de aprendizado de máquina. Para isso, são destacados os conceitos básicos nas áreas de visão computacional e processamento digital de imagens. Além disso, são citadas algumas aplicações nesses campos de estudo. Em seguida, são apresentadas as etapas do processo de classificação de imagens com *machine learning*: imagem de entrada, pré-processamento, extração de *features* e classificador.

2.1.1 Fundamentos de Imagens

Um sinal pode ser descrito como a manifestação de um fenômeno expresso de forma quantitativa [90]. Nesse sentido, um sinal é dado a partir de uma função de variáveis independentes. Assim, um sinal de imagem pode ser representado como uma função de duas variáveis: coordenadas espaciais x e y [90, 3].

Uma função pode ser utilizada para definir a intensidade luminosa em cada ponto de uma imagem. A Equação (2.1) representa essa função [90]:

$$f(x,y) = i(x,y)r(x,y), \tag{2.1}$$

em que, $f(x,y)$ é a intensidade de luz que aparece em um determinado local da imagem, $i(x,y)$ é a quantidade de luz na cena (iluminância) e $r(x,y)$ é quantidade de luz refletida pelos objetos na cena (reflectância).

Em procedimentos computacionais, como classificação, as imagens são utilizadas no formato digital. Assim, a função $f(x,y)$ é convertida para a forma discreta, obtendo uma imagem digital como um *grid* de pixels [90, 3]. O processo de digitalização é obtido em duas etapas: amostragem e quantização. A amostragem consiste em discretizar o domínio de definição da imagem, gerando uma matriz. Os elementos dessa matriz são pixels. Por outro lado, a quantização é responsável por definir o número inteiro L de níveis possíveis para cada pixel na imagem [90].

Geralmente, as imagens são representadas em escala de cinza (*grayscale*) ou coloridas. Em escala de cinza, apenas um valor é adotado para representar a intensidade luminosa no pixel. Por exemplo, quando $L = 256$, cada pixel pode ser associado a valores entre 0 e 255 [90]. Nesse caso, uma possível representação remete a atribuir: 0 para pixel preto, 255 para pixel branco e valores intermediários são os demais níveis da escala de cinza [3].

Outro formato comum de representação de imagens é pelo sistema RGB. No formato RGB, as imagens são descritas a partir de três canais: vermelho (*R - red*), verde (*G - green*) e azul (*B - blue*). A Figura 2.1 exemplifica a adoção do sistema RGB. A partir da Figura 2.1, percebe-se que esse tipo de imagem colorida é representada em três matrizes: intensidade de cada pixel em vermelho, verde e azul, respectivamente [3].

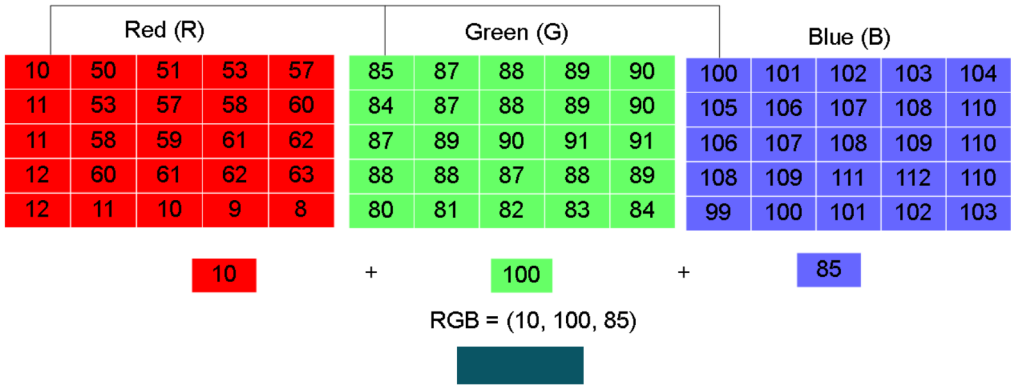


Figura 2.1: Representação de imagens no sistema RGB. Fonte: baseado em [3].

2.1.2 Sistemas de Visão Computacional e Processamento de Imagens

Os sistemas de visão computacional e processamento digital de imagens são responsáveis por permitir identificar padrões e objetos a partir de percepções visuais [3]. Destaca-se que a visão computacional atua na obtenção de respostas (predição ou classificação) a partir da entrada de dados gráficos (imagens ou vídeos). A Figura 2.2 apresenta um exemplo das etapas de um sistema com visão computacional.

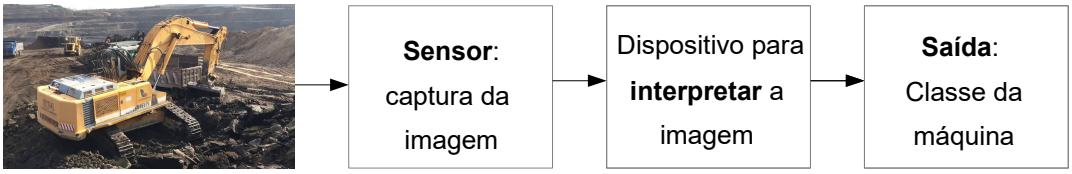


Figura 2.2: Etapas de um sistema de visão computacional. Fontes: etapas baseadas em [3] e fotografia da máquina da construção de ACID *dataset* [41].

A partir da Figura 2.2 percebe-se a importância dos dispositivos de detecção e interpretação para alcançar uma saída desejada. Nesse sentido, sensores (ex.: câmeras) podem ser utilizados para a aquisição de imagens ou vídeos. Em seguida, uma máquina deve examinar os dados adquiridos. Para isso, podem ser utilizados dispositivos de processamento da informação (ex.: computadores) dotados de algoritmos de inteligência artificial. Um exemplo de sistema de visão computacional aplicando aprendizado de máquina para detecção de objetos é mostrado a Figura 2.3.

(a) (b)

Figura 2.3: Exemplo de aplicação de detecção de objetos. Fonte das fotografias originais (sem detecção): ACID *dataset*. [40].

Por outro lado, o processamento digital de imagens consiste na aplicação de transformações nas imagens, ou seja, o resultado da operação também é uma imagem. Nessa linha, por exemplo, técnicas podem ser aplicadas para redução de ruídos, extração de bordas ou compressão de imagens [90]. Desse modo, a Figura 2.4 apresenta um exemplo de aplicação de técnicas de processamento de imagem. A partir de uma imagem de entrada (máquina na construção) são aplicadas transformações (*flip*, *zoom* e redimensionamento), obtendo então uma nova imagem processada.

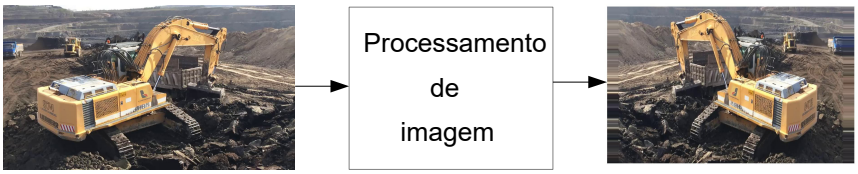


Figura 2.4: Exemplo de aplicação de técnicas de processamento digital de imagens (*flip*, *zoom* e redimensionamento). Fonte da fotografia original da máquina da construção: ACID *dataset* [41].

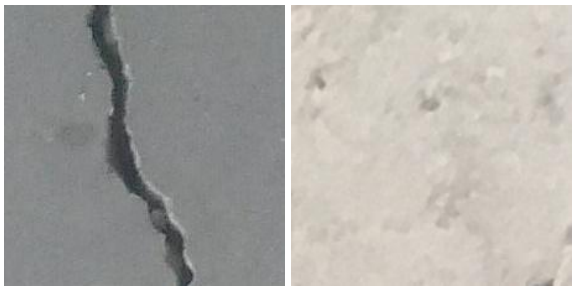
Nesse aspecto, as técnicas atuais de aprendizado de máquina aplicadas à análise de imagens congregam conceitos de visão computacional e processamento de imagens. Por exemplo, técnicas de processamento de imagens podem ser adotadas em etapas de pré-processamento, como na utilização de *data augmentation* [69]. Além disso, o processamento de imagens também é aplicado em fases de extração de características, a partir de operações com filtros convolucionais em *deep learning*. No entanto, conforme destacado por [3], processar uma imagem é distinto de interpretar uma imagem. Para isso, os algoritmos de *machine learning* devem possuir uma etapa adicional de visão computacional que possibilitam classificar ou detectar padrões.

2.1.3 Processo de Classificação de Imagens

O processo de classificação de imagens é responsável estabelecer conexões entre as propriedades das amostras com os respectivos rótulos [90]. Nesse sentido, um algoritmo classificador de imagens tem como uma entrada uma imagem e como saída a categoria (classe) que identifica

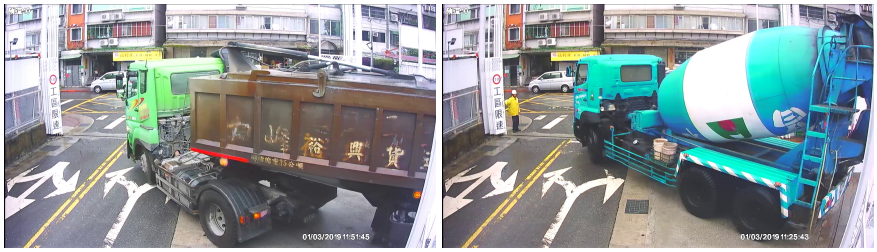
a amostra [3]. Assim, ao atribuir um mesmo rótulo à imagens distintas, conclui-se que essas amostras pertencem à mesma classe [90].

As Figuras 2.5 e 2.6 apresentam dois possíveis casos de aplicação de classificação de imagens na construção civil. O primeiro exemplo (Figura 2.5) refere-se a classificação de patologias de rachaduras em dois rótulos possíveis: (1) superfície com rachadura; (2) superfície sem rachadura. Por outro lado, a Figura 2.6 mostra um exemplo de classificação binária de máquinas da construção, em que, a primeira classe refere-se ao caminhão do tipo *dump truck* e a segunda para o veículo *concrete mixer truck*.



(a) (b)

Figura 2.5: Exemplo de aplicação de classificação de imagens: patologias na construção. (a) Superfície com rachadura. (b) Superfície sem rachadura. Fonte das fotografias: https://data.mendeley.com/datasets/5y9wdsg2zt/2



(a) (b)

Figura 2.6: Exemplo de aplicação de classificação de imagens: máquinas na construção. (a) Caminhão do tipo *dump truck*. (b) Caminhão do tipo *concrete mixer truck*. Fonte das fotografias: ACID *dataset* [40].

O processo de classificação de imagens é apresentado em etapas na Figura 2.7 e podem ser descritos como [3]:

1. **Entrada**: o computador recebe a entrada visual capturada por um dispositivo, como câmera.

2. **Pré-processamento**: a imagem é enviada para uma etapa de pré-processamento, como o objetivo de padronizar os dados. Exemplos de passos de pré-processamento são: redimensionamento da imagem ou transformação de cores (colorida para escala de cinza).

3. **Extração de características**: características (*features*) auxiliam a definir os objetos analisados nas imagens. Geralmente são informações sobre a forma ou a cor do objeto. Exemplo de *features* para um veículo: rodas, portas, largura, comprimento de entre-eixos e cor. Assim, uma lista de características extraídas são enviadas para a próxima etapa. A Figura 2.8 apresenta um exemplo de extração de características (*features*) de uma imagem.

4. **Classificador**: o classificador é responsável pela análise das *features* extraídas para predizer o rótulo da imagem.
5. **Saída**: finalmente, a saída é a definição da classe para a imagem analisada, ou ainda, a probabilidade para diferentes rótulos possíveis.

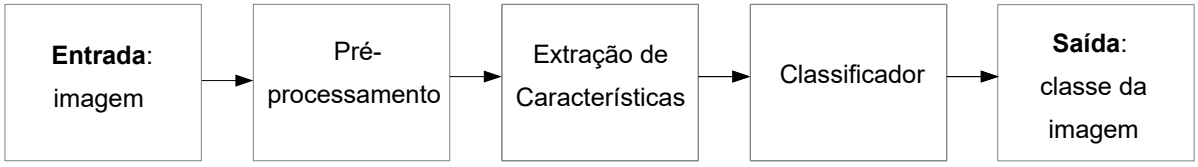
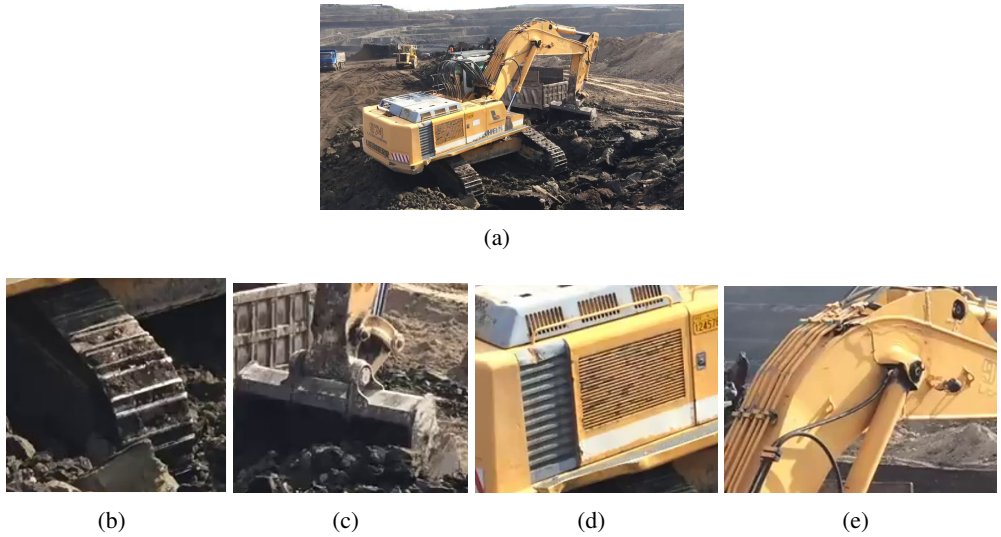


Figura 2.7: Etapas de um processo de classificação de imagens. Fonte: baseado em [3].



(a)

(b) (c) (d) (e)

Figura 2.8: Exemplo de extração de características (*features*) de uma imagem. (a) Imagem original. Fonte: ACID [40]. (b) à (e) Exemplos de *features*.

2.2 Redes Neurais Artificiais

Nesta seção, são apresentados conceitos sobre Redes Neurais Artificias. Para isso, são descritos fundamentos de um neurônio artificial e sua inspiração em neurônios biológicos. Além disso, são levantados aspectos de Redes Perceptron Multicamadas e seu processo de treinamento supervisionado.

2.2.1 Neurônio Artificial

Um neurônio artificial é a configuração mais simples de uma estrutura de Rede Neural Artificial [91, 3]. Além disso, pode-se dizer que os neurônios artificiais são modelos matemáticos simplificados de neurônios biológicos [91]. Os neurônios biológicos, por sua vez, são células do sistema nervoso central que permitem a condução e o processamento de impulsos elétricos. Isso somente é possível pois os neurônios possuem estruturas específicas que possibilitam o seu funcionamento em etapas: dendritos, corpo celular e axônio [91, 3]. Os dendritos são responsáveis

por captar sinais de entrada. Em seguida, o corpo celular processa essas informações e produz um potencial de ativação. Na sequência, o axônio possibilita conduzir os impulsos para outras células neurais. Destaca-se também nesse processo as sinapes (conexões entre neurônios) e os elementos neurotransmissores (ponderam as transmissões) [91].

McCulloch & Pitts (1943) [92] propôs um modelo de neurônio simples e contempla os principais aspectos de uma rede neural biológica [91]. Posteriormente, baseando-se nessa estrutura de um neurônio biológico, o neurônio artificial Perceptron foi idealizado por Rosenblatt (1958) [93]. O objetivo inicial do Perceptron foi implementar um modelo matemático computacional inspirado no funcionamento da retina [91]. A Figura 2.9 exemplifica um neurônio computacional do tipo Perceptron.

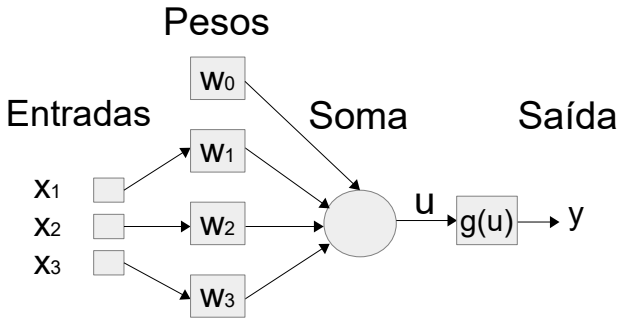


Figura 2.9: Representação de um neurônio artificial Perceptron. Fonte: baseado em [91].

A Figura 2.9 mostra uma Rede Neural Artificial do tipo mais simples, com apenas um neurônio e uma saída. Além disso, pode-se observar os seguintes elementos na representação [91]:

- Sinais de entrada (x_1, x_2, x_3): são os valores assumidos pelas variáveis.
- Pesos sinápticos (w_1, w_2, w_3): ponderam as respectivas variáveis de entrada do neurônio.
- Peso limiar (w_0): termo de ponderação que não multiplica por nenhuma entrada. Combinador linear (soma): somatório da multiplicação dos pesos pelas respectivas entradas.
- Potencial de ativação (u): resultado do combinador linear, conforme Equação 2.2:

$$u = \sum_{i=1}^{n} w_i x_i + w_0. \tag{2.2}$$

- Função de ativação (g): limita a saída do neurônio em um intervalo de valores específicos.
- Sinal de saída (y): valor final produzido pelo neurônio.

Nesse aspecto, a partir da Figura 2.9 de elementos de um neurônio artificial, também pode-se definir os passos de funcionamento de um Perceptron [91]:

1. Apresentação de valores de entrada.
2. Multiplicação dos sinais de entrada pelos respectivos pesos.
3. Cálculo do potencial de ativação.
4. Aplicação de uma função de ativação.

5. Cálculo da saída.
6. Cálculo do erro entre a saída e o valor desejado.
7. Ajuste dos pesos sinápticos.

Esse processo de aprendizagem de um neurônio artificial é denominado de treinamento supervisionado, em que, para cada amostra dos sinais de entrada se tem a respectiva saída desejada [91, 3]. Assim, os pesos sinápticos são ajustados se a saída produzida pelo Perceptron não é igual ao valor da saída desejada (amostra de treinamento). Por outro lado, os coeficientes (pesos) não são alterados se a saída do neurônio for coincidente com o valor esperado. Essas etapas são repetidas em uma sequência de passos para todas as amostras de treinamento. Assim, o objetivo é ajustar os pesos (treinamento) para que a saída do neurônio artificial seja finalmente igual ao valor desejado para todas as amostras [91].

2.2.2 Redes Perceptron Multicamadas

A resolução de problemas com apenas um neurônio Perceptron oferece limitações, por exemplo, aplicando-se principalmente na modelagem de classes linearmente separáveis. Nesse aspecto, as Redes Neurais Artificiais com mais neurônios e camadas foram propostas de modo a possibilitar a aplicação em tarefas mais complexas [91, 94, 3]. A Figura 2.10 exemplifica uma Rede Perceptron Multicamadas, em inglês, *Multi Layer Perceptron* (MLP).

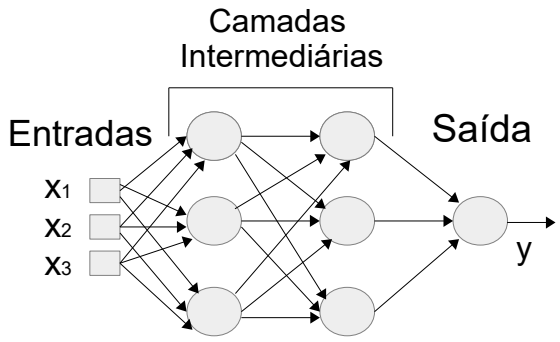


Figura 2.10: Representação de uma Rede Perceptron Multicamadas. Fonte: baseado em [91].

Conforme Figura 2.10 demonstra, diferentemente de uma rede com apenas um neurônio (Figura 2.9), as Redes Perceptron Multicamadas podem ser compostas por diversos neurônios artificiais e camadas. Nesse sentido, percebe-se na Figura 2.10 que a RNA exemplificada é composta por duas camadas intermédias de neurônios. Além disso, as saídas dos neurônios de uma camada são aplicadas como entradas dos neurônios da camada sequente [91]. Dessa forma, para cada conexão entre os neurônios é necessários ajustar os pesos da rede neural. Esse processo de treinamento de uma Rede Neural Multicamadas é descrito na próxima seção.

Processo de Treinamento

O processo de treinamento supervisionado de uma Rede Perceptron Multicamadas é realizado adotando-se o método de *Backpropagation* [91, 3]. O algoritmo de *Backpropagation* é um método de otimização baseado no gradiente descendente, em que, utiliza-se da estratégia de retropagação como forma de calcular o erro nas camadas ocultas. Assim, nesse algoritmo, os pesos da RNA são ajustados a partir da execução de duas etapas: propagação adiante (*forward*) e propagação reversa (*backward*).

Na propagação adiante, sinais de entrada são aplicados na RNA e são propagados (camada a camada) até a produção da resposta de saída [91]. O objetivo nesta fase é calcular as sinal de saída da rede, tomando-se os valores atuais para os pesos. Na sequência, utiliza-se as saídas desejadas (conjunto de treinamento) para comparar com os valores de resposta produzidos pela RNA [91].

Em seguida, é aplicada a propagação reversa pelo algoritmo de *Backpropagation*. Nesta fase, são realizados os ajuste de todos os pesos dos neurônios da RNA. Essa sintonia é efetuada na última camada em direção à primeira, levando-se em consideração o erro entre o valor desejado e saída produzida pela rede neural. Dessa forma, os pesos de ajustam cada interação (*forward* e *backward*) e aplicação sucessiva dessas duas etapas [91]. Na próxima seção, são descritos mais detalhes sobre a otimização dos pesos durante o processo de aprendizado.

Otimização e Aprendizado

O treinamento de RNAs envolve conceitos de otimização em passos de aprendizado [91, 3]. Destaca-se que uma rede neural deve sintonizar os pesos de forma a buscar minimizar o erro entre as respostas geradas e a saída desejada [3]. A Figura 2.11 ilustra esse processo.

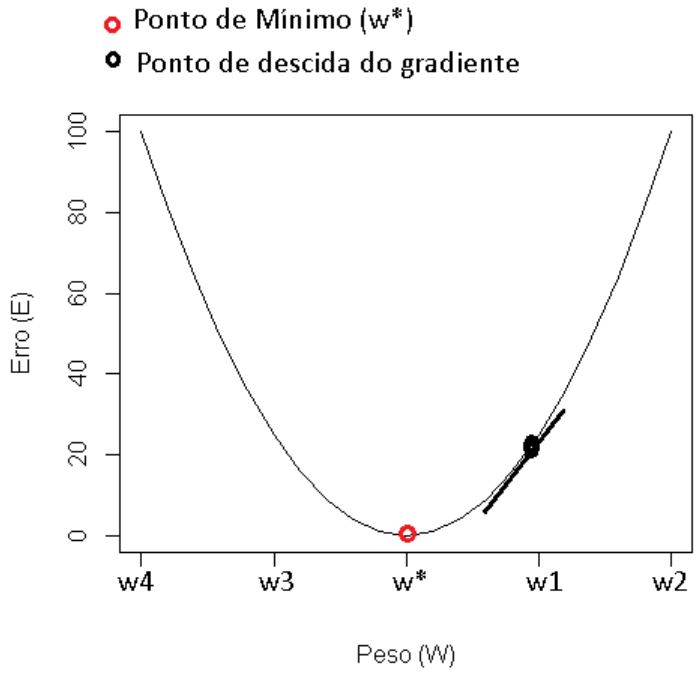


Figura 2.11: Otimização de pesos em uma Rede Neural Artificial. Fonte: baseado em [91].

A Figura 2.11 exemplifica uma curva de erro produzido pela RNA em função dos pesos adotados (w). Pode-se observar que o menor erro é gerado ao adotar os valores de pesos ótimos ($w*$ - ponto em vermelho no gráfico). Esses pesos ótimos podem ser encontrados iterativamente adotando métodos de otimização baseados no Gradiente Descendente [91, 3]. O gradiente é dado em função da inclinação da linha que é tangente a curva em um ponto [3]. Assim, para calcular o gradiente deriva-se o erro em relação aos pesos, conforme Equação (2.3):

$$G = \frac{dE}{dw}. \tag{2.3}$$

Além do operador gradiente (Equação (2.3)), para descer a curva do erro é necessário a definição do tamanho do passo (ou, taxa de aprendizado). A taxa de aprendizado (α) é um importante

hiperparâmetro no treinamento de uma Rede Neural Artificial [3]. Essa taxa é responsável por regular a velocidade do aprendizado. Assim, multiplicando a taxa de aprendizado pelo gradiente (direção), têm-se o ajuste dos pesos por iteração i, conforme Equação (2.4) [3]:

$$w_{i+1} = w_i - \alpha \frac{dE}{dw_i}. \tag{2.4}$$

Nesse aspecto, em razão da importância da taxa de aprendizado no processo de treinamento e otimização dos pesos de uma RNA, esse hiperparâmetro deve ser criteriosamente definido. Por exemplo, um tamanho de passo muito pequeno pode levar a uma convergência demorada, e assim, exigir muito tempo de treinamento [3]. Por outro lado, quando α é muito alto pode-se gerar divergência no cálculo dos pesos, ou seja, resultando em valores de w mais distantes do ótimo. Assim, uma taxa de aprendizado ideal possibilita descrever o erro consistentemente em busca dos pesos ótimos [3].

Métodos de Otimização

Na literatura, existem diferentes métodos de otimização desenvolvidos para atualização de pesos de RNAs [95]. Destacam-se os algoritmos baseados em gradiente descendente, como adadelta [96], adagrad [97], adam [98], adamax [98] e sgd [94].

Em geral, esses métodos incluem aprimoramentos em relação ao algoritmo do gradiente descendente tradicional. Por exemplo, o gradiente descendente estocástico (*stochastic gradient descent - sgd)*) seleciona aleatoriamente pontos de partida na curva de otimização [3]. Desse modo, isso fornece várias soluções iniciais com pesos diferentes, e assim, são calculados os mínimos locais. Na sequência, a partir desses valores mínimos é encontrado o mínimo global.

Além disso, o gradiente descendente tradicional necessita de todo o conjunto de treinamento para dar um passo em direção ao mínimo, conforme sequência à seguir [3]:

1. Selecione todos os dados.
2. Calcule o gradiente.
3. Atualize os pesos e realize a descida de degrau.
4. Repetir por várias interações.

Por outro lado, o sgd seleciona aleatoriamente uma instância no conjunto de treinamento e calcula o gradiente com base nessa única instância [3]:

1. Embaralhe as amostras aleatoriamente no conjunto de treinamento.
2. Selecione um instância de dados.
3. Calcule o gradiente.
4. Atualize os pesos e realize a descida de degrau.
5. Selecione outra instância de dados.
6. Repetir por várias interações.

Vale destacar também algumas caracaterísticas dos seguintes otimizadores [95]:

- adagrad [97]: é um algoritmo baseado em gradiente descendente que consegue adaptar a taxa de aprendizado em relação aos parâmetros. Nesse aspecto, realiza atualizações distintas de acordo com a frequência de utilização dos parâmetros do modelo. Assim, o valor para a taxa de aprendizado é alterado em cada interação de tempo para cada parâmetro.

- adadelta [96]: é uma extensão do método adagrad e busca aprimorar o ajuste da taxa de aprendizado adaptativa. Além disso, os autores ressaltam que esse método é menos sensível às variações de hiperparâmetros em relação à outros otimizadores.
- adam [98]: é uma abreviação para *adaptive moment estimation* e também é um método baseado no otimizador sgd. Além disso, utiliza conceitos no ajuste de parâmetros similar a técnica denominada de *momentum*. O *momentum* possibilita que o gradiente navegue em direções relevantes e suavize a oscilação em direções irrelevantes para otimização. Isso possibilita uma convergência mais rápida e reduz oscilações [3]. Os autores do método adam também apresentaram outra variação para esse otimizador baseando-se na norma do infinito, denominado de adamax [98].

2.3 Redes Neurais Convolucionais

Nesta seção, é realizada uma breve descrição sobre Redes Neurais Convolucionais. Para isso, são apresentados aspectos sobre o funcionamento desses métodos. Além disso, são levantados fundamentos de camadas convolucionais e outros tipos de camadas geralmente adotadas nessas estruturas.

2.3.1 Arquitetura de uma Rede Neural Convolucional

As Redes Neurais Convolucionais, em inglês, *Convolutional Neural Networks*, (CNN) são métodos de aprendizado de máquina profundo (*deep learning*). Basicamente, as CNNs possuem a capacidade de extrair características das imagens a partir da aplicação de filtros convolucionais. A estrutura básica de uma Rede Neural Convolucional é representada na Figura 2.12.

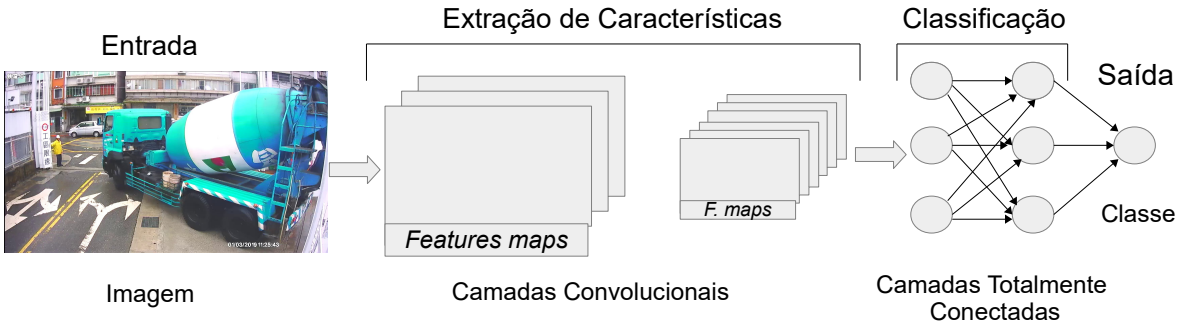


Figura 2.12: Representação de uma Rede Neural Convolucional. Fonte: baseado em [3]. Imagem de entrada: Acid *dataset* [41].

A arquitetura mostrada da Figura 2.12 pode ser sintetizada em quatro partes principais [3]:

- Camada de entrada: recebe a imagem a ser classificada.
- Camadas convolucionais: responsável pela extração de características (*features*) das imagens.
- Camadas totalmente conectadas: atua de forma semelhante às redes tradicionais MLP.
- Camada de saída: realiza a predição final para imagem.

Nas próxima seção, o funcionamento das camadas convolucionais é explicado em mais detalhes.

2.3.2 Camadas Convolucionais

Na matemática, a convolução é uma operação entre duas funções que produz uma terceira função modificada. A operação de convolução pode ser representada pela Equação (2.5) [99]:

$$s(t) = (x * w)(t), \tag{2.5}$$

em que, t é o instante de tempo, x é a função de entrada, w é a função de ponderação e s é o resultado da convolução.

Seguindo essa linha, em termos de terminologia para Redes Neurais Convolucionais em classificação de imagens pode-se dizer que: o primeiro argumento (função x na Equação (2.5)) da convolução refere-se a imagem de entrada; o segundo argumento (função w na Equação (2.5)) é o filtro convolucional (ou *kernel*); por fim, a saída é denominada de mapa de características, em inglês, *feature map* [99, 3].

Os filtros em camadas convolucionais são aplicados no domínio espacial, ou seja, no próprio plano da imagem (conjunto de pixels) [90]. O resultado de uma operação de filtragem em um determinado pixel da imagem depende do nível de cinza atual (imagem em *grayscale*), dos valores dos pixels da vizinhança e também dos pesos do filtro aplicado, também conhecido como máscara ou *kernel* [90, 3]. O resultado dessa operação é exemplificado na Equação (2.6) para uma máscara de ordem 3×3 [90]:

$$R = \sum_{i=1}^{9} w_i z_i, \tag{2.6}$$

em que, z_i são os níveis de cinza $f(x,y)$ da imagem sob o filtro; w_i são os pesos (coeficientes) da máscara e; R é o novo valor para o pixel da posição (x, y) da imagem. Esse processo se repete até o filtro se movimentar por todas as posições da imagem. Então o objetivo de um filtro convolucional é deslizar pela imagem (pixel por pixel) de forma a extrair recursos significativos para identificar os objetos [3]. A Figura 2.13 exemplifica o funcionamento de um filtro convolucional.

A Figura 2.13 retrata a operação entre uma imagem de entrada, um filtro convolucional (3×3) e um resultado de saída para um pixel ($R = -3$) na imagem convoluída (*feature map*). Ressalta-se que cada camada convolucional possui um ou mais filtros. Além disso, cada filtro convolucional produz seu próprio mapa de características. Os neurônios, por sua vez, são as unidades da matriz do *kernel*, ou seja, um filtro (3×3) possui 9 neurônios [3]. Dessa forma, as camadas convolucionais podem ser associadas como o núcleo de uma estrutura CNN [3].

2.3.3 Outras Camadas e Operações

As camadas convolucionais representam a principal estrutura de uma arquitetura de uma Rede Neural Convolucional, conforme descrito na última seção. No entanto, uma CNN também pode possuir outros tipos de camadas e operações, como: camadas totalmente conectadas, camada de *pooling* e operação de *dropout* [94, 3].

As camadas totalmente conectadas representam um sistema tradicional MLP na arquitetura CNN. Essas estruturas são utilizadas após a extração das *features* pelas camadas convolucionais. Nesse sentido, as caraterísticas são então usadas no processo de classificação de imagens [3].

As camadas de *pooling*, por sua, vez, têm como objetivo reduzir a complexidade nos cálculos de extração de características. Para isso, essas estruturas aplicam operações estatísticas simples (máximo ou média) em regiões de pixels [94, 3]. Por exemplo, se a operação de máximo *pooling* for aplicada em um conjunto de 4 pixels (*pool size* = 2×2), a saída para a próxima camada será apenas o maior valor entre esses pixels. Nesse sentido, é comum adicionar cama-

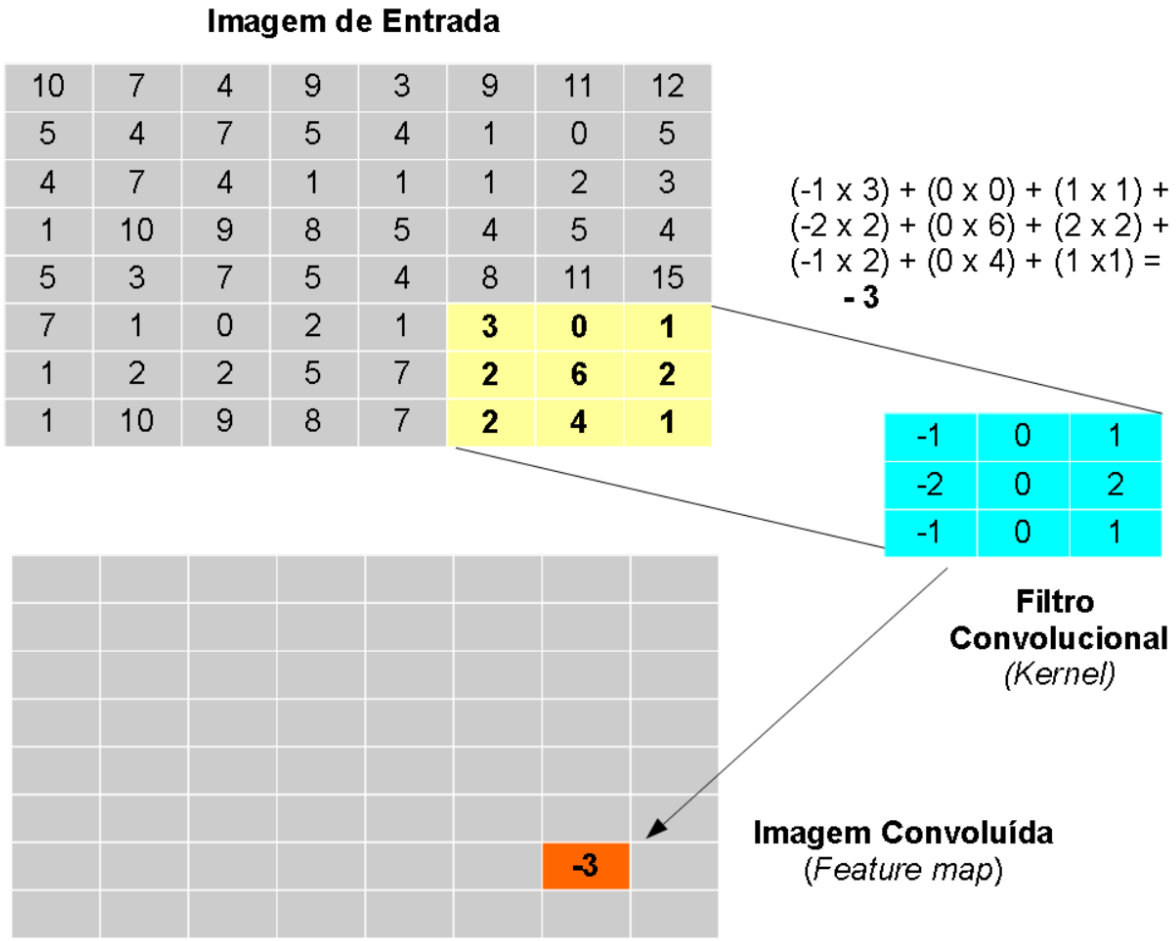


Figura 2.13: Funcionamento de um filtro convolucional. Fonte: baseado em [3].

das de *pooling* após as camadas convolucionais, reduzindo o mapa de características (número de parâmetros processados) [3].

Outra técnica comum em Redes Neurais Convolucionais é o *dropout*. Essa operação de regularização é aplicada na tentativa de evitar *overfitting* [94, 3]. Para isso, a técnica de *dropout* desliga uma porcentagem de neurônios durante os passos de treinamento da rede. Assim, o modelo tenderá aumentar sua capacidade de generalização em uma fase de teste [3].

2.4 Recomendação de Hiperparâmetros

Os hiperparâmetros são variáveis que devem ser definidas pelo projetista do sistema de aprendizado de máquina antes de iniciar o treinamento [94, 59, 3]. Esse processo é distinto do ajuste de parâmetros (ou pesos) realizado em algoritmos de *machine learning*, como pelo método de *Backpropagation*. Assim, os parâmetros são variáveis otimizadas e aprendidas durante o treinamento. Por outro lado, os hiperparâmetros não são aprendidos diretamente pelos métodos de *machine learning*, como por exemplo, taxa de aprendizado, número de neurônios e quantidade de épocas. Nesse aspecto, [3] apresenta uma divisão de hiperparâmetros de Redes Neurais Artificiais em três grandes grupos:

- Arquitetura neural: responsáveis por definir diretamente a estrutura da RNA, como número de camadas, neurônios e função de ativação.

- Aprendizado e otimização: adotados no processo de treinamento (atualização dos pesos), como taxa de aprendizado, algoritmos de otimização e número de épocas.
- Técnicas de regularização: adotados na tentativa de evitar *overfitting*, como camadas de *dropout* e *data augmentation*.

Os métodos baseados em *deep learning* dependem de uma variedade de hiperparâmetros [59]. Isso torna o problema de recomendação ou otimização de hiperparâmetros muito desafiador em vários aspectos [94]. Destaca-se, por exemplo, que avaliar o ajuste de hiperparâmetros torna-se muito difícil para modelos profundos ou em grandes *datasets* [59]. Além disso, o espaço configuração pode ser bem complexo, envolvendo valores inteiros (ex.: número de camadas), reais (ex.: taxa de aprendizado) e categóricos (ex.: usar ou não usar *dropout*). Outra característica é que os hiperparâmetros podem apresentar bons desempenhos na forma condicional. Por exemplo, um hiperparâmetro pode ser relevante apenas se outro hiperparâmetro (ou combinações de hiperparâmetros) tiver determinados valores [59]. Nesse aspecto, métodos para recomendação de hiperparâmetros dependem da execução de experimentos e observação do modelo [3]. Além disso, uma boa definição de hiperparâmetros está sujeito ao *dataset* utilizado e da tarefa analisada [3].

O autor de [94] define o processo de otimização de hiperparâmetros em etapas:

1. Selecionar um conjunto de valores de hiperparâmetros.
2. Construir o modelo correspondente.
3. Realizar o treinamento e medir o desempenho no conjunto de validação.
4. Selecionar o próximo conjunto de valores de hiperparâmetros.
5. Repetir o processo.
6. Eventualmente, medir o desempenho nos dados de teste.

Assim, as etapas desse processo podem ser aplicadas de acordo com o formato da seleção do conjunto de valores dos hiperparâmetros ou medida de desempenho avaliada. Por exemplo, podem ser adotados métodos *Grid Search* ou *Random Search* [59]. Nas técnicas de *Grid Search* é utilizado o planejamento de experimentos fatorial [79], em que, é especificado um conjunto finitos de valores para cada hiperparâmetro [59]. Por outro lado, os métodos baseados em *Random Search* adotam pesquisas aleatórias nas configurações, ou seja, os valores para os hiperparâmetros são gerados aleatoriamente [59].

CAPÍTULO 3

Revisão Sistemática da Literatura

A Indústria 4.0 dispõe de ferramentas e métodos que inovam a construção civil [100, 101]. Essa nova fase, denominada Construção 4.0, representa a digitalização da indústria da construção [102, 103]. Nesse sentido, alguns exemplos de tecnologias digitais utilizadas na Construção 4.0 são: *Building Information Modelling (BIM)*, realidade aumentada, internet das coisas e inspeção com drones [104, 105].

A Inteligência Artificial (IA) é outra importante tecnologia digital adotada na Construção 4.0 [102, 103]. Destaca-se a aplicação de técnicas de Aprendizado de Máquina em sistemas de visão artificial da construção civil [16, 17, 39]. Seguindo essa linha, o objetivo deste capítulo é apresentar uma revisão sistemática da aplicação de *deep learning* em sistemas de visão computacional na construção civil. Para isso, foram analisadas algumas aplicações, como na análise de patologias, inspeção de construções, segurança das obras, reconhecimento de equipamentos de construções e identificação de ações de trabalhadores. Além disso, foi investigado como a literatura da área de Construção 4.0 tem lidado com relevantes desafios da aplicação de *machine learning*, como seleção de hiperparâmetros e adoção de aumento artificial de bases de dados (*data augmentation*).

Na literatura recente, outros trabalhos realizaram revisões sobre a aplicação de Aprendizado Profundo na construção, tais como [106, 107, 108, 109]. No entanto, a presente revisão sistemática inova principalmente nos seguintes aspectos: metodologia de pesquisa observando os estudos mais citados e mais recentes de periódicos da área da construção civil; análise de dados das publicações, como instituições e países dos autores; análise da seleção de hiperparâmetros e; investigação da aplicação de *data augmentation* na Construção 4.0.

Este capítulo está estruturado em cinco seções. O método de pesquisa é proposto na primeira Seção. Na sequência, dados preliminares desses trabalhos são listados na Seção 2. Em seguida, as Seções 3 e 4 apresentam considerações da literatura sobre a adoção de métodos de *deep learning* e aplicações na construção civil, respectivamente. Finalmente, as discussões e perspectivas futuras para a área são apresentadas na Seção 5.

3.1 Método de Pesquisa

O objetivo desta revisão sistemática foi analisar estudos com a temática de *deep learning* em sistemas de visão computacional da construção civil. Para isso, sete perguntas de pesquisa (*Research Questions* - RQ) foram determinadas para análise dos artigos:

RQ1: Qual a principal área da construção civil abordada do artigo?

RQ2: Quais as principais aplicações de visão computacional?

RQ3: Quais as técnicas de *deep learning* foram aplicadas?

RQ4: O artigo descreve abordagens para a definição de hiperparâmetros?
RQ5: O artigo apresenta estratégias para *data augmentation*?
RQ6: Quais os principais resultados dos artigos?
RQ7: Quais as limitações dos estudos e perspectivas para trabalhos futuros?

A base de dados *Scopus*[1] foi utilizada para a realização da busca dos artigos. Em seguida, foram selecionados cinco periódicos indexados com publicações na área de Construção 4.0 e fator de impacto (JCR) maior que 1,0 (referente à 2020). Os periódicos foram selecionados pela relevância no cenário científico internacional na área de construção civil, sendo:

1. *Advances in Civil Engineering* (ACE) (JCR = 1,924).
2. *Automation and Construction* (AutCon) (JCR = 7,700).
3. *Computer-Aided Civil and Infrastructure Engineering* (CACIE) (JCR = 11,775).
4. *Construction and Building Materials* (CBM) (JCR = 6,141).
5. *Journal of Computing in Civil Engineering* (JCCE) (JCR = 4,640).

Após a delimitação dos periódicos, foi fixada a *string* de busca na base *Scopus* como sendo: "*deep learning*" "*image*". Para cada um dos cinco periódicos foram salvos trabalhos presentes em dois *rankings*:

- 10 artigos mais **citados** em cada periódico.
- 10 artigos mais **recentes** em cada periódico.

Esperava-se que, os artigos com mais citações retratem possivelmente os tópicos com maior relevância científica para área. Por outro lado, a adoção de um conjunto de trabalhos mais recentes visa analisar as novidades e direcionamentos futuros para essa área de pesquisa.

Assim, foram definidos os seguintes princípios para pesquisa e inclusão de artigos:

- Artigos publicados na base Scopus.
- Manuscritos de cinco periódicos da construção civil: ACE, AutCon, CACIE, CBM ou JCCE.
- *Rankings* com 10 mais citados ou 10 mais recentes artigos em cada periódico.
- Estudos direcionados na adoção de *deep learning* na análise de imagens da construção.
- Artigos publicados entre 2017 e 2021.

Além disso, também foram definidos três critérios para a exclusão de artigos da revisão sistemática:

- Estudos que não analisaram imagens.
- Trabalhos que não aplicaram técnicas de *deep learning*.
- Artigos repetidos nos dois *rankings* (mais citados e mais recentes).

[1] http://www.scopus.com

Nesse sentido, ao detectar uma dessas características no manuscrito, o artigo seria retirado dos trabalhos analisados na presente revisão de literatura.

A Figura 3.1 apresenta um diagrama sobre o procedimento de seleção (pesquisa, inclusão e exclusão) dos artigos. O número inicial esperado de publicações para análise seria de 100 trabalhos (20 artigos por revista). No entanto, 24 estudos foram excluídos: dois artigos repetidos nos *rankings* (mais citados e mais recentes); onze estudos que não analisavam imagens; onze trabalhos que não utilizavam *deep learning*. Dessa forma, o número de publicações consideradas nesta revisão sistemática passou para 76 artigos.

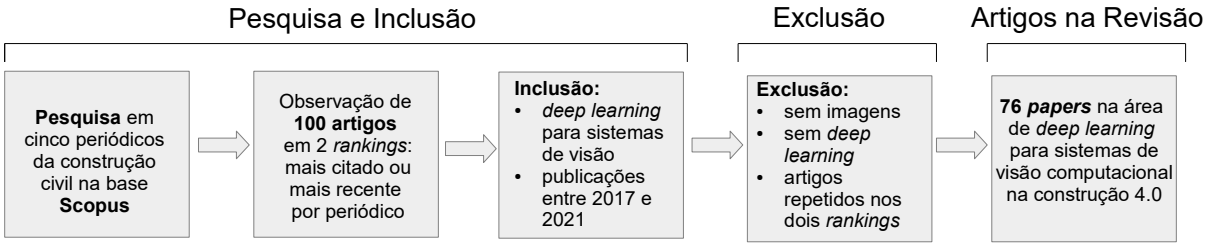


Figura 3.1: Diagrama de seleção de artigos desta revisão sistemática.

3.2 Análise Preliminar

Nesta seção, são apresentados resultados preliminares gerais sobre os artigos incluídos na revisão sistemática. Inicialmente foi realizada uma análise da origem territorial e acadêmica dos estudos. Para isso, foi observada a instituição principal dos primeiros autores de cada um dos artigos. Além disso, foram armazenados os países sedes dessas universidades/centros de pesquisa. Nesse sentido, vale destacar as quatro instituições com mais trabalhos presentes nesta revisão (observando o primeiro autor):

1. *Huazhong University of Science and Technology* (China): 5 artigos.
2. *University of Manitoba* (Canadá): 4 artigos.
3. *Yonsei University* (Coreia do Sul): 4 artigos.
4. *Hong Kong Polytechnic University* (Hong Kong): 3 artigos.

A Tabela 3.1, por sua vez, apresenta o número de artigos selecionados por periódico e o *ranking* do número de publicações por país.

Nesse aspecto, pode-se observar que os países com mais artigos incluídos nesta revisão sistemática são China, EUA e Canadá. Além disso, a Ásia é o continente com mais publicações: 44 artigos (57,89%). O continente asiático também contempla o maior número de países presentes na Tabela 3.1, sendo oito distintas nacionalidades para o primeiro autor. Por outro lado, os demais continentes possuem apenas dois (América), um (Europa e Oceania) ou nenhum país (África) representando. A análise da Tabela 3.1 revela ainda que a revista AutCon possui mais artigos (19 trabalhos) incluídos na revisão sistemática, seguida do periódico JCCE com 18 publicações. Dessa forma, indicando a alta aderência desses dois periódicos ao tema desta pesquisa. Por outro lado, o periódico com menos trabalhos é o CBM (11). Além disso, China e EUA são os únicos países com publicações em todos os cinco periódicos. Vale ressaltar também que, a China concentrou 68% das suas 25 publicações em apenas dois periódicos: ACE (10 artigos) e CBM (7 artigos).

Tabela 3.1: Número de artigos selecionados por periódico e *ranking* do número de publicações por país (considerando a instituição principal do primeiro autor).

Continente	**País**	**ACE**	**AutCon**	**CACIE**	**CBM**	**JCCE**	**Total**	*Ranking*
América	Canadá	0	1	4	1	3	9	**3º**
	EUA	1	5	5	3	5	19	**2º**
Ásia	China	10	5	2	7	1	25	**1º**
	Coreia do Sul	0	1	1	0	5	7	4º
	Hong Kong	0	2	1	0	2	5	5º
	Japão	0	2	1	0	0	3	6º
	Singapura	0	0	1	0	0	1	10º
	Vietnã	1	1	0	0	0	2	7º
	Taiwan	0	0	0	0	1	1	10º
Europa	Holanda	0	2	0	0	0	2	7º
Oceania	Austrália	0	0	1	0	1	2	7º
	Total	12	19	16	11	18	76	

A Tabela 3.2, por sua vez, apresenta a porcentagem de artigos selecionados de acordo com alguns critérios observados nesta revisão sistemática. Por exemplo, quanto ao tipo de método de visão computacional, grande parte dos estudos analisados (50 trabalhos) aplicaram técnicas para detecção de objetos. Em seguida, 23,7% utilizaram métodos para tarefas de classificação e 19,7% para segmentação. Ressalta-se que alguns estudos adotaram mais de um tipo de abordagem de visão computacional. Em relação a área da construção civil, a maioria das publicações (50) dedicam-se à tarefa de análise de patologias e inspeção das construção. Por outro lado, 17 pesquisas foram classificadas da área de segurança e reconhecimento de equipamentos das construções. Outros campos de estudos somam nove trabalhos.

Tabela 3.2: Porcentagem de artigos selecionados de acordo com critérios observados.

Análise	**Critério**	%
Aplicação de Visão Computacional	Classificação	23,7
	Detecção	65,7
	Segmentação	19,7
Área da Construção	Patologias	65,8
	Segurança/Equipamentos	22,4
	Outras	11,8
Análise de Hiperparâmetros	Sim	18,4
	Não	81,6
Data Augmentation	Sim	25,0
	Não	75,0

A partir da Tabela 3.2 também é possível observar uma certa carência de pesquisas na área de *deep learning* para a Construção 4.0 que considerem dois importantes aspectos: análise de hiperparâmetros e aplicação de *data augmentation*. Nesse sentido, em apenas 18,4% dos estudos analisados foi encontrada alguma metodologia para a seleção de hiperparâmetros/arquiteturas dos modelos de *machine learning* utilizados no processamento das imagens. Além disso, somente 25% das publicações desta revisão sistemática utilizaram alguma técnica para geração de imagens artificiais e o aumento de bases de dados.

Nas próximas seções, serão aprofundados aspectos apontados na Tabela 3.2 e observados nos estudos analisados da revisão sistemática.

3.3 Métodos de Visão Computacional e Desafios do *Deep Learning*

Nesta seção, são apresentados alguns métodos de visão computacional e desafios de *deep learning* na seguinte sequência: (i) aplicações de visão computacional; (ii) seleção de hiperparâmetros; e (iii) aplicação de *data augmentation*. Além disso, é importante destacar também que o escopo desta revisão limitou-se apenas aos métodos de *deep learning*, ou seja, quando for utilizado o termo *machine learning*, refere-se apenas aos modelos de aprendizado profundo.

3.3.1 Aplicações de Visão Computacional

A Tabela 3.3 agrupa os estudos de acordo com a aplicação de visão computacional adotada: classificação, detecção ou segmentação. Vale destacar também que os métodos de *deep learning* são exemplificados de acordo com o tipo de tarefa. Além disso, a Tabela 3.3 apresenta 83 citações, número superior aos 76 trabalhos analisados, tendo em vista que alguns artigos apresentam mais de uma aplicação.

Tabela 3.3: Métodos de visão computacional com *deep learning* na construção civil.

Técnica	**Trabalhos**	**Exemplos de métodos**
Classificação	[110], [111], [112], [23], [18], [113], [17], [24], [114], [32], [27], [115], [116], [117], [118], [119], [120], [121]	AlexNet, ResNet e VGG-16
Detecção	[122], [123], [29], [16], [19], [111], [35], [31], [28], [124], [113], [37], [34], [20], [125], [126], [127], [25], [128], [129], [39], [130], [38], [131], [132], [133], [134], [30], [27], [135], [136], [137], [138], [139], [140], [26], [36] [141], [142], [143], [40], [41], [144], [119], [145], [146], [33], [147], [148] [149], [150]	Faster R-CNN, SSD e YoLO-V3
Segmentação	[110], [23], [18], [113], [127], [151], [152], [27], [115], [153], [138], [154], [140], [22], [155]	CrackSegNet e U-Net

A partir da Tabela 3.3, pode-se observar que 18 trabalhos aplicaram algum método de *deep learning* para classificação de imagens da construção civil. Nesses estudos, destaca-se a utilização de arquiteturas de Redes Neurais Convolucionais relevantes da literatura, como: VGG-16 [17, 18], AlexNet [113, 121], ResNet [114, 24]. Por outro lado, em 15 estudos foi observada a utilização de técnicas para segmentação de imagens. Para isso, foram abordadas métodos de *deep learning* como U-Net [27] e CrackSegNet [140].

No entanto, grande parte das pesquisas investigadas (50 trabalhos) nesta revisão sistemática utilizou métodos para detecção de objetos da construção. Nesse sentido, esses estudos utilizaram diversas técnicas de *machine learning*, destacando várias relevantes na área de visão computacional: Faster R-CNN [31, 37], SSD [29, 26], e YoLO [41, 33]. Para exemplificar, a Figura 3.2 apresenta um exemplo de detecção de equipamentos de proteção individual (capacetes) utilizando o método SSD-Mobilenet [133].

3.3.2 Seleção de Hiperparâmetros

A seleção dos hiperparâmetros é uma etapa relevante na realização de experimentos com modelos de *machine learning* [59]. No entanto, entre os trabalhos analisados nesta revisão, em

Figura 3.2: Exemplo de detecção de equipamentos de proteção usando *deep learning* (Fonte: [133]).

apenas 14 estudos foi verificada a apresentação de alguma metodologia para a definição de pelo menos um tipo de hiperparâmetro:

- Treinamento - ex.: taxa de aprendizado ou otimizador [148, 22, 123, 131, 17, 120, 125, 128].
- Arquitetura - ex.: número de camadas ou neurônios [31, 125, 150].
- Regularização - ex.: taxa de *dropout* [32, 38, 21, 143].

Os autores de [22] realizaram experimentos com uma *Fully Convolutional Network* (FCN) para segmentação de rachaduras em uma base de dados com aproximadamente 800 imagens. Assim, foram investigados diferentes valores para taxa de aprendizado (lr): 10^{-6}, 10^{-5}, 10^{-4}, e 10^{-3}. Para isso, são analisadas graficamente as métricas de acurácia, *recall*, precisão e *F-score* ao longo de 20 épocas de treinamento e validação. De acordo com [22], utilizar lr com valor inicial de 10^{-4} é razoável para a tarefa de *crack recognition* para o conjunto de dados adotado. Nessa mesma linha, o estudo de [123] avaliou valores de taxa de aprendizado no processo de *crack detection* com arquiteturas conhecidas da literatura, como VGG-16, ResNet-50, ResNet-101. Os autores de [123] concluem que a taxa de aprendizado de $0{,}00001$ obteve os melhores resultados para *F-score* em uma base de dados com 527 imagens. Por outro lado, o trabalho de [131] selecionou $lr = 0{,}01$ ao alcançar $99{,}07\%$ de acurácia para *crack detection*. Os autores investigaram a acurácia na validação com seis valores de taxa de aprendizado (0,1; 0,05; 0,01; 0,005; 0,001; 0,0005; 0,0001) com uma arquitetura baseada na AlexNet em uma base de dados com 60 mil imagens.

Outro importante hiperparâmetro de treinamento analisado nos trabalhos é o otimizador [17, 120, 128]. As pesquisadores de [17] avaliaram quatro otimizadores (adam, adadelta, sgd e adamax) para a tarefa de *pavement distress detection*. Os resultados dos diferentes classificadores testados por [17], revelaram que o melhor desempenho foi alcançado por uma rede neural com otimizador adam, pré-treinada com a base ImageNet e arquitetura VGG-16 para um conjunto de dados com 1056 imagens.

A literatura também apresenta estudos sobre a seleção da arquitetura da rede neural [31, 150]. Os autores de [31] conduzem experimentos para investigar o desempenho de três tipos de redes neurais na detecção de defeitos em tubulações de esgoto (*sewer pipe defects*). Nesse sentido, concluem que redes neurais com mais camadas convolucionais contribuem para aumentar a acurácia na tarefa analisada. No entanto, o estudo ressalta que arquiteturas mais complexas podem influenciar no custo computacional. Assim, de acordo com [31], ao selecionar uma estrutura de rede neural, deve ser balanceado as métricas de acurácia e velocidade de detecção requerida na detecção dos objetos (patologias). O trabalho de [150], por sua vez, avalia arquiteturas para uma rede neural recorrente no processo de reconhecimento de ações de equipamentos

da construção. Os autores de [150] ressaltam que o número de camadas gerou grande impacto no desempenho de detecção no banco de dados adotado.

Vale ressaltar também a análise de hiperparâmetros de regularização [32, 38, 21]. Nesse aspecto, em [32] foram analisados os efeitos de taxas de *dropout* no treinamento de uma CNN para classificação de defeitos em tubulações de esgoto em um conjunto com 12 mil imagens. Os autores de [32] concluíram que valores de *dropout* menores que $0{,}5$ resultaram na diminuição da acurácia na validação para a base de dados adotada. Por outro lado, taxas maiores que $0{,}5$ levaram ao aumento de *overfitting*. Assim, o hiperparâmetro de *dropout* foi definido por [32] em 50%. Já o trabalho de [38] investiga os efeitos dos valores de *dropout* e regularização L2 na detecção de guarda-corpos (*guardrail*). Para isso, utilizam a técnica de *grid search*. Os autores concluíram que os melhores desempenhos foram alcançados ao adotar *dropout* de $0{,}2$ e L2 de $0{,}01$ em uma base de dados com 4000 imagens.

Ressalta-se que os resultados alcançados nesses trabalhos refletem a seleção de hiperparâmetros em problemas específicos. Assim, deve ser observado que, em geral, as conclusões desses estudos são restritas às bases de dados adotadas nesses trabalhos. Dessa maneira, esse fator reforça a importância da proposta de abordagens para recomendação de hiperparâmetros com metodologias criteriosas que possam ser replicadas em estudos de casos distintos.

3.3.3 Aplicação de *Data Augmentation*

Um dos principais desafios para aplicação de *deep learning* na área da construção civil é o possível problema de bases de dados pequenas e pouca variedade de imagens [31, 155, 136]. De fato, o treinamento de CNNs com poucos exemplos pode influenciar na capacidade de generalização do modelo neural [94, 3]. Nesse aspecto, uma estratégia adotada na Construção 4.0 e também em outras áreas é a geração de imagens artificiais para o aumento dos dados (*data augmentation*) [38, 26, 144].

A literatura apresenta diferentes estratégias de *data augmentation* aplicadas em conjuntos de dados da construção [110, 156, 151]. Alguns exemplos de transformações utilizadas para a geração dos novas imagens são: rotação [31, 38], *horizontal flip* [32, 19], *vertical flip* [110, 156], ajuste de cores (*color tuning*) [115, 154], borrão (*blur*) [32, 156], translação [140, 155], corte em largura (*width shift*) [38, 136], corte em altura (*heigth shift*) [38, 136], cisalhamento (*shear*) [140, 136] e *zoom* [154, 136]. A Tabela 3.4 resume as principais transformações de *data augmentation* selecionadas pelos estudos analisados neste trabalho.

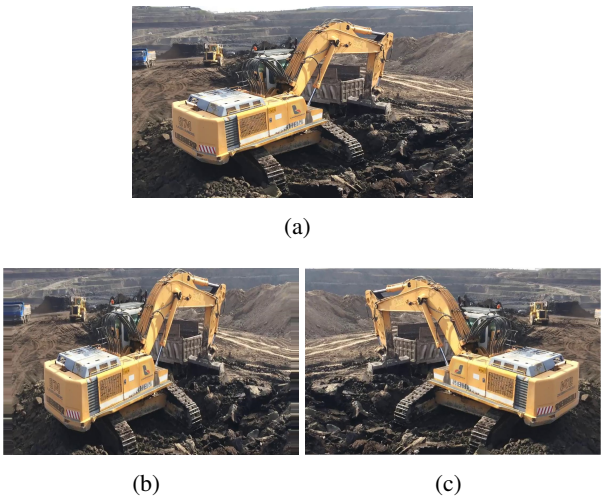
Tabela 3.4: Transformações de *data augmentation* utilizadas nos trabalhos analisados.

Artigo	*Rotation*	*Horizontal Flip*	*Vertical Flip*	*Color*	*Blur*	*Translation*	*Width Shift*	*Heigth Shift*	*Shear*	*Zoom*	Outros
[110]		✓	✓								
[29]	✓	✓									
[19]		✓									
[31]	✓	✓	✓	✓							✓
[114]		✓									✓
[151]						✓					
[38]	✓	✓					✓	✓	✓		✓
[39]		✓									
[32]		✓	✓	✓	✓						
[136]	✓	✓					✓	✓	✓	✓	✓
[115]	✓										
[137]				✓							
[154]				✓	✓					✓	✓
[140]	✓					✓			✓		✓
[26]		✓									
[156]	✓	✓	✓	✓	✓						✓
[144]	✓	✓	✓								✓
[121]	✓										✓
[155]	✓					✓					✓

A partir da Tabela 3.4 é possível observar que as principais estratégias adotadas na aplicação de *data augmentation* são *flip (horizontal ou vertical)* e rotação. A transformação de rotação foi utilizada em dez dos 18 trabalhos analisados na Tabela 3.4, ou seja, $55{,}6\%$. Além disso, em $66{,}7\%$ dos estudos foi aplicada os processamentos de *horizontal flip* ou *vertical flip* na geração das novas imagens. Destaca-se também a utilização do ajuste de cores (ex.: brilho e contraste) em cinco estudos avaliados. Por outro lado, ressalta-se que as transformações de recorte em largura (*width shift*), recorte em altura (*heigth shift*) e *zoom* foram apontadas em apenas dois trabalhos, cada uma. Também cabe ressaltar que a coluna "Outros" refere-se a outros métodos diversos aplicados, como reflexão, distorção e *pixel dropout*.

Também pode-se observar a viabilidade de aplicação de *data augmentation* nas áreas de detecção de patologias da construção [140] e reconhecimento de equipamentos de segurança [38]. Nesse sentido, os autores de [140] utilizaram estratégias de geração de imagens para detecção de fissuras em túneis com CNNs. Para isso, em [140] foram aplicados ângulos de rotação entre 0º e 20º, translação de 0 a 5%, cisalhamento de 0 a $0{,}2$ radianos e mudança de escala em até 5%. Em outra via, [38] utilizaram *data augmentation* no treinamento de uma rede neural para o reconhecimento de guarda-corpos. As seguintes transformações foram adotadas por [38]: rotação (-15º a 15º), *width shift* (-15% a 15%), *heigth shift* (-15% a 15%), *shear shift* (-20% a 20%), *channel shift* (-20% a 20%) e *horizontal flip*.

Outro importante aspecto é a utilização de *frameworks* livres para a geração das imagens artificiais. Nesse sentido, destaca-se a biblioteca Keras, disponível para as linguagens Python e R [94]. A biblioteca Keras possui a função $image\_data\_generator()$ que possibilita a adoção de diversas transformações de *data augmentation*. Por exemplo, [136] utilizaram Keras para treinamento de modelos de *deep learning* no processo de detecção de arranhões em painéis de vidros. Desse modo, os seguintes parâmetros de transformações foram definidos por [136] no Keras: *Rotation (60)*; *Width shift range (0.4)*; *Heigth shift range (0.4)*, *Shear range (0.4)*, *Zoom range (0.2)*; *Horizontal Flip (true)*, *Fill mode (Nearest)*. A Figura 3.3 apresenta exemplos de imagens de geradas por *data augmentation* adotando a biblioteca Keras. Para isso, foi adotado uma imagem disponível no *Alberta Construction Image Dataset* (ACID) [40].



(a) (b) (c)

Figura 3.3: Exemplos de imagens com *data augmentation* aplicadas no banco de dados ACID. (a) Imagem original (Fonte: [40]). (b) e (c) Imagens geradas usando a biblioteca Keras.

3.4 Aplicações de *Deep Learning* na Construção Civil

Nesta seção, são apresentados algumas aplicações de *deep learning* em Construção 4.0 nas seguintes áreas: (i) patologias e inspeções de construções; e (ii) segurança e reconhecimento de equipamentos nas construções.

3.4.1 Patologias e Inspeções de Construções

A Tabela 3.5 apresenta os trabalhos classificados na área de reconhecimento de patologias e inspeções das construções, divididos em seis grupos de aplicações: edificações, pavimentação, pontes, tubulações de esgoto, túneis e diversos.

Tabela 3.5: Estudos na linha de pesquisa de reconhecimento de patologias e inspeções de construções com *deep learning*.

Aplicação	**Trabalhos**
Edificações	[16], [112], [23], [18], [24], [136], [142], [22], [145]
Pavimentação	[123], [17], [25], [151], [134], [27], [115], [26], [141], [117], [157], [143], [148], [149], [155],
Pontes	[122], [29], [28], [113], [131], [30], [139] [21]
Tubulações de esgoto	[31], [32], [33]
Túneis	[140], [119], [121]
Diversas	[110], [19], [20], [125], [127], [128], [114], [129], [152], [137], [116], [156]

A partir da Tabela 3.5 pode-se observar que parte dos estudos dedicaram-se a análise de edificações. Por exemplo, os autores de [16] propõem uma abordagem de *deep learning* para a detecção de rachaduras (*crack detection*) em um edifício da Universidade de Manitoba (Canadá). Para isso, foi adotada uma base de dados com 40 mil imagens. Os resultados de [16] apontaram acurácia de até $97{,}95\%$ na etapa de validação. Em outra linha, o trabalho de [24] adota a arquitetura ResNet-101 no processo de classificação de defeitos em fachadas de edifícios residenciais de Singapura. Nesse aspecto, foram definidas sete classes: *crack*, *blistering*, *biological-growth*, spalling, *delamination*, *peeling* e *no-defects*. Os resultados de [24] apontaram acurácia máxima $84{,}36\%$ na etapa de validação. A Figura 3.4, por sua vez, apresenta amostras de imagens que pode ser adotadas para detecção ou segmentação de rachaduras em edificações, disponíveis na base de dados de [22].

(a) (b) (c) (d)

Figura 3.4: Exemplos de imagens para detecção/segmentação de rachaduras em edificações. Fonte: [22]).

Outra importante vertente é a inspeção de superfícies pavimentadas, como ruas ou rodovias. Destaca-se que essa aplicação (pavimentação) é a com mais trabalhos (15) na Tabela 3.5. Em um desses estudos, os autores de [148] propõem a arquitetura *CrackNet*, com o objetivo de automatizar a detecção de rachaduras em superfícies asfálticas. Nesse sentido, a *CrackNet* foi treinada com 1800 imagens de pavimentos. Os resultados na etapa de teste (200 imagens) apontaram precisão de $90{,}13\%$, *recall* de $87{,}63\%$ e *F-score* de $88{,}86\%$. Por outro lado, os autores

de [151] aplicam diferentes métodos de *deep learning* para o reconhecimento de patologias em estradas não-pavimentadas. Para isso, foram utilizadas arquiteturas da literatura no processo de segmentação de 715 imagens: VGG16, ResNet16, ResNet50 e Mobilenetv2. Os experimentos de [151] demonstraram que foi alcançado para a métrica de IoU valores maiores que 86% na segmentação de buracos (*pothole*) nas fotografias das rodovias.

A Tabela 3.5 mostra ainda a utilização de *deep learning* na análise de patologias em pontes. Nessa linha, os autores de [139] apresentam um método para reconhecimento de rachaduras em inspeções de pontes com drones. Para isso, os autores descrevem uma abordagem combinando CNNs e *Support Vector Machine*. A partir disso, foram realizados experimentos com uma base de dados 1600 imagens dividido em quatro patologias: *water seepage*, *concrete spalling*, *reinforcing bars*, e *cracks*. Os resultados do método proposto por [139] apontam acurácias superiores a 90%. Outro trabalho que também aplica veículos aéreos não-tripulados em conjunto com métodos de *deep learning* na inspeção de pontes é o estudo de [29]. Os autores de [29] propõem um conjunto de dados com 774 imagens para análise de patologias em pontes. A Figura 3.5 apresenta exemplos de imagens disponíveis na base de dados *Coco Bridge*.

(a) (b) (c)

Figura 3.5: Exemplos de imagens disponíveis na base de dados *Coco Bridge*. Fonte: [29].

Também vale ressaltar a realização de estudos para inspeção de tubulações de esgoto [31, 32, 33] e túneis [140, 119, 121]. Por fim, a Tabela 3.5 apresenta ainda uma seleção de trabalhos que utilizaram *deep learning* no reconhecimento de imagens em inspeções de mais de um tipo de construção.

3.4.2 Segurança e Reconhecimento de Equipamentos nas Construções

Uma das principais vertentes da aplicação de *deep learning* na Construção 4.0 é o campo de segurança e reconhecimento de equipamentos [34, 39, 124]. Neste estudo de revisão, 17 trabalhos foram identificados como pertencentes dessa linha de pesquisa. Assim, esses artigos foram divididos em três grupos:

- Detecção de equipamentos de proteção: detecção da utilização equipamentos de proteção individual (EPI) ou coletiva (EPC) no canteiro de obras, como capacetes, cintos e guarda-corpos.
- Reconhecimento de máquinas: detecção de máquinas da construção ou de condições seguras para a utilização desses dispositivos, como caminhões e escavadeiras.
- Identificação de ações de trabalhadores: reconhecimento de ações de sinalização ou inseguras/inadequadas praticadas por trabalhadores no canteiro de obras, como sinalização manual ou riscos de atropelamento.

A Tabela 3.6 apresenta os estudos do campo de segurança e reconhecimento de equipamentos nas construções de acordo com o respectivo grupo em análise.

Tabela 3.6: Estudos no campo de detecção de condições de segurança e reconhecimento de equipamentos nas construções com *deep learning*.

Área	**Trabalhos**	**Exemplos**
Detecção de EPIs e EPCs	[35], [37], [34], [38], [133], [36]	Capacetes e cintos
Reconhecimento de máquinas	[127], [39], [41], [40], [120], [150]	Caminhões
Identificação de ações de trabalhadores	[124], [130], [146], [118], [147]	Sinalizações manuais

A partir da Tabela 3.6 pode-se observar que parte dos estudos dedica-se a detecção de equipamentos de proteção com *deep learning*. Nessa linha, os autores de [34] aplicam Redes Neurais Convolucionais para detecção da utilização de cintos de proteção por trabalhadores. Para isso, foram utilizadas 770 imagens de trabalhadores em canteiros de obras. Os resultados apontam que o método Faster R-CNN alcançou até 99% de precisão e 95% de *recall* na detecção dos cintos de segurança nas obras [34]. Outro equipamento de proteção individual frequentemente abordado nessa área é o capacete [37, 36, 133]. Por exemplo, o estudo de [133] utilizou o algoritmo SSD-Mobilenet para detecção de capacetes de segurança em uma base de dados com 3261 imagens. Os autores adotaram *framework TensorFlow* e alcançaram 95% de precisão e 77% de *recall*. A literatura contempla ainda a adoção de *machine learning* no processo de reconhecimento de outros dispositivos de proteção, como guarda-corpos [38] e óculos de proteção [35].

Outra vertente apontada na Tabela 3.6 é o reconhecimento de máquinas na construção civil [39, 40]. Nessa linha, os autores de [39] desenvolveram um estudo para detecção de equipamentos da construção usando *Region-Based Fully Convolutional Network* (R-FCN) e transferência de aprendizado. Para isso, os autores de [39] utilizaram a arquitetura ResNet treinada com imagens da base ImageNet para classificar fotografias de tipos de caminhões da construção, como caminhão basculante (*dump truck*), escavadeira (*excavator*), carregadeira (*loader*), caminhão betoneira (*concrete mixer truck*) e rolo-compactador (*road roller*). Os resultados apontaram $96{,}3\%$ de mAP (*mean average precision*) na detecção das máquinas analisadas. Seguindo esse campo de pesquisa, o trabalho de [40] apresenta uma base de dados de imagens para detecção de máquinas da construção com *deep learning*. O *Alberta Construction Image Dataset* (ACID)[2] possui 10 mil fotografias de dez tipos de máquinas da construção. Nesse sentido, a Figura 3.6 apresenta exemplos de fotografias disponíveis na base ACID. Vale ressaltar também que, os autores de [40] comparam diferentes técnicas para detecção de objetos na ACID, sendo que os resultados para métrica mAP foram: $83{,}0\%$ (Inception-SSD), $87{,}8\%$ (YOLO-v3), $88{,}8\%$ (R-FCN-ResNet101), e $89{,}2\%$ (Faster-RCNN-ResNet101).

A Tabela 3.6 revela ainda estudos para identificação de ações de trabalhadores no canteiro de obras. Por exemplo, os autores de [124] propõem um modelo híbrido de *deep learning* para detectar comportamentos inseguros usando CNNs e *Long Short-Term Memory*. Para isso, os autores utilizam uma rede Inception-v3 pré-treinada com imagens base de dados ImageNet. A acurácia do sistema proposto foi de 97% na detecção de ações seguras e 92% no reconhecimento de atividades inseguras. O trabalho de [118], por sua vez, propõem um sistema de visão computacional para reconhecimento sinais manuais na construção. Nesse sentido, foram utilizados dois métodos de *machine learning* para classificação das imagens: ResNeXt-101 e Res3D + ConvLSTM+MobileNet. Assim, os autores alcançaram até $93{,}3\%$ de acurácia com a arquitetura ResNeXt-101 e $84{,}8\%$ com a técnica Res3D + ConvLSTM+MobileNet.

[2] https://www.acidb.ca/dataset

(a) (b)

(c) (d)

Figura 3.6: Exemplos de imagens disponíveis no *Alberta Construction Image Dataset* (Fonte: [40]).

3.5 Considerações Finais

Na sequência, é apresentada uma breve discussão e também perspectivas futuras quanto à temática da aplicação de aprendizado profundo em aplicações de visão computacional da construção civil.

3.5.1 Discussão

O objetivo deste capítulo foi desenvolver uma revisão sistemática da aplicação de *deep learning* em sistemas de visão computacional da construção civil. Para isso, foram selecionados cinco periódicos da área de Construção 4.0. Além disso, foram investigados sete pontos principais: área do trabalho, técnicas de análise de imagens adotadas, métodos de *deep learning*, seleção de hiperparâmetros, *data augmentation*, principais resultados e perspectivas futuras.

A metodologia proposta buscou investigar dois grupos de trabalhos: mais citados e mais recentes de cada periódico. Assim, no total, foram avaliados 76 artigos publicados entre 2017 e 2021. Os resultados preliminares apontaram que os países com mais publicações nesta revisão sistemática foram China (25), EUA (19) e Canadá (9), considerando a instituição de pesquisa do primeiro autor do trabalho. Além disso, nessa mesma linha, a Ásia é o continente com mais publicações incluídas (44 artigos). Dessa forma, indicando uma carência do desenvolvimento de pesquisas no Brasil e América Latina na área de aprendizado profundo aplicado à construção civil.

Outra análise realizada foi a de métodos de visão computacional e técnicas de *deep learning*. Destaca-se que em $65{,}7\%$ dos trabalhos foram realizadas experimentos para detecção de objetos, como: equipamentos de proteção individual, máquinas ou patologias. Nesse sentido, é importante ressaltar a adoção de importantes técnicas de *machine learning* para detecção, como Faster R-CNN, SSD e YoLO [31, 41, 33]. Também cabe ressaltar a utilização de relevantes arquiteturas CNN para a classificação de imagens da construção, como VGG-16 e ResNet [17, 121, 24].

A revisão sistemática também investigou dois aspectos relacionados ao treinamento de modelos de *deep learning*: (i) seleção dos hiperparâmetros [148, 22, 123] e (ii) adoção de *data augmentation* [38, 26, 144]. Ressalta-se que apenas $18{,}4\%$ das publicações avaliadas apresentaram alguma análise para a definição de hiperparâmetros. Nesse sentido, os principais

hiperparâmetros avaliados foram: taxa de aprendizado, otimizador, arquitetura neural e taxa de *dropout*. A metodologia sistemática revelou ainda que somente 25% das publicações utilizaram alguma técnica de *data augmentation*. Nesse conjunto de trabalhos, foram adotadas diferentes combinações de transformações para a geração artificiais, sendo que as mais citadas foram: rotação e *flip*.

Na sequência foram investigadas as aplicações de *deep learning* de acordo com duas grandes áreas: (i) patologias e inspeções de construções e (ii) segurança e reconhecimento de equipamentos. Quando a temática envolve patologias, destaca-se um conjunto de estudos direcionados para a tarefa de detecção de rachaduras (*Crack Detection*). Além disso, existem estudos em distintas aplicações, como: edificações [16, 112], pavimentação [123, 17], pontes [122, 29], tubulações [31, 32] e túneis [140, 119]. Por outro lado, também são vários os campos de estudos de *deep learning* na área de segurança [35, 37, 34] ou reconhecimento de equipamentos [127, 39, 41]. Por exemplo, destaca-se a detecção de equipamentos de proteção (ex.: capacetes e guarda-corpos) e o reconhecimento de máquinas da construção (ex.: caminhões e escavadeiras).

3.5.2 Perspectivas Futuras

As perspectivas futuras quanto aos possíveis estudos da área de *deep learning* na análise de imagens da Construção 4.0 foram elencadas a partir da análise dos trabalhos mais recentes presentes nesta revisão. Nesse sentido, destaca-se os seguintes direcionamentos apontados:

1. Aprimorar os métodos de *deep learning* [29, 142, 155].
2. Melhorar a qualidade das bases de dados [134, 118, 121].
3. Investigar a generalidade das técnicas em outras aplicações [23, 135].
4. Otimizar a capacidade de processamento [134, 126].

Destaca-se inicialmente a importância da constante evolução das técnicas e metodologias para aplicação de *deep learning*. Nesse aspecto, o trabalho de [155] enfatiza a necessidade de melhorar as arquiteturas para produzir maior acurácia. Por outro lado, os autores de [142] ressaltam a relevância em aumentar a aplicabilidade e precisão do método proposto para implementações práticas. Vale ressaltar que, em apenas 14 estudos analisados nesta revisão sistemática foi verificada a utilização de alguma análise para a definição de hiperparâmetros. Além disso, pesquisas distintas demonstram que os hiperparâmetros e modelos de *machine learning* podem influenciar diretamente nos resultados das métricas de desempenho. Desse modo, uma possível vertente de estudo nessa área é o desenvolvimento de novos métodos para o ajuste de hiperparâmetros e seleção de arquiteturas neurais específicas para aplicações na Construção 4.0.

Outra perspectiva indicada nos estudos recentes desta revisão sistemática é o aperfeiçoamento dos conjuntos adotados para as etapas de treinamento. Por exemplo, os autores de [121] destacam a necessidade da melhoria da qualidade das imagens coletadas, tendo em vista vários aspectos que envolvem o ambiente da construção, como poeira e luminosidade inadequada. O trabalho de [134], por sua vez, apontam como um dos indicativos para pesquisas futuras, o aumento do conjunto de dados. Nesse sentido, destaca-se a importância do continuidade dos avanços da aplicação de *data augmentation* para o treinamento de modelos da construção civil. Os modelos de *data augmentation* possibilitam a geração de imagens artificiais e aumentar a diversidade do conjunto de dados, adotando transformações diversas (ex.: rotação, *flip* e *zoom*) na criação de novos dados. Nessa linha, outro caminho para a aplicação de *data augmentation* é a adoção das redes adversárias generativas (*generative adversarial networks*) [69], ainda pouco exploradas na Construção 4.0.

Também vale destacar a importância de investigar a viabilidade dos métodos em outras aplicações, diferente das inicialmente propostas. Por exemplo, os autores de [23] ressaltam como possibilidade de trabalhos futuros, implementar as melhores arquiteturas analisadas para *crack detection* em outros tipos de superfícies. Nesse aspecto, cabe comentar também sobre a adoção de transferência de aprendizado (*transfer learning*), campo em crescente expansão na área de *machine learning* [20, 3]. Técnicas de *transfer learning* podem ser adotadas, por exemplo, para minimizar o esforço computacional com novos treinamentos voltados para detecção de patologias semelhantes. Nessa mesma vertente, inclui a otimização do processamento dos modelos de *deep learning*. Conforme apontado por [134], uma saída é a adoção de GPUs (*Graphics Processing Unit*) de altos desempenhos. No entanto, também é válido a abordagem em conjunto da busca por modelos de *deep learning* mais simples que garantam a mesma bons níveis de eficiência e eficácia, principalmente visando aplicações de computação móvel [126].

CAPÍTULO 4

Materiais e Estudos de Casos

Neste capítulo, são apresentados os materiais e estudos de casos propostos para esta tese. Inicialmente são descritos os principais recursos de *software* e *hardware* adotados. Em seguida, são apresentadas informações sobre os tipos de hiperparâmetros analisados e arquiteturas de Redes Neurais Convolucionais utilizadas. Posteriormente, são descritas informações dos quatro estudos de casos de classificação de imagens da construção civil abordados.

4.1 Materiais de *Software* e *Hardware*

Os experimentos foram conduzidos adotando a linguagem R (versão 4.0.3) [158] no *software* RStudio. A linguagem R em conjunto com RStudio reúnem importantes aspectos para o desenvolvimento desta pesquisa, como ambiente para visualização gráfica, funções estatísticas, bibliotecas para *machine learning* e análise de dados. Desse modo, a Tabela 4.1 apresenta as principais funções da linguagem R utilizadas nos experimentos.

Tabela 4.1: Principais funções da linguagem R utilizadas nos experimentos.

Etapa Experimental	**Biblioteca**	**Objetivo**	**Funções no R**
Deep Learning	Keras	Arquitetura DenseNet121	application_densenet121()
	Keras	Arquitetura MobileNet	application_mobilenet()
	Keras	Arquitetura VGG16	application_vgg16()
	Keras	Adicionar camadas	keras_model_sequential()
	Keras	Aplicar *data augmentation*	image_data_generator()
	Keras	Leitura dos dados nos diretórios	flow_images_from_directory()
	Keras	Treinamento e Validação	compile(); fit_generator()
	Keras	Teste - Acurácia	evaluate_generator()
	Keras	Teste - Predição de classes	predict_classes()
Recomendação de Hiperparâmetros	–	Modelos de Análise de Variância	aov(); anova()
	–	Normalidade dos resíduos	ks.test()
	–	Homogeneidade das variâncias	bartlett.test()
	ScottKnott	Método de Scott-Knott	SK()

A Tabela 4.1 divide os métodos adotados de acordo a etapa experimental: (i) aplicação de *deep learning* na classificação de imagens e (ii) recomendação de hiperparâmetros usando métodos estatísticos. Na etapa de experimentos com *deep learning*, ressalta-se a utilização da biblioteca Keras (versão 2.3.0.0) [94]. O ambiente do Keras reúne um conjunto de métodos que

possibilitam o desenvolvimento de modelos de Redes Neurais Convolucionais e Redes Neurais Recorrentes, a partir da plataforma TensorFlow (versão 2.2.0). Além disso, essa biblioteca é gratuita e disponível para a utilização nas linguagens Python e R. Ressalta-se também a possibilidade da execução das operações em CPU ou GPU. Nesse sentido, conforme Tabela 4.1 a biblioteca Keras contempla funções que permitem a execução de experimentos com *deep learning*, como utilização de arquiteturas da literatura (ex.: DenseNet121, MobileNet, VGG16), *data augmentation*, treinamento, validação e teste.

A Tabela 4.1 apresenta também as principais funções adotadas nesta tese para recomendação de hiperparâmetros em linguagem R. Nesse aspecto, ressalta-se os métodos estatísticos para ajuste de modelos de ANOVA e também análise de medidas de adequação (normalidade dos resíduos e homogeneidade). Também vale destacar a adoção da biblioteca ScottKnott (versão 1.2-7) [81]. A biblioteca ScottKnott implementa o algoritmo de agrupamento hierárquico para o método de Scott-Knott [80], usado quando o modelo de ANOVA indica que existe diferença estatística entre os tratamentos (hiperparâmetros).

Além disso, os experimentos foram realizados com um processador Intel Core i7-8565 e 8GB de memória RAM. O *hardware* conta também com uma placa GPU NVIDIA GeForce MX110. A Tabela 4.2 resume as principais configurações da máquina utilizada nos experimentos.

Tabela 4.2: Principais configurações da máquina utilizada nos experimentos.

Item	**Configurações**
Modelo	Lenovo Ideapad S145 - 81S90003BR
Sistema operacional	Windows 10 Home
Processador	Intel Core i7-8565U, 4 núcleos, frequência de 1,8 GHz até 4,6 GHz e 8 MB de Cache
Placa de vídeo	NVIDIA GeForce MX110, 2 GB de memória de vídeo dedicada (GDDR5), 256 Cores CUDA e frequência de 978 MHz
Memória RAM	8GB, DDR4 e frequência de 2400 MHz
Disco rígido (HD)	1 TB e 5400 RPM

4.2 Hiperparâmetros

Nesta tese, são propostos métodos para recomendação e análise de três tipos de hiperparâmetros: otimizador, taxa de aprendizado e *data augmentation*.

4.2.1 Otimizador e Taxa de Aprendizado

O processo de treinamento de uma rede neural artificial contempla o ajuste de pesos com o objetivo de minimizar o valor de erro produzido na saída. Para isso, em tarefas de aprendizado profundo são adotados algoritmos de otimização estocástica. Nesta tese, seis otimizadores da literatura foram selecionados para análise: adadelta [96], adagrad [97], adam [98], adamax [98] e sgd [94]. Esses métodos foram adotados a partir da biblioteca Keras [94]:

- adadelta: *optimizer_adadelta()*.
- adagrad: *optimizer_adagrad()*.
- adam: *optimizer_adam()*.
- adamax: *optimizer_adamax()*.
- sgd: *optimizer_sgd()*.

Outro hiperparâmetro abordado nesta tese foi a taxa de aprendizado, em inglês, *learning rate* (lr). A taxa de aprendizado é responsável por controlar a velocidade do processo de otimização

de pesos da rede neural artificial. Assim, a definição conjunta do otimizador com a taxa de aprendizado tem papel relevante no treinamento de sistemas de *deep learning*. Nesse sentido, parte dos métodos para ajuste de hiperparâmetros propostos nesta tese, visam a recomendação de combinações de algoritmos de otimização e valores para taxa de aprendizado.

4.2.2 *Data Augmentation*

O pré-processamento dos dados é uma etapa importante do campo de *machine learning*. Isso porque, esta etapa pode diminuir a complexidade para o aprendizado e melhorar os resultados de acurácia [3]. Nessa linha, uma técnica que pode ser utilizada é o *data augmentation* [94, 69, 71, 3].

Data augmentation é uma abordagem aplicada principalmente para o aprendizado em *small datasets*. Nesse sentido, métodos de *data augmentation* geram mais dados para treinamento, a partir das imagens existentes [94]. O objetivo é aumentar a capacidade de generalização do modelo CNN e evitar *overfitting* [94, 3]. Para isso, as imagens artificiais são criadas a partir de transformações aleatórias nos dados originais [94]. *Zoom*, rotação e *flip* são alguns exemplos de transformações possíveis para a geração das imagens aumentadas [3].

Nesta tese, os métodos de *data augmentation* foram aplicados a partir da biblioteca Keras no *software R* [94]. Para isso, foi utilizado um método para geração de imagens artificiais: *image_data_generator()* [94]. A função *image_data_generator()* gera lotes de dados com novas imagens modificadas a partir das originais. Nesse aspecto, foram utilizadas as seguintes transformações aleatórias para o aumento dos dados de treinamento[1]:

- **Rotação** (*Rotation Range - R*): a rotação é um movimento circular em torno de um ponto fixo. Nesse sentido, um número inteiro define o intervalo de graus para rotações aleatórias. As imagens de processamento terão rotações aleatórias em uma faixa predefinida de graus de acordo com a entrada de dados.

- **Flip no eixo horizontal** (*Horizontal flip - H*): se esta entrada for "TRUE", as imagens serão espelhadas aleatoriamente na direção horizontal (esquerda-direita).

- **Flip no eixo vertical** (*Vertical flip - V*): se esta entrada for "TRUE", as imagens serão espelhadas aleatoriamente na direção vertical (*up-bottom*).

- **Cisalhamento** (*Shear range - S*): distorce a imagem ao longo de um eixo para criar ou retificar os ângulos de percepção. Existem duas transformações de cisalhamento, *X-Shear* que muda os valores das coordenadas X e *Y-Shear* que muda os valores das coordenadas Y.

- **Deslocamento em largura** (*Width shift range - W*): desloca a imagem aleatoriamente para a esquerda ou direita (deslocamentos horizontais). Se o valor for flutuante e menor ou igual a um (≤ 1), a porcentagem da largura total será considerada como intervalo. Por exemplo, em uma imagem com largura de 100 pixels (px) e se *width_shift_range = 1.0*, a imagem será deslocada aleatoriamente entre -100% a 100% ou $-100px$ a $100px$. Os valores positivos deslocarão a imagem para o lado direito e os valores negativos deslocarão a imagem para o lado esquerdo.

- **Deslocamento em altura** (*Height shift range - He*): desloca a imagem aleatoriamente para cima ou para baixo (deslocamento vertical). Se o valor for flutuante e menor ou igual a um (≤ 1), a porcentagem da altura total será considerada como intervalo. Por exemplo, em uma imagem com altura de 100 pixels e se *height_shift_range = 1.0*, a imagem será

[1] https://keras.io/api/preprocessing/image/

deslocada aleatoriamente entre -100% a 100% ou $-100px$ a $100px$. Os valores positivos deslocarão a imagem para o lado superior e os valores negativos deslocarão a imagem para o lado inferior.

- **Zoom** (*Zoom range - Z*): realiza um aumento aleatório na imagem, adicionando novos valores de pixels. Essa transformação pode ser especificada com a porcentagem do *zoom* como flutuação única ou um intervalo como uma matriz. Por exemplo, se *zoom_range = 0.4*, o intervalo será [0,6; 1,4] entre 60% (*zoom in*) e 140% (*zoom out*).

A Figura 4.1 apresenta exemplos de imagens geradas por *data augmentation* com a biblioteca Keras.

Figura 4.1: Exemplos de imagens geradas com data augmentation (biblioteca Keras): (a) imagem original; (b) - (d) rotação ($R = 40$); (e) flip horizontal ($H = TRUE$); (f) flip vertical ($V = TRUE$); (g) e (h) deslocamento em altura ($He = 0.2$); (l) cisalhamento ($S = 0.2$); (j) - (l) deslocamento em largura ($W = 0.2$); (m) - (p) *zoom* ($Z = 0.2$).

O número de imagens geradas artificialmente depende de definições do treinamento: tamanho do lote (*batch_size*), número de passos por época (*steps_per_epoch*) e quantidade de

épocas (*epoch*). Por exemplo, se esses parâmetros forem definidos como: *batch_size* = 32; *steps_per_epoch* = 100; e *epoch* = 5, então, para cada simulação serão geradas aleatoriamente em torno de 16 mil novas imagens para treinamento ($32 \times 100 \times 5$).

Assim, nesta tese, também foram propostos métodos para recomendação de hiperparâmetros de *data augmentation*, ou seja, a seleção e ajuste das transformações de processamento de imagens adotadas na geração de dados sintéticos para treinamento de modelos de *deep learning*

4.3 Arquiteturas de *Deep Learning*

A metodologia proposta adotou três arquiteturas do estado da arte de CNNs: DenseNet121 [159], MobileNet [160] e VGG16 [161]. Essas estruturas são frequentemente discutidas na literatura de *deep learning* [162, 163, 11, 6, 8, 164]. Além disso, esses modelos são disponibilizados no ambiente Keras, conforme descrito na sequência.

A DenseNet121 (*Dense Convolutional Network*) é uma arquitetura proposta por [159]. Essa estrutura é caracterizada por conter blocos, em que, cada camada está conectada à todas as outras camadas (blocos densos). Além disso, a DenseNet121 possui 8.062.504 parâmetros. A Figura 4.2 mostra uma representação da DenseNet121. Para a utilização dessa arquitetura, foi adotado o método *application_densenet121()* da biblioteca Keras com *include_top* = *FALSE* (sem a última camada totalmente conectada).

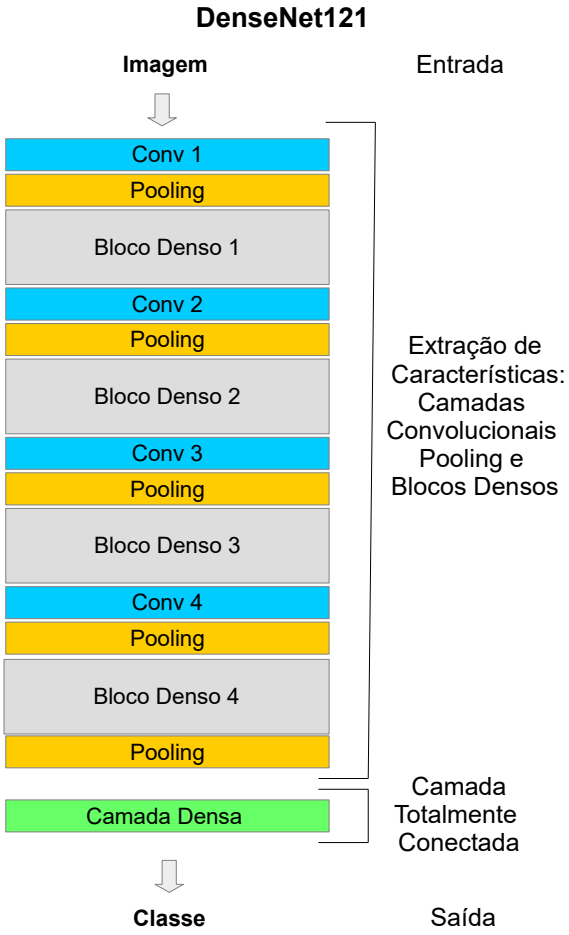


Figura 4.2: Arquitetura Densenet121. Fonte: baseado em [159].

A MobileNet é uma arquitetura CNN proposta para aplicações móveis e embarcadas de visão computacional por pesquisadores do Google [160]. A estrutura usa convoluções fatorizadas, denominadas de *deptwise separable convolutions* (Conv dw). Além disso, possui 28 camadas e 4.253.864 parâmetros. A Figura 4.2 mostra uma representação da MobileNet. Foi utilizado o método *application_mobilenet()* da biblioteca Keras com *include_top = FALSE* (sem a camada densa).

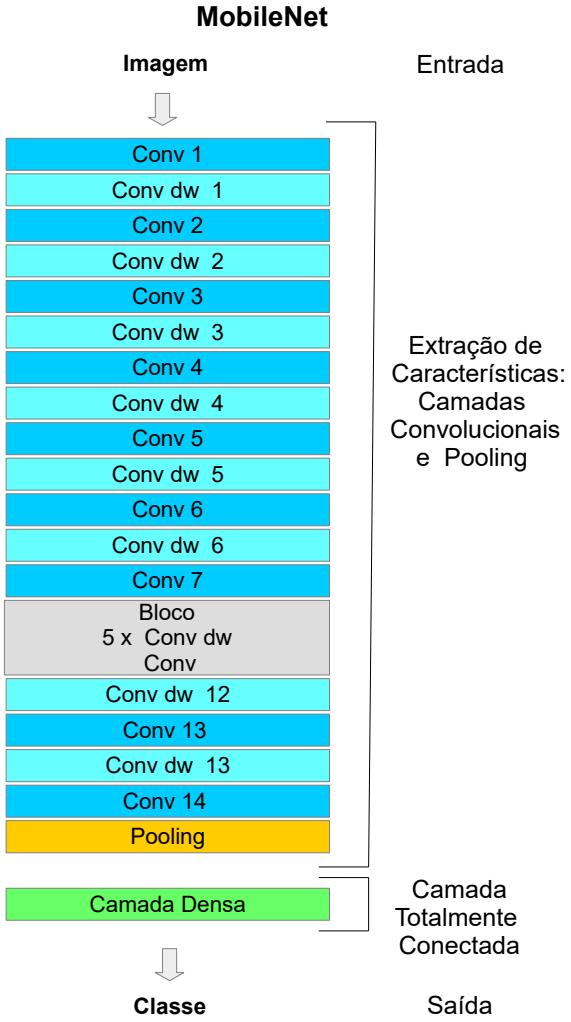


Figura 4.3: Arquitetura MobileNet. Fonte: baseado em [160].

A arquitetura VGG16 é uma estrutura CNN proposta por [161] no *Visual Geometry Group* da Universidade de Oxford. Refere-se a configuração D da arquitetura geral VGGNet com 138 milhões de parâmetros para treinamento. Além disso, a VGG16 consiste de 16 camadas, sendo 13 camadas convolucionais, 3 camadas totalmente conectadas e operações de *pooling*. A Figura 4.4 mostra uma representação da VGG16. Ressalta-se que, para a adoção dessa estrutura, foi adotado o método *application_vgg16()* com *include_top = FALSE*, ou seja, retirando as últimas camadas totalmente conectadas.

A Tabela 4.3 resume algumas características das arquiteturas convolucionais adotadas: número de parâmetros e quantidade de memória necessária para armazenamento da estrutura final em Mega Byte (MB). O número de parâmetros da estrutura padrão refere-se a quantidade de pesos na arquitetura incluindo as camadas densas e dimensões de entradas originais ($224 \times 224 \times 3$). Por outro lado, para as estruturas finais utilizadas foi padronizado como entrada da rede neural

Figura 4.4: Arquitetura VGG16. Fonte: baseado em [161].

as dimensões $50 \times 50 \times 3$: *input_shape = c(50, 50, 3))*. Também ressalta-se que todas as três arquiteturas foram configuradas com as últimas duas camadas como totalmente conectadas, sendo que a última possui o neurônio classificador binário com função de ativação *sigmoid* [94]. Para adicionar essas camadas foi utilizado a função *keras_model_sequential()* da biblioteca Keras.

Tabela 4.3: Características das arquiteturas convolucionais adotadas: número de parâmetros e quantidade de memória necessária para armazenamento da estrutura final.

Arquitetura	**Parâmetros: estrutura padrão**	**Parâmetros: estrutura final**	**Armazenamento** (MB)
DenseNet121	8.062.504	7.562.817	29,5
MobileNet	4.253.864	3.754.177	14,4
VGG16	138.000.000	14.977.857	57,1

4.4 Estudos de Casos

Nesta seção, são apresentados os quatro estudos de casos na área classificação de imagens da construção civil adotados neste trabalho:

1. Reconhecimento de vegetação em fachadas.
2. Detecção de patologias em calhas.
3. Classificação de máquinas na construção.
4. Classificação de rachaduras em edificações.

Ressalta-se que para os quatro estudos de casos foram adotadas bases de dados binárias e balanceadas, ou seja, mesmo número de imagens entre as duas classes. Essa definição tem como objetivo retirar o possível impacto do desbalanceamento nos problemas de classificações investigados, e assim, conduzir a investigação buscando avaliar os efeitos dos hiperparâmetros nas métricas de desempenho.

Nas próximas seções são abordados aspectos das bases de dados e importância desses problemas na literatura.

4.4.1 Estudo de Caso 1: Reconhecimento de Vegetação em Fachadas

Estudos sobre a análise de manifestações patológicas biológicas em edificações tem papel relevante na literatura [165, 166, 167, 168]. Isso porque, o crescimento de vegetações em construções é um problema que pode levar ao enfraquecimento das estruturas [165]. Além disso, o reconhecimento de vida vegetal em edificações pode ser relevante na identificação da degradação e abandono de edifícios históricos [165, 166, 169, 170]. Nesse aspecto, o trabalho de [166] investiga a presença de vegetação em edifícios históricos do Rio de Janeiro. Por outro lado, os trabalhos de [165] e [167] analisam a degradação de marquises (como por vegetação parasitária) em Recife (PE) e Campina Grande (PB), respectivamente. Nessa mesma linha, o estudo de [169] identifica que o crescimento de vegetação é uma das principais causas de degradação de mesquitas Ottomanas.

Nesse aspecto, o primeiro estudo de caso refere-se ao reconhecimento da patologia de vegetação em fachadas de edificações. Para isso, foram aplicados modelos de *deep learning* no aprendizado de duas classes:

1. com vegetação na fachada da edificação;
2. sem vegetação na fachada da edificação.

Nesta aplicação, foram utilizados dados do *The Zurich Urban Micro Aerial Vehicle Dataset* (ZUDataset) [171] para treinamento e validação de Redes Neurais Convolucionais. O ZUDataset é um conjunto de dados que contém 81.169 imagens gravadas por um vant das ruas urbanas de Zurique (Suíça). Um micro veículo aéreo (tipo quadrorrotor) registrou os dados em janeiro de 2015 com uma câmera GoPro Hero 4. As imagens foram registradas em alta resolução (1920 $\times$ 1080 $\times$ 24 bits). Além disso, as informações foram coletadas em baixas altitudes (5 a 15 m acima do solo) ao longo de percurso de 2 km de extensão.

O banco de dados também possui informações de GPS, medição inercial (IMU) e imagens do *Google Street View* ao nível do solo. Conforme informações no site do projeto[2], esse banco de dados é liberado sem restrições e pode ser usado para fins de pesquisa, avaliação e

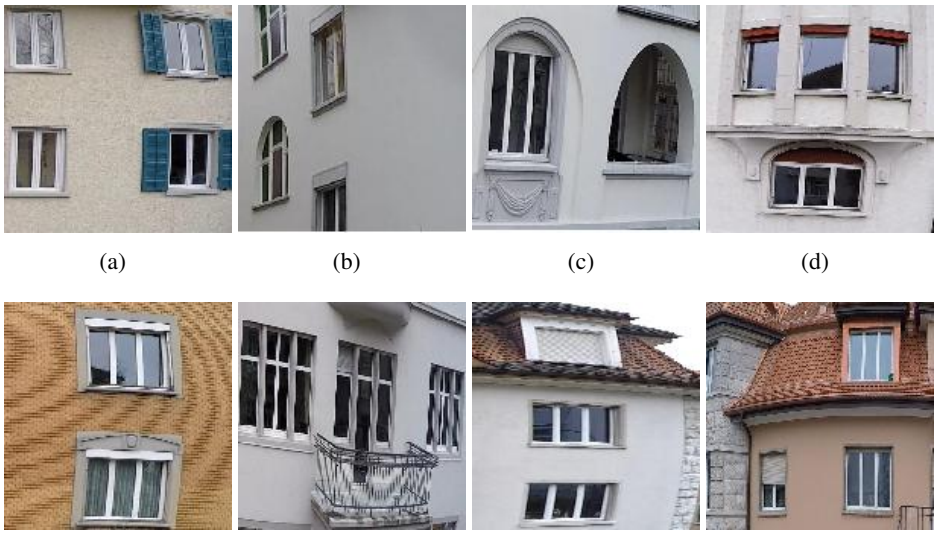
(a) (b)

Figura 4.5: Exemplos de imagens do ZUDataset. Fonte: *The Zurich Urban Micro Aerial Vehicle Dataset* (ZUDataset) [171].

comerciais. A Figura 4.5 apresenta exemplos de imagens capturadas pelo vant e disponíveis no ZUDataset.

De acordo com [171], o ZUDataset é ideal para aplicação em tarefas de navegação autônoma, como odometria visual, localização e mapeamento. No entanto, para a utilização nesta tese, um subconjunto das imagens do ZUDataset foi selecionado e adaptado para a divisão das amostras em duas classes: (i) com vegetação na fachada da edificação; (ii) sem vegetação na fachada da edificação.

No primeiro momento, foram analisadas as fotografias que continham vida vegetal nas paredes e realizado o procedimento de *zoom*, gerando novas imagens para um *dataset* adaptado para a aplicação nesta tese. Assim, foram obtidas 150 novas imagens para a classe 1 (com vegetação). Em seguida, foram selecionadas as fotografias das edificações que não continham vegetação nas fechadas das construções. Também foi aplicado o procedimento de *zoom* gerando 150 novas imagens para a classe 2 (sem vegetação). As Figuras 4.6 e 4.7 apresentam exemplos de imagens para as classes 0 (sem vegetação) e 1 (com vegetação), respectivamente.



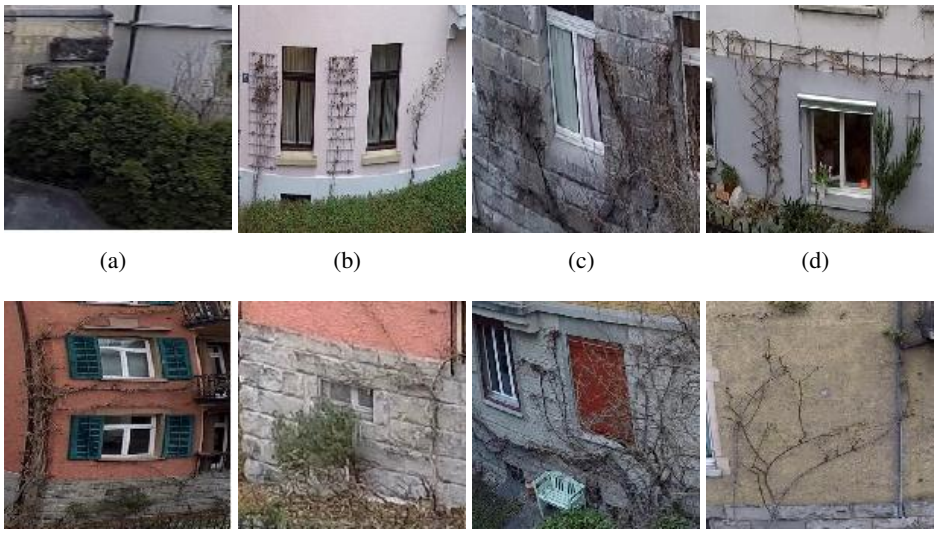
(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.6: Exemplos de imagens para a classe 0 - sem vegetação na fachada. Fonte: Adaptado de *The Zurich Urban Micro Aerial Vehicle Dataset* [171].

Na etapa de teste, foram selecionadas 90 novas imagens de domínio público da internet. As

[2]http://rpg.ifi.uzh.ch/zurichmavdataset.html



(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.7: Exemplos de imagens para a classe 1 - com vegetação na fachada. Fonte: Adaptado de *The Zurich Urban Micro Aerial Vehicle Dataset* [171].

fotografias de acesso livre e gratuitas foram obtidas no site Pixabay[3], a partir de pesquisas de termos relacionados ao estudo de caso, como: "*vegetation building*", "vegetação construção", "edifício abandonado", "*building*" e "casa". Essas imagens continham, em geral, edificações abandonadas com vegetação na fachada (45 imagens de teste para a classe 1). Além disso, outras 45 imagens com construções sem vegetação para a compor a classe 0 no teste.

Assim, a base de dados (390 imagens) para o primeiro estudo de caso foi estruturada na seguinte forma:

- Treinamento (250 imagens): 125 imagens na classe 0 (sem vegetação) e 125 imagens na classe 1 (com vegetação).
- Validação (50 imagens): 25 imagens na classe 0 (sem vegetação) e 25 imagens na classe 1 (com vegetação).
- Teste (90 imagens): 45 imagens na classe 0 (sem vegetação) e 45 imagens na classe 1 (com vegetação).

4.4.2 Estudo de Caso 2: Detecção de Patologias em Calhas

A proposta de estudos para o desenvolvimento de análises eficientes de estruturas de telhados é frequente na literatura [172, 173, 174, 175]. Conforme apontando por [174], inspeções de telhados possuem limitações por acessibilidade e, assim, apresenta-se como tarefa desafiadora. Além disso, a carência de análise e manutenção dessas estruturas pode levar ao aparecimento de diversas patologias [172, 174, 175].

Nesse aspecto, os autores de [174] apresentam uma lista com 28 patologias para a análise de telhados, sendo integridade e limpeza da calha (*gutter integrity and cleanliness*) como um desses pontos. O trabalho de [174] destaca que um dos desafios nas inspeções com vant é a dificuldade para identificar itens relacionados à ferrugem de calhas (*gutters rust*). Além disso, o acúmulo de sujeira em calhas é uma patologia recorrente, devido a falta de regularidade de limpeza de telhados.

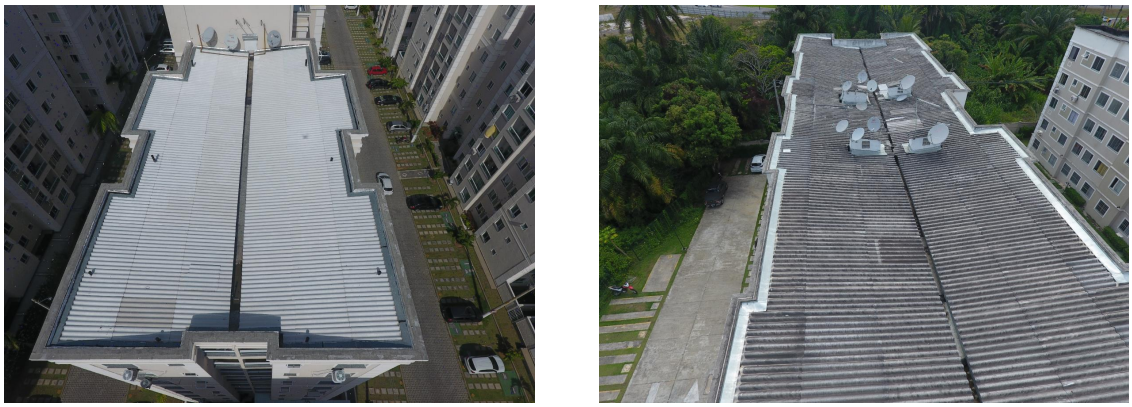
[3] www.pixabay.com

Assim, o segundo estudo caso refere-se à classificação de defeitos em calhas de telhados. Para isso, foram aplicados modelos de *deep learning* no aprendizado de duas classes:

1. calha não-íntegra ou suja.

2. calha íntegra e limpa.

Foi utilizado um subconjunto de imagens do banco de dados apresentado e descrito por [174, 175] e disponibilizado pelo Grupo de Pesquisa e Extensão em Gestão e Tecnologia das Construções (GETEC) da Escola Politécnica da UFBA. Essas imagens foram capturadas a partir de inspeções em telhados com um veículo aéreo não-tripulado (vant). Para isso, foi utilizado o dispositivo DJI Phanton 4[4] com câmera acoplada de 20 megapixels. As imagens foram coletadas em voos mantendo 5 metros acima do prédio analisado. Assim, foram registradas 1661 fotografias referentes a 61 telhados de 6 condomínios residenciais, localizados na região metropolitana da cidade de Salvador (BA). A Figura 4.8 apresenta exemplos de imagens desse banco de dados.



(a) (b)

Figura 4.8: Exemplos de imagens do *dataset* adotado no segundo estudo de caso. Fonte: Base de dados do GETEC/UFBA.

Foram realizados procedimentos de pré-processamento nas imagens da base de dados. Inicialmente, as imagens foram selecionadas e divididas em duas classes: (0) telhados com calhas íntegras e limpas e (1) telhados com calhas não-íntegras ou sujas. Em seguida, foi aplicado o procedimento de *zoom* nas regiões das calhas, gerando um novo conjunto de imagens, conforme exemplos da Figura 4.9. A base de dados gerada nesta etapa totaliza 220 imagens, separadas para as fases de treinamento, validação e teste:

- Treinamento (160 imagens): 80 imagens na classe 0 (calhas íntegras e limpas) e 80 imagens na classe 1 (calhas não-íntegras ou sujas).

- Validação (30 imagens): 15 imagens da classe 0 (calhas íntegras e limpas) e 15 imagens da classe 1 (calhas não-íntegras ou sujas).

- Teste (30 imagens): 15 imagens da classe 0 (calhas íntegras e limpas) e 15 imagens da classe 1 (calhas não-íntegras ou sujas).

[4]https://www.dji.com/br/phantom-4

(a) Exemplo de imagem da classe de telhado com calha íntegra e limpa.

(b) Exemplo de imagem da classe de telhado com calha não-íntegra ou suja.

Figura 4.9: Exemplos de imagens das classes (0) e (1) do segundo estudo de caso. Fonte: Adaptado da base de dados do GETEC/UFBA.

4.4.3 Estudo de Caso 3: Classificação de Máquinas na Construção

A literatura também apresenta estudos sobre a aplicação de aprendizado profundo no reconhecimento de máquinas da construção [39, 40, 41]. Certamente um canteiro de obras pode apresentar diferentes tipos de máquinas, como caminhões e tratores [40]. Dessa forma, no processo de automatização da Construção 4.0 torna-se importante o desenvolvimento de técnicas de alto desempenho para detectar e classificar veículos em imagens da construção. Além disso, o estudo de [39] destaca a relevância no reconhecimento desses objetos para gerar informações adequadas sobre a construção, como *status* de segurança dos colaboradores, progresso e qualidade das obras.

Nesse sentido, no terceiro estudo de caso foram utilizados modelos de Redes Neurais Convolucionais no aprendizado de imagens de dois tipos de veículos:

1. Máquina do tipo escavadora (*excavator*).

2. Máquina do tipo *mixer* de concreto.

Nesta aplicação, foram utilizadas dados do *Alberta Construction Image Dataset* (ACID) [41]. O ACID (formato básico) contém 2850 imagens de máquinas da construção. De acordo com a página do dataset[5], a base de dados contém imagens extraídas do YouTube e também capturadas pela equipe do projeto. Além disso, o ACID deve ser utilizado apenas para fins de pesquisa e não é permitido a adoção para atividades comerciais.

As imagens disponibilizadas no formato básico do ACID possuem em geral três tipos de máquinas da construção: escavadoras (*excavator*), *mixer* de concreto e *dump truck*. Dessa forma, a princípio foi analisada quais desses tipos de máquinas possuíam mais exemplares de fotografias no *dataset*. Além disso, para possibilitar o processo de classificação, foram utilizadas apenas as fotografias com apenas um tipo veículo em cada imagem. Assim, foram selecionadas 800 imagens e divididas em duas classes: máquina escavadora e caminhão *mixer*. As Figuras 4.10 e 4.11 apresentam exemplos de imagens para as classes 0 (máquina escavadora) e 1 caminhão *mixer*), respectivamente.

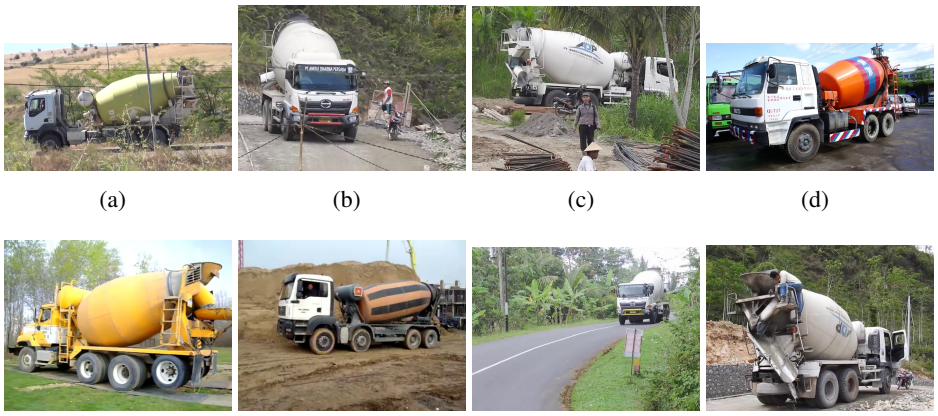
Assim, a base de dados (800 imagens) para o terceiro estudo de caso foi estruturada na seguinte forma:

[5] https://www.acidb.ca/dataset

(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.10: Exemplos de imagens para a classe 0 - máquina do tipo escavadora (*excavator*). Fonte: ACID *dataset* [41].



(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.11: Exemplos de imagens para a classe 1 - máquina do tipo caminhão *mixer* para concreto. Fonte: ACID *dataset* [41].

- Treinamento (640 imagens): 320 imagens na classe 0 (máquina escavadora) e 320 imagens na classe 1 (caminhão *mixer*).
- Validação (80 imagens): 40 imagens na classe 0 (máquina escavadora) e 40 imagens na classe 1 (caminhão *mixer*).
- Teste (80 imagens): 40 imagens na classe 0 (máquina escavadora) e 40 imagens na classe 1 (caminhão *mixer*).

4.4.4 Estudo de Caso 4: Classificação de Rachaduras em Edificações

O surgimento de rachaduras é um importante sinal de degradação de uma construção civil [176, 177, 178]. Essa patologia pode causar significantes danos e afetar a vida útil de edificações, rodovias e pontes [179, 180, 181]. Nesse aspecto, rachaduras podem aparecer por diferentes fatores, como sobrecarga da estrutura, fadiga, acidentes naturais e processos químicos [179, 180, 176]. Assim, a detecção de rachaduras é essencial para assegurar a boa manutenção de estruturas e prevenir possíveis desastres [181, 182]. Para isso, a detecção automática des-

sas patologias é uma vertente da literatura, por exemplo, a partir de técnicas de processamento digital de imagens e aprendizado de máquina [177, 183, 181].

Redes Neurais Convolucionais são frequentemente adotadas na tarefa de classificação de rachaduras em imagens. As aplicações são diversas, como na análise de superfícies de rodovias [115], classificação de defeitos em fachadas [24], inspeção de pontes e detecção de fissuras em paredes de concreto de edificações [18].

Nesse sentido, o quarto estudo de caso abordado nesta teste é o de classificação de rachaduras em edificações. Para isso, foi adotado a base de dados "*Concrete Crack Images for Classification*" [184]. Esse banco de dados de imagens está disponível em domínio publico na internet link[6] e já foi utilizado em outros estudos da literatura [185, 18, 186, 187].

Esse conjunto de dados contém 40.000 imagens (227 $\times$ 227 pixels com canais RGB) referentes a fotografias de paredes e pisos de edifícios de concreto da Universidade Técnica do Oriente Médio (*Middle East Technical University*) na Turquia. Além disso, as imagens são divididas em duas classes (20.000 imagens por classe): (i) negativa (sem rachadura) e (ii) positiva (com rachadura).

As Figuras 4.12 e 4.13 mostram exemplos de imagens das classes negativa e positiva, respectivamente.

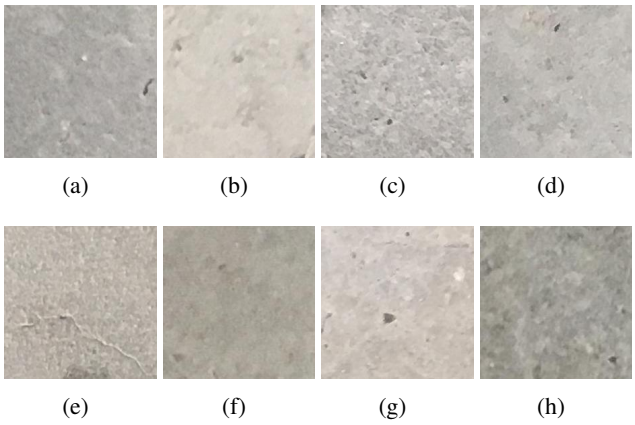


Figura 4.12: Exemplo de imagens da classe negativa (0) - sem rachadura. Fonte: [184].

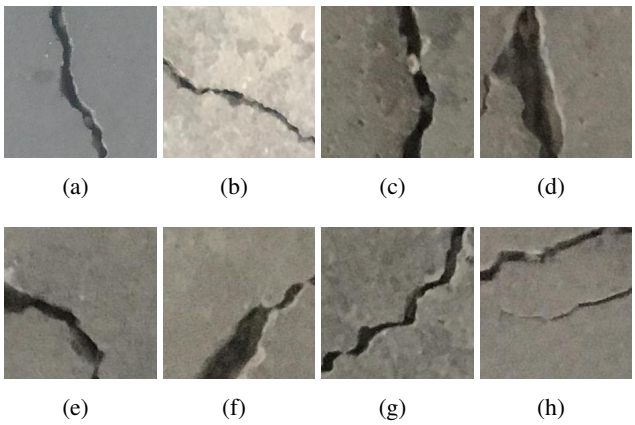


Figura 4.13: Exemplos de imagens da classe positiva (1) - com rachadura. Fonte: [184].

Nesta tese, essa base de dados foi estruturada em três conjuntos para experimentos com *deep*

[6]https://data.mendeley.com/datasets/5y9wdsg2zt/2

learning: treinamento (80%), validação (10%) e teste (10%). Assim, o número total de imagens (40,000) foi dividido da seguinte maneira:

- Treinamento (32.000 imagens): 16.000 imagens na classe negativa e 16.000 imagens na classe positiva.
- Validação (4.000 imagens): 2.000 imagens na classe negativa e 2.000 imagens na classe positiva.
- Teste (4.000 imagens): 2.000 imagens na classe negativa e 2.000 imagens na classe positiva.

CAPÍTULO 5

Método HyperTuningSK para Recomendação de Hiperparâmetros

Neste capítulo, o método HyperTuningSK para recomendação de hiperparâmetros é proposto em etapas: análise preliminar, análise de variância, *rankings* de recomendação, algoritmo e testes. Além disso, também é proposta um procedimento para análise de hiperparâmetros de *data augmentation*. Posteriormente, são analisados os resultados da aplicação da abordagem proposta em três estudos de casos da área de classificação de imagens da construção civil: reconhecimento de vegetação em fachadas, detecção de patologias em calhas e classificação de máquinas da construção.

5.1 Método HyperTuningSK

Nesta seção, é proposto o método HyperTuningSK para recomendação de hiperparâmetros. Em seguida, essa metodologia é descrita em etapas (ver Figura 5.1):

1. Experimentos com arquiteturas CNN para classificação de imagens da construção civil.
2. Análise preliminar dos resultados.
3. HyperTuningSK (fase 1): ajuste do modelo estatístico de ANOVA;
4. HyperTuningSK (fase 2): *ranking* de recomendação de hiperparâmetros usando o método de Scott-Knott;
5. Testes (experimentos e análise).

Conforme Figura 5.1, a primeira etapa consiste no treinamento e validação das arquiteturas *deep learning* adotadas (VGG16 e DenseNet121) em estudos de casos da classificação de imagens da construção civil. Nesse sentido, nesta fase são experimentadas diferentes combinações de hiperparâmetros. Na sequência, é realizada uma análise preliminar dos resultados para realizar um pré-processamento dos resultados dos experimentos. Em seguida, é aplicado o método proposto para recomendação de hiperparâmetros (HyperTuningSK) em duas fases: ajuste do modelo estatístico de Análise de Variância (ANOVA) e *ranking* de hiperparâmetros usando Scott-Knott *Clustering* [80, 81]. Após a seleção dos hiperparâmetros, é realizada uma nova rodada de experimentos para teste. Essas etapas são explanadas em mais detalhes nas próximas seções.

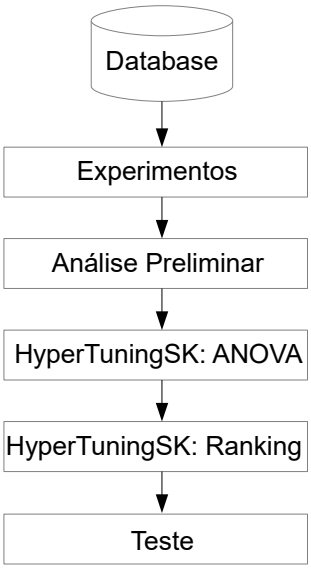


Figura 5.1: Etapas para aplicação do método HyperTuningSK para recomendação de hiperparâmetros.

5.1.1 Experimentos para Recomendação de Hiperparâmetros

Nesta seção, é proposto o planejamento do experimentos para recomendação de hiperparâmetros. Nesse sentido, foram conduzidas simulações para treinamento e validação de duas arquiteturas *deep learning*: DenseNet121 e VGG16.

Além disso, foram utilizados três estudos de casos de classificação de imagens da construção civil para análise:

- Reconhecimento de vegetação em fachadas.
- Detecção de patologias em calhas.
- Classificação de máquinas da construção.

Por outro lado, os hiperparâmetros selecionados para o ajuste foram otimizador e taxa de aprendizado, baseando-se na importância do ajuste desses elementos na literatura. Assim, os valores para ajuste são apresentados a seguir:

- otimizadores: adam, adamax, adagrad e sgd.
- taxa de aprendizado: [0,001; 0,005; 0,010; 0,015; 0,020; 0,025].

Inicialmente, foram definidas um total de 12 combinações de hiperparâmetros (2 otimizadores $\times$ 6 taxas de aprendizado): adagrad001, adagrad005, adagrad010, adagrad015, adagrad020, adagrad025, sgd001, sgd005, sgd010, sgd015, sgd020 e sgd025. A notação utilizada obdece a seguinte forma: adagrad001 (otimizador adagrad + $lr = 0{,}001$). Além disso, mais duas combinações de hiperparâmetros foram definidas sem a variação de taxa de aprendizado: adam001 e adamax001. Dessa forma, totalizando 14 tratamentos para análise.

Desse modo, para cada estudo de caso e arquitetura foram realizados experimentos com os 14 tratamentos de taxa de aprendizado e otimizador abordados. Para cada uma dessas combinações foram treinados 5 modelos CNN (repetições) com 5 épocas por estudo de caso/arquitetura. O número de épocas foi fixado a partir de análises prévias dos experimentos, sendo definido

o menor valor onde já era possível visualizar graficamente diferenças entre as definições dos diferentes valores de hiperparâmetros.

A métrica de desempenho observada foi a acurácia nas etapas de validação e teste. A Equação (5.1) apresenta o cálculo da acurácia (Acc):

$$Acc = \frac{VP + VN}{VP + VN + FP + FN}, \tag{5.1}$$

em que, VP representa as classificações corretas na classe positiva; VN são as classificações corretas na classe negativa; FP são a classificações incorretas na classe positiva e; FN são as classificações incorretas na classe negativa.

5.1.2 Análise Preliminar

Os experimentos realizados com diversas combinações de hiperparâmetros e arquiteturas nos três estudos de casos geram dados que devem ser pré-processados antes da aplicação do método de HyperTuningSK. Por isso, nesta etapa da metodologia é realizada uma análise prévia dos resultados.

Inicialmente, é calculada a média dos resultados de acurácia na validação para cada repetição de combinação de hiperparâmetro, conforme Equação (5.2):

$$M_{acc} = \frac{Acc_1 + Acc_2 + \ldots + Acc_N}{N}, \tag{5.2}$$

em que, N é o número de épocas para cada combinação de hiperparâmetro e Acc_N é a acurácia na validação (Equação (5.1)).

Na sequência, o objetivo é identificar possíveis combinações de hiperparâmetros que levam o sistema de classificação à resultados de *underfitting* (média acurácia de 50% na validação). Assim, quando para pelo menos uma das repetições do tratamento, se $M_{acc} = 50\%$, então essa combinação de hiperparâmetro é retirada das próximas etapas. Dessa forma, nas etapas posteriores de aplicação do método HyperTuningSK são utilizados apenas hiperparâmetros adequados nesta análise preliminar.

5.1.3 Análise de Variância

Nesta seção, são descritas as formulações estatísticas da primeira etapa do método de recomendação de hiperparâmetros proposto (HyperTuningSK). Sendo assim, o modelo linear estatístico de Análise de Variância (ANOVA) dos experimentos é apresentado na Equação (5.3) [79]:

$$y_{ij} = \mu_i + \epsilon_{ij} \tag{5.3}$$

em que, i representa o tratamento (combinação de hiperparâmetro) ($i = 1, 2, \ldots, 14$); j é o número da repetição do treinamento ($j = 1, 2, \ldots, 5$); y_{ij} é a observação, ou seja, a média de acurácia de um tratamento i na repetição j (ou seja, M_{acc}, conforme Equação (5.2)); μ_i é a média i-ésimo tratamento; ϵ_{ij} é o componente de erro (diferença entre uma observação e a respectiva média). No modelo apresentado, a combinação (otimizador e taxa de aprendizado) representa o fator do experimento. Além disso, cada uma das 14 combinações é um tratamento analisado.

O objetivo da análise de variância é verificar se existe significativa diferença entre as médias de acurácia ($\mu_1, \mu_2, \ldots, \mu_a$) dos a tratamentos analisados. Assim, são observadas duas hipóteses [79]:

$$\begin{cases} H_0: & \mu_1 = \mu_2 = \ldots = \mu_a, \\ H_1: & \mu_i \neq \mu_j \quad \text{para pelo menos um par } (i, j). \end{cases}$$

A hipótese inicial (H_0) é aceita se a médias dos tratamentos são estatisticamente iguais, ou seja, o valor $p > 0{,}05$ (adotando um nível de significância de 5%). No entanto, se H_0 é rejeitada e a hipótese alternativa é aceita (H_1), então pelo menos uma das médias das combinações de hiperparâmetros é diferente das demais ($p < 0{,}05$).

Também são realizadas análises em duas medidas de adequação do modelo estatístico de ANOVA. O primeiro critério observado é a normalidade dos resíduos. Um resíduo (ϵ_{ij}) é a diferença entre uma observação e o valor ajustado pelo modelo [79]. Assim, nesta metodologia proposta, é utilizado o teste de Kolmogorov-Smirov (KS) [188] e as seguintes hipóteses:

$$\begin{cases} H_{ks0}: & \text{resíduos normais,} \\ H_{ks1}: & \text{resíduos não-normais,} \end{cases}$$

em que, se a hipótese inicial (H_{ks0}) é aceita, os resíduos são normais e a medida de adequação é satisfeita ($p_{ks} \geq 0{,}05$). Por outro lado, se H_{ks0} é rejeitada e a hipótese alternativa é aceita (H_{ks1}), ou seja, os resíduos não são normais ($p_{ks} < 0{,}05$).

A segunda métrica de avaliação do modelo ANOVA é a homogeneidade das variâncias. Nesta tese, foi adotado o teste de Bartlett [189], observando as seguintes hipóteses:

$$\begin{cases} H_{bt0}: & \text{variâncias homogêneas,} \\ H_{bt1}: & \text{variâncias não-homegêneas,} \end{cases}$$

sendo que, aceitar hipótese inicial (H_{bt0}) assegura a medida de adequação de variâncias homogêneas ($p_{bt} \geq 0{,}05$). No entanto, e H_{bt0} é rejeitada e a hipótese alternativa é aceita (H_{bt1}), assume-se que as variâncias são não-homogêneas ($p_{bt} < 0{,}05$).

Dessa forma, caso a Análise de Variância confirme que existe diferença significativa entre os hiperparâmetros e as medidas de adequação forem satisfeitas, é iniciado o processo de *ranking* de recomendação de hiperparâmetros (*hyperparameter tuning ranking*), conforme descrito na próxima seção.

5.1.4 Rankings de Recomendação

Nesta tese, a metodologia propõe a adoção de *rankings* de recomendação de hiperparâmetros gerados a partir do Scott-Knott *Clustering Algorithm* [80] e a biblioteca ScottKnott no R [81]. Basicamente, o método de Scott-Knott (SK) é um algoritmo que gera grupos dos tratamentos, quando existe diferença estatística significativa entre as médias na Análise de Variância, ou seja, H_1 é aceita.

O objetivo do método de Scott-Knott é particionar os tratamentos em grupos de modo a maximizar a soma de quadrados (B_0). Inicia-se o processo ordenando os tratamentos pelas médias. Em seguida, são avaliadas as partições possíveis, definindo sempre dois grupos com k_1 e k_2 tratamentos em cada um. Na sequência, são calculadas as somas totais de ambos os grupos (T_1 e T_2) e a soma de quadrados (B_0), conforme Equações de (5.4) a (5.6):

$$T_1 = \sum_{i=1}^{k_1} \mu_i, \tag{5.4}$$

$$T_2 = \sum_{i=k_1+1}^{k_1+k_2} \mu_i, \tag{5.5}$$

$$B_0 = \frac{T_1^2}{k_1} + \frac{T_2^2}{k_2} - \frac{(T_1 + T_2)^2}{k_1 + k_2}, \tag{5.6}$$

em que, k = $(k_1 + k_2)$ é o total de tratamentos e μ_i é a média do tratamento i. Assim, para a partição que maximize a soma de quadrados (B_0), são analisadas duas hipóteses que avaliam se os tratamentos nos grupos são homogêneos ou heterogêneos, adotando as estatísticas de máxima verossimilhança e qui-quadrado [81]:

$$\begin{cases} H_{sk0}: & \text{grupo homogêneo (médias iguais)} \\ H_{sk1}: & \text{grupo heterogêneo (médias diferentes).} \end{cases}$$

Caso a hipótese inicial (H_{sk0}) seja rejeitada e a hipótese alternativa (H_{sk1}) aceita, então os tratamentos naquela partição são separados, efetivando a formação de dois novos grupos. Desse modo, o processo de particionamento é repetido (cálculos das Eqs. 5.4 a 5.6 e testes de hipóteses) para os novos grupos formados até a hipótese inicial ser aceita (grupos homogêneos), ou seja, não é mais necessária a divisão dos tratamentos.

Assim, são criados grupos de hiperparâmetros a partir dos resultados médios de acurácia na classificação de imagens da construção civil. A Tabela 5.1 apresenta um exemplo de *ranking* de recomendação de hiperparâmetros.

Tabela 5.1: Exemplo de *ranking* de recomendação de hiperparâmetros com o método HyperTuningSK.

Grupo	**Hiperparâmetro**	**Média (%)**
A	Hiperparâmetro-A1	72,0
A	Hiperparâmetro-A2	71,8
B	Hiperparâmetro-B1	63,6
B	Hiperparâmetro-B2	60,4
C	Hiperparâmetro-C1	56,0
D	Hiperparâmetro-D1	48,0

No exemplo da Tabela 5.1 foram gerados quatro grupos de hiperparâmetros: A, B, C e D. As combinações de hiperparâmetros no Grupo A (*top position*) são as indicadas pela metodologia proposta, por possuírem os maiores valores de acurácia: Hiperparâmetro-A1 e Hiperparâmetro-A2. Os outros tratamentos não são recomendados para a utilização na fase de teste, pois foram distribuídos nos demais grupos (B, C e D).

Mais detalhes sobre formulação matemática do método Scott-Knott, podem ser encontrados em [80, 81]. Nesta tese, o Scott-Knott *Clustering* é adotado como procedimento computacional para geração de *rankings* de hiperparâmetros, conforme descrito na próxima seção.

5.1.5 Algoritmo HyperTuningSK

Nesta seção, é proposto um algoritmo computacional em linguagem R para recomendação de hiperparâmetros, denominado HyperTuningSK. O Algoritmo 1 demonstra a sequência de passos para geração de *rankings* de hiperparâmetros usando ANOVA e o Scott-Knott *Clustering Algorithm* [81].

Nas linhas de 1 a 6 são realizadas definições iniciais, como a declaração da biblioteca ScottKnott [81] e leitura de dados de acurácia. Já nas linhas 6 à 22 é realizado o processo de Análise de Variância. As funções $aov()$ e $anova()$ são adotados para ajuste do modelo. Já os métodos $ks.test()$ e $bartlett.test()$ realizam os testes de normalidade dos resíduos (teste de Kolmogorov Smirnov) [188] e homogeneidade das variâncias (teste de Bartlett) [189].

Assim, na linha 12 inicia-se a verificação das medidas de adequação (normalidade dos resíduos e homogeneidade das variâncias). Se confirmada a adequação é analisado a estatística

```
1  library(ScottKnott)
2  dados <- read.delim("data.txt")
3  attach(dados)
4  h<-c("adagrad001", "adagrad005", "adagrad010", "adagrad015", "sgd001", "sgd005",
     "sgd010", "sgd015","sgd020")
5  sizeh<-length(h)
6  # Stage 1: Analysis of Variance
7  model<-aov(y ~ factor(hyp))
8  paov <- anova(model)
9  pks<-ks.test(resid(model),'pnorm', mean(resid(model)),sd(resid(model)))
10 pbt<-bartlett.test(y ~ factor(hyp))
11 H1 <- 0
12 if ((pks$p.value < 0.05) || (pbt$p.value< 0.05)){
13       print("Not adequate model")
14 }else{
15       print("Adequate model")
16       if(paov$`Pr(>F)`[1]< 0.05){
17            print("Anova: H1 confirmed")
18            H1 <- 1
19       }else{
20            print("Anova: H0 confirmed")
21       }
22 }
23 # Stage 2: Hyperparameter Tuning Ranking
24 if (H1==1){
25       print("Hyperparameter Tuning of CNN")
26       print("Residue Normality (pks):")
27       print(pks$p.value)
28       print("Homoscedasticity (pbt):")
29       print(pbt$p.value)
30       print("Anova, p-value:")
31       print(paov$`Pr(> F)`[1])
32       sk1 <- SK(model)
33       print("HP Ranking")
34       sk1s <- summary(sk1)
35       gr <- sk1$groups[1]
36       group_h <- which(sk1$groups==gr)
37       size_group <- length(group_h)
38       ind <- sk1$ord[1: size_group]
39       hsk <- h[ind]
40       print("Recommended hyperparameters:")
41       print(hsk)
42 }else{
43       print("Hyperparameters have no statistical difference or not adequate model")
44 }
```

Algoritmo 1: HyperTuningSK - Método para recomendação de hiperparâmetros usando ANOVA e Scott-Knott em linguagem R.

(valor p) do teste ANOVA. Nesse caso, se $p < 0{,}05$, a hipótese alternativa é confirmada (linha 16), ou seja, existe diferença estatística significativa entre as médias de acurácia dos tratamentos

(hiperparâmetros).

Nas linhas de 24 a 41 é realizado o *ranking* de recomendação de hiperparâmetros (*hyperparameter tuning ranking*) ($if H1 == 1$). São apresentados os resultados de ANOVA (linhas 25 a 31), Na sequência, é utilizado o comando $SK()$ da biblioteca ScottKnott para gerar o *ranking* e agrupamentos. Em seguida, nas linhas 35 à 39 são separados os hiperparâmetros do grupo A para recomendação. Assim, os hiperparâmetros recomendados são apresentados nas linhas 40 e 41.

Finalmente, se a hipótese inicial (H_0) não for rejeitada ou as medidas de adequação não forem satisfeitas, então é apresentada uma mensagem na linha 43.

5.1.6 Etapa de Teste

Nesta etapa, foram propostos experimentos para analisar o desempenho dos hiperparâmetros recomendados pelo algoritmo HyperTuningSK. Para isso, foram treinados novos modelos CNN com os otimizadores e taxas de aprendizado selecionadas. Assim, nesta fase foram utilizadas no processo de treinamento: 3 repetições com 20 épocas para cada conjunto de hiperparâmetros (otimizador + lr) recomendados.

Além disso, para fins comparativos também foram treinados modelos CNN com hiperparâmetros adotados na literatura [13, 190, 131]. Essas combinações de hiperparâmetros são de trabalhos que aplicaram *deep learning* na área de processamento de imagens da construção civil:

- adam e $lr = 0{,}001$ [13].
- sgd e $lr = 0{,}001$ [190].
- sgd e $lr = 0{,}010$ [131].

5.2 Metodologia para Análise de *Data Augmentation*

Nesta tese, também foi proposta a aplicação do método HyperTuningSK na análise de hiperparâmetros de *data augmentation*. O objetivo foi investigar como as transformações adotadas na geração de imagens artificiais podem influenciar no desempenho dos modelos de *deep learning* na classificação de imagens da construção civil.

5.2.1 Planejamento dos Experimentos

Inicialmente, foram definidas sete transformações básicas (hiperparâmetros) de *data augmentation* na biblioteca Keras para ajuste, sendo que, cada uma com dois níveis de tratamentos (0 - sem transformação e 1 - com transformação):

- *Rotation Range* (R): 0 ou 40.
- *Horizontal Flip* (H): FALSE ou TRUE.
- *Vertical Flip* (V): FALSE ou TRUE.
- *Height Shift Range* (He): 0 ou 0,2.
- *Shear Range* (S): 0 ou 0,2.
- *Width Shift Range* (W): 0 ou 0,2.

- *Zoom Range* (Z): 0 ou 0,2

Além disso, de modo a explorar as potencialidades do arranjo das transformações nas imagens para *data augmentation*, também foram avaliadas as combinações de hiperparâmetros. Sendo assim, um total de 128 (2^7) combinações de hiperparâmetros de *data augmentation* foram analisadas para os três estudos de casos abordados nesta tese. Para cada configuração foram treinados 5 modelos de CNN (repetições) em 10 épocas com 100 passos (*steps por epoch*) adotando a arquitetura MobileNet [160].

Assim, o objetivo é encontrar as combinações de hiperparâmetros de *data augmentation* que tendem a maximizar a acurácia no processo de classificação com *deep learning*. As Equações (5.7) e (5.8) apresentam o modelo de otimização para esse problema, sendo função objetivo e restrição, respectivamente:

$$\max Acc = \sum_{i=1}^{N} c_i x_i \tag{5.7}$$

sujeito à:

$$x_i \in \{0,1\} \tag{5.8}$$

sendo que, as variáveis binárias do modelo (x_1, x_2, x_3, x_4, x_5, x_6 e x_7) representam a decisão de utilizar cada uma das transformações básicas na geração das imagens (0 - sem transformação e 1 - com transformação):

$$\begin{cases} x_1: & \text{Rotation Range (R)} \\ x_2: & \text{Horizontal Flip (H)} \\ x_3: & \text{Vertical Flip (V)} \\ x_4: & \text{Height Shift Range (He)} \\ x_5: & \text{Shear Range (S)} \\ x_6: & \text{Width Shift Range (W)} \\ x_7: & \text{Zoom Range (Z)} \end{cases}$$

Além disso, na Equação (5.7), o termo c_i representa a contribuição no valor de acurácia de acordo com a transformação i utilizada. A Equação (5.8) garante que as variáveis de decisão são binárias.

As Tabelas 5.2 a 5.5, por sua vez, apresentam as 128 combinações de hiperparâmetros de *data augmentation* analisadas. A primeira configuração ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$) é a combinação de referência, pois não utiliza nenhuma transformação nas imagens ao gerar os dados artificiais. Além disso, também pode-se observar nessas tabelas, as sete transformações básicas (apenas um tipo de processamento de imagem adotado), sendo na ordem:

- *Zoom Range (Z)*: # 2 ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 1$).
- *Width Shift Range (W)*: # 3 ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$).
- *Shear Range (S)*: # 5 ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 0$ e $x_7 = 0$).
- *Height Shift Range (He)*: # 9 ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$).
- *Vertical Flip (V)*: # 17 ($x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$).
- *Horizontal Flip (H)*: # 33 ($x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$).
- *Rotation Range (R)*: # 65 ($x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$).

Tabela 5.2: Combinações de *data augmentation* analisadas (Parte I).

#	x_1 (R)	x_2 (H)	x_3 (V)	x_4 (He)	x_5 (S)	x_6 (W)	x_7 (Z)
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	0	0	0	1	0
4	0	0	0	0	0	1	1
5	0	0	0	0	1	0	0
6	0	0	0	0	1	0	1
7	0	0	0	0	1	1	0
8	0	0	0	0	1	1	1
9	0	0	0	1	0	0	0
10	0	0	0	1	0	0	1
11	0	0	0	1	0	1	0
12	0	0	0	1	0	1	1
13	0	0	0	1	1	0	0
14	0	0	0	1	1	0	1
15	0	0	0	1	1	1	0
16	0	0	0	1	1	1	1
17	0	0	1	0	0	0	0
18	0	0	1	0	0	0	1
19	0	0	1	0	0	1	0
20	0	0	1	0	0	1	1
21	0	0	1	0	1	0	0
22	0	0	1	0	1	0	1
23	0	0	1	0	1	1	0
24	0	0	1	0	1	1	1
25	0	0	1	1	0	0	0
26	0	0	1	1	0	0	1
27	0	0	1	1	0	1	0
28	0	0	1	1	0	1	1
29	0	0	1	1	1	0	0
30	0	0	1	1	1	0	1
31	0	0	1	1	1	1	0
32	0	0	1	1	1	1	1

As demais configurações presentes nas Tabelas 5.2 a 5.5 são referentes a adoção combinada de técnicas de processamento de imagens para *data augmentation*. Para exemplificar, o quarto arranjo ($x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 1$) combina as transformações de *Zoom Range* e *Width Shift Range*. Por outro lado, a configuração número 128 utiliza todas as sete transformações básicas na geração de novas artificiais.

5.2.2 Aplicação do Método HyperTuningSK

O método HyperTuningSK foi aplicado na análise dos resultados do experimentos com as combinações de hiperparâmetros de *data augmentation*. Para isso, foi adicionada uma fase de pré-processamento automático nos dados, seguindo as seguintes etapas:

1. Leitura dos resultados das 128 combinações de hiperparâmetros.

2. Definição das combinações básicas: 1, 2, 3, 5, 9, 17, 33 e 65, conforme Tabelas 5.2 a 5.4.

Tabela 5.3: Combinações de *data augmentation* analisadas (Parte II).

#	x_1 (R)	x_2 (H)	x_3 (V)	x_4 (He)	x_5 (S)	x_6 (W)	x_7 (Z)
33	0	1	0	0	0	0	0
34	0	1	0	0	0	0	1
35	0	1	0	0	0	1	0
36	0	1	0	0	0	1	1
37	0	1	0	0	1	0	0
38	0	1	0	0	1	0	1
39	0	1	0	0	1	1	0
40	0	1	0	0	1	1	1
41	0	1	0	1	0	0	0
42	0	1	0	1	0	0	1
43	0	1	0	1	0	1	0
44	0	1	0	1	0	1	1
45	0	1	0	1	1	0	0
46	0	1	0	1	1	0	1
47	0	1	0	1	1	1	0
48	0	1	0	1	1	1	1
49	0	1	1	0	0	0	0
50	0	1	1	0	0	0	1
51	0	1	1	0	0	1	0
52	0	1	1	0	0	1	1
53	0	1	1	0	1	0	0
54	0	1	1	0	1	0	1
55	0	1	1	0	1	1	0
56	0	1	1	0	1	1	1
57	0	1	1	1	0	0	0
58	0	1	1	1	0	0	1
59	0	1	1	1	0	1	0
60	0	1	1	1	0	1	1
61	0	1	1	1	1	0	0
62	0	1	1	1	1	0	1
63	0	1	1	1	1	1	0
64	0	1	1	1	1	1	1

3. Geração de um *ranking* inicial de hiperparâmetros utilizando o método de Scott-Knott.

4. Definição das três melhores e três piores configurações no *ranking* inicial.

5. Geração de um novo grupo de resultados hiperparâmetros, unindo os dados das combinações básicas, melhores e piores no *ranking* inicial.

6. Aplicação do método HyperTuningSK, conforme Algoritmo 1.

Dessa forma, a partir do método HyperTuningSK são gerados *rankings* de recomendação de hiperparâmetros para *data augmentation*. As configurações básicas e as piores configurações no *ranking* inicial podem ser utilizadas como referência, e assim, verificar o quanto as melhores combinações podem influenciar na otimização da acurácia.

Tabela 5.4: Combinações de *data augmentation* analisadas (Parte III).

#	x_1 (R)	x_2 (H)	x_3 (V)	x_4 (He)	x_5 (S)	x_6 (W)	x_7 (Z)
65	1	0	0	0	0	0	0
66	1	0	0	0	0	0	1
67	1	0	0	0	0	1	0
68	1	0	0	0	0	1	1
69	1	0	0	0	1	0	0
70	1	0	0	0	1	0	1
71	1	0	0	0	1	1	0
72	1	0	0	0	1	1	1
73	1	0	0	1	0	0	0
74	1	0	0	1	0	0	1
75	1	0	0	1	0	1	0
76	1	0	0	1	0	1	1
77	1	0	0	1	1	0	0
78	1	0	0	1	1	0	1
79	1	0	0	1	1	1	0
80	1	0	0	1	1	1	1
81	1	0	1	0	0	0	0
82	1	0	1	0	0	0	1
83	1	0	1	0	0	1	0
84	1	0	1	0	0	1	1
85	1	0	1	0	1	0	0
86	1	0	1	0	1	0	1
87	1	0	1	0	1	1	0
88	1	0	1	0	1	1	1
89	1	0	1	1	0	0	0
90	1	0	1	1	0	0	1
91	1	0	1	1	0	1	0
92	1	0	1	1	0	1	1
93	1	0	1	1	1	0	0
94	1	0	1	1	1	0	1
95	1	0	1	1	1	1	0
96	1	0	1	1	1	1	1

5.3 Resultados para o Estudo de Caso 1

Neste primeiro estudo de caso, o método HyperTuningSK de recomendação hiperparâmetros de CNNs foi adotado na tarefa de reconhecimento da patologia de vegetação em fachadas de edificações. Para isso, foram aplicados modelos de *deep learning* no aprendizado de duas classes:

1. com vegetação na fachada da edificação;

2. sem vegetação na fachada da edificação.

Os resultados para esse estudo de caso são apresentados na seguinte sequência: (i) recomendação de hiperparâmetros (otimizador e taxa de aprendizado); (ii) testes; e (iii) análise de *data augmentation*.

Tabela 5.5: Combinações de *data augmentation* analisadas (Parte IV).

#	x_1 (R)	x_2 (H)	x_3 (V)	x_4 (He)	x_5 (S)	x_6 (W)	x_7 (Z)
97	1	1	0	0	0	0	0
98	1	1	0	0	0	0	1
99	1	1	0	0	0	1	0
100	1	1	0	0	0	1	1
101	1	1	0	0	1	0	0
102	1	1	0	0	1	0	1
103	1	1	0	0	1	1	0
104	1	1	0	0	1	1	1
105	1	1	0	1	0	0	0
106	1	1	0	1	0	0	1
107	1	1	0	1	0	1	0
108	1	1	0	1	0	1	1
109	1	1	0	1	1	0	0
110	1	1	0	1	1	0	1
111	1	1	0	1	1	1	0
112	1	1	0	1	1	1	1
113	1	1	1	0	0	0	0
114	1	1	1	0	0	0	1
115	1	1	1	0	0	1	0
116	1	1	1	0	0	1	1
117	1	1	1	0	1	0	0
118	1	1	1	0	1	0	1
119	1	1	1	0	1	1	0
120	1	1	1	0	1	1	1
121	1	1	1	1	0	0	0
122	1	1	1	1	0	0	1
123	1	1	1	1	0	1	0
124	1	1	1	1	0	1	1
125	1	1	1	1	1	0	0
126	1	1	1	1	1	0	1
127	1	1	1	1	1	1	0
128	1	1	1	1	1	1	1

5.3.1 Resultados para a Recomendação dos Hiperparâmetros

Nesta seção, são apresentados os resultados para a recomendação de hiperparâmetros para o primeiro estudo de caso. Conforme etapas apresentadas na Seção 5.1, o primeiro passo da metodologia proposta é realizar uma análise preliminar dos experimentos. Nesse sentido, a Tabela 5.6 apresenta os resultados de acurácia média na validação para cada repetição de arquitetura e método (otimizador + taxa de aprendizado).

A partir da Tabela 5.6 pode-se observar para a arquitetura DenseNet121 que todos os métodos analisados alcançaram o valor médio de acurácia limiar ($\neq 50\%$) para a sequência da recomendação dos hiperparâmetros. Por outro lado, ao adotar a arquitetura VGG16, percebe-se que as duas combinações (adam001 e adagrad025) limitaram-se a acurácia média na validação de apenas 50% em pelo menos uma das repetições. Por isso, os método adam001 e adagrad025 foram retirados do ajuste do modelo ANOVA para a arquitetura VGG16.

Em seguida, foi adotado o algoritmo HyperTuningSK para recomendação de hiperparâmetros para as duas arquiteturas analisadas. A Tabela 5.7 apresenta os principais resultados para

Tabela 5.6: Resultados para análise preliminar dos experimentos para o primeiro estudo de caso. Valores médios de acurácia (%) para cada repetição de arquitetura e método (otimizador + taxa de aprendizado).

Arquitetura	**Método**	**1**	**2**	**3**	**4**	**5**
	adagrad001	56,0	58,8	54,0	54,4	57,2
	adagrad005	56,8	66,4	66,8	61,2	57,0
	adagrad010	60,8	63,2	63,6	64,4	66,0
	adagrad015	74,4	68,0	74,8	70,4	72,8
DenseNet121	adagrad020	72,4	63,6	70,0	68,0	75,6
	adagrad025	65,2	73,2	68,8	67,6	66,4
	sgd001	65,6	58,0	63,2	56,8	58,0
	sgd005	61,6	76,8	69,2	68,0	74,8
	sgd010	64,8	76,4	70,4	68,4	77,6
	sgd015	74,8	73,6	64,4	68,8	76,8
	sgd020	68,0	68,4	63,2	80,4	64,4
	sgd025	64,8	82,4	66,4	70,0	76,4
	adam001	61,6	55,2	58,4	64,4	63,6
	adamax001	54,8	50,8	53,6	54,0	55,6
	adagrad001	67,2	72,0	67,2	69,2	74,0
	adagrad005	73,6	68,4	73,2	70,0	70,0
	adagrad010	66,4	71,6	58,0	66,4	54,8
	adagrad015	69,2	66,4	65,2	73,2	60,8
VGG16	adagrad020	58,0	60,4	55,6	51,2	62,8
	adagrad025	62,0	*50,0*	54,4	60,0	55,6
	sgd001	58,8	51,2	60,0	51,6	58,0
	sgd005	73,6	73,0	72,0	72,8	72,4
	sgd010	75,0	74,0	76,0	69,0	73,0
	sgd015	70,0	74,0	70,8	72,0	70,8
	sgd020	73,2	74,4	63,6	69,2	72,8
	sgd025	72,4	74,0	61,6	70,8	70,0
	adam001	*50,0*	*50,0*	*50,0*	*50,0*	*50,0*
	adamax001	82,4	80,4	79,2	79,2	73,2

aplicação do HyperTuningSK no primeiro estudo de caso. A análise da Tabela 5.7 pode-se iniciar observando que as medidas de adequação do modelo ANOVA foram satisfeitas para ambas as arquiteturas. A suposição de normalidade dos resíduos foi aceita ($p_{ks} > 0{,}05$) adotando o teste de Kolmogorov-Smirnov [188], com $p_{ks} = 0{,}86$ (DenseNet121) e $p_{ks} = 0{,}53$ (VGG16). Além disso, a premissa de homogeneidade das variâncias também foi cumprida ($p_{bt} > 0{,}05$) com o teste de Bartlett [189], com $p_{bt} = 0{,}21$ (DenseNet121) e $p_{bt} = 0{,}08$ (VGG16). Na sequência, verifica-se que foi rejeitada a hipótese inicial (H_0) e aceita a hipótese alternativa (H_a) para a significância dos modelos ANOVA ($p < 0{,}05$), pois estatística do teste é menor que $0{,}001$ para as duas arquiteturas. Ou seja, existe diferença significativa entre os desempenhos dos métodos simulados na classificação de imagens da construção civil (primeiro estudo de caso) para os experimentos com as duas estruturas (DenseNet121 e VGG16).

A Tabela 5.7 apresenta ainda os tratamentos selecionados pelo algoritmo HyperTuningSK para as duas arquiteturas analisadas. Nesse sentido, percebe-se que oito combinações foram recomendadas para a estrutura DenseNet121 e apenas uma para a adoção de VGG16. Esse resultado reforça hipótese que os hiperparâmetros devem ser ajustados de acordo com a arquitetura de *deep learning* adotada.

A Tabela 5.8, por sua vez, apresenta o *ranking* de hiperparâmetros para a arquitetura DenseNet121. Pode-se observar que o algoritmo HyperTuningSK, a partir do método de Scott-Knott

Tabela 5.7: Resultados do Algoritmo HyperTuningSK para o primeiro estudo de caso.

Arquitetura	p_{ks}	p_{bt}	p	**Métodos Recomendados**
DenseNet121	0,86	0,21	<0,001	adagrad015 sgd025 sgd015 sgd010 sgd005 adagrad020 sgd020 adagrad025
VGG16	0,53	0,08	<0,001	adamax001

Clustering [81], distribuiu os tratamentos em três grupos: A, B e C. No grupo A encontram-se os métodos recomendados, ou seja, combinações de hiperpararâmetros que alcançaram os melhores resultados médios de acurácia. Por outro lado, no grupo C estão os hiperparâmetros com os piores desempenhos. Nesse sentido, a diferença de acurácia entre um método recomendado pelo HyperTuningSK (ex.: adagrad015 - $72{,}0\%$) e um tratamento não selecionado (ex.: adamax001 - $54{,}0\%$) pode ser até de 18%.

Tabela 5.8: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o primeiro estudo de caso e arquitetura DenseNet121.

Grupo	**Método**	**Média (%)**
A	adagrad015	72,0
A	sgd025	71,8
A	sgd015	71,8
A	sgd010	71,4
A	sgd005	70,2
A	adagrad020	70,0
A	sgd020	68,6
A	adagrad025	68,2
B	adagrad010	63,6
B	adagrad005	61,6
B	adam001	60,6
B	sgd001	60,4
C	adagrad001	56,0
C	adamax001	54,0

A Figura 5.2 reforça as diferenças de desempenhos entre os hiperparâmetros e apresenta os grupos por cores, sendo: vermelho (grupo A), verde (grupo B) e azul (grupo C). Além disso, a partir desse gráfico é possível analisar o valor médio de acurácia de cada método (ponto central) e também variabilidade entre as medições (comprimento da linha). Para exemplificar, o método adagrad015 (Grupo A) possui acurácia média de $72{,}0\%$ (ou 0.72 no gráfico) e variação entre $68{,}0\%$ (0.68) e $74{,}8\%$ (0.748). Por outro lado, a combinação adamax001 (Grupo C) possui acurácia média de $54{,}0\%$ (ou 0.54 no gráfico) e variação entre $50{,}8\%$ (0.508) e $55{,}6\%$ (0.556).

A Tabela 5.9, por sua vez, apresenta o *ranking* de recomendação de hiperparâmetros para o primeiro estudo de caso com a arquitetura VGG16. Pode-se observar que o algoritmo HyperTuningSK, a partir do método de Scott-Knott, dividiu os hiperparâmetros em quatro grupos (A, B, C e D). Nesse caso, apenas a combinação adamax001 foi recomendada, pois é a única no grupo

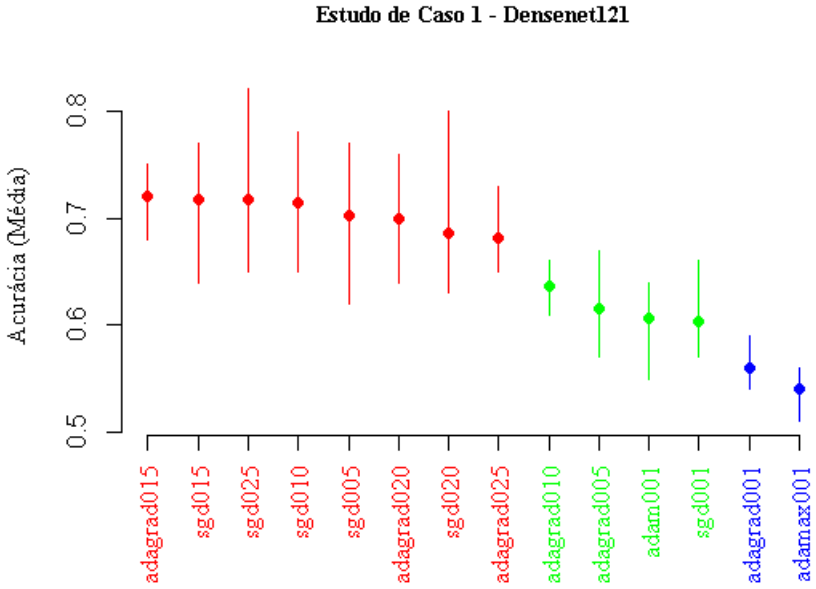


Figura 5.2: Gráfico de agrupamento de hiperparâmetros para o primeiro estudo de caso e arquitetura DenseNet121. As cores identificam os grupos: vermelho (grupo A - métodos recomendados), verde (grupo B) e azul (grupo C).

A. Nesse sentido, verifica-se a diferença de média de acurácia de até $22{,}6\%$ entre os hiperparâmetros classificados nos grupos A (adamax001 - $78{,}6\%$) e D (sgd001 - $56{,}0\%$). Além disso, também é possível analisar os distintos desempenhos entre os hiperparâmetros ao observar a Figura 5.3.

Tabela 5.9: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o primeiro estudo de caso e arquitetura VGG16.

Grupo	**Método**	**Média (%)**
A	adamax001	78,6
B	sgd010	73,4
B	sgd005	72,8
B	sgd015	71,6
B	adagrad005	71,0
B	sgd020	70,6
B	adagrad001	69,8
B	sgd025	69,8
B	adagrad015	66,8
C	adagrad010	63,4
D	adagrad020	57,6
D	sgd001	56,0

Dessa forma, após as análises da recomendação do algoritmo HyperTuningSK, combinações de hiperparâmetros do grupo A foram selecionadas para a etapa de experimentos de teste: (i) adagrad015 e sgd025 para a arquitetura DenseNet121 e (ii) adamax001 para VGG16.

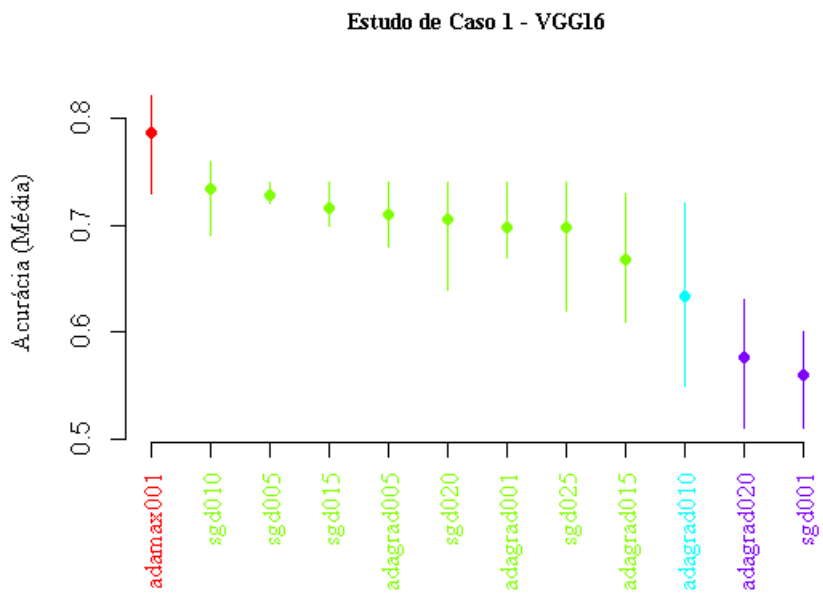


Figura 5.3: Gráfico de agrupamento de hiperparâmetros para o primeiro estudo de caso e arquitetura VGG16. As cores identificam os grupos: vermelho (grupo A - métodos recomendados), verde (grupo B), azul (grupo C) e roxo (grupo D).

5.3.2 Resultados para a Etapa de Teste

Nesta seção, são analisados os desempenhos dos tratamentos recomendados pelo método proposto: (i) adagrad015 e sgd025 (Desenet121) e; adamax001 (VGG16). Para fins comparativos, são utilizados hiperparâmetros adotados em outros trabalhos recentes na área de processamento de imagens na construção civil: adam001 [13], sgd001 [190] and sgd010 [131]. A Tabela 5.10 apresenta os resultados de validação e teste para a arquitetura DenseNet121.

Tabela 5.10: Resultados para a arquitetura DenseNet121 nos experimentos de teste no primeiro estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	**Método**	**Fonte**	**1**	**2**	**3**	**Média**
Validação	adagrad015	R	92,0	94,0	92,0	92,7
	sgd025	R	92,0	96,0	96,0	**94,7**
	adam001	L	94,0	90,0	92,0	92,0
	sgd001	L	86,0	90,0	88,0	88,0
	sgd010	R/L	92,0	92,0	92,0	92,0
Teste	adagrad015	R	78,9	76,7	73,3	76,3
	sgd025	R	76,7	70,0	76,7	74,5
	adam001	L	74,4	86,7	66,7	75,9
	sgd001	L	61,1	68,9	63,3	64,4
	sgd010	R/L	67,8	81,1	84,5	**77,8**

Pode-se observar na Tabela 5.10 que os hiperparâmetros selecionados alcançaram as maiores médias de acurácia na validação: sgd025 (94,7%) e adagrad015 (92,7%). Também vale destacar que um dos métodos recomendados (sgd025) foi o único a alcançar 96,0% de taxa de acertos na validação nas repetições. A Tabela 5.10 revela ainda que as duas maiores médias de acurácia na

etapa de teste foram obtidas pelos métodos sgd010 ($77{,}8\%$) e adagrad015 ($76{,}3\%$). Nesse caso, destaca-se que a combinação sgd010 foi recomendada pelo HP *Ranking* e também utilizada na literatura.

A Tabela 5.11, por sua vez, apresenta os resultados para a arquitetura VGG16. Vale destacar que o método recomendado (adamax001) alcançou as maiores médias de acurácia nas etapas de validação e teste. Além disso, a combinação do otimizador adamax com $lr = 0{,}001$ obteve os maiores valores máximos na validação ($94{,}0\%$) e teste ($87{,}8\%$) em comparação com os hiperparâmetros da literatura.

Tabela 5.11: Resultados para a arquitetura VGG16 nos experimentos de teste no primeiro estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	**Método**	**Fonte**	**1**	**2**	**3**	**Média**
Validação	adamax001	R	90,0	94,0	94,0	**92,7**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	76,0	78,0	80,0	78,0
	sgd010	L	80,0	80,0	82,0	80,7
Teste	adamax001	R	87,8	82,2	85,5	**85,2**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	61,1	62,2	64,5	62,6
	sgd010	L	78,9	77,8	76,7	77,8

Para exemplificar a classificação das imagens da construção civil no primeiro estudo de caso, as Figuras 5.4 e 5.5 apresentam as imagens adotadas na fase de teste. Além disso, é possível observar quando o modelo treinado com VGG16 e um método selecionado (adamax001) realizou a classificação correta. As imagens com um retângulo verde, significam classificação correta. Por outro lado, as figuras com um retângulo vermelho mostram as classificações incorretas. É possível observar que o modelo CNN classificou corretamente $95{,}6\%$ das imagens da classe positiva (com vegetação na fechada), como visto na Figura 5.4. Além disso, o modelo selecionado errou apenas nove imagens (de um total de 45) da classe negativa (sem vegetação na fachada), como observado na Figura 5.5.

5.3.3 Resultados para Análise de *Data Augmentation*

Nesta seção, são apresentados os resultados da análise de transformações de *data augmentation* para o primeiro estudo de caso. Nesse sentido, o método HyperTuningSK foi aplicado e foram satisfeitas as medidas de adequação: $p < 0{,}001$, $p_{ks} = 0{,}84$ e $p_{bt} = 0{,}11$. Na sequência, foi gerado o *ranking* de hiperparâmetros, conforme Tabela 5.12.

A partir da Tabela 5.12, percebe-se que três configurações foram recomendadas para o grupo A:

- *Height Shift Range* + *Width Shift Range* (11).
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$.
- *Width Shift Range* + *Zoom Range* (4)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 1$:
- *Horinzontal Flip* + *Height Shift Range* + *Width Shift Range* + *Zoom Range* (44)

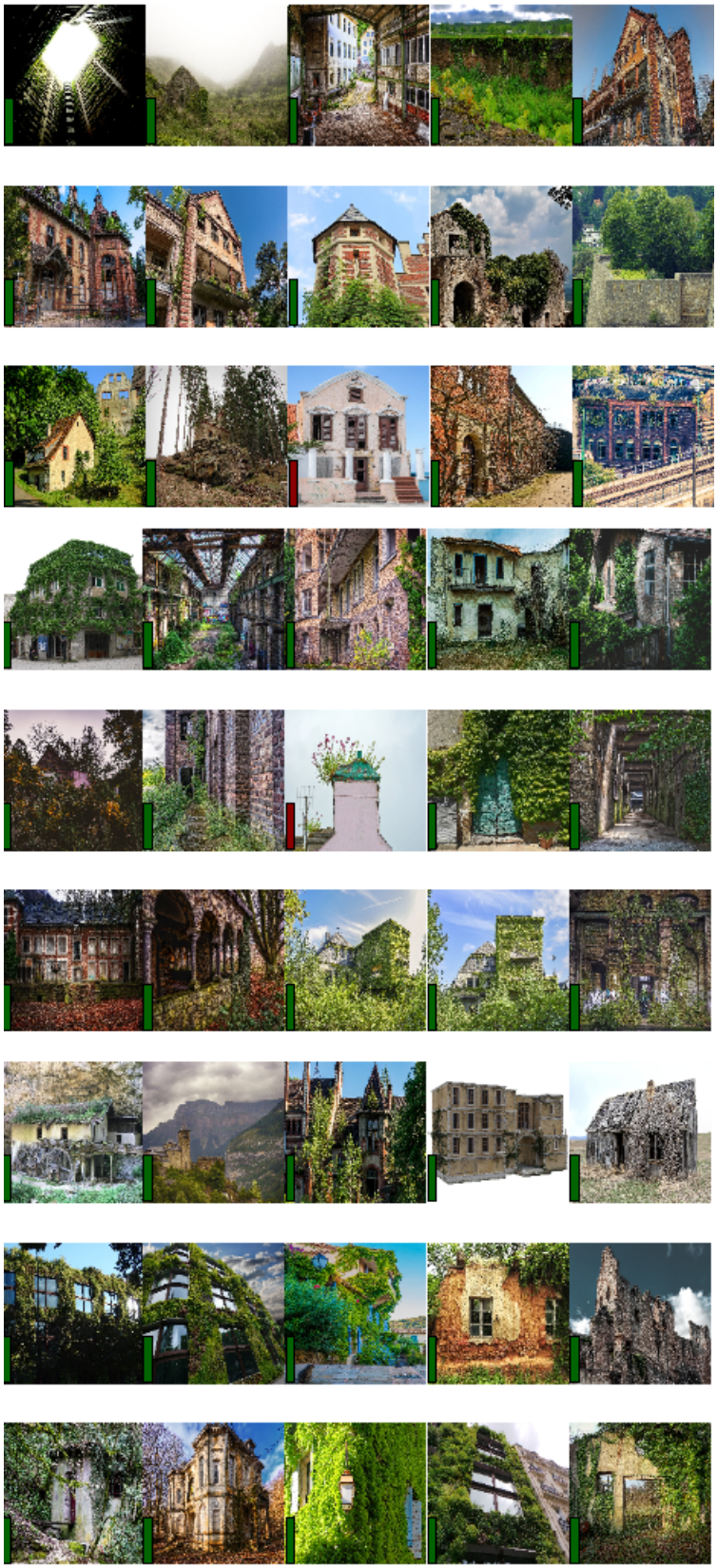


Figura 5.4: Resultado de classificação no conjunto de teste (classe 1 - com vegetação na fachada) a partir do treinamento com VGG16 e os hiperparâmetros recomendados (adamax e $lr = 0{,}001$) para EC1. Retângulo verde: classificação correta; retângulo vermelho: classificação incorreta.

Figura 5.5: Resultado de classificação no conjunto de teste (classe 0 - sem vegetação na fachada) a partir do treinamento com VGG16 e os hiperparâmetros recomendados (adamax e $lr = 0{,}001$) para EC1. Retângulo verde: classificação correta; retângulo vermelho: classificação incorreta.

– $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 1$.

Essas três combinações de DA obtiverem de $15{,}2\%$ até $18{,}4\%$ médias de acurácia superiores

Tabela 5.12: *Ranking* de recomendação de hiperparâmetros de *data augmentation* para o primeiro estudo de caso e arquitetura Mobilenet.

Grupo	Combinação de DA	Média (%)	Diferença (%)
A	11	92,0	+18,4
A	4	89,6	+16,0
A	44	88,8	+15,2
B	3	86,4	+12,8
B	2	82,8	+9,2
B	65	82,8	+9,2
C	9	81,2	+7,6
C	33	77,2	+3,6
C	5	76,8	+3,2
C	17	76,0	+2,4
C	119	75,6	+2,0
C	1 (*referência*)	73,6	–

a combinação de referência (sem aplicação de transformações nas imagens), alocada no grupo C do *ranking* de hiperparâmetros. Ressalta-se também que outras configurações também foram agrupadas no grupo C do *ranking*. Destaca-se, por exemplo, que a configuração 119 utiliza seis transformações possíveis ($x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_6 = 1$ e $x_7 = 0$), porém, seu resultado médio foi apenas 2% maior que a combinação 1. Dessa forma, reforçando a relevância da otimização de hiperparâmetros de *data augmentation* para o primeiro estudo de caso.

5.4 Resultados para o Estudo de Caso 2

No segundo estudo de caso, o método HyperTuningSK de recomendação hiperparâmetros de CNNs foi adotado na tarefa de detecção de patologias em calhas presentes em telhados de edificações. Para isso, foram aplicados modelos de *deep learning* no aprendizado de duas classes:

1. calha não-íntegra ou suja.
2. calha íntegra e limpa.

Os resultados para esse estudo de caso são apresentados na seguinte sequência: (i) recomendação de hiperparâmetros (otimizador e taxa de aprendizado); (ii) testes; e (iii) análise de *data augmentation*.

5.4.1 Resultados para a Recomendação dos Hiperparâmetros

Nesta seção, são apresentados os resultados da recomendação de hiperparâmetros para o segundo estudo de caso. A Tabela 5.13 apresenta os valores de acurácia média na validação para cada uma das repetições de arquitetura e métodos analisados. Pode-se observar que todas as combinações alcançaram médias de acurácia superiores a 50% ao adotar a arquitetura DenseNet121. Por outro lado, os métodos adam001 e sgd001 não alcançaram convergência na acurácia (limitadas em 50%) em pelo menos uma das repetições da arquitetura VGG16. Assim, essas duas combinações (adam + $lr = 0{,}001$ e sgd + $lr = 0{,}001$) foram retiradas da sequência de ajuste de hiperparâmetros de VGG16. A Tabela 5.14, por sua vez, apresenta um resumo dos resultados do HyperTuningSK algoritmo para as duas arquiteturas.

Tabela 5.13: Resultados para análise preliminar dos experimentos para o segundo estudo de caso. Valores médios de acurácia (%) para cada repetição de arquitetura e método (otimizador + taxa de aprendizado).

Arquitetura	**Método**	**1**	**2**	**3**	**4**	**5**
DenseNet121	adam001	64,7	62,7	64,7	56,7	60,0
	adamax001	58,6	54,7	54,0	54,0	60,7
	adagrad001	54,7	52,7	51,3	54,0	55,3
	adagrad005	61,3	58,0	58,7	58,0	59,3
	adagrad010	65,3	64,0	67,3	64,7	60,7
	adagrad015	68,7	67,3	62,0	68,7	67,3
	adagrad020	68,7	65,3	68,0	67,3	66,7
	adagrad025	71,3	70,7	71,3	74,0	72,0
	sgd001	58,0	53,3	60,7	63,3	61,3
	sgd005	62,0	61,3	56,7	64,0	60,0
	sgd010	62,7	68,7	67,3	68,0	65,3
	sgd015	68,7	67,3	62,0	68,7	67,3
	sgd020	72,0	75,3	68,0	68,7	61,3
	sgd025	66,0	66,7	62,7	70,0	74,0
VGG16	adam001	*50,0*	*50,0*	*50,0*	*50,0*	*50,0*
	adamax001	74,2	64,0	70,0	76,7	72,7
	adagrad001	67,3	78,0	70,7	75,3	54,7
	adagrad005	70,7	63,3	66,7	56,0	70,7
	adagrad010	66,7	68,0	74,0	66,0	60,0
	adagrad015	58,7	60,0	68,0	75,3	64,7
	adagrad020	65,3	52,0	59,0	62,0	60,7
	adagrad025	62,7	65,3	55,3	51,3	60,0
	sgd001	*50,0*	70,7	63,3	*50,0*	79,3
	sgd005	53,3	55,3	73,3	55,3	67,3
	sgd010	70,7	79,3	74,0	71,3	70,0
	sgd015	60,7	57,3	65,3	58,7	75,3
	sgd020	64,0	72,0	70,7	54,7	72,0
	sgd025	67,3	72,0	74,0	66,7	68,7

Tabela 5.14: Resultados do Algoritmo HyperTuningSK para o segundo estudo de caso.

Arquitetura	p_{ks}	p_{bt}	p	**Métodos Recomendados**
DenseNet121	0,52	0,19	<0,001	adagrad025 sgd020
VGG16	0,93	0,74	0,01	sgd010 adamax001 sgd025 adagrad001 sgd020 adagrad005 adagrad015

Pode-se observar que os ambos os modelos atenderam as suposições de normalidade ($p_{ks} > 0{,}05$) e homogeneidade ($p_{bt} > 0{,}05$). Além disso, para as duas arquiteturas a hipótese alternativa (H_1) da Análise de Variância foi aceita ($p < 0{,}05$), ou seja, existe diferença estatística

entre as médias de acurácia dos hiperparâmetros. A Tabela 5.14 apresenta ainda os métodos recomendados para cada estrutura referentes ao Grupo A do *ranking* de recomendação de hiperparâmetros.

Na sequência, a Tabela 5.15 apresenta o *ranking* de recomendação de hiperparâmetros para o segundo estudo de caso e arquitetura DenseNet121. Nesse aspecto, verifica-se que os hiperparâmetros foram distribuídos em quatro grupos adotando o algoritmo HyperTuningSK: A, B, C e D. Ressalta-se que a diferença de acurácia média entre os resultados dos grupos A (adagrad025 - 71,8%) e D (adagrad001 - 53,6%) pode chegar até 18,2%. Também é possível visualizar os distintos desempenhos entre os grupos de hiperparâmetros na Figura 5.6, sendo: vermelho (grupo A - métodos recomendados), verde (grupo B), azul (grupo C) e roxo (grupo D).

Tabela 5.15: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o segundo estudo de caso e arquitetura DenseNet121.

Grupo	**Método**	**Média (%)**
A	adagrad025	71,8
A	sgd020	69,0
B	sgd025	68,0
B	adagrad020	67,2
B	adagrad015	66,8
B	sgd015	66,8
B	sgd010	66,4
B	adagrad010	64,4
C	adam001	62,0
C	sgd005	60,8
C	sgd001	59,2
C	adagrad005	59,0
D	adamax001	56,6
D	adagrad001	53,6

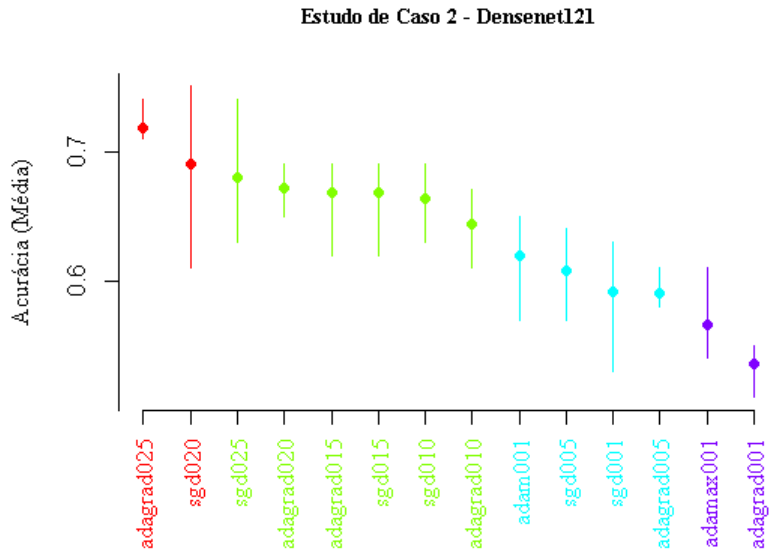


Figura 5.6: Gráfico de agrupamento de hiperparâmetros para o segundo estudo de caso e arquitetura DenseNet121. As cores identificam os grupos: vermelho (grupo A - métodos recomendados), verde (grupo B), azul (grupo C) e roxo (grupo D).

Os resultados do *ranking* de recomendação de hiperparâmetros para o segundo estudo de caso e arquitetura VGG16 são apresentados na Tabela 5.16. Nessa situação, o algoritmo HyperTuningSK, a partir do método de Scott-Knott, distribuiu os hiperparâmetros em dois grupos: A e B. Pode-se observar que a diferença média de acurácia foi de até $14{,}2\%$ entre os hiperparâmetros do grupo A (sgd010 - $73{,}0\%$) e B (adagrad025 - $58{,}8\%$). A Figura 5.7, por sua vez, apresenta graficamente os resultados de acurácia média para os hiperparâmetros analisados e respectivos grupos (A - vermelho; B - azul).

Tabela 5.16: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o segundo estudo de caso e arquitetura VGG16.

Grupo	**Método**	**Média (%)**
A	sgd010	73,0
A	adamax001	71,6
A	sgd025	69,8
A	adagrad001	69,2
A	adagrad010	67,0
A	sgd020	66,8
A	adagrad005	65,6
A	adagrad015	65,4
B	sgd015	63,4
B	sgd005	60,6
B	adagrad020	59,8
B	adagrad025	58,8

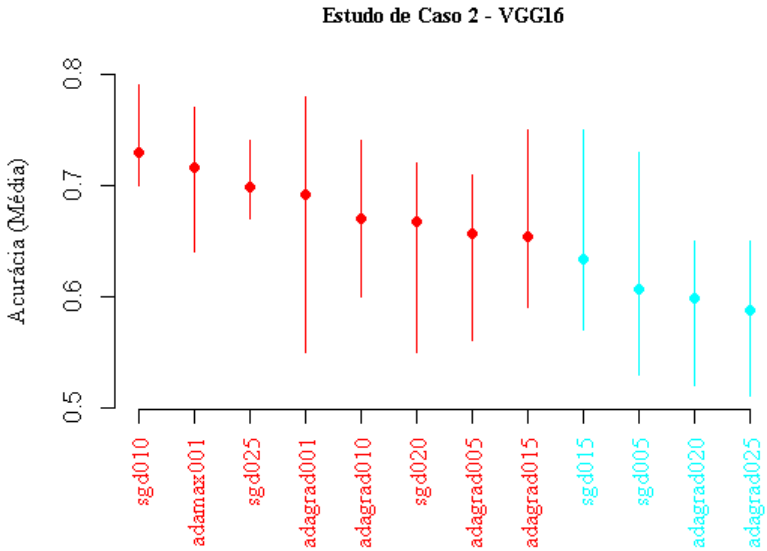


Figura 5.7: Gráfico de agrupamento de hiperparâmetros para o segundo estudo de caso e arquitetura VGG16. As cores identificam os grupos: vermelho (grupo A - métodos recomendados) e azul (grupo B).

5.4.2 Resultados para a Etapa de Teste

Na sequência, dois métodos recomendados por arquitetura (DenseNet121 - adagrad025 e sgd020; VGG16 - sgd010 e adamax001) foram analisados em uma nova rodada de experimentos

de testes e comparados com hiperparâmetros da literatura (mesmos passos da Seção 5.1.3). Nesse sentido, a Tabela 5.17 apresenta os resultados de validação e teste para a arquitetura Densenet.

Tabela 5.17: Resultados para a arquitetura DenseNet121 nos experimentos de teste no segundo estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	**Método**	**Fonte**	**1**	**2**	**3**	**Média**
Validação	adagrad025	R	93,3	96,7	96,7	**95,6**
	sgd020	R	96,7	96,7	93,3	**95,6**
	adam001	L	93,3	93,3	96,7	94,4
	sgd001	L	90,0	86,7	90,0	88,9
	sgd010	L	93,3	93,3	96,7	94,4
Teste	adagrad025	R	90,0	90,0	76,7	**85,7**
	sgd020	R	83,3	63,3	76,7	74,4
	adam001	L	63,3	53,4	80,0	65,7
	sgd001	L	83,3	76,7	80,0	80,0
	sgd010	L	83,3	80,0	83,3	82,2

A partir da Tabela 5.17, pode-se observar que as maiores médias de acurácia na validação foram alcançadas pelos hiperparâmetros recomendados: adagrad025 ($95{,}6\%$) e sgd020 ($95{,}6\%$). Além disso, a combinação adagrad025 também atingiu a maior média de acurácia na etapa de teste para arquitetura Densenet: $85{,}7\%$. Ressalta-se também que o método adagrad025 foi o único a obter 90% de acertos em uma repetição na etapa de teste.

A Tabela 5.18, por sua vez, apresenta os resultados para a arquitetura VGG16. Nesse aspecto, destaca-se que um método recomendado pelo algoritmo HyperTuningSK também alcançou as maiores médias de acurácia na validação ($96{,}7\%$) e teste ($84{,}4\%$), em comparação com hiperparâmetros adotados na literatura. Também é importante ressaltar que a combinação selecionada (adamax001) foi a única a alcançar 90% ao adotar a arquitetura VGG16 na base de teste.

Tabela 5.18: Resultados para a arquitetura VGG16 nos experimentos de teste no segundo estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	**Método**	**Fonte**	**1**	**2**	**3**	**Média**
Validação	adamax001	R	96,7	100,0	93,3	**96,7**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	80,0	63,3	60,0	67,8
	sgd010	R/L	90,0	86,7	83,3	86,7
Teste	adamax001	R	80,0	90,0	83,3	**84,4**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	66,7	63,3	60,0	63,3
	sgd010	R/L	70,0	53,3	53,3	58,8

5.4.3 Resultados para Análise de *Data Augmentation*

Nesta seção, são apresentados os resultados da análise de transformações de *data augmentation* para o segundo estudo de caso. Nesse sentido, o método HyperTuningSK foi aplicado e foram satisfeitas as medidas de adequação: $p < 0{,}001$, $p_{ks} = 0{,}13$ e $p_{bt} = 0{,}29$. Na sequência, foi gerado o *ranking* de hiperparâmetros, conforme Tabela 5.19.

Tabela 5.19: *Ranking* de recomendação de hiperparâmetros de *data augmentation* para o segundo estudo de caso e arquitetura Mobilenet.

Grupo	**Combinação de DA**	**Média (%)**	**Diferença (%)**
A	29	90,7	+8,0
A	57	90,7	+8,0
A	7	90,0	+7,3
A	3	88,0	+5,3
B	2	85,3	+2,6
B	9	83,3	+0,6
B	33	83,3	+0,6
B	5	83,3	+0,6
B	1 (*referência*)	82,7	–
C	17	80,7	–2,0
C	65	80,0	–2,7
D	75	74,7	–8,0
D	121	74,7	–8,0
D	67	73,3	–9,4

A partir da Tabela 5.19, percebe-se que quatro configurações foram recomendadas para o grupo A:

- *Vertical Flip* + *Height Shift Range* + *Shear Range* (29).
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_6 = 0$ e $x_7 = 0$.
- *Horinzontal Flip* + *Vertical Flip* + *Height Shift Range* (57)
  - $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$:
- *Shear Range* + *Width Shift Range* (7)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 1$ e $x_7 = 0$.
- *Width Shift Range* (3)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$.

As combinações recomendadas no grupo A alcançaram entre $88{,}0\%$ e $90{,}7\%$ de média de acurácia, ou seja, uma diferença de até $8{,}0\%$ em relação ao método básico (configuração 1). Ressalta-se também para o segundo estudo de caso as configurações presentes dos grupos C e D, com desempenhos inferiores a aplicação de *data augmentation* sem transformação. Nessa linha, a combinação 65 ($x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$) alcançou apenas $80{,}0\%$ de média de acurácia. Outro exemplo é a configuração 67 ($x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 1$ e $x_7 = 0$), obtendo um resultado de $9{,}4\%$ menor que a combinação que não adota transformações nas imagens.

5.5 Resultados para o Estudo de Caso 3

No terceiro estudo de caso, o método de recomendação de hiperparâmetros foi aplicado na classificação de máquinas na construção civil. Nesse sentido, foram utilizados modelos de Redes Neurais Convolucionais no aprendizado de imagens de dois tipos de veículos:

- Máquina do tipo escavadora (*excavator*).
- Máquina do tipo *mixer* de concreto.

Os resultados para esse estudo de caso são apresentados na seguinte sequência: (i) recomendação de hiperparâmetros (otimizador e taxa de aprendizado); (ii) testes; e (iii) análise de *data augmentation*.

5.5.1 Resultados para a Recomendação dos Hiperparâmetros

Nesta seção, é descrito o processo de recomendação de hiperparâmetros para o terceiro estudo de caso. A Tabela 5.20 apresenta os resultados para a análise preliminar dos experimentos.

Tabela 5.20: Resultados para análise preliminar dos experimentos para o terceiro estudo de caso. Valores médios de acurácia (%) para cada repetição de arquitetura e método (otimizador + taxa de aprendizado).

Arquitetura	**Método**	**1**	**2**	**3**	**4**	**5**
	adagrad001	56,3	55,0	55,8	55,0	53,8
	adagrad005	62,0	61,3	59,3	63,3	59,0
	adagrad010	62,3	65,5	65,3	62,8	68,0
	adagrad015	69,0	67,5	67,5	71,5	66,0
DenseNet121	adagrad020	70,8	64,0	71,3	74,8	76,5
	adagrad025	72,0	73,5	71,5	74,5	63,0
	sgd001	53,0	53,5	56,3	56,3	59,0
	sgd005	57,5	58,8	57,5	61,0	62,5
	sgd010	62,0	64,3	65,8	66,8	68,8
	sgd015	70,8	71,5	61,0	67,5	61,5
	sgd020	64,8	58,8	66,0	66,5	68,0
	sgd025	64,8	74,3	65,8	71,5	56,8
	adam001	63,0	57,0	58,5	67,3	60,3
	adamax001	61,3	53,8	56,5	52,8	56,9
	adagrad001	53,8	53,0	56,3	55,5	52,3
	adagrad005	59,3	56,5	64,5	58,0	52,5
	adagrad010	56,5	65,8	*50,0*	60,0	56,0
	adagrad015	57,8	58,3	55,0	53,8	51,8
VGG16	adagrad020	*50,0*	52,8	53,0	54,8	56,0
	adagrad025	50,8	56,8	54,3	56,5	*50,0*
	sgd001	54,5	*50,0*	53,8	55,5	*50,0*
	sgd005	58,5	60,5	55,0	59,5	70,8
	sgd010	57,8	61,8	52,8	56,3	60,5
	sgd015	56,3	58,8	58,3	59,5	56,0
	sgd020	56,0	*50,0*	54,0	58,3	53,5
	sgd025	55,0	50,8	56,0	51,0	58,8
	adam001	*50,0*	*50,0*	*50,0*	*50,0*	*50,0*
	adamax001	57,3	77,5	74,0	53,3	*50,0*

Nota-se que todas as combinações de hiperparâmetros alcançaram médias de acurácia maiores que 50% ao adotar a arquitetura DenseNet121. No entanto, sete métodos não ultrapassaram esse valor limiar em pelo menos uma das repetições com a estrutura VGG16, sendo: adagrad010, adagrad020, adagrad025, sgd001, sgd020, adam001 e adamax001. Assim, essas combinações foram retiradas das próximas fases de seleção de hiperparâmetros para o terceiro estudo de caso.

Na sequência, foi aplicado o algoritmo HyperTuningSK na recomendação de hiperperparâmetros para ambas arquiteturas. Nesse aspecto, a Tabela 5.21 apresenta um resumo dos resultados da adoção do HyperTuningSK no terceiro estudo de caso. Pode-se observar que a hipótese de normalidade dos resíduos ($p_{ks} > 0{,}05$) foi satisfeita para as duas arquiteturas. Além disso, a premissa de homogeneidade das variâncias também foi satisfeita ($p_{bt} > 0{,}05$), sendo DenseNet121 ($p_{bt} = 0{,}051$) e VGG16 ($p_{bt} = 0{,}22$). Também vale ressaltar que a hipótese alternativa (H_1) de Análise de Variância foi aceita para o modelo da arquitetura DenseNet121 ($p < 0{,}05$). Para o modelo ANOVA da estrutura VGG16, o resultado foi $p = 0{,}05196$. Nesse caso, para apenas nessa situação, foi alterado o critério de significância para 10% ($p < 0{,}10$) no algoritmo HyperTuningSK.

Tabela 5.21: Resultados do Algoritmo HyperTuningSK para o terceiro estudo de caso.

Arquitetura	p_{ks}	p_{bt}	p	**Métodos Recomendados**
DenseNet121	0,59	0,051	<0,001	adagrad020 adagrad025 adagrad015
VGG16	0,96	0,22	0,05196	sgd005 adagrad005 sgd010 sgd015

A Tabela 5.21 apresenta ainda os hiperparâmetros recomendados para cada uma das arquiteturas analisadas. Nesse caso, vale ressaltar que os três métodos recomendados para a DenseNet121 utilizam o otimizador adagrad: adagrad020, adagrad025 e adagrad015. Por outro lado, para a arquitetura VGG16, três combinações recomendadas adotam o otimizador sgd (sgd005, sgd010 e sgd015) e apenas um método selecionado utiliza a otimização por adagrad (adagrad005). Nesse sentido, esses resultados reforçam a relevância de selecionar os hiperparâmetros (otimizador e taxa de aprendizado) de acordo com a arquitetura adotada no terceiro estudo de caso.

A Tabela 5.22, por sua vez, apresenta o *ranking* de recomendação de hiperparâmetros para o terceiro estudo de caso e arquitetura DenseNet121. Pode-se notar que o método HyperTuningSK utilizando o Scott-Knott *Clustering* separou as combinações em quatro grupos (A, B, C e D). Cabe ressaltar que a diferença de acurácia média entre os hiperparâmetros foi de até $16{,}4\%$ entre o grupo A (adagrad020 - $71{,}6\%$) e D (adagrad001 - $55{,}2\%$). Também é importante destacar que todos os métodos agrupados nos dois melhores grupos (A e B) possuem $lr \geq 0{,}010$. A Figura 5.8 retrata graficamente a divisão em grupos de hiperparâmetros para o terceiro estudo de caso e arquitetura DenseNet121, sendo: vermelho (A), verde (B), azul (C) e roxo (D).

A Tabela 5.23 e a Figura 5.9, por sua vez, apresentam para o terceiro estudo de caso e VGG16, o *ranking* de recomendação e o gráfico de grupos, respectivamente. Nesse caso, os métodos foram distribuídos em apenas dois grupos (A - vermelho e B - azul). Além disso, a diferença máxima de acurácia média entre os hiperparâmetros foi de até $7{,}0\%$ (sgd005 e adagrad001).

Tabela 5.22: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o terceiro estudo de caso e arquitetura DenseNet121.

Grupo	Método	Média (%)
A	adagrad020	71,6
A	adagrad025	71,2
A	adagrad015	68,6
B	sgd015	66,8
B	sgd025	66,8
B	sgd010	65,6
B	sgd020	65,0
B	adagrad010	64,8
C	adam001	61,2
C	adagrad005	60,8
C	sgd005	59,8
D	adamax001	56,4
D	sgd001	55,6
D	adagrad001	55,2

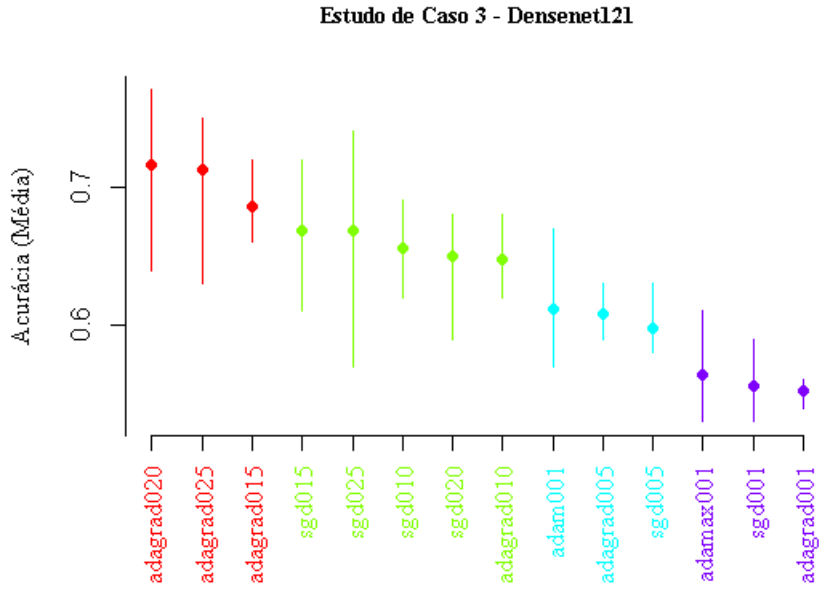


Figura 5.8: Gráfico de agrupamento de hiperparâmetros para o terceiro estudo de caso e arquitetura Densenet. As cores identificam os grupos: vermelho (grupo A - métodos recomendados), verde (grupo B), azul (grupo C) e roxo (grupo D).

5.5.2 Resultados para a Etapa de Teste

Na sequência, duas combinações de hiperparâmetros recomendadas por arquitetura (DenseNet121 - adagrad020 e adagrad025; VGG16 - sgd005 e adagrad005) foram adotadas na etapa de teste do terceiro estudo de caso. Conforme passos retratados na abordagem proposta, esses tratamentos selecionados foram comparados com hiperparâmetros da literatura.

A Tabela 5.24 apresenta os resultados para validação e teste da arquitetura DenseNet121. Nesse sentido, nota-se que a maior acurácia na validação foi alcançada por um método recomendado: adagrad025 ($94{,}6\%$). Ao avaliar a etapa de teste, a maior média de acurácia foi alcançada por um método da literatura (sgd010 - $92{,}5\%$). No entanto, ressalta-se que a segunda

Tabela 5.23: *Ranking* de recomendação de hiperparâmetros (HP Ranking) para o terceiro estudo de caso e arquitetura VGG16.

Grupo	**Método**	**Média (%)**
A	sgd005	61,2
A	adagrad005	58,4
A	sgd010	58,0
A	sgd015	57,8
B	adagrad015	55,4
B	sgd025	54,4
B	adagrad001	54,2

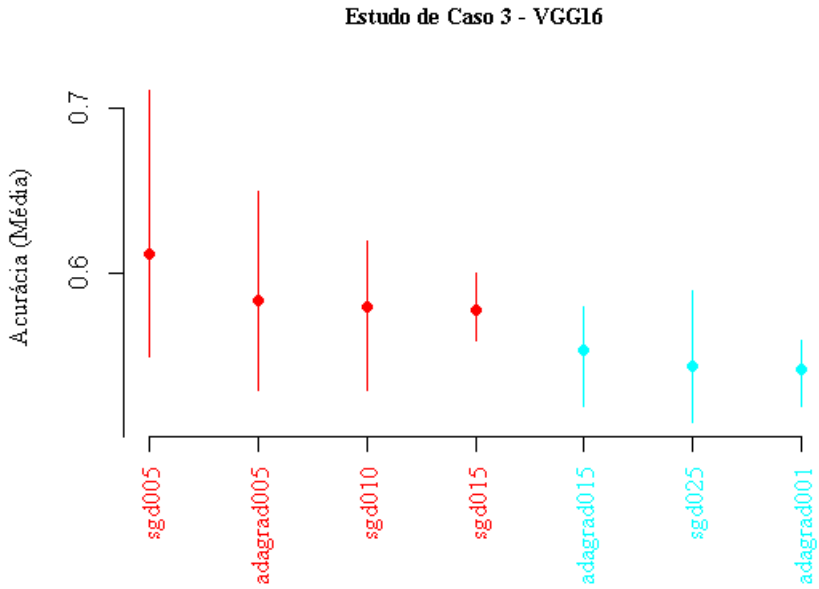


Figura 5.9: Gráfico de agrupamento de hiperparâmetros para o terceiro estudo de caso e arquitetura VGG16. As cores identificam os grupos: vermelho (grupo A - métodos recomendados) e azul (grupo B).

maior média foi de uma combinação recomendada adagrad025 ($91{,}3\%$), ou seja, uma diferença de apenas $1{,}2\%$ entre os desempenhos de sgd010 e adagrad025.

Em seguida, a Tabela 5.25 retrata os resultados nas etapas de validação e teste para a arquitetura VGG16. Vale ressaltar o desempenho do método recomendado adagrad005, essa configuração obteve os melhores resultados médios de acurácia nas duas etapas: validação ($85{,}4\%$) e teste ($80{,}4\%$). Também cabe destacar que outro método recomendado (sgd010) também atingiu o mesmo patamar de acurácia média na etapa de teste ($80{,}4\%$).

Para exemplificar a classificação das imagens da construção civil no terceiro estudo de caso, as Figuras 5.10 e 5.11 apresentam as fotografias adotadas na fase de teste. Além disso, é possível observar quando o modelo treinado com DenseNet121 e um método selecionado (adagrad020) realizou classificações corretas (imagens com retângulo verde). Por outro lado, as figuras com um retângulo vermelho representam classificações incorretas. É possível observar que o modelo CNN classificou corretamente $95{,}0\%$ das imagens da classe 0 (máquina do tipo escavadeira), como visto na Figura 5.10. Além disso, o modelo selecionado errou apenas três imagens (de um total de 40) da classe 1 (máquina do tipo *mixer*), como observado na Figura 5.11.

Tabela 5.24: Resultados para a arquitetura DenseNet121 nos experimentos de teste no terceiro estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	Método	Fonte	1	2	3	Média
Validação	adagrad020	R	95,0	93,8	93,8	94,2
	adagrad025	R	96,3	93,8	93,8	**94,6**
	adam001	L	82,5	87,5	83,8	84,6
	sgd001	L	85,0	78,8	75,0	79,6
	sgd010	L	91,3	88,8	88,8	89,6
Teste	adagrad020	R	93,8	81,3	81,3	85,5
	adagrad025	R	92,5	91,3	90,0	91,3
	adam001	L	88,8	81,3	85,0	85,0
	sgd001	L	63,8	81,3	71,3	72,1
	sgd010	L	91,3	92,5	93,8	**92,5**

Tabela 5.25: Resultados para a arquitetura VGG16 nos experimentos de teste no terceiro estudo de caso. Acurácia máxima (%) nas etapas de validação e teste em cada repetição de experimento e respectiva acurácia média. Comparação entre os métodos recomendados pelo HyperTuningSK algoritmo (HP *Ranking* - R) e hiperparâmetros usados na literatura (L).

Etapa	Método	Fonte	1	2	3	Média
Validação	sgd005	R	71,3	78,8	78,8	76,3
	adagrad005	R	87,5	81,3	87,5	**85,4**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	60,0	57,5	51,3	57,1
	sgd010	R/L	82,5	81,3	81,3	81,7
Teste	sgd005	R	62,5	66,3	76,3	68,4
	adagrad005	R	65,0	85,0	91,3	**80,4**
	adam001	L	50,0	50,0	50,0	50,0
	sgd001	L	71,3	66,3	60,0	65,9
	sgd010	R/L	71,2	86,2	83,8	**80,4**

5.5.3 Resultados para Análise de *Data Augmentation*

Nesta seção, são apresentados os resultados da análise de transformações de *data augmentation* para o terceiro estudo de caso. Nesse sentido, o método HyperTuningSK foi aplicado e foram satisfeitas as medidas de adequação: $p < 0{,}001$, $p_{ks} = 0{,}60$ e $p_{bt} = 0{,}77$. Na sequência, foi gerado o *ranking* de hiperparâmetros, dividindo as combinações em dois grupos (A e B), conforme Tabela 5.26. A partir da Tabela 5.26, percebe-se que sete configurações foram recomendadas para o grupo A:

- *Horinzontal Flip* + *Shear Range* + *Width Shift Range* (39)
  - $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $x_6 = 1$ e $x_7 = 0$.
- *Rotation Range* + *Horinzontal Flip* (97)
  - $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$):
- *Vetical Flip* + *Shear Range* + *Width Shift Range* (23)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$, $x_5 = 1$, $x_6 = 1$ e $x_7 = 0$.

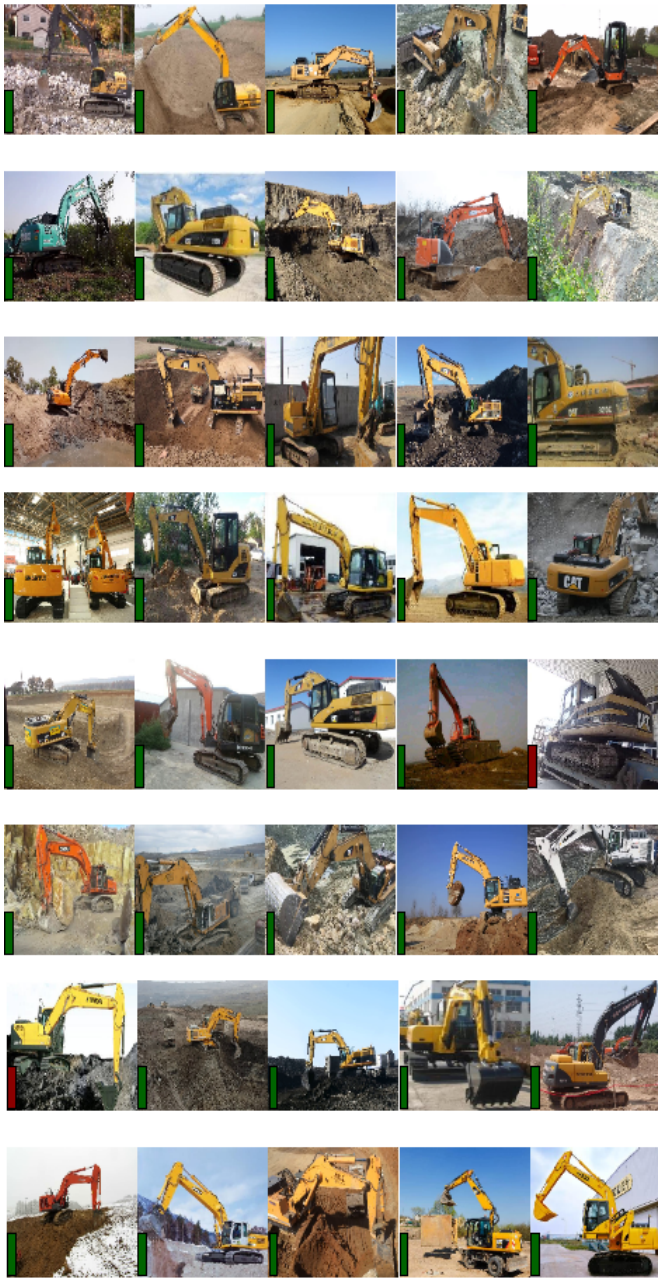


Figura 5.10: Resultado da classificação das imagens no conjunto de teste (classe 0 - máquina do tipo escavadeira) a partir do treinamento com a arquitetura DenseNet121 e os hiperparâmetros recomendados (adagrad e $lr = 0{,}020$) para o EC3. Retângulo verde: classificação correta; retângulo vermelho: classificação incorreta.

- *Zoom Range* (2)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 1$.
- *Height Shift Range* (9)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$.
- *Width Shift Range* (3)
  - $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$.

Figura 5.11: Resultado da classificação das imagens no conjunto de teste (classe 1 - máquina do tipo *mixer*) a partir do treinamento com a arquitetura DenseNet121 e os hiperparâmetros recomendados (adagrad e $lr = 0{,}020$) para o EC3. Retângulo verde: classificação correta; retângulo vermelho: classificação incorreta.

- *Rotation Range* (65)
  - $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$ e $x_7 = 0$.

As combinações recomendadas no grupo A alcançaram médias de acurácia entre $81{,}5\%$ e $86{,}0\%$ na classificação de imagens do terceiro estudo de caso. Por outro lado, as configurações agrupadas em B obtiveram acurácias médias no intervalo de no máximo $76{,}5\%$. Além disso, ressalta-se que o método referência (sem aplicação de transformação nas imagens) para *data augmentation* (configuração 1) foi ranqueado no grupo B pelo HyperTuningSK.

Tabela 5.26: *Ranking* de recomendação de hiperparâmetros de *data augmentation* para o terceiro estudo de caso e arquitetura Mobilenet.

Grupo	Combinação de DA	Média (%)	Diferença (%)
A	39	86,0	+11,7
A	97	85,5	+11,1
A	23	84,8	+10,5
A	2	84,0	+9,7
A	9	83,3	+9,0
A	3	83,0	+8,7
A	65	81,5	+7,2
B	33	76,5	+2,2
B	5	76,3	+2,0
B	1 (*referência*)	74,3	–
B	127	74,0	–0,3
B	64	73,3	–1,0
B	17	71,3	–3,0

5.6 Discussão

Nesta seção, são discutidos aspectos gerais sobre a recomendação de hiperparâmetros para a classificação de imagens da construção civil nos três estudos de casos abordados. Além disso, na primeira etapa desta discussão é realizada a análise dos resultados por um novo índice proposto nesta tese para avaliação de máximos grupos dos hiperparâmetros, denominado *HyperScore*. O *HyperScore* representa o melhor grupo alocado para o método nos três estudos de casos. Essa métrica também enfatiza quantas vezes os hiperparâmetros alcançaram o topo dos *rankings* de recomendação. Nesse aspecto, a Tabela 5.27 mostra os resultados do *HyperScore* de acordo com o estudo de caso e arquitetura adotada na seleção dos hiperparâmetros otimizador e taxa de aprendizado.

Observa-se na Tabela 5.27 que o método adagrad025 tem $HyperScore = AAA$ para a arquitetura DenseNet121, ou seja, foi agrupado em A em todos os três estudos de casos. Nessa mesma linha, os hiperparâmetros adagrad015, adagrad020 e sgd020 obtiveram $HyperScore = AA$ para a arquitetura DenseNet121, pois foram alocados duas vezes no grupo A. Por outro lado, adagrad001 e adamax001 possuem $HyperScore = C$ (DenseNet121), ou seja, esses hiperparâmetros obtiveram no máximo a indicação de um grupo C nos estudos de casos. Dessa forma, adagrad025 foi o método com maior regularidade de desempenho nas três aplicações com adoção da arquitetura DenseNet121 ($HyperScore = AAA$).

Os resultados do *HyperScore* para a arquitetura VGG16 também são apresentados na Tabela 5.27. Pode-se notar que, nesse caso nenhum método obteve a métrica AAA nas recomendações. No entanto, três combinações de hiperparâmetros alcançaram $HyperScore = AA$: adagrad005, sgd010 e adamax001. Por outro lado, o método sgd001 conseguiu apenas $HyperScore = D$ com a adoção de VGG16.

Vale ressaltar também que, as combinações de hiperparâmetros (taxa de aprendizado e otimizador) demonstraram desempenhos distintos de acordo com arquitetura de *deep learning* utilizada. Em seguida, dois exemplos:

- adagrad025: DenseNet121 (AAA) e VGG16 (B),
- adamax001: DenseNet121 (C) e VGG16 (AA).

Dessa forma, os exemplos de desempenhos dessas configurações (adagrad025 e adamax001)

Tabela 5.27: Resumo dos resultados de recomendação dos hiperparâmetros taxa de aprendizado e otimizador para os três estudos de casos (EC).

Arquitetura	**Método**	**EC1**	**EC2**	**EC3**	***HyperScore***
	adagrad001	C	D	D	C
	adagrad005	B	C	C	B
	adagrad010	B	B	B	B
	adagrad015	A	B	A	AA
DenseNet121	adagrad020	A	B	A	AA
	adagrad025	A	A	A	**AAA**
	sgd001	B	C	D	B
	sgd005	A	C	C	A
	sgd010	A	B	B	A
	sgd015	A	B	B	A
	sgd020	A	A	B	AA
	sgd025	A	B	B	A
	adam001	B	C	C	B
	adamax001	C	D	D	C
	adagrad001	B	A	B	A
	adagrad005	B	A	A	**AA**
	adagrad010	C	A	–	A
	adagrad015	B	A	B	A
VGG16	adagrad020	D	B	–	B
	adagrad025	–	B	–	B
	sgd001	D	–	–	D
	sgd005	B	B	A	A
	sgd010	B	A	A	**AA**
	sgd015	B	B	A	A
	sgd020	B	A	–	A
	sgd025	B	A	B	A
	adam001	–	–	–	–
	adamax001	A	A	–	**AA**

reforçam a importância do ajuste dos hiperparâmetros de acordo arquitetura de CNN empregada.

A Tabela 5.28, por sua vez, apresenta as transformações de *data augmentation* recomendadas pelo algoritmo HyperTuningSK para cada um dos estudos de casos. Observa-se na Tabela 5.28 que as duas transformações mais recomendadas nos estudos de casos são: *Width Shift* (8 $\times$) e *Heigth Shift* (5 $\times$). Além disso, esses dois tipos de métodos para processamento de imagens em *data augmentation* foram selecionados para os três estudos de casos, pelo menos uma vez. Por outro lado, a maioria das demais transformações foram recomendadas para dois ou apenas um estudo de caso. Por exemplo, o método de rotação (*Rotation*) foi selecionado (grupo A) apenas em duas combinações indicadas no terceiro estudo de caso. A Tabela 5.28 revela ainda a relevância em adotar configurações de *data augmentation* com mais de uma técnica de processamento de imagem. Percebe-se nessa tabela que $64{,}3\%$ das combinações recomendadas pelo algoritmo HyperTuningSK adotam dois ou mais métodos de transformações de *data augmentation*. Nessa linha, para o Estudo de Caso 1, essa hipótese ainda torna-se ainda mais relevante, pois todas as três combinações recomendadas utilizam pelo menos duas técnicas de processamento de imagens.

Por fim, cabe ressaltar também que a Tabela 5.28 reforça a importância da seleção dos hiperparâmetros de *data augmentation* de acordo com a base de dados adotada. Isso porque, de

Tabela 5.28: Transformações de *data augmentation* recomendadas (grupo A) para os três estudos de casos (EC).

Aplicação	Comb.	*Rotation*	*Horizontal Flip*	*Vertical Flip*	*Heigth Shift*	*Shear Range*	*Width Shift*	*Zoom*
	11				✓		✓	
EC1	4						✓	✓
	44		✓		✓		✓	✓
	29			✓	✓	✓		
EC2	57		✓	✓	✓			
	7					✓	✓	
	3						✓	
	39		✓			✓	✓	
	97	✓	✓					
EC3	23			✓		✓	✓	
	2							✓
	9				✓			
	3						✓	
	65	✓						
Total		2	4	3	5	4	8	3

todas configurações recomendadas na Tabela 5.28, apenas a combinação 3 (*Width Shift*: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$) foi indicada para pelo menos dois estudos de casos: EC2 e EC3.

CAPÍTULO 6

Método AutoHyperTuningSK para Ajuste Automático de Hiperparâmetros

Neste capítulo, o método AutoHyperTuningSK para ajuste automático de hiperparâmetros é proposto. O AutoHyperTuningSK é um novo método de aprendizado de máquina automatizado (AutoML). Essa abordagem utiliza técnicas estatísticas para recomendação de hiperparâmetros e apresenta avanços em relação ao algoritmo HyperTuningSK (Capítulo 5). Para avaliar o desempenho do método proposto, foi utilizado o estudo de caso de classificação de rachaduras em construções.

6.1 Método AutoHyperTuningSK

AutoHyperTuningSK é um método de AutoML que realiza a integração automática entre experimentos de *deep learning* para classificação de imagens e ajuste de hiperparâmetros. A Figura 6.1 resume as etapas do AutoHyperTuningSK. A partir da Figura 6.1, cinco fases do processo de recomendação automática de hiperparâmetros são destacadas:

1. *Database*: imagens de entrada para treinamento e validação dos modelos.

2. DL-run(): execução dos experimentos com *deep learning*. Para isso, a arquitetura MobileNet foi adotada [160].

3. Resultados de DL: resultados dos experimentos na classificação de imagens com modelos de *deep learning*.

4. HyperTuningSK(): aplicação do algoritmo HyperTuningSK para análise variância e geração de *rankings* de hiperparâmetros com o método de Scott-Knott [80, 81].

5. Hiperparâmetros: representa os hiperparâmetros e o espaço de busca para otimização. Esses valores são usados em uma nova etapa de ajuste de hiperparâmetros.

Nas próximas subseções, mais detalhes sobre a abordagem para recomendação automática de hiperparâmetros é apresentada. Além disso, duas variações do AutoHypertTuningSK também são propostas: AutoHyperTuningSK-test (experimentos de teste) e recomendação de transformações de *data augmentation* (AutoHyperTuningSK-DA).

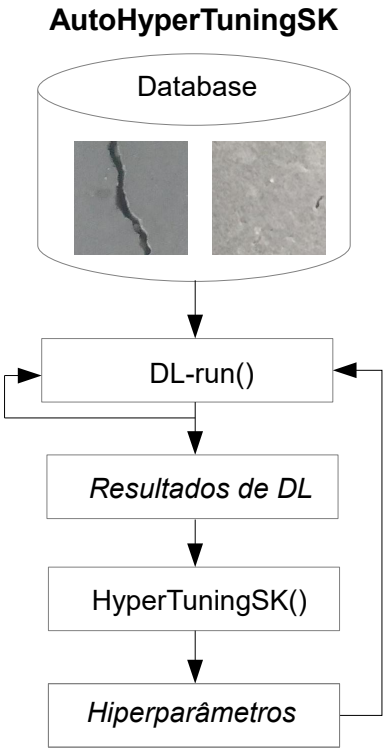


Figura 6.1: AutoHyperTuningSK: ajuste automático de hiperparâmetros na classificação de imagens da construção civil com *deep learning*.

6.1.1 Módulo para *Deep Learning*

Neste estudo, modelos de aprendizado de máquina profundo são usados para classificação binária, por exemplo, classe 0 (negativo: sem rachadura na edificação) e classe 1 (positivo: com rachadura na edificação). Para isso, foi proposta a função DL-run() como um módulo do AutoHyperTuningSK. O Algoritmo 2 apresenta uma sequência de passos do processo de aprendizado de máquina para classificação de imagens. Vale ressaltar a adoção de métodos da biblioteca Keras para adoção da arquitetura, otimização, *data augmentation* e simulação.

```
1 DL-run(){
2       Input: hyperparameters;
3       Set architecture: application_mobilenet();
4       Set optimizer and learning rate: compile();
5       Set data augmentation: image_data_generator();
6       Set directory: flow_images_from_directory();
7       Simulation: fit_generator();
8       Return: results of simulation;
9 }
```

Algoritmo 2: Função para *deep learning*. Fonte: Baseado em [94].

A estrutura de rede neural usada foi a arquitetura MobileNet [160], disponível no *framework* Keras como *application_mobilenet()* [94]. MobileNet é uma rede neural convolucional para aplicações móveis e embarcadas de visão computacional. O trabalho de [160] demonstrou a eficiência da MobileNet em várias aplicações, como, detecção de objetos e reconhecimento facial. Além disso, essa arquitetura é frequentemente adotada na literatura [191, 192, 193, 194,

195].

Assim, a arquitetura MobileNet foi selecionada pelos relevantes resultados na literatura e também pelo baixo custo computacional. Nesse sentido, a Tabela 6.1 compara o tempo computacional médio da estrutura MobileNet em relação a outras duas estruturas (VGG16 e Densenet). Para isso, a tarefa de recomendação automática de hiperparâmetros para o estudo de caso 4 (classificação de rachaduras) é observada no ajuste de 55 modelos (10 épocas) para cada arquitetura. É possível observar que a arquitetura MobileNet obteve um tempo total de 11.550 segundos (210 seg. por modelo em média). Por outro lado, as estruturas VGG16 e Densenet alcançaram tempos computacionais muito superiores, respectivamente 25.850 segundos (+123,8%) e 36.850 segundos (+219,0%). Dessa forma, esses resultados indicam que a arquitetura MobileNet apresenta condições eficientes para uma execução com baixo custo computacional no processo de ajuste automático de hiperparâmetros em relação a outras estruturas da literatura analisadas.

Tabela 6.1: Comparação entre os tempos computacionais médios de arquiteturas de *deep learning* para recomendação automática de hiperparâmetros. Parâmetros: número de pesos na rede neural artificial. Diferença: diferença percentual entre os tempos totais.

Arquitetura	**Paramêtros**	**Tempo por modelo (s)**	**Total (s)**	**Diferença** (%)
Mobilenet	3.754.177	210	11.550	–
VGG16	14.977.857	470	25.850	+123,8
Densenet	7.562.817	670	36.850	+219,0

As dimensões das imagens para a camada de entrada foram padronizadas como: $50 \times 50 \times 3$. Além disso, foram adicionadas duas camadas totalmente conectadas a estrutura para classificação. A última camada contém o neurônio classificador binário com função de ativação *sigmoid*.

6.1.2 Módulo para o HyperTuningSK

O método proposto para ajuste automático de hiperparâmetros utiliza um módulo do HyperTuningSK, apresentado no Capítulo 5. Conforme descrito anteriormente, o HyperTuningSK adota métodos estatísticos para recomendação de hiperparâmetros, como ANOVA [79] e o método clusterização de Scott-Knott [80]. Para isso, o Algortimo 3 apresenta o HyperTuningSK na forma de função para adoção como módulo do AutoHyperTuningSK.

Dessa forma, a função HyperTuningSK para recomendação de hiperparâmetros é aplicada em duas fases: análise de variância (linhas 3 até 18) e geração de *ranking* de hiperparâmetros usando o método de Scott–Knott (linhas 19 até 31) [80, 81]. Além disso, a função apresenta uma modificação na recomendação de hiperparâmetros, em relação ao algoritmo proposto no Capítulo 5. Nesse aspecto, a geração de valores aleatórios de hiperparâmetros foi adicionada na linha 29 (*hsk* $\leftarrow$ *rand()*), quando não existe diferença entre os tratamentos ou modelo não está adequado. Essa alteração é importante para manter a dinâmica do processo de ajuste automático de hiperparâmetros.

6.1.3 Algoritmo AutoHyperTuningSK

Nesta seção, o algoritmo AutoHyperTuningSK é proposto, conforme Algoritmo 4. Esse algoritmo de AutoML reúne conceitos da execução de *deep learning* (módulo DL-run) e recomendação de hiperpararâmetros (função HyperTuningSK). O objetivo é ajustar dois hiperparâmetros (otimizador e taxa de aprendizado) em estudos de casos da classificação de imagens da construção civil.

```
1  HyperTuningSK(){
2      Input: hyperparameters; results of simulations;
3      /*Stage 1: Analysis of Variance*/
4      Adjust the ANOVA model: aov() and anova();
5      Calculate p-value: p;
6      Residual normality test (p_ks): ks.test();
7      Homogeneity of variances (p_bt): bartllet.test();
8      H_1 ← 0;
9      if (p_ks < 0.05) or (p_bt < 0.05) {
10         print("Not adequate model");
11     }else{
12         if (p < 0.05) {
13             H_1 ← 1;
14             print("ANOVA: H1 confirmed");
15         }else{
16             print("ANOVA: H0 confirmed");
17         }
18     }
19     /*Stage 2: Hyperparameter Tuning Ranking*/
20     if (H_1 == 1){
21         print(p, p_ks, p_bt);
22         /*Ranking with Scott-Knott method */
23         ScottKnott clustering algorithm: SK();
24         /*Recommended hyperparameters: */
25         hsk ← Group A from SK();
26         print("Recommended hyperparameters:"hsk) ;
27     }else{
28         print("No difference or not adequate model") ;
29         hsk ← rand();
30     }
31     Return: hsk;
32 }
```

Algoritmo 3: Módulo do HyperTuningSK.

```
1  AutoHyperTuningSK(){
2      Libraries: keras (2.3.0.0) and
3                 ScottKnott (1.2-7);
4      Dataset directories: train and validation;
5      run ← number of repetitions;
6      /*Stage 1: Tuning of optimization methods*/
7      hyper ←optimizers;
8      length_opt ←number of optimizers;
9      for k in 1 : length_opt
10         for z in 1 : run
11             results[k,z] ←DL-run(arguments);
12         }
13     }
14     hskopt ←HyperTuningSK(hyper, results);
15     /*Stage 2: Tuning of learning rate values */
16     hyper1 ←random learning rate values;
17     length_lr1 ←number of hyper1 values;
18     for k in 1 : length_lr1
19         for z in 1 : run
20             results1[k,z] ←DL-run(arguments);
21         }
22     }
23     hsk1 ←HyperTuningSK(hyper1, results1);
24     /*Stage 3: Tuning in local search region (lsr)*/
25     lsr ←[min(hsk1)−0.001,max(hsk1)+0.001];
26     hyper2 ←random values in lsr;
27     length_lr2 ←number of hyper2 values;
28     for k in 1 : length_lr2
29         for z in 1 : run
30             results2[k,z] ←DL-run(arguments);
31         }
32     }
33     hsk2 ←HyperTuningSK(hyper2, results2);
34     /*Results*/
35     print("Stage 1:", hskopt);
36     print("Stage 2:", hsk1);
37     print("Stage 3:", hsk2);
38 }
```

Algoritmo 4: AutoHyperTuningSK.

Inicialmente, as bibliotecas (Keras e ScottKnott) e o endereço do conjunto de dados (treino e validação) são declarados (linhas 2 até 5). Além disso, o número de repetições (run) também fixados. Nos experimentos desta tese, essa variável foi padronizada em cinco repetições. Na sequência, o AutoHyperTuningSK é executado em três estágios:

1. Recomendação de métodos de otimização (linhas 6 até 14).

2. Recomendação de valores de taxa de aprendizado (linhas 15 até 23).

3. Recomendação de valores de taxa de aprendizado em busca local (linhas 24 até 33).

A primeira etapa do AutoHyperTuningSK refere-se à recomendação de otimizadores. Foram definidos dois otimizadores da literatura para seleção: adagrad [97] e adadelta [96]. Nas linhas 9 até 13, experimentos de *deep learning* para classificação de imagens são realizados com os dois otimizadores, usando a função DL-run(). Posteriormente, os resultados dessas simulações são usadas pela função do HyperTuningSK (linha 14) na recomendação dos otimizadores. A variável $hskopt$ armazena os hiperparâmetros selecionados para serem aplicados na segunda fase do AutoHyperTuningSK.

O segundo estágio refere-se ao ajuste da taxa de aprendizado. Para isso, na linha 16, valores aleatórios de taxa de aprendizado são gerados a partir de seis intervalos pré-definidos [min, máx]:

1. $[0{,}001; 0{,}005]$;

2. $[0{,}005; 0{,}010]$;

3. $[0{,}010; 0{,}015]$;

4. $[0{,}015; 0{,}020]$;

5. $[0{,}020; 0{,}025]$;

6. $[0{,}025; 0{,}030]$.

Assim, uma nova fase de experimentos de *deep learning* são executados com a função DL-run() (linhas 18 até 22). Na linha 23, as taxas de aprendizado geradas são avaliadas pelo HyperTuningSK e recomendadas para o terceiro estágio de execução do AutoHyperTuningSK.

Na terceira fase, é realizado ajuste da taxa de aprendizado em busca local (linhas 24 até 33). O objetivo é otimizar os hiperparâmetros na região de valores recomendados na segunda etapa do AutoHyperTuningSK. Além disso, foi definido um valor de incremento de $0{,}001$ para aumentar a região de busca. Por exemplo, se na segunda etapa, quatro valores são recomendados ($0{,}014$; $0{,}018$; $0{,}024$; $0{,}028$), então a região de busca será: [min($hsk1$)$-0{,}001$; max($hsk1$)$+0{,}001$] = [$0{,}014 - 0{,}001$; $0{,}028 + 0{,}001$] = [$0{,}013$; $0{,}029$]. Posteriormente, são gerados valores aleatórios de taxa de aprendizado usando a região de busca local (linha 26). Nas linhas 28 até 32, novos experimentos de *deep learning* são executados usando a função DL-run(). Na sequência, os novos valores de hiperparâmetros são recomendados usando novamente o módulo HyperTuningSK() (linha 33).

Finalmente, os resultados de ajuste automático de hiperparâmetros para classificação de imagens usando *deep learning* são apresentados nas linhas 34 até 37. Assim, os hiperparâmetros recomendados (otimizador e taxa de aprendizado) podem ser aplicados no conjunto de imagens de teste, como descrito na próxima seção.

6.1.4 Algoritmo AutoHyperTuningSK-test

Nesta seção, o algoritmo AutoHyperTuningSK-test é proposto, conforme Algoritmo 5. O objetivo deste método é realizar experimentos de teste com os hiperparâmetros recomendados pelo AutoHyperTuningSK.

```
1  AutoHyperTuningSK-test(){
2      Libraries: keras (2.3.0.0) and
3                 ScottKnott (1.2-7);
4      Dataset directories: train, validation and test;
5      run ← number of repetitions;
6      /*Test of hyperparameters*/
7      hyper ←AutoHyperTuningSK();
8      length_hyper ←number of hyper;
9      for k in 1 : length_hyper
10         for z in 1 : run
11             results[k,z] ←DL-run(arguments);
12         }
13     }
14     hsk_test ←HyperTuningSK(hyper, results);
15     /*Results*/
16     print("Test:", hsk_test);
17 }
```

Algoritmo 5: AutoHyperTuningSK-test.

O algoritmo AutoHyperTuningSK-test executa simulações de *deep learning* no conjunto de imagens de teste usando a função DL-run (linhas 9 até 13). Assim, a acurácia no teste é adotada pelo método HyperTuningSK para realizar a recomendação de hiperparâmetros (linha 14). Finalmente, os resultados do ajuste de hiperparâmetros no conjunto de teste é mostrado na linha 16.

6.1.5 Algoritmo AutoHyperTuningSK-DA

Neste estudo, uma versão do AutoHyperTuningSK também foi proposta para ajuste de hiperparâmetros de *data augmentation*, denominado AutoHyperTuningSK-DA (Algoritmo 6). Conforme descrito anteriormente, *data augmentation* é uma abordagem para gerar mais dados artificialmente a partir de transformações aleatórias nas imagens originais [94, 69]. Para isso, foi usado o método *image_data_generator()* a partir da biblioteca Keras (versão 2.3.0.0) [94]. As seguintes transformações foram ajustadas na geração de imagens artificiais:

- *height_shift_range*: He.
- *shear_range*: S.
- *width_shift_range*: W.
- *zoom_range*: Z.

O AutoHyperTuningSK-DA é executado em dois estágios: (1) recomendação de valores de *data augmentation*; (2) recomendação de valores de *data augmentation* em busca local.

Na primeira etapa, valores aleatórios são gerados para cada transformação de *data augmentation* (lines 7 to 10): *height shift* (He), *shear* (S), *width shift* (W) and *zoom* (Z). Para isso,

```
AutoHyperTuningSK-DA(){
    Libraries: keras (2.3.0.0) and
                ScottKnott (1.2-7);
    Dataset directories: train and validation;
    run ← number of repetitions;
    /*Stage 1: Tuning of data augmentation values */
    He ←random values;
    S ←random values;
    W ←random values;
    Z ←random values;
    length_1 ←number of (He, S, W, Z) values;
    for k in 1 : length_1
        for z in 1 : run
            res1[k,z] ←DL-run(arguments);
        }
    }
    hsk1 ←HyperTuningSK(He, S, W, Z, res1);
    /*Stage 2: Tuning in local search region (lsr)*/
    lsr ←local search region from hsk1;
    He_2 ←random values in lsr;
    S_2 ←random values in lsr;
    W_2 ←random values in lsr;
    Z_2 ←random values in lsr;
    length_2 ←number of (He_2, S_2, W_2, Z_2) values;
    for k in 1 : length_2
        for z in 1 : run
            res2[k,z] ←DL-run(arguments);
        }
    }
    hsk2 ←HyperTuningSK(He_2, S_2, W_2, Z_2, res2);
    /*Results*/
    print("Stage 1:", hsk1);
    print("Stage 2:", hsk2);
}
```

Algoritmo 6: AutoHyperTuningSK-DA.

quatro intervalos foram definidos [min, max]: $[0{,}000; 0{,}250]$; $[0{,}250; 0{,}500]$; $[0{,}500; 0{,}750]$ e $[0{,}750; 1{,}000]$.

Além disso, uma configuração sem transformações também é analisada: $He = 0{,}000$, $S = 0{,}000$, $W = 0{,}000$ e $Z = 0{,}000$. Dessa forma, os experimentos de aprendizado de máquina são realizados com essas diferentes configurações de *data augmentation* (linhas 12 até 16). Posteriormente, o método HyperTuningSK é aplicado no ajuste de hiperparâmetros na linha 17. Assim, os valores de hiperparâmetros selecionados são adotados na segunda fase.

No segundo estágio, uma busca local é realizada na região de valores de hiperparâmetros recomendados no primeira etapa de execução do AutoHyperTuningSK-DA. Para isso, novos valores aleatórios são gerados com variação de até 20% em relação aos hiperparâmetros recomendados no primeira fase (linhas 20 até 23). Na sequência, novos experimentos de *deep learning* na classificação de imagens são realizados DL-run (linhas 25 até 29). Assim, os resultados são novamente avaliados pelo HyperTuningSK (linha 30). Finalmente, as recomendações de hiperparâmetros de *data augmentation* (fases 1 e 2) são apresentados nas linhas 32 e 33.

6.2 Resultados para o Estudo de Caso 4: Classificação de Rachaduras

Nesta seção, são apresentados os resultados da aplicação do AutoHyperTuningSK no ajuste automático de hiperparâmetros no estudo de caso 4 (classificação de rachaduras em edificações) nas seguintes subseções: (i) recomendação de otimizador e taxa de aprendizado; (ii) recomendação de hiperparâmetros de *data augmentation*; (iii) análise de tempo computacional; e (iv) comparação com outros métodos.

6.2.1 Recomendação de Otimizador e Taxa de Aprendizado

Os resultados do ajuste automático de hiperparâmetros (otimizador e taxa de aprendizado) são apresentados de acordo com as etapas do AutoHyperTuningSK:

1. Recomendação de métodos de otimização.
2. Recomendação de valores de taxa de aprendizado.
3. Recomendação de valores de taxa de aprendizado em busca local.

A Tabela 6.2 mostra os resultados da recomendação de otimizadores no primeiro estágio.

Tabela 6.2: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK na primeira etapa (recomendação de métodos de otimização). Média: média de acurácia na validação (%). Recomendação: hiperparâmetros recomendados para o próximo estágio.

	Grupo	**Otimizador**	**Média**	**Recomendação**
Ranking	A	adagrad	69,2	✓
	B	adadelta	60,7	–
Medidas estatísticas	p_{ks} 0,716	p_{bt} 0,122	p <0,001	

Observa-se na Tabela 6.2 que as medidas de adequação foram satisfeitas: normalidade dos resíduos ($p_{ks} > 0{,}05$) e homogeneidade das variâncias ($p_{bt} > 0{,}05$). Além disso, a hipótese inicial da ANOVA foi rejeitada ($p < 0{,}05$). Assim, a análise de variância indica que existe diferença estatística entre os desempenhos (acurácia) dos métodos de otimização (hipótese alternativa aceita). Além disso, é possível observar que o algoritmo AutoHyperTuningSK distribuiu os otimizadores em dois grupos: A e B. No grupo A está o tratamento recomendado (adagrad:

$69{,}2\%$), ou seja, o hiperparâmetro que alcançou a maior média de acurácia na validação. Por outro lado, no grupo B está o otimizador com o pior desempenho médio: adadelta ($60{,}7\%$). Nesse aspecto, o otimizador adagrad foi selecionado pelo AutoHyperTuningSK para a segunda etapa de ajuste de hiperparâmetros.

Os resultados para a segunda etapa de ajuste automático de hiperparâmetros (recomendação de taxa de aprendizado) são apresentados na Tabela 6.3. Pode-se observar que as medidas de adequação estatísticas foram satisfeitas: $p_{ks} > 0{,}05$, $p_{bt} > 0{,}05$ e $p < 0{,}05$. Além disso, seis valores aleatórios para taxa de aprendizado foram gerados. Nesse aspecto, dois tratamentos não foram recomendados (grupo B): $0{,}00542$ ($65{,}6\%$) e $0{,}00367$ ($64{,}0\%$). Em contrapartida, quatro valores de taxa de aprendizado foram selecionados no grupo A: $0{,}02833$ ($72{,}7\%$), $0{,}02443$ ($70{,}9\%$), $0{,}01457$ ($69{,}5\%$) e $0{,}01892$ ($69{,}2\%$). Então o intervalo de lr = $[0{,}01457; 0{,}02833]$ foi recomendado para busca local no próximo estágio.

Tabela 6.3: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK na segunda etapa (recomendação de taxa de aprendizado). Média: média de acurácia na validação (%). Recomendação: hiperparâmetros recomendados para o próximo estágio.

	Grupo	**lr**	**Média**	**Recomendação**
Ranking	A	0,02833	72,7	✓
	A	0,02443	70,9	✓
	A	0,01457	69,5	✓
	A	0,01892	69,2	✓
	B	0,00542	65,6	–
	B	0,00367	64,0	–
Medidas estatísticas	p_{ks} 0,816	p_{bt} 0,631	p <0,001	

A Tabela 6.4 apresenta os resultados do AutoHyperTuning na execução no terceiro estágio. Nesta fase, é realizado o ajuste de valores de taxa de aprendizado em busca local, gerando três valores aleatórios na região intervalar recomendada na segunda etapa: lr = $[0{,}01457 - 0{,}001;$ $0{,}02833 + 0{,}001]$ = $[0{,}01357; 0{,}02933]$. Novamente as medidas de adequação estatísticas foram satisfeitas ($p_{ks} > 0.05$ e $p_{bt} > 0.05$). Por outro lado, a hipótese inicial (H_0) da ANOVA não foi rejeitada, indicando que não existe diferença estatística significativa entre as taxas de aprendizado nessa etapa, ou seja, todos os tratamentos foram agrupados em A: $0{,}02029$ ($71{,}7\%$), $0{,}02220$ ($71{,}3\%$) e $0{,}01560$ ($70{,}8\%$). Assim, o algoritmo AutoHyperTuningSK realizou uma recomendação aleatória entre os valores candidatos nessa etapa, sendo sorteado o valor de $0{,}02220$.

Tabela 6.4: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK na terceira etapa (recomendação de taxa de aprendizado na região de busca local). Média: média de acurácia na validação (%). Recomendação: hiperparâmetros recomendados para o próximo estágio.

	Grupo	**lr**	**Média**	**Recomendação**
Ranking	A	0,02029	71,7	–
	A	0,02220	71,3	✓
	A	0,01560	70,8	–
Medidas estatísticas	p_{ks} 0,998	p_{bt} 0,340	p 0,882	

Dessa forma, o AutoHyperTuningSK recomendou os seguintes valores de hiperparâmetros para o estudo de caso de classifcação de rachaduras: otimizador adagrad e taxa de aprendizado de $0{,}02220$.

6.2.2 Resultados dos Testes: Otimizador e Taxa de Aprendizado

Nesta seção, são apresentados os resultados da etapa de testes dos hiperparâmetros recomendados (otimizador e taxa de aprendizado) para o estudo de caso de classificação de imagens de rachaduras. Para isso, é analisada o desempenho dos tratamento selecionados pelo algoritmo AutoHyperTuningSK: adagrad e lr = $0{,}02220$. Além disso, para fins comparativos, dois valores de taxa de aprendizado não recomendados no estágio 2 também são testados (veja grupo B na Tabela 6.3): $0{,}00542$ e $0{,}00367$.

A Tabela 6.5 mostra os resultados da recomendação de hiperparâmetros usando o algoritmo AutoHyperTuningSK-test no conjunto de dados de teste. Esses resultados reforçam o bom desempenho dos hiperparâmetros selecionados também nas imagens de teste. Nesse sentido, a análise de variância confirmou que existe diferença significativa entre os três valores de taxa de aprendizado simulados ($p < 0{,}05$). Destaca-se também que as medidas de adequação estatísticas foram satisfeitas ($p_{ks} > 0{,}05$ e $p_{bt} > 0{,}05$). A taxa de aprendizado de $0{,}02220$ alcançou a maior média de acurácia ($99{,}1\%$) e foi recomendada pelo algoritmo AutoHyperTuningSK-test. Em contraste, os outros dois valores de taxa de aprendizado analisados foram novamente alocados no grupo B: $0{,}00542$ ($97{,}2\%$) e $0{,}00367$ ($96{,}8\%$).

Tabela 6.5: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK-test nos experimentos de teste (otimizadores e taxa de aprendizado). Média: média de acurácia na validação (%). Recomendação: hiperparâmetros recomendados na etapa de teste.

	Grupo	**lr**	**Média**	**Recomendação**
Ranking	A	0,02220	99,1	✓
	B	0,00542	97,2	–
	B	0,00367	96,8	–
Medidas estatísticas	p_{ks} 0,520	p_{bt} 0,198	p 0,006	

Os resultados da Tabela 6.6 reforçam o indicativo de bom desempenho dos hiperparâmetros recomendados na classificação de imagens no conjunto de testes. A taxa de aprendizado selecionada pelo algoritmo AutoHyperTuningSK alcançou a acurácia máxima dos testes: $99{,}48\%$. Essa porcentagem é equivalente a classificação correta de 3.979 imagens (total de 4.000) na base de dados de teste do quarto estudo de caso. Por outro lado, os hiperparâmetros não recomendados alcançaram apenas $98{,}58\%$ (3.943 imagens) e $97{,}78\%$ (3.911 imagens) de acurácia máxima no testes.

Tabela 6.6: Resultados de acurácia (%) nos testes para cada repetição (1 ... 5) de hiperparâmetro analisado (valor de taxa de aprendizado). Max.: acurácia máxima nos testes (%).

lr	**1**	**2**	**3**	**4**	**5**	**Max.**
0,02220	99,30	98,25	99,35	99,15	99,48	**99,48**
0,00542	97,88	98,58	98,05	95,68	95,83	98,58
0,00367	95,35	96,70	96,80	97,25	97,78	97,78

6.2.3 Recomendação de Hiperparâmetros de *Data Augmentation*

Os resultados da recomendação de hiperparâmetros de *data augmentation* são apresentados na seguinte sequência: (i) recomendação de valores de *data augmentation* no primeiro estágio do AutoHyperTuningSK-DA; (ii) recomendação de hiperparâmetros de *data augmentation* em busca local no segundo estágio do AutoHyperTuningSK-DA; e (iii) experimentos de teste.

A Tabela 6.7 mostra os resultados do ajuste automático de hiperparâmetros de *data augmentation* no primeiro estágio. Ressalta-se que as medidas de adequação estatísticas foram satisfeitas: $p_{ks} = 0{,}952$ (normalidade dos resíduos) e $p_{bt} = 0{,}882$ (homogeneidade das variâncias). Além disso, a análise de variância indica que existe diferença estatística significante entre os desempenhos das configurações analisadas ($p < 0{,}001$). Nesse aspecto, o algoritmo AutoHyperTuningSK-DA distribuiu os hiperparâmetros em dois grupos: A e B. No grupo B está a combinação com pior desempenho nesta etapa ($64{,}1\%$): (5) He = $0{,}835$, S = $0{,}755$, W = $0{,}885$, Z = $0{,}832$. Por outro lado, as configurações recomendadas estão no grupo A: (2) He = $0{,}160$, S = $0{,}012$, W = $0{,}193$, Z = $0{,}096$; (3) He = $0{,}423$, S = $0{,}389$, W = $0{,}442$, Z = $0{,}277$; (1) He = $0{,}000$, S = $0{,}000$, W = $0{,}000$, Z = $0{,}000$; e (4) He = $0{,}571$, S = $0{,}555$, W = $0{,}711$, Z = $0{,}671$. Assim, quatro combinações foram selecionadas pelo AutoHyperTuningSK-DA para o segundo estágio de ajuste automático de hiperparâmetros de *data augmentation*.

Tabela 6.7: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK-DA no primeiro estágio. Comb.: combinação de *data augmentation*. Média: média de acurácia na validação (%). Recom.: hiperparâmetros recomendados para a próxima etapa.

	Grupo	**Comb.**	**He**	**S**	**W**	**Z**	**Média**	**Recom.**
Ranking	A	2	0,160	0,012	0,193	0,096	71,5	✓
	A	3	0,423	0,389	0,442	0,277	71,2	✓
	A	1	0,000	0,000	0,000	0,000	70,8	✓
	A	4	0,571	0,550	0,711	0,671	68,7	✓
	B	5	0,835	0,755	0,885	0,832	64,1	–
Medidas estatísticas		p_{ks} 0,952	p_{bt} 0,882	p <0,001				

A Tabela 6.8 mostra os resultados da segunda etapa de execução do AutoHyperTuningSK-DA. Nessa linha, o ajuste de hiperparâmetros de *data augmentation* na região de busca local. Para isso, quatro novas configurações de *data augmentation* são geradas a partir das combinações de hiperparâmetros selecionadas na primeira etapa, sendo: (2) $\rightarrow$ (6), (3) $\rightarrow$ (7), (1) $\rightarrow$ (8) and (4) $\rightarrow$ (9). Por exemplo, a configuração 6 (He = $0{,}179$, S = $0{,}013$, W = $0{,}215$, Z = $0{,}105$) foi gerada na região local da combinação 2 (He = $0{,}160$, S = $0{,}012$, W = $0{,}193$, Z = $0{,}096$).

Tabela 6.8: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK-DA no segundo estágio (busca local). Comb.: combinação de *data augmentation*. Média: média de acurácia na validação (%). Recom.: hiperparâmetros recomendados.

	Grupo	**Comb.**	**He**	**S**	**W**	**Z**	**Média**	**Recom.**
Ranking	A	6	0,179	0,013	0,215	0,105	71,0	✓
	A	7	0,454	0,402	0,526	0,279	70,4	✓
	A	8	0,000	0,000	0,000	0,000	69,7	✓
	B	9	0,602	0,621	0,828	0,770	65,8	–
Medidas estatísticas		p_{ks} 0,707	p_{bt} 0,209	p 0,046				

Os resultados da 6.8 mostram que existe diferença significativa entre as configurações ($p = 0{,}019$) e as medidas estatísticas foram adequadas ($p_{ks} = 0{,}707$ e $p_{bt} = 0{,}209$). Assim, as configurações de *data augmentation* recomendas foram: (6) He = $0{,}179$, S = $0{,}013$, W = $0{,}215$, Z = $0{,}105$; (7) He = $0{,}454$, S = $0{,}402$, W = $0{,}526$, Z = $0{,}279$; e (8) He = $0{,}000$, S = $0{,}000$, W = $0{,}000$, Z = $0{,}000$.

Os resultados dos experimentos de teste são apresentados na Tabela 6.9. Ressalta-se que nesta etapa, os desempenhos de duas configurações de *data augmentation* foram comparados:

- Combinação 6 (recomendada, conforme Tabela 6.8): He = 0,179, S = 0,013, W = 0,215, Z = 0,105.
- Combinação 5 (não recomendada, conforme Tabela 6.7): He = 0,835, S = 0,755, W = 0,885, Z = 0,832.

Tabela 6.9: Resultados do ajuste automático de hiperparâmetros pelo AutoHyperTuningSK-DA nos experimentos de testes. Comb.: combinação de *data augmentation*. Média: média de acurácia na validação (%). Recom.: hiperparâmetros recomendados.

	Grupo	**Comb.**	**He**	**S**	**W**	**Z**	**Média**	**Recom.**
Ranking	A	6	0,179	0,013	0,215	0,105	99,2	✓
	B	5	0,835	0,755	0,885	0,832	96,9	–
Medidas estatísticas		p_{ks} 0,828	p_{bt} 0,002	p 0,019				

Nesse aspecto, a combinação 6 alcançou a maior média de acurácia nos testes: 99,2%. Por outro lado, a configuração não recomendada obteve apenas 96,9%. Assim, esses resultados reforçam a eficiência do algoritmo AutoHyperTuningSK-DA no ajuste de hiperparâmetros de *data augmentation*.

A Figura 6.2 apresenta a acurácia obtida ao longo do processo de treinamento e também reforça as diferenças de desempenho dessas duas configurações de *data augmentation*. Pode-se observar que a combinação 6 obteve convergência em torno da oitava época, obtendo resultados de aproximadamente 99,0% (acurácia na validação). Em contraste, a validação com a configuração 5 (não recomendada) não alcançou convergência em 20 épocas, variando de 75,5% até 98,0% nas últimas cinco épocas.

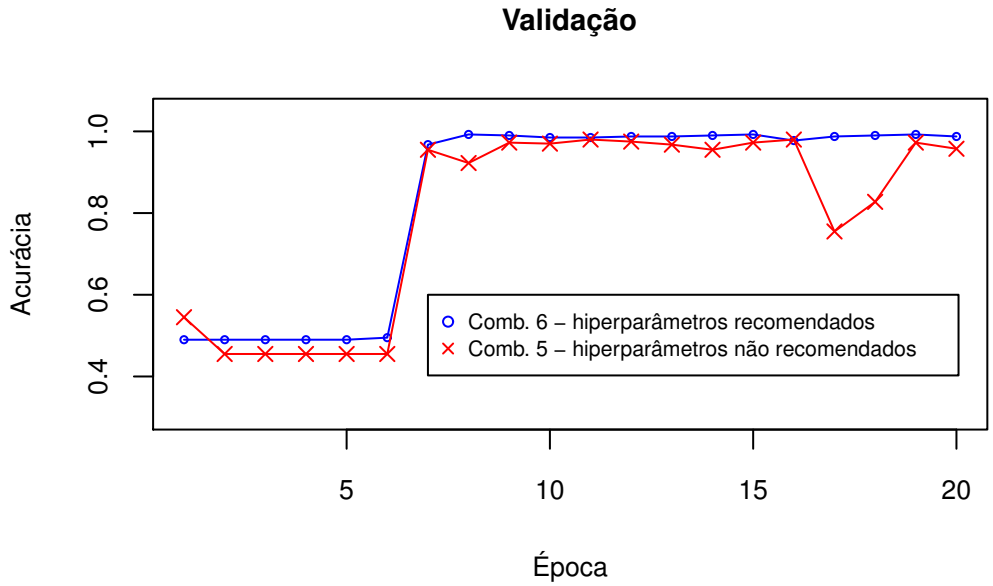


Figura 6.2: Acurácia na etapa de validação (20 épocas) para duas configurações de *data augmentation*: 6 (hiperparâmetros recomendados) e 5 (hiperparâmetros não recomendados).

Outros exemplos da relevância do ajuste de hiperparâmetros de *data augmentation* com AutoHyperTuningSK-DA são apresentados na Figuras 6.3 e 6.4.

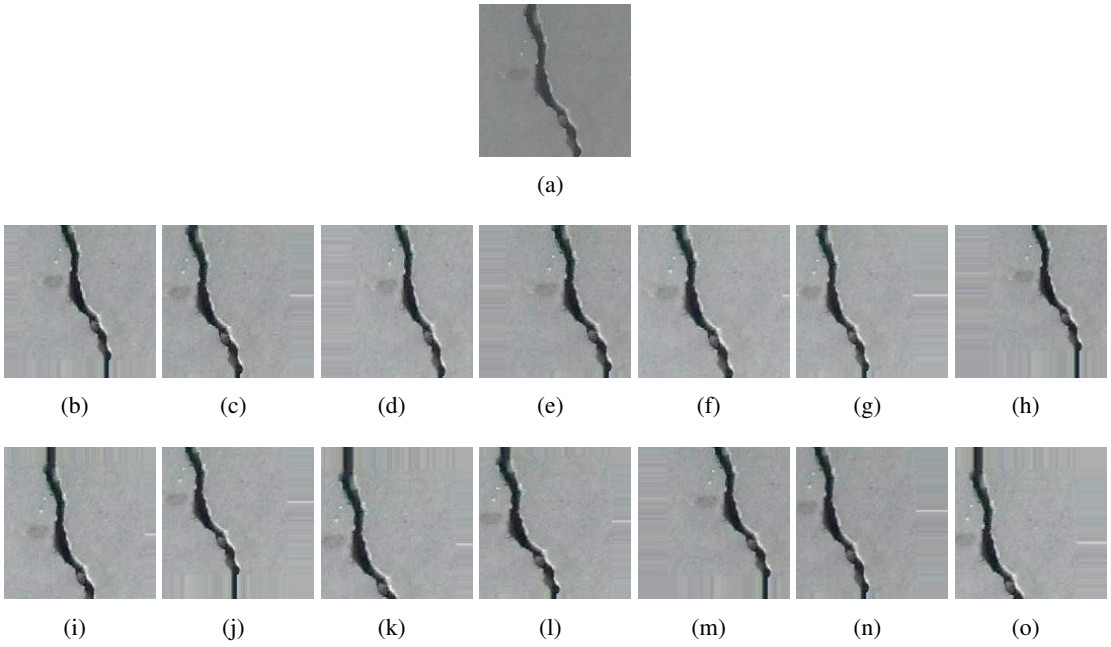


Figura 6.3: Exemplos de imagens geradas com combinações recomendadas de *data augmentation* (combinação 6): He = 0,179; S = 0,013; W = 0,215; Z = 0,105. (a) Imagem original (Fonte: [184]). (b) Imagens artificiais com *data augmentation*.

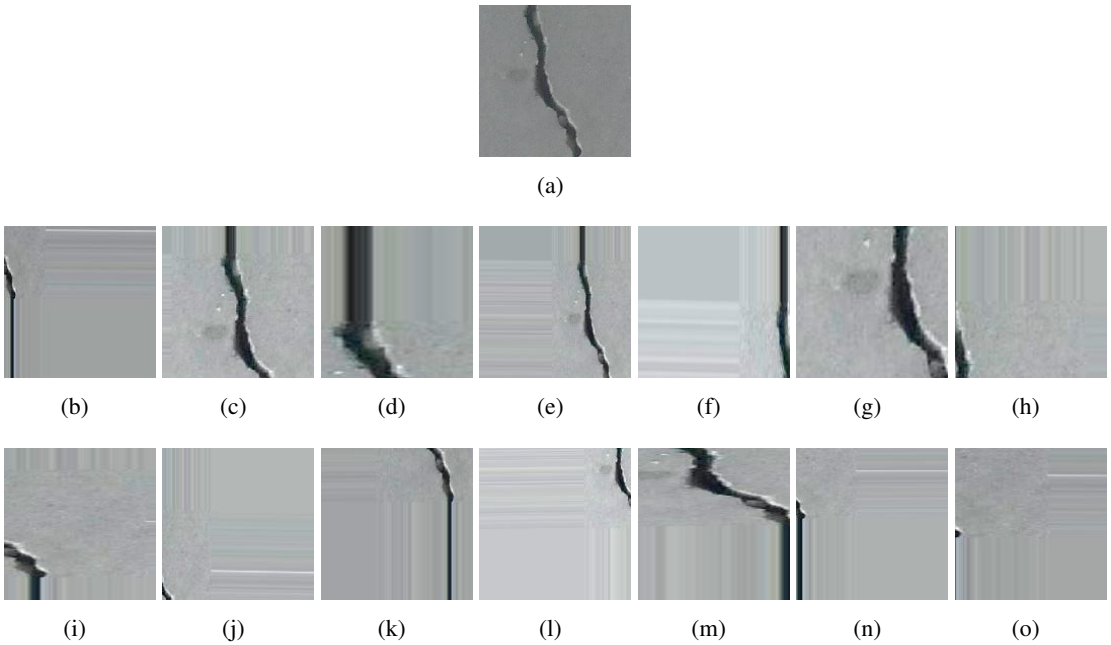


Figura 6.4: Exemplos de imagens geradas com combinações não recomendadas de *data augmentation* (combinação 5): He = 0,835; S = 0,755; W = 0,885; Z = 0,832. (a) Imagem original (Fonte: [184]). (b) Imagens artificiais com *data augmentation*.

Nesse aspecto, a Figura 6.3 mostra exemplos de imagens geradas com as transformações recomendadas (combinação 6): He = 0,179; S = 0,013; W = 0,215; Z = 0,105. Observa-se que as novas imagens geradas artificialmente (Figura 6.3: (b) até (o)) apresentam alterações em relação à fotografia original (Figura 6.3: (a)), porém, visualmente mantém o mesmo rótulo da imagem inicial, ou seja, classe positiva (com rachadura). No entanto, as imagens geradas com hiperparâmetros não recomendados (combinação 5) revelam os efeitos da aplicação de configurações

inadequadas de *data augmentation*, conforme Figura 6.4. Por exemplo, em algumas imagens geradas (ex.: Figura 6.4: (j) e (o)), o rótulo original (com rachadura) não é mais evidente. Assim, o desempenho do processo de treinamento com a adoção dessas imagens artificiais pode ficar comprometido, tendo em vista que parte dos novos dados sintéticos não corresponde à classe original.

6.2.4 Análise de Tempo Computacional

A Tabela 6.10 apresenta os tempos computacionais necessários para execução de cada algoritmo proposto neste capítulo e aplicados na recomendação de hiperparâmetros para o quarto estudo de caso (classificação de rachaduras).

Tabela 6.10: Tempos computacionais para execução de cada algoritmo proposto no quarto estudo de caso: AutoHyperTuningSK, AutoHyperTuningSK-test e AutoHyperTuningSK-DA. Média: média de tempo para três execuções (t_1, t_2 e t_3) em **horas (h)**.

Algoritmo	t_1	t_2	t_3	**Média**
AutoHyperTuningSK	5,58	5,19	4,45	5,07
AutoHyperTuningSK-test	2,07	2,28	2,51	2,29
AutoHyperTuningSK-DA	2,90	3,57	2,29	2,92

Percebe-se na Tabela 6.10 que o AutoHyperTuningSK é o algoritmo proposto neste capítulo com maior custo computacional, tendo obtido a média de processamento de 5,07 horas (5 h e 4 min). Por outro lado, os algoritmos AutoHyperTuningSK-test e AutoHyperTuningSK-DA alcançaram médias de 2,29 horas (2 h e 17 min) e 2,92 horas (2 h e 55 min), respectivamente.

A Tabela 6.11, por sua vez, apresenta os tempos computacionais médios para execução de testes para classificação de imagens após o treinamento e recomendação de hiperparâmetros no quarto estudo de caso.

Tabela 6.11: Tempos computacionais para execução de testes de classificação de imagens para o quarto estudo de caso após o treinamento e recomendação de hiperparâmetros. Média: média de tempo para três execuções (t_1, t_2 e t_3) em **segundos (s)**.

Número de imagens	**Porcentagem da base de teste (%)**	t_1	t_2	t_3	**Média**
1	0,025	0,057	0,055	0,057	0,056
40	1	0,100	0,098	0,100	0,099
400	10	0,460	0,469	0,503	0,477
4000	100	4,098	4,000	4,016	4,038

Pode-se observar na Tabela 6.11 que para classificar o número total de imagens (4000) na base de teste foi necessário 4,038 segundos em média usando a função *evaluate_generator()* do Keras. Além disso, ao classificar apenas uma imagem, o tempo médio gasto foi de 0,056 segundos usando o método *predict_classes()* do Keras. A Tabela 6.11 apresenta ainda os tempos computacionais médios para a classificação de 1% (40 imagens) e 10% (400 imagens) na base de teste do estudo de caso 4, sendo esses valores de 0,099 s e 0,477 s, respectivamente.

6.2.5 Comparação com Outros Métodos de AutoML

A Tabela 6.12 apresenta uma comparação entre o método proposto (AutoHyperTuningSK) e outros algoritmos de AutoML: Hyperband [196] and Random Search [52]. Os métodos de otimização de hiperparâmetros Hyperband e Random Search foram aplicados a partir do Keras

Tuner API[1] na linguagem Python. Os resultados desses algoritmos são analisados no ajuste de dois hiperparâmetros (otimizador e taxa de aprendizado) no estudo de caso de classificação de rachaduras.

Tabela 6.12: Comparação do método proposto (AutoHyperTuningSK) com outros algoritmos de AutoML. Otimizador e lr: hiperparâmetros recomendados. Média: média de acurácia na etapa de teste (%). Máximo: valor máximo de acurácia na etapa de teste (%).

Método	**Otimizador**	**lr**	**Média**	**Máximo**
AutoHyperTuningSK	adagrad	0,02220	99,1	99,5
Hyperband	adagrad	0,02100	98,3	99,6
Random Search	adagrad	0,02600	99,2	99,4

Os resultados da Tabela 6.12 mostram que os três métodos analisados recomendaram otimizador adagrad. Além disso, os valores de taxa de aprendizado ajustados foram próximos: $0{,}02220$ (AutoHyperTuningSK), $0{,}02100$ (Hyperband) e $0{,}02600$ (Random Search). Ressalta-se também que os valores de lr selecionados pelos algoritmos de AutoML da literatura estão na região de recomendação na primeira fase de execução do AutoHyperTuningSK, conforme Seção 6.2.1. Assim, esses resultados indicam a consistência do ajuste automático de hiperparâmetros pelo método proposto em relação aos algoritmos da literatura.

Nesse sentido, como esperado, devido as similaridades na recomendação de hiperparâmetros pelos três métodos, a acurácia nos testes é bem similar. Também é importante destacar que os hiperparâmetros recomendados pelo AutoHyperTuningSK e Random Search alcançaram métricas de desempenho (média e máximos) superiores a 99%.

[1] https://keras.io/api/keras_tuner/

CAPÍTULO 7

Conclusões

Neste capítulo, são apresentadas as conclusões desta tese. Nesse aspecto, a primeira seção descreve as considerações finais sobre as propostas e análises da tese. Em seguida, é realizado um estudo comparativo da presente proposta com outros trabalhos da literatura. Por fim, são apontadas as possibilidades para trabalhos futuros.

7.1 Considerações Finais

O objetivo principal desta tese foi desenvolver métodos para recomendação de hiperparâmetros de aprendizado de máquina na classificação de imagens da construção civil. Sendo assim, a principal contribuição desta tese foi a proposta de um método criterioso com Análise de Variância e Scott-Knott *clustering algorithm* para a geração de *rankings* de hiperparâmetros. Para isso, o algoritmo HyperTuningSK é proposto. Outra contribuição desta tese é a proposta de um método para aprendizado de máquina automatizado, denominado AutoHyperTuningSK. Essa técnica realiza a integração automática entre experimentos de aprendizado profundo e ajuste de hiperparâmetros. Assim, a partir dessas abordagens propostas, é possível analisar a influência da definição hiperparâmetros no desempenho de acurácia em tarefas de processamento de imagens da Construção 4.0.

Além disso, pode-se destacar outros tópicos relevantes da presente tese: (i) realização de revisão de literatura sistemática da aplicação *deep learning* na Construção 4.0; (ii) aplicação e análise de arquiteturas de *deep learning* da literatura na classificação de imagens da construção; (iii) análise de quatro distintos estudos de casos da construção civil; (iv) análise de hiperparâmetros de treinamento de CNNs (otimizadores e taxas de aprendizado); (v) planejamento de experimentos com base de dados com poucas imagens; (vi) utilização da técnica de geração de imagens (*data augmentation*) para aumentar a variedade de imagens no *dataset* de treinamento; (vii) análise de 128 combinações de transformações de *data augmentation*; (viii) proposta de índice *HyperScore* para análise dos resultados da recomendação de hiperparâmetros; (ix) proposta de três novos algoritmos para a abordagem de AutoML: AutoHyperTuningSK, AutoHyperTuningSK-test e AutoHyperTuningSK-DA.

Os resultados obtidos pelo método HyperTuningSK demonstraram que os hiperparâmetros analisados influenciaram diretamente no desempenho dos modelos CNN na classificação binária de imagens dos três estudos de casos analisados: (1) reconhecimento de vegetação em fachadas; (2) detecção de patologias em calhas; e (3) classificação de máquinas da construção. Além disso, em geral, os métodos de treinamento recomendados (taxa de aprendizado e otimizador) pelo algoritmo HyperTuningSK alcançaram bons resultados de acurácia em relação à hiperparâmetros adotados na literatura. Também vale ressaltar que as configurações

de hiperparâmetros obtiveram recomendações distintas de acordo com arquitetura de *deep learning* utilizada. Nesse sentido, a combinação adagrad025 alcançou $HyperScore = AAA$ para a arquitetura Densenet121, ou seja, foi agrupado em A em todos os três estudos de casos. Por outro lado, para a arquitetura VGG16, três combinações de hiperparâmetros alcançaram $HyperScore = AA$, sendo adagrad005, sgd010 e adamax001.

Os resultados para análise de *data augmentation*, por sua vez, mostram que duas transformações foram mais recomendadas pelo HyperTuningSK: *Width Shift* e *Heigth Shift*. Cabe ressaltar também que a maioria das configurações de *data augmentation* recomendadas adotam duas ou mais transformações de processamento de imagem na geração dos dados artificiais. Além disso, os resultados do método proposto reforçam a importância de selecionar hiperparâmetros de *data augmentation* de acordo com a base de dados analisada, pois apenas uma configuração (*Width Shift*: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$ e $x_7 = 0$) foi recomendada para pelo menos dois estudos de casos.

Vale ressaltar também os resultados da abordagem de aprendizado de máquina automatizado no quarto estudo de caso: classificação de rachaduras em edificações. Os resultados para o método AutoHyperTuningSK também podem resumidos para o ajuste de dois grupos de hiperparâmetros: (i) otimizador e taxa de aprendizado; e (ii) transformações de *data augmentation*. Nesse sentido, o algoritmo AutoHyperTuningSK recomendou otimizador adagrad e taxa de aprendizado de $0{,}02220$. Essa configuração de hiperparâmetros alcançou acurácia média de $99{,}1\%$ nos testes. Além disso, essa combinação conseguiu acurácia máxima de $99{,}48\%$, ou seja, o equivalente a classificação correta de 3,979 imagens (total de 4,000) no conjunto de teste na base de dados de classificação de rachaduras. Os resultados para recomendação de hiperparâmetros de *data augmentation* também reforçam eficiência do algoritmo AutoHyperTuningSK-DA. Nesse aspecto, a combinação recomendada (He = 0,179; S = 0,013; W = 0,215; Z = 0,105) alcançou acurácia média de $99{,}2\%$ nos experimentos de teste.

7.2 Comparação com Outros Trabalhos

Nesta seção, um estudo comparativo é realizado entre a presente proposta e outros trabalhos da literatura que realizaram análises da seleção de hiperparâmetros. Os estudos foram divididos em dois grupos: aplicação de métodos estatísticos para recomendação de hiperparâmetros (I - [49], II - [84], III - [83]) ou adoção de *deep learning* na construção civil (IV - [38], V - [13], VI - [131] e VII - [47]). Além disso, seis características foram observadas: técnica de aprendizado de máquina, área da aplicação, estudos de casos na construção civil, hiperparâmetros analisados e métodos de recomendação adotados. A Tabela 7.1 apresenta os resultados dessa comparação.

A partir da Tabela 7.1, destaca-se que a maioria dos trabalhos utilizados na comparação utilizaram Redes Neurais Artificiais. Esse fator é importante, pois os métodos de aprendizado de máquina possuem diferentes tipos de hiperparâmetros. Assim, ao analisar estudos que também adotaram modelos neurais, a comparação pode ser mais direta entre as possíveis classes de hiperparâmetros, como de treinamento, regularização e arquitetura.

Outro ponto relevante refere-se aos estudos de casos pelos trabalhos concentrados na área da construção civil. Nesse aspecto, a presente proposta inova em realizar experimentos em quatro distintas aplicações: reconhecimento de vegetação em fachadas, detecção de integridade de calhas, classificação de máquinas e classificação de rachaduras. Geralmente, os estudos correlatos dedicam-se a apenas a uma única utilização, como para detecção de rachaduras [13, 131, 47]. Vale ressaltar que, também foram selecionados estudos de outras áreas, como em saúde [84] e otimização combinatória [49].

A abordagem proposta também destaca-se por sua aplicabilidade em dois grupos de hiperparâmetros de redes neurais: treinamento (taxa de aprendizado e otimizador) e regularização

Tabela 7.1: Comparação da presente proposta com diferentes estudos que realizaram análises da seleção de hiperparâmetros. Os trabalhos estão divididos em dois grupos: aplicação de métodos estatísticos para recomendação de hiperparâmetros (I , II e III) ou adoção de *deep learning* na construção civil (IV a VII).

		Proposta	I [49]	II [84]	III [83]	IV [38]	V [13]	VI [131]	VII [47]
Técnica de Aprendizado de Máquina	Redes Neurais Artificiais	✓	✓	✓		✓	✓	✓	✓
	Outras		✓		✓				
Área da aplicação	Construção civil	✓				✓	✓	✓	✓
	Saúde			✓					
	Otimização combinatória		✓		✓				
Estudos de casos na construção	Reconhecimento de vegetação	✓							
	Detecção de integridade de calhas	✓							
	Classificação de máquinas	✓							
	Classificação/Detecção de rachaduras	✓					✓	✓	✓
	Segurança					✓			
Hiperparâmetros analisados	Taxa de aprendizado	✓						✓	✓
	Otimizador	✓					✓		
	Data Augmentation	✓							
	Arquitetura neural		✓	✓					✓
	Outros		✓		✓	✓			✓
Métodos para recomendação de hiperparâmetros	ANOVA	✓			✓				
	fANOVA		✓						
	Scott-Knott	✓		✓	✓				
	HyperTuningSK	✓							
	AutoHyperTuningSK	✓							
	AutoHyperTuningSK-DA	✓							
	AutoHyperTuningSK-test	✓							
	Grid Search					✓			✓
	Outros			✓			✓	✓	✓

(*data augmentation*). Por outro lado, os demais trabalhos focam principalmente na seleção da arquitetura neural [84, 47] ou em único tipo de hiperparâmetro, como apenas taxa de aprendizado [131] ou otimizador [13].

Nessa linha, a principal contribuição desta tese é a proposta de métodos recomendação de hiperparâmetros para classificação de imagens da construção civil. Para isso, foram adotados conceitos de estatísticos de Análise de Variância e algoritmo de agrupamento de Scott-Knott, resultando nos métodos HyperTuningSK, AutoHyperTuningSK e variações. Embora outros estudos da literatura também utilizaram técnicas estatísticas, ressalta-se que as abordagens e aplicações propostas são distintas. Por exemplo, os autores de [84] aplicam o algoritmo de Scott-Knott para ranquear arquiteturas de redes neurais em um processo de diagnóstico médico com imagens. Em outra linha, os métodos de ANOVA e Scott-Knott foram adotados por [83] para ajustar hiperparâmetros de algoritmos de Aprendizado por Reforço em uma aplicação de otimização combinatória.

7.3 Trabalhos Futuros

Os estudos realizados nesta tese apresentam limitações e possíveis vertentes para a sua sequência. Dessa forma, são levantados os seguintes tópicos como indicações para a continuidade desta tese:

1. Analisar o desempenho dos métodos propostos em outras aplicações de classificação de imagens e em tarefas mais complexas de visão computacional, como segmentação e detecção. Nesta tese, as abordagens foram testadas apenas em estudos de casos de classifica-

ção binária de imagens. Sendo assim, existem diversos outros problemas relevantes para análise de imagens na construção civil, como detecção de equipamentos de segurança e segmentação de rachaduras. Do mesmo modo, julga-se relevante avaliar os métodos propostos em áreas distintas da Construção 4.0, como em problemas variados da engenharia elétrica, telecomunicações e ciência da computação.

2. Abordar o ajuste de outros tipos de hiperparâmetros, como a seleção da arquitetura neural. Os métodos propostos nesta tese foram aplicados na recomendação de dois tipos de hiperparâmetros: treinamento (taxa de aprendizado e otimizador) e regularização (*data augmentation*). No entanto, a busca por arquiteturas convolucionais adequadas também é um significativo desafio da literatura, e assim, merece ser investigado.

3. Extender os métodos propostos com a adoção de outras técnicas estatísticas e de otimização para o desenvolvimento de novos algoritmos de recomendação de hiperparâmetros. A literatura apresenta diferentes abordagens que podem ser úteis para o aprimoramento e avanço dos métodos HyperTuningSK e AutoHyperTuningSK, como algoritmos evolucionários, técnicas de otimização de busca local, regressão logística e regressão linear múltipla.

4. Aplicar os métodos propostos na recomendação de hiperparâmetros visando problemas de classificação de imagens em tempo real para sistemas embarcados. Nesse aspecto, um importante desafio é a seleção de modelos neurais que apresentem respostas precisas, rápidas e com baixo custo computacional.

Referências Bibliográficas

[1] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60 – 88, 2017.

[3] Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning Publications, 2020.

[4] A. Agrawal and N. Mittal. Using cnn for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. *The Visual Computer*, 36(2):405–412, 2020.

[5] R. Singh, A. Goel, and D.K. Raghuvanshi. Computer-aided diagnostic network for brain tumor classification employing modulated gabor filter banks. *The Visual Computer*, 37(8):2157–2171, 2021.

[6] R. Ünlü and R. Kiriş. Detection of damaged buildings after an earthquake with convolutional neural networks in conjunction with image segmentation. *The Visual Computer*, pages 1–10, 2021.

[7] X. Ni, C. Li, H. Jiang, and F. Takeda. Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. *Horticulture Research*, 7(1):1–14, 2020.

[8] Heekyung Yang and Kyungha Min. Classification of basic artistic media based on a deep convolutional approach. *The Visual Computer*, 36(3):559–578, 2020.

[9] Paulo Chagas, Luiz Souza, Ikaro Araújo, Nayze Aldeman, Angelo Duarte, Michele Angelo, Washington LC Dos-Santos, and Luciano Oliveira. Classification of glomerular hypercellularity using convolutional features and support vector machine. *Artificial intelligence in medicine*, 103:101808, 2020.

[10] M.Y. Abbass, K.-C. Kwon, N. Kim, S.A. Abdelwahab, F.E.A. El-Samie, and A.A.M. Khalaf. Efficient object tracking using hierarchical convolutional features model and correlation filters. *Visual Computer*, 37(4):831–842, 2021.

[11] C. Liu, J. Ying, H. Yang, X. Hu, and J. Liu. Improved human action recognition approach based on two-stream convolutional neural network model. *The Visual Computer*, 37(6):1327–1341, 2021.

[12] Kuan Li, Yi Jin, Muhammad Waqar Akram, Ruize Han, and Jiongwei Chen. Facial expression recognition with convolutional neural networks via a new face cropping and rotation strategy. *The Visual computer*, 36(2):391–404, 2020.

[13] Maryam Kouzehgar, Yokhesh Krishnasamy Tamilselvam, Manuel Vega Heredia, and Mohan Rajesh Elara. Self-reconfigurable façade-cleaning robot equipped with deep-learning-based crack detection based on convolutional neural networks. *Automation in Construction*, 108:102959, 2019.

[14] Xinghua Li, Meizhen He, Huifang Li, and Huanfeng Shen. A combined loss-based multiscale fully convolutional network for high-resolution remote sensing image change detection. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2021.

[15] S. G. E. Gökstorp and T. P. Breckon. Temporal and non-temporal contextual saliency analysis for generalized wide-area search within unmanned aerial vehicle (uav) video. *The Visual Computer*, pages 1–8, 2021.

[16] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017.

[17] Kasthurirangan Gopalakrishnan, Siddhartha K Khaitan, Alok Choudhary, and Ankit Agrawal. Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, 157:322–330, 2017.

[18] Cao Vu Dung and Le Duc Anh. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, 99:52–58, 2019.

[19] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[20] Yuqing Gao and Khalid M Mosalam. Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768, 2018.

[21] Seyed Omid Sajedi and Xiao Liang. Uncertainty-assisted deep vision structural health monitoring. *Computer-Aided Civil and Infrastructure Engineering*, 36(2):126–142, 2021.

[22] Xincong Yang, Heng Li, Yantao Yu, Xiaochun Luo, Ting Huang, and Xu Yang. Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1090–1109, 2018.

[23] Dimitris Dais, Ihsan Engin Bal, Eleni Smyrou, and Vasilis Sarhosis. Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning. *Automation in Construction*, 125:103606, 2021.

[24] Jingjing Guo, Qian Wang, and Yiting Li. Semi-supervised learning based on convolutional neural network and uncertainty filter for façade defects classification. *Computer-Aided Civil and Infrastructure Engineering*, 36(3):302–317, 2021.

[25] Hoang Nhat-Duc, Quoc-Lam Nguyen, and Van-Duc Tran. Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network. *Automation in Construction*, 94:203–213, 2018.

[26] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, and Hiroshi Omata. Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1127–1141, 2018.

[27] Hamed Majidifard, Yaw Adu-Gyamfi, and William G Buttlar. Deep machine learning approach to develop a new asphalt pavement condition index. *Construction and building materials*, 247:118513, 2020.

[28] Guojun Deng, Zhixiang Zhou, Xi Chu, and Shuai Shao. Identification of behavioral features of bridge structure based on static image sequences. *Advances in Civil Engineering*, 2020, 2020.

[29] Eric Bianchi, Amos Lynn Abbott, Pratap Tokekar, and Matthew Hebdon. Coco-bridge: Structural detail data set for bridge inspections. *Journal of Computing in Civil Engineering*, 35(3):04021003, 2021.

[30] Jacob J Lin, Amir Ibrahim, Shubham Sarwade, and Mani Golparvar-Fard. Bridge inspection with aerial robots: Automating the entire pipeline of visual data capture, 3d mapping, defect detection, analysis, and reporting. *Journal of Computing in Civil Engineering*, 35(2):04020064, 2021.

[31] Jack CP Cheng and Mingzhu Wang. Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, 95:155–171, 2018.

[32] Srinath S Kumar, Dulcy M Abraham, Mohammad R Jahanshahi, Tom Iseley, and Justin Starr. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction*, 91:273–283, 2018.

[33] Xianfei Yin, Tianxin Ma, Ahmed Bouferguene, and Mohamed Al-Hussein. Automation for sewer pipe assessment: Cctv video interpretation algorithm and sewer pipe video assessment (spva) system development. *Automation in Construction*, 125:103622, 2021.

[34] Weili Fang, Lieyun Ding, Hanbin Luo, and Peter ED Love. Falls from heights: A computer vision-based approach for safety harness detection. *Automation in Construction*, 91:53–61, 2018.

[35] Shi Chen and Kazuyuki Demachi. Towards on-site hazards identification of improper use of personal protective equipment using deep learning-based geometric relationships and hierarchical scene graph. *Automation in Construction*, 125:103619, 2021.

[36] Jie Shen, Xin Xiong, Ying Li, Wei He, Peng Li, and Xiaoyu Zheng. Detecting safety helmet wearing on construction sites with bounding-box regression and deep transfer learning. *Computer-Aided Civil and Infrastructure Engineering*, 36(2):180–196, 2021.

[37] Qi Fang, Heng Li, Xiaochun Luo, Lieyun Ding, Hanbin Luo, Timothy M Rose, and Wangpeng An. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. *Automation in Construction*, 85:1–9, 2018.

[38] Zdenek Kolar, Hainan Chen, and Xiaowei Luo. Transfer learning and deep convolutional neural networks for safety guardrail detection in 2d images. *Automation in Construction*, 89:58–70, 2018.

[39] Hongjo Kim, Hyoungkwan Kim, Yong Won Hong, and Hyeran Byun. Detecting construction equipment using a region-based fully convolutional network and transfer learning. *Journal of Computing in Civil Engineering*, 32(2):04017082, 2018.

[40] Bo Xiao and Shih-Chung Kang. Development of an image data set of construction machines for deep learning object detection. *Journal of Computing in Civil Engineering*, 35(2):05020005, 2021.

[41] Bo Xiao and Shih-Chung Kang. Vision-based method integrating deep learning detection for tracking multiple construction machines. *Journal of Computing in Civil Engineering*, 35(2):04020071, 2021.

[42] Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. Reinforcement learning for neural architecture search: A review. *Image and Vision Computing*, 89:57 – 66, 2019.

[43] Dilyara Baymurzina, Eugene Golikov, and Mikhail Burtsev. A review of neural architecture search. *Neurocomputing*, 474:82–93, 2022.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[46] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[47] S. Zhou and W. Song. Deep learning-based roadway crack classification using laser-scanned range images: A comparative study on hyperparameter selection. *Automation in Construction*, 114, 2020.

[48] Ghada Atteia, Nagwan Abdel Samee, El-Sayed M El-Kenawy, and Abdelhameed Ibrahim. Cnn-hyperparameter optimization for diabetic maculopathy diagnosis in optical coherence tomography and fundus retinography. *Mathematics*, 10(18):3274, 2022.

[49] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of International Conference on Machine Learning 2014 (ICML 2014)*, pages 754–762, June 2014.

[50] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[51] Dário Passos and Puneet Mishra. A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks. *Chemometrics and Intelligent Laboratory Systems*, page 104520, 2022.

[52] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[53] Rafael G Mantovani, André LD Rossi, Edesio Alcobaça, Joaquin Vanschoren, and André CPLF de Carvalho. A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers. *Information Sciences*, 501:193–221, 2019.

[54] K Krishnakumari, E Sivasankar, and Sam Radhakrishnan. Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification (htcnn-dasc). *Soft Computing*, 24(5):3511–3527, 2020.

[55] Cheng-Jian Lin, Shiou-Yun Jeng, and Chin-Ling Lee. Hyperparameter optimization of deep learning networks for classification of breast histopathology images. *Sensors and Materials*, 33(1):315–325, 2021.

[56] K. Shankar, Yizhuo Zhang, Yiwei Liu, Ling Wu, and Chi-Hua Chen. Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification. *IEEE Access*, 8:118164–118173, 2020.

[57] Ciaran Cooney, Attila Korik, Raffaella Folli, and Damien Coyle. Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech eeg. *Sensors*, 20(16):4629, 2020.

[58] Ugur Erkan, Abdurrahim Toktas, and Deniz Ustun. Hyperparameter optimization of deep cnn classifier for plant species identification using artificial bee colony algorithm. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12, 2022.

[59] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. In press, available at http://automl.org/book.

[60] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[61] Tao Han, Francisco Nauber Bernardo Gois, Ramsés Oliveira, Luan Rocha Prates, and Magda Moura de Almeida Porto. Modeling the progression of covid-19 deaths using kalman filter and automl. *Soft Computing*, pages 1–16, 2021.

[62] Márcio P Basgalupp, Rodrigo C Barros, Alex Guimarães Cardoso de Sá, Gisele L Pappa, Rafael Gomes Mantovani, ACPLF de Carvalho, and Alex Alves Freitas. An extensive experimental evaluation of automated machine learning methods for recommending classification algorithms. *Evolutionary Intelligence*, 14(4):1895–1914, 2021.

[63] Catherine Wong, Neil Houlsby, Yifeng Lu, and Andrea Gesmundo. Transfer learning with neural automl. *Advances in neural information processing systems*, 31, 2018.

[64] Robson P Bonidia, Anderson P Avila Santos, Breno LS de Almeida, Peter F Stadler, Ulisses N da Rocha, Danilo S Sanches, and André CPLF de Carvalho. Bioautoml: automated feature engineering and metalearning to predict noncoding rnas in bacteria. *Briefings in Bioinformatics*, 23(4):bbac218, 2022.

[65] Y.-Q. Hu and Y. Yu. A technical view on neural architecture search. *International Journal of Machine Learning and Cybernetics*, 11(4):795–811, 2020.

[66] Heungseok Park, Yoonsoo Nam, Ji-Hoon Kim, and Jaegul Choo. Hypertendril: Visual analytics for user-driven hyperparameter optimization of deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1407–1416, 2021.

[67] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

[68] Pornntiwa Pawara, Emmanuel Okafor, Lambert Schomaker, and Marco Wiering. Data augmentation for plant classification. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 615–626. Springer, 2017.

[69] C. Shorten and T.M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 2019.

[70] Randall Balestriero, Ishan Misra, and Yann LeCun. A data-augmentation is worth a thousand samples: Exact quantification from analytical augmented sample moments. *arXiv preprint arXiv:2202.08325*, 2022.

[71] Mohammed Chachan Younis and Edward Keedwell. Semantic segmentation on small datasets of satellite images using convolutional neural networks. *Journal of Applied Remote Sensing*, 13(4):046510, 2019.

[72] Zirui Wang, Jingjing Yang, Haonan Jiang, and Xueling Fan. Cnn training with twenty samples for crack detection via data augmentation. *Sensors*, 20(17):4849, 2020.

[73] Markus Bayer, Marc-André Kaufhold, Björn Buchhold, Marcel Keller, Jörg Dallmeyer, and Christian Reuter. Data augmentation in natural language processing: a novel text generation approach for long and short text classifiers. *International Journal of Machine Learning and Cybernetics*, 14:135–150, 2023.

[74] Chunhe Song, Wenxiang Xu, Zhongfeng Wang, Shimao Yu, Peng Zeng, and Zhaojie Ju. Analysis on the impact of data augmentation on target recognition for uav-based transmission line inspection. *Complexity*, 2020, 2020.

[75] Daisuke Hirahara, Eichi Takaya, Taro Takahara, and Takuya Ueda. Effects of data count and image scaling on deep learning training. *PeerJ Computer Science*, 6:e312, 2020.

[76] Randall Balestriero, Leon Bottou, and Yann LeCun. The effects of regularization and data augmentation are class dependent. *arXiv preprint arXiv:2204.03632*, 2022.

[77] Maram Mahmoud A Monshi, Josiah Poon, Vera Chung, and Fahad Mahmoud Monshi. Covidxraynet: Optimizing data augmentation and cnn hyperparameters for improved covid-19 detection from cxr. *Computers in Biology and Medicine*, 133:104375, 2021.

[78] Diane Wagner, Fabio Ferreira, Danny Stoll, Robin Tibor Schirrmeister, Samuel Müller, and Frank Hutter. On the importance of hyperparameters and data augmentation for self-supervised learning. *arXiv preprint arXiv:2207.07875*, 2022.

[79] D. C. Montgomery. *Design and analysis of experiments*. New York: John Wiley & Sons, 9th, 2017.

[80] Andrew Jhon Scott and M Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3):507–512, 1974.

[81] Enio G Jelihovschi, José Cláudio Faria, and Ivan Bezerra Allaman. Scottknott: a package for performing the scott-knott clustering algorithm in r. *TEMA - SBMAC*, 15(1):3–17, 2014.

[82] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1533–1540, 2016.

[83] André L. C. Ottoni, Erivelton G. Nepomuceno, Marcos S. de Oliveira, and Daniela C. R. de Oliveira. Tuning of reinforcement learning parameters applied to sop using the scott–knott method. *Soft Computing*, 24:4441–4453, 2020.

[84] Chaymaa Lahmar and Ali Idri. On the value of deep learning for diagnosing diabetic retinopathy. *Health and Technology*, pages 1–17, 2021.

[85] Hasnae Zerouaoui and Ali Idri. Deep hybrid architectures for binary classification of medical breast cancer images. *Biomedical Signal Processing and Control*, 71:103226, 2022.

[86] A. L. C. Ottoni, M. S. Novo, and D. B. Costa. Hyperparameter tuning of convolutional neural networks for building construction image classification. *The Visual Computer*, pages 1–15, 2022.

[87] A. L. C. Ottoni, R. M. Amorim, M. S. Novo, and D. B. Costa. Tuning of data augmentation hyperparameters in deep learning to building construction image classification with small datasets. *International Journal of Machine Learning and Cybernetics*, 14:171–186, 2023.

[88] André L. C. Ottoni, Marcela S. Novo, and Dayana B. Costa. Deep learning for vision systems in construction 4.0: a systematic review. *Signal, Image and Video Processing*, pages 1–9, 2022.

[89] André L. C. Ottoni and Marcela S. Novo. A deep learning approach to vegetation images recognition in buildings: a hyperparameter tuning case study. *IEEE Latin America Transactions*, 19(12):2062–2070, 2021.

[90] Hélio Pedrini and William Robson Schwartz. *Análise de imagens digitais: princípios, algoritmos e aplicações*. Cengage Learning, 2008.

[91] I. N. Silva, D. H. Spatti, and R. A. Flauzino. *Redes Neurais Artificiais para Engenharias e Ciências Aplicadas: conceitos teóricos e aspectos práticos*. 2016.

[92] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[93] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[94] François Chollet and J. J. Allaire. *Deep Learning With R*. Manning Publications, 2018.

[95] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[96] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[97] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7):2121–2159, 2011.

[98] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[99] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[100] Orsolya Nagy, Ilona Papp, and Roland Zsolt Szabó. Construction 4.0 organisational level challenges and solutions. *Sustainability*, 13(21):1–18, 2021.

[101] N. Perrier, A. Bled, M. Bourgault, N. Cousin, C. Danjou, R. Pellerin, and T. Roland. Construction 4.0: A survey of research trends. *Journal of Information Technology in Construction*, 25:416–437, 2020.

[102] Eric Forcael, Isabella Ferrari, Alexander Opazo-Vega, and Jesús Alberto Pulido-Arcas. Construction 4.0: A literature review. *Sustainability*, 12(22), 2020.

[103] Anil Sawhney, Michael Riley, and Javier Irizarry. *Construction 4.0: An innovation platform for the built environment*. Routledge, 2020.

[104] Pia Schönbeck, Malin Löfsjögård, and Anders Ansell. Quantitative review of construction 4.0 technology presence in construction project research. *Buildings*, 10(10), 2020.

[105] Rafaela Oliveira Rey, Roseneia Rodrigues Santos de Melo, and Dayana Bastos Costa. Design and implementation of a computerized safety inspection system for construction sites using uas and digital checklists–smart inspecs. *Safety science*, 143:105430, 2021.

[106] Taofeek D Akinosho, Lukumon O Oyedele, Muhammad Bilal, Anuoluwapo O Ajayi, Manuel Davila Delgado, Olugbenga O Akinade, and Ashraf A Ahmed. Deep learning in the construction industry: A review of present status and future innovations. *Journal of Building Engineering*, page 101827, 2020.

[107] W. Fang, P.E.D. Love, H. Luo, and L. Ding. Computer vision for behaviour-based safety in construction: A review and future directions. *Advanced Engineering Informatics*, 43, 2020.

[108] L. Hou, H. Chen, G.K. Zhang, and X. Wang. Deep learning-based applications for safety management in the aec industry: A review. *Applied Sciences (Switzerland)*, 11(2):1–18, 2021.

[109] R. Khallaf and M. Khallaf. Classification and analysis of deep learning applications in construction: A systematic literature review. *Automation in Construction*, 129, 2021.

[110] Mohamad Alipour, Devin K Harris, and Gregory R Miller. Robust pixel-level crack detection using deep fully convolutional neural networks. *Journal of Computing in Civil Engineering*, 33(6):04019040, 2019.

[111] Jingdao Chen, Zsolt Kira, and Yong K Cho. Deep learning approach to point cloud scene understanding for automated scan to 3d reconstruction. *Journal of Computing in Civil Engineering*, 33(4):04019027, 2019.

[112] P Shane Crawford, Mohammad A Al-Zarrad, Andrew J Graettinger, Alexander M Hainen, Edward Back, and Lawrence Powell. Rapid disaster data dissemination and vulnerability assessment through synthesis of a web-based extreme event viewer and deep learning. *Advances in Civil Engineering*, 2018, 2018.

[113] Sattar Dorafshan, Robert J Thomas, and Marc Maguire. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials*, 186:1031–1045, 2018.

[114] Youjin Jang, Yonghan Ahn, and Ha Young Kim. Estimating compressive strength of concrete using deep convolutional neural networks with digital microscope images. *Journal of Computing in Civil Engineering*, 33(3):04019018, 2019.

[115] Somin Park, Seongdeok Bang, Hongjo Kim, and Hyoungkwan Kim. Patch-based crack detection in black box images using convolutional neural networks. *Journal of Computing in Civil Engineering*, 33(3):04019017, 2019.

[116] Xuebin Qin, Shifu Cui, Lang Liu, Pai Wang, Mei Wang, and Jie Xin. Prediction of mechanical strength based on deep learning using the scanning electron image of microscopic cemented paste backfill. *Advances in Civil Engineering*, 2018, 2018.

[117] Zheng Tong, Jie Gao, and Haitao Zhang. Innovative method for recognizing subgrade defects based on a convolutional neural network. *Construction and Building Materials*, 169:69–82, 2018.

[118] Xin Wang and Zhenhua Zhu. Vision-based hand signal recognition in construction: A feasibility study. *Automation in Construction*, 125:103625, 2021.

[119] Yadong Xue and Yicheng Li. A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Computer-Aided Civil and Infrastructure Engineering*, 33(8):638–654, 2018.

[120] Yang Yang, Lin Yang, Bo Wu, Gang Yao, Hang Li, and Soltys Robert. Safety prediction using vehicle safety evaluation model passing on long-span bridge with fully connected neural network. *Advances in Civil Engineering*, 2019, 2019.

[121] Zhe Yang, Boning He, Yang Liu, Di Wang, and Guoli Zhu. Classification of rock fragments produced by tunnel boring machine using convolutional neural networks. *Automation in Construction*, 125:103612, 2021.

[122] Rahmat Ali and Young-Jin Cha. Subsurface damage detection of a steel bridge using deep learning and uncooled micro-bolometer. *Construction and Building Materials*, 226:376–387, 2019.

[123] Seongdeok Bang, Somin Park, Hongjo Kim, and Hyoungkwan Kim. Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering*, 34(8):713–727, 2019.

[124] Lieyun Ding, Weili Fang, Hanbin Luo, Peter ED Love, Botao Zhong, and Xi Ouyang. A deep hybrid learning model to detect unsafe behavior: Integrating convolution neural networks and long short-term memory. *Automation in Construction*, 86:118–124, 2018.

[125] Nur Sila Gulgec, Martin Takáč, and Shamim N Pakzad. Convolutional neural network approach for robust structural damage detection and localization. *Journal of computing in civil engineering*, 33(3):04019005, 2019.

[126] Feng Guo, Yu Qian, and Yuefeng Shi. Real-time railroad track components inspection based on the improved yolov4 framework. *Automation in Construction*, 125:103596, 2021.

[127] Feng Guo, Yu Qian, Yunpeng Wu, Zhen Leng, and Huayang Yu. Automatic railroad track components inspection using real-time instance segmentation. *Computer-Aided Civil and Infrastructure Engineering*, 36(3):362–377, 2021.

[128] Nhat-Duc Hoang and Quoc-Lam Nguyen. A novel approach for automatic detection of concrete surface voids using image texture analysis and history-based adaptive differential evolution optimized support vector machine. *Advances in Civil Engineering*, 2020, 2020.

[129] Dongho Kang and Young-Jin Cha. Autonomous uavs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging. *Computer-Aided Civil and Infrastructure Engineering*, 33(10):885–902, 2018.

[130] Daeho Kim, Meiyin Liu, SangHyun Lee, and Vineet R Kamat. Remote proximity monitoring between mobile construction resources using camera-mounted uavs. *Automation in Construction*, 99:168–182, 2019.

[131] Shengyuan Li and Xuefeng Zhao. Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Advances in Civil Engineering*, 2019, 2019.

[132] Yonglong Li, Hua Zhang, Shuang Wang, Haoran Wang, and Jialong Li. Image-based underwater inspection system for abrasion of stilling basin slabs of dam. *Advances in Civil Engineering*, 2019, 2019.

[133] Yange Li, Han Wei, Zheng Han, Jianling Huang, and Weidong Wang. Deep learning-based safety helmet detection in engineering management based on convolutional neural networks. *Advances in Civil Engineering*, 2020, 2020.

[134] Shuwei Li, Xingyu Gu, Xiangrong Xu, Dawei Xu, Tianjie Zhang, Zhen Liu, and Qiao Dong. Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm. *Construction and Building Materials*, 273:121949, 2021.

[135] Pablo Martinez, Beda Barkokebas, Farook Hamzeh, Mohamed Al-Hussein, and Rafiq Ahmad. A vision-based approach for automatic progress tracking of floor paneling in offsite construction facilities. *Automation in Construction*, 125:103620, 2021.

[136] Zhufeng Pan, Jian Yang, Xing-er Wang, Feiliang Wang, Iftikhar Azim, and Chenyu Wang. Image-based surface scratch detection on architectural glass panels using deep learning approach. *Construction and Building Materials*, 282:122717, 2021.

[137] Ju An Park, Chul Min Yeum, and Trevor D Hrynyk. Learning-based image scale estimation using surface textures for quantitative visual inspection of regions-of-interest. *Computer-Aided Civil and Infrastructure Engineering*, 36(2):227–241, 2021.

[138] Somin Park, Francis Baek, Jiu Sohn, and Hyoungkwan Kim. Computer vision–based estimation of flood depth in flooded-vehicle images. *Journal of Computing in Civil Engineering*, 35(2):04020072, 2021.

[139] Xiong Peng, Xingu Zhong, Chao Zhao, Y Frank Chen, and Tianyu Zhang. The feasibility assessment study of bridge crack width recognition in images based on special inspection uav. *Advances in Civil Engineering*, 2020, 2020.

[140] Yupeng Ren, Jisheng Huang, Zhiyou Hong, Wei Lu, Jun Yin, Lejun Zou, and Xiaohua Shen. Image-based concrete crack detection in tunnels using deep fully convolutional networks. *Construction and Building Materials*, 234:117367, 2020.

[141] Zheng Tong, Jie Gao, and Haitao Zhang. Recognition, location, measurement, and 3d reconstruction of concealed cracks using convolutional neural networks. *Construction and Building Materials*, 146:775–787, 2017.

[142] Wei Wei, Lieyun Ding, Hanbin Luo, Chen Li, and Guowei Li. Automated bughole detection and quality performance assessment of concrete using image processing and deep convolutional neural networks. *Construction and Building Materials*, 281:122576, 2021.

[143] Jasper S Wijnands, Haifeng Zhao, Kerry A Nice, Jason Thompson, Katherine Scully, Jingqiu Guo, and Mark Stevenson. Identifying safe intersection design through unsupervised feature extraction from satellite imagery. *Computer-Aided Civil and Infrastructure Engineering*, 36(3):346–361, 2021.

[144] Yongzhi Xu, Xuesong Shen, and Samsung Lim. Cordet: Corner-aware 3d object detection networks for automated scan-to-bim. *Journal of Computing in Civil Engineering*, 35(3):04021002, 2021.

[145] Gang Yao, Fujia Wei, Yang Yang, and Yujia Sun. Deep-learning-based bughole detection for concrete surface image. *Advances in Civil Engineering*, 2019, 2019.

[146] Xiaochun Luo, Heng Li, Dongping Cao, Fei Dai, J Seo, S Lee, et al. Recognizing diverse construction activities in site images via relevance networks of construction-related objects detected by convolutional neural networks. *Journal of Computing in Civil Engineering*, 32(3):04018012, 2018.

[147] Yantao Yu, Heng Li, Waleed Umer, Chao Dong, Xincong Yang, Martin Skitmore, and Arnold YL Wong. Automatic biomechanical workload estimation for construction workers by computer vision and smart insoles. *Journal of Computing in Civil Engineering*, 33(3):04019010, 2019.

[148] Allen Zhang, Kelvin CP Wang, Baoxian Li, Enhui Yang, Xianxing Dai, Yi Peng, Yue Fei, Yang Liu, Joshua Q Li, and Cheng Chen. Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering*, 32(10):805–819, 2017.

[149] Allen Zhang, Kelvin CP Wang, Yue Fei, Yang Liu, Siyu Tao, Cheng Chen, Joshua Q Li, and Baoxian Li. Deep learning–based fully automated pavement crack detection on 3d asphalt surfaces with an improved cracknet. *Journal of Computing in Civil Engineering*, 32(5):04018041, 2018.

[150] Jinyue Zhang, Lijun Zi, Yuexian Hou, Mingen Wang, Wenting Jiang, and Da Deng. A deep learning-based approach to enable action recognition for construction equipment. *Advances in Civil Engineering*, 2020, 2020.

[151] T. N. Khilji, Luana Lopes Amaral Loures, and Ehsan Rezazadeh Azar. Distress recognition in unpaved roads using unmanned aerial systems and deep learning segmentation. *Journal of Computing in Civil Engineering*, 35(2):04020061, 2021.

[152] Zhenqing Liu, Yiwen Cao, Yize Wang, and Wei Wang. Computer vision-based concrete crack detection using u-net fully convolutional networks. *Automation in Construction*, 104:129–139, 2019.

[153] Gwanyong Park, Minhyung Lee, Hyangin Jang, and Changmin Kim. Thermal anomaly detection in walls via cnn-based segmentation. *Automation in Construction*, 125:103627, 2021.

[154] Yalong Pi, Nipun D Nath, and Amir H Behzadan. Detection and semantic segmentation of disaster damage in uav footage. *Journal of Computing in Civil Engineering*, 35(2):04020063, 2021.

[155] Shanglian Zhou and Wei Song. Crack segmentation through deep convolutional neural networks and heterogeneous image fusion. *Automation in Construction*, 125:103605, 2021.

[156] Alireza Valikhani, Azadeh Jaberi Jahromi, Samira Pouyanfar, Islam M Mantawy, and Atorod Azizinamini. Machine learning and image processing approaches for estimating concrete surface roughness using basic cameras. *Computer-Aided Civil and Infrastructure Engineering*, 36(2):213–226, 2021.

[157] Zheng Tong, Zhenjun Wang, Xiaofeng Wang, Yuwei Ma, Haoyan Guo, and Cunqiang Liu. Characterization of hydration and dry shrinkage behavior of cement emulsified asphalt composites using deep learning. *Construction and Building Materials*, 274:121898, 2021.

[158] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[159] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[160] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[161] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[162] S. Cheng, H. Lai, L. Wang, and J. Qin. A novel deep hashing method for fast image retrieval. *The Visual Computer*, 35(9):1255–1266, 2019.

[163] Z. Chen, Z. Hu, B. Sheng, P. Li, J. Kim, and E. Wu. Simplified non-locally dense network for single-image dehazing. *The Visual Computer*, 36(10-12):2189–2200, 2020.

[164] Mingming Yang, Tinghuai Ma, Qing Tian, Yuan Tian, Abdullah Al-Dhelaan, and Mohammed Al-Dhelaan. Aggregated squeeze-and-excitation transformations for densely connected convolutional networks. *The Visual Computer*, pages 1–14, 2021.

[165] ECB Monteiro, MQ Oliveira, KS Almeida, JR Carvalho, TO Chaves, and E ARIMATEIA. Estudo da degradação nas marquises de edificações do centro histórico do recife. In *Congresso Internacional sobre Patología y Recuperación de Estructuras*, 2010.

[166] Marcus Plaisant, Gustavo Almeida, and Assed Naked Haddad. Patologias biológicas–tratamento de vida vegetal nos edifícios históricos do rio de janeiro iv cirmare–rio de janeiro. *IV Congresso Internacional na "Recuperação, Manutenção e Restauração de Edifícios (CIRMARE)*, 2015.

[167] Fábio Remy de Assunção Rios, Dalva Damiana Estevam da Silva, Jonatas Nascimento da Costa, and Bruna Jakeline Soares Souza. Análise das manifestações patológicas das marquises de concreto armado no centro de campina grande-pb. *Revista de Geociências do Nordeste*, 5:12–22, 2019.

[168] M. Kaamin, N.F.A. Ahmad, S.N.M. Razali, M. Mokhtar, N. Ngadiman, D.M. Masri, I.A. Hussin, and L.H.M. Asri. Visual inspection of heritage mosques using unmanned aerial vehicle (uav) and condition survey protocol (csp) 1 matrix: A case study of tengkera mosque and kampung kling mosque, melaka. volume 1529, 2020.

[169] M. Loukma and M. Stefanidou. Causes of deterioration of ottoman mosques. *WIT Transactions on The Built Environment*, 177:173–180, 2018.

[170] EA Rocha, JVS Macedo, P Correia, and ECB Monteiro. Adaptation of a damage map to historical buildings with pathological problems: Case study at the church of carmo in olinda, pernambuco. *Revista ALCONPAT*, 8(1):51–63, 2018.

[171] András L Majdik, Charles Till, and Davide Scaramuzza. The zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3):269–273, 2017.

[172] Nuno Garcez, Nuno Lopes, Jorge de Brito, and José Silvestre. System of inspection, diagnosis and repair of external claddings of pitched roofs. *Construction and Building Materials*, 35:1034–1044, 2012.

[173] J Conceição, B Poça, J De Brito, I Flores-Colen, and A Castelo. Inspection, diagnosis, and rehabilitation system for flat roofs. *Journal Of Performance Of Constructed Facilities*, 31(6):04017100, 2017.

[174] Bruno Silveira, Roseneia Melo, and Dayana Bastos Costa. Using uas for roofs structure inspections at post-occupational residential buildings. In Eduardo Toledo Santos and Sergio Scheer, editors, *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*, pages 1055–1068, Cham, 2021. Springer International Publishing.

[175] L. B. Staffa, L. S. V. Sá, M. I. S. C. Lima, and D. B. Costa. Use of image processing techniques for inspection of building roof structures for technical assistance purposes (in portuguese). *ENTAC - National Meeting of the Built Environment Technology*, 2020.

[176] Luqman Ali, Fady Alnajjar, Wasif Khan, Mohamed Adel Serhani, and Hamad Al Jassmi. Bibliometric analysis and review of deep learning-based crack detection literature published between 2010 and 2022. *Buildings*, 12(4):432, 2022.

[177] Arun Mohan and Sumathi Poobal. Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, 57(2):787–798, 2018.

[178] Wissam Kaddah, Marwa Elbouz, Yousri Ouerhani, Ayman Alfalou, and Marc Desthieux. Automatic darkest filament detection (adfd): a new algorithm for crack extraction on two-dimensional pavement images. *The Visual Computer*, 36(7):1369–1384, 2020.

[179] Yao Yao, Shue-Ting Ellen Tung, and Branko Glisic. Crack detection and characterization techniques—an overview. *Structural Control and Health Monitoring*, 21(12):1387–1413, 2014.

[180] Hafiz Suliman Munawar, Ahmed WA Hammad, Assed Haddad, Carlos Alberto Pereira Soares, and S Travis Waller. Image-based crack detection methods: A review. *Infrastructures*, 6(8):115, 2021.

[181] Raza Ali, Joon Huang Chuah, Mohamad Sofian Abu Talip, Norrima Mokhtar, and Muhammad Ali Shoaib. Structural crack detection using deep convolutional neural networks. *Automation in Construction*, 133:103989, 2022.

[182] Md Zawad, Rahat Shahriar, Md Zawad, Fahad Shahriar, Md Rahman, Sudipto Nath Priyom, et al. A comparative review of image processing based crack detection techniques on civil engineering structures. *Journal of Soft Computing in Civil Engineering*, 5(3):58–74, 2021.

[183] Yuhua Zhang, Jiaqi Zheng, Wei Sun, and Liang Shan. Image recognition method of building wall cracks based on feature distribution. *Soft Computing*, 24(11):8285–8294, 2020.

[184] Ç F Özgenel. *Concrete Crack Images for Classification*, v2 edition, 2019. Mendeley Data.

[185] Ç F Özgenel and A Gönenç Sorguç. Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *Isarc. proceedings of the international symposium on automation and robotics in construction*, volume 35, pages 1–8. IAARC Publications, 2018.

[186] Haitong Tang, Jun Shi, Xia Lu, Zhichao Yin, Lixue Huang, Dongbao Jia, and Nizhuan Wang. Comparison of convolutional sparse coding network and convolutional neural network for pavement crack classification: A validation study. In *Journal of Physics: Conference Series*, page 012016. IOP Publishing, 2020.

[187] Bubryur Kim, Natarajan Yuvaraj, KR Sri Preethaa, and R Arun Pandian. Surface crack detection using deep learning with shallow cnn architecture for enhanced computation. *Neural Computing and Applications*, 33(15):9289–9305, 2021.

[188] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[189] Maurice Stevenson Bartlett. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, 160(901):268–282, 1937.

[190] Jingjing Guo, Qian Wang, Yiting Li, and Pengkun Liu. Façade defects classification from imbalanced dataset using meta learning-based convolutional neural network. *Computer-Aided Civil and Infrastructure Engineering*, 35(12):1403–1418, 2020.

[191] Javed Imran and Balasubramanian Raman. Deep motion templates and extreme learning machine for sign language recognition. *The Visual Computer*, 36(6):1233–1246, 2020.

[192] Junde Chen, Defu Zhang, Md Suzauddola, Yaser Ahangari Nanehkaran, and Yuandong Sun. Identification of plant disease images via a squeeze-and-excitation mobilenet model and twice transfer learning. *IET Image Processing*, 15(5):1115–1127, 2021.

[193] Yahui Nan, Jianguo Ju, Qingyi Hua, Haoming Zhang, and Bo Wang. A-mobilenet: An approach of facial expression recognition. *Alexandria Engineering Journal*, 61(6):4435–4444, 2022.

[194] V Tangudu, Jagadeesh Kakarla, and Isunuri Bala Venkateswarlu. Covid-19 detection from chest x-ray using mobilenet and residual separable convolution block. *Soft Computing*, 26(5):2197–2208, 2022.

[195] Elhoucine Elfatimi, Recep Eryigit, and Lahcen Elfatimi. Beans leaf diseases classification using mobilenet models. *IEEE Access*, 10:9471–9482, 2022.

[196] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.