

**UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E
ENGENHARIAS**

LEONARDO LACERDA BURATO

**GERAÇÃO DE TRAJETÓRIAS PARA MANUTENÇÃO PREVENTIVA
EM USINAS FOTOVOLTAICAS DE GRANDE PORTE UTILIZANDO
VISÃO COMPUTACIONAL E ALGORITMOS CLÁSSICOS DE
OTIMIZAÇÃO**

CAXIAS DO SUL

2026

LEONARDO LACERDA BURATO

**GERAÇÃO DE TRAJETÓRIAS PARA MANUTENÇÃO PREVENTIVA
EM USINAS FOTOVOLTAICAS DE GRANDE PORTE UTILIZANDO
VISÃO COMPUTACIONAL E ALGORITMOS CLÁSSICOS DE
OTIMIZAÇÃO**

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do título de Bacharel em
Engenharia de Computação na Área
do Conhecimento de Ciências Exatas e
Engenharias da Universidade de Caxias
do Sul.

Orientador: Prof. Me. Tiago Cassol
Severo

CAXIAS DO SUL

2026

LEONARDO LACERDA BURATO

**GERAÇÃO DE TRAJETÓRIAS PARA MANUTENÇÃO PREVENTIVA
EM USINAS FOTOVOLTAICAS DE GRANDE PORTE UTILIZANDO
VISÃO COMPUTACIONAL E ALGORITMOS CLÁSSICOS DE
OTIMIZAÇÃO**

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do título de Bacharel em
Engenharia de Computação na Área
do Conhecimento de Ciências Exatas e
Engenharias da Universidade de Caxias
do Sul.

Aprovado(a) em 01/07/2026

BANCA EXAMINADORA

Prof. Me. Tiago Cassol Severo
Universidade de Caxias do Sul - UCS

Prof. Dr. André Luis Martinotto
Universidade de Caxias do Sul - UCS

Profª. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

RESUMO

O crescimento acelerado da energia solar no Brasil, evidenciado pela expansão das usinas fotovoltaicas de geração centralizada, impõe desafios técnicos significativos relacionados ao monitoramento e à manutenção de extensas áreas operacionais. As inspeções manuais tradicionais nos ambientes demandam elevado tempo, recursos humanos e expõem os trabalhadores a riscos de segurança. Este trabalho propõe um sistema baseado em visão computacional para automatizar a detecção das fileiras de módulos fotovoltaicos em imagens aéreas e gerar rotas de cobertura para as equipes de manutenção. A arquitetura de detecção de objetos YOLOv8m é treinada para localizar as mesas fotovoltaicas, cujas regiões são tratadas como obstáculos na construção de uma máscara navegável dos corredores. Sobre a máscara aplica-se a esqueletização, que extrai o eixo central das vias, posteriormente convertido em um grafo ponderado com distâncias expressas em pixels, dada a ausência de informação de *Ground Sample Distance* (GSD) nas imagens. A rota de inspeção é obtida pela formulação do Problema do Carteiro Rural, na qual os corredores entre as fileiras constituem as arestas de percurso obrigatório e os trechos de conexão são empregados apenas para ligar os corredores e ajustar a paridade do grafo, utilizando o algoritmo de Dijkstra no cálculo dos caminhos mínimos das ligações. O modelo de detecção alcançou mAP@0,5 de valor médio de 0,81, e a rota gerada cobriu entre 96% e 100% das fileiras detectadas nas usinas avaliadas, com cobertura média de 97,6% e economia média de 23,4% no percurso em relação à formulação do Carteiro Chinês.

Palavras-chave: Usinas Fotovoltaicas. Visão Computacional. Detecção de Objetos. Planejamento de Cobertura. Otimização de Rotas.

ABSTRACT

The accelerated growth of solar energy in Brazil, evidenced by the expansion of centralized-generation photovoltaic plants, poses significant technical challenges related to the monitoring and maintenance of large operational areas. Traditional manual inspections in these environments demand considerable time and human resources and expose workers to safety risks. This work proposes a computer-vision-based system to automate the detection of photovoltaic module rows in aerial images and to generate coverage routes for maintenance teams. The YOLOv8m object detection architecture is trained to locate the photovoltaic tables, whose regions are treated as obstacles in the construction of a navigable mask of the corridors. Skeletonization is applied to this mask to extract the central axis of the paths, which is then converted into a weighted graph with distances expressed in pixels, given the absence of Ground Sample Distance (GSD) information in the images. The inspection route is obtained by formulating the Rural Postman Problem, in which the corridors between rows constitute the required edges and the connecting segments are used only to link the corridors and adjust the parity of the graph, using Dijkstra's algorithm to compute the shortest paths of these connections. The detection model achieved a mAP@0.5 of approximately 0.81, and the generated route covered between 96% and 100% of the detected rows in the evaluated plants, with an average coverage of 97.6% and an average travel savings of 23.4% compared with the Chinese Postman formulation.

Keywords: Photovoltaic Power Plants. Computer Vision. Object Detection. Coverage Path Planning. Route Optimization.

LISTA DE FIGURAS

Figura 1 – Evolução da Potência Solar Instalada no Brasil	9
Figura 2 – Componentes de um sistema fotovoltaico.	12
Figura 3 – Curva P-V de um módulo fotovoltaico	13
Figura 4 – Componentes de um módulo fotovoltaico	13
Figura 5 – Disposição dos módulos em uma usina solar de grande porte	16
Figura 6 – Estrutura de uma CNN	17
Figura 7 – Filtro convolucional deslizando sobre a imagem de entrada	17
Figura 8 – Campo receptivo do filtro convolucional	18
Figura 9 – <i>Max Pooling</i> com janela 2×2 e <i>stride</i> 2	20
Figura 10 – Processo de achatamento dos mapas de características	21
Figura 11 – Camada totalmente conectada e camada de <i>Dropout</i>	22
Figura 12 – Ciclo de treinamento <i>Backpropagation</i>	23
Figura 13 – Resumo da arquitetura do modelo R-CNN	26
Figura 14 – Arquitetura completa do YOLOv8: <i>Backbone</i> , <i>Neck</i> e <i>Head</i>	28
Figura 15 – Módulos internos da arquitetura YOLOv8: C2f, Bottleneck, SPPF, Conv e cabeça desacoplada	29
Figura 16 – Cálculo do <i>Intersection over Union</i> (IoU) entre a caixa predita e a caixa real	31
Figura 17 – Exemplo de mapa de ocupação binário.	33
Figura 18 – Exemplo de grafo ponderado com caminho calculado pelo algoritmo de Dijkstra	36
Figura 19 – Exemplo de instância do Problema do Carteiro Rural	37
Figura 20 – Resultados de detecção de múltiplas classes de sujidade: (a,d) YOLOv5; (b,e) HIC-YOLOv5; (c,f) outros modelos	39
Figura 21 – Pipeline do método proposto por Pérez-González <i>et al.</i> (2021): imagem de entrada, segmentação semântica, máscara pós-processada e rota CPP calculada	41
Figura 22 – Fluxograma do <i>pipeline</i> de implementação	43
Figura 23 – Exemplo de anotação no formato <i>bounding box</i> por fileira de módulos foto- voltaicos	45
Figura 24 – Resultado da detecção das fileiras de módulos fotovoltaicos pelo YOLOv8m, com as <i>bounding boxes</i> sobrepostas à imagem aérea da usina.	49
Figura 25 – Máscara navegável, sobreposta à imagem da usina.	51
Figura 26 – Esqueleto extraído após a poda, sobreposto à imagem da usina.	52
Figura 27 – Grafo de corredores extraído, com nós e arestas sobrepostos à imagem da usina.	53
Figura 28 – Rota de cobertura gerada, com ponto de início (círculo verde) e ponto de fim (quadrado vermelho) destacados.	55

LISTA DE TABELAS

Tabela 1 – Divisão do conjunto de dados	44
Tabela 2 – Variantes da arquitetura YOLOv8 para detecção e número de parâmetros. . .	47
Tabela 3 – Hiperparâmetros de treinamento.	48
Tabela 4 – Comparação de desempenho entre os otimizadores SGD e AdamW (YOLOv8m).	56
Tabela 5 – Comparação de desempenho entre as variantes YOLOv8n, YOLOv8m e YOLOv8l.	57
Tabela 6 – Resultados da geração da rota em usinas representativas.	59

SUMÁRIO

1	INTRODUÇÃO	9
1.1	ESTRUTURA DO TRABALHO	10
1.2	OBJETIVOS E JUSTIFICATIVAS	11
1.2.1	OBJETIVO GERAL	11
1.2.2	OBJETIVOS ESPECÍFICOS	11
1.2.3	JUSTIFICATIVAS	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	Sistema Fotovoltaico	12
2.1.1	Módulo Fotovoltaico	12
2.2	Usinas Fotovoltaicas de Grande Porte	14
2.2.1	Características das Usinas de Grande Porte	15
2.3	Redes Neurais Convolucionais	16
2.3.1	Camada de Convolução	17
2.3.2	Função de Ativação	19
2.3.3	Camada de <i>pooling</i>	20
2.3.4	Camada de Achatamento	21
2.3.5	Camada Totalmente Conectada e Classificação	21
2.3.6	Treinamento e <i>Backpropagation</i>	23
2.3.7	Generalização e Aumento de Dados	24
2.3.8	Aprendizado por Transferência e <i>Fine-Tuning</i>	25
2.3.9	Deteção de Objetos	25
2.3.10	Arquitetura YOLOv8	27
2.3.11	Métricas de Avaliação de Desempenho	30
2.4	Planejamento de Rotas com Algoritmos de Busca em Grafos	32
2.4.1	Mapas de Ocupação Binários	32
2.4.2	Representação em Grafo e Circuito de Euler	34
2.4.3	Planejamento de Cobertura	34
2.4.4	Algoritmo de Dijkstra	35
2.4.5	Problema do Carteiro Chinês e Emparelhamento de Custo Mínimo . . .	36
2.4.6	Problema do Carteiro Rural	36
2.5	Revisão Bibliográfica	38
2.5.1	<i>Aerial Imaging-Based Soiling Detection System for Solar Photovoltaic Panel Cleanliness Inspection</i> (NAEEM et al., 2025)	38
2.5.2	<i>YOLO-Based Photovoltaic Panel Detection: A Comparative Study</i> (MOUS-SAOUI et al., 2025)	39

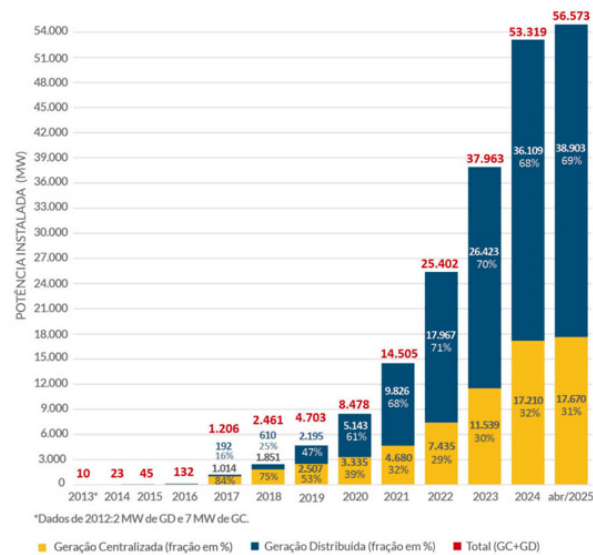
2.5.3	<i>Coverage Path Planning with Semantic Segmentation for UAV in PV Plants</i> (PÉREZ-GONZÁLEZ <i>et al.</i> , 2021)	40
3	IMPLEMENTAÇÃO DESENVOLVIDA	42
3.1	Dataset	43
3.1.1	Fonte e Composição das Imagens	43
3.1.2	Divisão do Conjunto de Dados	43
3.1.3	Anotação das Imagens	44
3.1.4	Pré-processamento e Aumento de Dados	45
3.2	Treinamento do Modelo	46
3.2.1	Ambiente de Treinamento	46
3.2.2	Estratégia de Treinamento	46
3.2.3	Configuração e Hiperparâmetros	47
3.2.4	Métricas de Avaliação	48
3.3	Geração da Máscara Navegável	49
3.4	Extração e Poda do Esqueleto	51
3.5	Construção do Grafo de Corredores	52
3.6	Geração da Rota de Cobertura pelo Carteiro Rural	54
4	TESTES E RESULTADOS OBTIDOS	56
4.1	Resultados do Treinamento e Detecção	56
4.1.1	Comparação entre Otimizadores	56
4.1.2	Comparação entre Variantes de Escala	57
4.1.3	Análise das Métricas e Limitações	57
4.2	Critérios de Avaliação da Rota	58
4.3	Resultados da Geração da Rota	59
4.4	Limitações	60
5	CONSIDERAÇÕES FINAIS	62
5.1	Trabalhos Futuros	63
	REFERÊNCIAS	64

1 INTRODUÇÃO

Nas últimas décadas, a sociedade tem observado uma transformação na forma como a energia é produzida e consumida. Com o avanço da tecnologia e a crescente pressão por soluções sustentáveis, a busca por sistemas energéticos mais eficientes vem ganhando relevância em escala global. Em tal cenário, as fontes renováveis têm assumido um papel de destaque devido aos benefícios ambientais e ao aumento de 15,1% na capacidade global instalada em 2024 (WORLD ECONOMIC FORUM, 2025).

No Brasil, a transição para fontes renováveis tem se destacado pelo avanço da energia solar, que correspondeu a mais de 50% da expansão da matriz elétrica brasileira em 2024, e representa, em 2025, uma potência instalada de 56,5 GW em sistemas fotovoltaicos no modelo de geração centralizada e de geração distribuída (PORTAL SOLAR, 2024a). Na Figura 1, observa-se o crescimento da geração solar ao longo dos últimos anos, especialmente a partir de 2022. O avanço reflete a adoção de usinas solares como parte da estratégia nacional de diversificação energética.

Figura 1 – Evolução da Potência Solar Instalada no Brasil



Fonte: ABSOLAR (2025)

Contudo, o crescimento traz desafios técnicos. As usinas solares no modelo de geração centralizada ocupam áreas extensas e exigem monitoramento constante para assegurar o funcionamento adequado dos módulos. As inspeções manuais, além de demandarem tempo e recursos humanos consideráveis, também apresentam riscos à segurança dos trabalhadores.

A complexidade associada à inspeção tradicional dos sistemas evidencia a necessidade de soluções inteligentes e escaláveis. Tecnologias como visão computacional e aprendizado de

máquina oferecem alternativas inovadoras para automatizar o processo, permitindo a análise de imagens aéreas para identificar e mapear módulos fotovoltaicos com exatidão.

Este trabalho desenvolve uma solução computacional baseada em visão computacional e algoritmos clássicos de otimização para detectar automaticamente fileiras de módulos fotovoltaicos em imagens aéreas e gerar rotas de cobertura para equipes de manutenção em usinas solares de grande porte. A abordagem proposta utiliza a detecção das mesas fotovoltaicas como base para a construção de uma máscara navegável dos corredores, posteriormente convertida em um grafo ponderado sobre o qual é calculada a rota de inspeção. Assim, o trabalho busca contribuir para a redução do deslocamento das equipes em campo e para a otimização das operações de manutenção preventiva em usinas fotovoltaicas.

1.1 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- O Capítulo 1 apresenta o panorama do crescimento das usinas fotovoltaicas, os problemas associados às inspeções manuais em usinas de grande porte e os objetivos e justificativas do trabalho.
- O Capítulo 2 reúne a fundamentação teórica, abordando os sistemas fotovoltaicos, as redes neurais convolucionais, a detecção de objetos e a arquitetura YOLOv8, as métricas de avaliação de desempenho e o planejamento de rotas de cobertura em grafos, além da revisão de trabalhos correlatos.
- O Capítulo 3 descreve a implementação desenvolvida, detalhando o conjunto de dados, o treinamento do modelo de detecção, a geração da máscara navegável, a extração do esqueleto dos corredores, a construção do grafo ponderado e o cálculo da rota de cobertura.
- O Capítulo 4 apresenta os testes e os resultados obtidos em cada etapa do pipeline, desde a detecção das mesas fotovoltaicas até a geração da rota de manutenção.
- O Capítulo 5 apresenta as considerações finais e as perspectivas de trabalhos futuros.

1.2 OBJETIVOS E JUSTIFICATIVAS

1.2.1 OBJETIVO GERAL

Desenvolver um algoritmo de otimização de rotas baseado em visão computacional e modelos clássicos de otimização, capaz de reduzir distância de deslocamento de equipes de manutenção em grandes usinas solares.

1.2.2 OBJETIVOS ESPECÍFICOS

Alinhado ao objetivo geral, definem-se como objetivos específicos:

- a) Coletar e organizar um banco de dados com imagens aéreas de usinas solares de grande porte;
- b) Treinar um modelo de detecção de objetos;
- c) Detectar e delimitar automaticamente as fileiras de módulos fotovoltaicos nas imagens;
- d) Extrair os corredores de circulação entre as fileiras a partir das máscaras navegáveis;
- e) Gerar uma rota de cobertura que percorra os corredores entre as fileiras de módulos, minimizando a distância total de deslocamento.

1.2.3 JUSTIFICATIVAS

As usinas solares exigem monitoramento e manutenção constantes para garantir o funcionamento e a eficiência na geração de energia. No caso das usinas de grande porte, a necessidade torna-se evidente devido à vasta área ocupada, o que aumenta a complexidade das operações de inspeção e manutenção. No entanto, a maioria dos processos é realizada de forma manual, exigindo tempo, esforço físico e colocando os trabalhadores em situações de risco, como a exposição prolongada ao Sol.

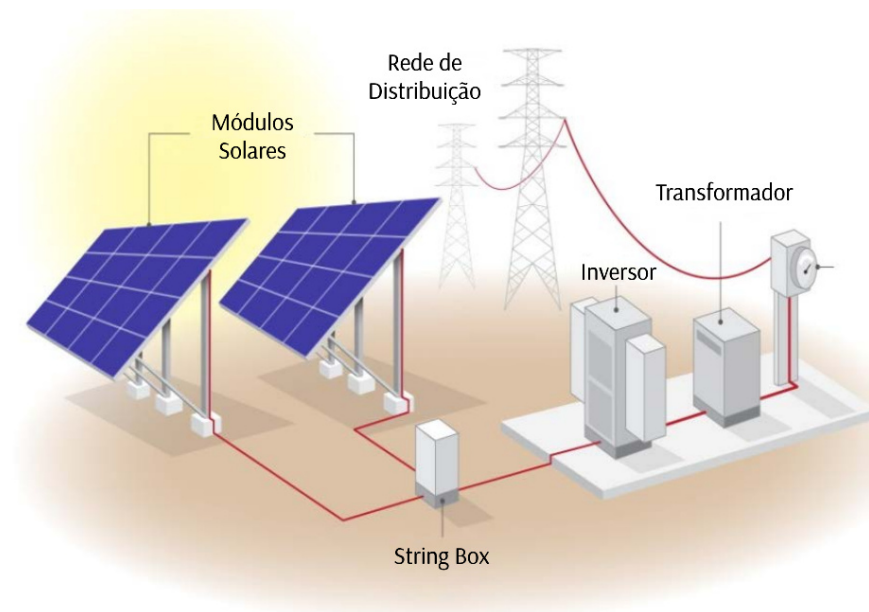
Diante do crescente número de usinas solares sendo instaladas, justifica-se otimizar os processos utilizados para manter o pleno funcionamento dos sistemas de geração. A aplicação de tecnologias inteligentes na identificação de módulos e na definição de rotas dentro de usinas solares apresenta uma alternativa para otimização do tempo de deslocamento de equipes, redução de falhas humanas e dos custos de processos de operação e manutenção.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMA FOTOVOLTAICO

Sistemas fotovoltaicos são conjuntos de componentes responsáveis por converter a luz solar em energia elétrica. A eletricidade gerada pode ser utilizada diretamente no sistema elétrico local ou armazenada em baterias para uso posterior. Os sistemas desempenham um papel fundamental na geração de energia limpa e renovável. O presente capítulo irá tratar dos componentes presentes em usinas solares de solo e de grande escala, conforme esquematizado na Figura 2 que necessitam de operações de rotina e de manutenção constante para geração de energia elétrica.

Figura 2 – Componentes de um sistema fotovoltaico.

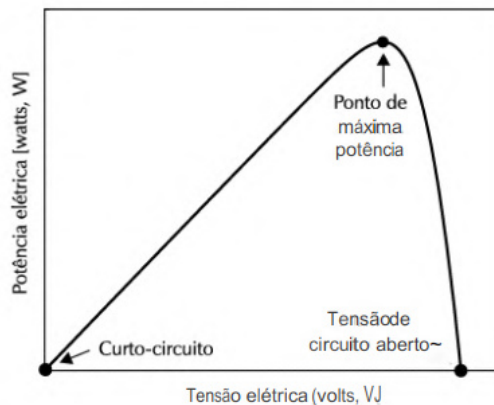


Fonte: Adaptado de: Kiatreungwattana, VanGeet e Stoltenberg (2016)

2.1.1 Módulo Fotovoltaico

O módulo fotovoltaico é formado por um conjunto de células solares interligadas eletricamente e montadas sobre uma estrutura rígida, geralmente em série, com o objetivo de elevar a tensão de saída. A principal função é converter a energia proveniente da radiação solar, captada pelas células, em tensão e corrente elétrica. A relação entre a tensão e a potência geradas por um módulo fotovoltaico é representada pela chamada curva P-V (potência versus tensão), que ilustra a potência gerada pelo módulo em função da tensão, proporcionando uma visão do desempenho do equipamento, conforme a Figura 3.

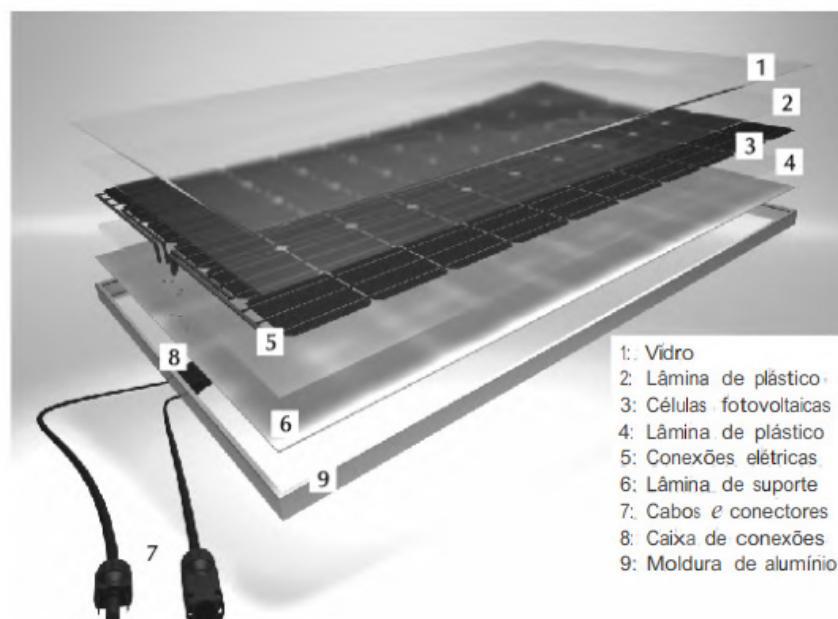
Figura 3 – Curva P-V de um módulo fotovoltaico



Fonte: Villalva e Gazoli (2012)

A estrutura física de um módulo fotovoltaico é resultado da combinação de diversos materiais e tecnologias projetados para garantir eficiência e resistência ao longo do tempo. As células solares, responsáveis pela conversão da energia luminosa em elétrica, são constituídas de silício encapsulado entre camadas protetoras, como o vidro temperado na parte frontal e o *backsheet* na parte posterior. As camadas protegem contra umidade, impactos e variações térmicas. A moldura, geralmente em alumínio anodizado, confere rigidez mecânica e facilita a fixação do módulo, prevenindo danos estruturais como trincas nas células (VILLALVA; GAZOLI, 2012). A qualidade dos materiais e da construção tem influência direta na durabilidade e no desempenho do módulo frente às condições ambientais e operacionais. A configuração de construção de um módulo fotovoltaico pode ser vista na Figura 4.

Figura 4 – Componentes de um módulo fotovoltaico



Fonte: Villalva e Gazoli (2012)

A energia gerada no módulo fotovoltaico é produzida em corrente contínua (CC) e necessita ser convertida em corrente alternada (CA), para ser utilizada em residências, comércios, prédios públicos, empresas e indústrias (WEG, 2025). O processo é realizado pelo inversor, um dispositivo eletrônico que transforma a corrente contínua em corrente alternada, compatível com os padrões da rede elétrica, tendo uma tensão de saída de 110V e 220V. Além de realizar a conversão, os inversores modernos também desempenham funções adicionais, como monitoramento do desempenho do sistema, proteção contra curto-circuito, proteção de reversão de polaridade.

Os módulos fotovoltaicos possuem dimensões métricas que variam conforme a potência e a tecnologia utilizada. Os modelos atuais de aproximadamente 500 Wp apresentam, em média, 2,20 m de comprimento, 1,10 m de largura e cerca de 0,04 m de espessura, resultando em uma área próxima de 2,40 m². Já módulos de maior potência, como os de 650 Wp a 700 Wp, têm dimensões superiores, alcançando tipicamente 2,30 m a 2,40 m de comprimento e 1,13 m a 1,20 m de largura, totalizando cerca de 2,70 a 2,90 m². A definição das dimensões está relacionada a aplicação, onde fatores como a área disponível, o tipo de estrutura, o objetivo do projeto e a estratégia de otimização energética são parâmetros importantes para caracterização (PORTAL ENERGIA, 2024).

2.2 USINAS FOTOVOLTAICAS DE GRANDE PORTE

Uma usina solar de grande porte, também conhecida como parque solar, é um sistema de geração de energia em larga escala, composto por milhares de módulos fotovoltaicos interligados e organizados em áreas extensas, projetadas para a produção e comercialização de energia elétrica.

As usinas solares podem ser catalogadas em dois modelos, Geração Distribuída (GD) e Geração Centralizada (GC), cada um apresentando características técnicas e operacionais distintas. A GD refere-se a sistemas fotovoltaicos de menor porte, até 5 MW de módulos solares, conectados diretamente à rede de distribuição, cuja energia é gerada próxima ao ponto de consumo ou no próprio local. A modalidade visa descentralizar a produção de energia, reduzindo perdas por transmissão e promovendo maior eficiência energética nas áreas urbanas e rurais (PORTAL SOLAR, 2024b).

De acordo com a regulamentação vigente, a GD é subdividida em duas categorias:

- Microgeração distribuída: sistemas com potência instalada de até 75 kW, utilizando fontes renováveis ou cogeração qualificada;
- Minigeração distribuída: sistemas com potência superior a 75 kW e inferior a 5 MW, também baseados em fontes renováveis ou cogeração qualificada.

Já a GC, se refere a instalações solares de produção de energia conectadas diretamente à rede de transmissão. O modelo é caracterizado pela concentração da geração em locais específicos, com posterior distribuição da energia por longas distâncias até os centros de consumo. Geralmente, as usinas solares em GC têm grande porte, extensas áreas e potenciais superiores a 5 MW de módulos solares, podendo chegar a centenas de MW e até GW de capacidade de geração elétrica (ABSOLAR, 2024).

A vantagem do sistema é a otimização de custos de implementação e na geração, contudo, apresenta algumas desvantagens (SOLARVOLT ENERGIA, 2024a):

- Custos e perdas na transmissão da energia.
- Impacto ambiental e social na implementação da planta.

Para reduzir perdas e manter a eficiência global do sistema, são necessárias rotinas de manutenção preventiva e corretiva, envolvendo atividades como a verificação da integridade dos módulos, a limpeza das superfícies e a análise do desempenho elétrico dos inversores e de outros componentes. A execução periódica das intervenções é fundamental para preservar a capacidade de geração, evitar falhas prematuras, prolongar a vida útil dos componentes e garantir a sustentabilidade técnica e econômica da planta ao longo de toda a operação (PORTAL SOLAR, 2024c).

2.2.1 Características das Usinas de Grande Porte

A estrutura de uma usina de grande porte utiliza os componentes básicos de um sistema fotovoltaico, porém, aplicados em grandes proporções.

Os equipamentos são divididos em três blocos principais (SOLARVOLT ENERGIA, 2024b):

- Bloco gerador: composto pelos módulos solares, estruturas de suporte e cabeamento. Os módulos são organizados em fileiras sobre estruturas fixas ao solo, com espaçamento entre elas que varia, em média, de 5,5 a 7,6 metros, formando corredores que permitem a circulação de equipes de manutenção, ventilação adequada e redução de sombreamento entre as fileiras.
- Bloco de condicionamento da potência: inclui os inversores, controladores de carga e transformadores, responsáveis por converter e adequar a energia gerada aos padrões da rede elétrica. O bloco tem a aplicação condicionada ao modelo de negócio implementado ou da necessidade do investidor, podendo não estar presente nas usinas solares de grande porte.
- Bloco de armazenamento: composto por baterias, utilizado em sistemas com necessidade de autonomia energética, *backup* ou operação em locais isolados.

A implantação dos blocos requer áreas de topografia regular, evitando-se relevos acentuados que causem sombreamento nas mesas. Além disso, a vegetação densa ou arbórea é eliminada para desobstruir os corredores de manutenção e não afetar a captação de luz solar, resultando em um cenário padronizado.

Os sistemas fotovoltaicos são construídos de forma modular, o que permite que novas seções possam ser adicionadas ou retiradas para manutenção, sem causar interrupções na operação da usina. A Figura 5 apresenta usina solar de solo e de grande porte, onde podem ser observados os módulos instalados em mesas e as travessas e ruas onde as equipes de manutenção operam.

Figura 5 – Disposição dos módulos em uma usina solar de grande porte



Fonte: PORTAL SOLAR (2025)

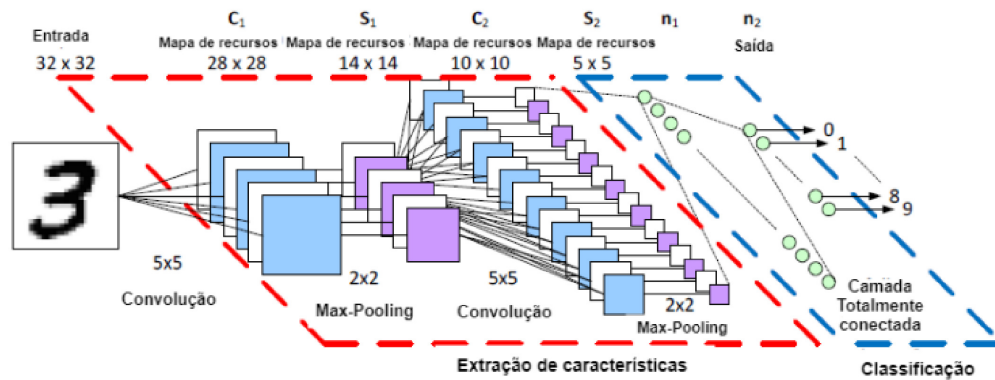
2.3 REDES NEURAIAS CONVOLUCIONAIS

As Redes Neurais Convolucionais (do inglês *Convolutional Neural Networks*) têm se consolidado como uma das abordagens mais representativas no campo do aprendizado profundo (LI *et al.*, 2020). As aplicações abrangem domínios variados, desde o reconhecimento facial, veículos autônomos, mercados com auto serviço e o diagnóstico médico assistido (LI *et al.*, 2020). O que distingue as CNNs de outras arquiteturas é a capacidade de extrair automaticamente representações hierárquicas a partir dos dados brutos: nas camadas iniciais, a rede identifica elementos primitivos como bordas, texturas e variações de intensidade, nas camadas mais profundas, os elementos se combinam para formar representações progressivamente mais abstratas, como formas, objetos e padrões estruturais complexos (GOODFELLOW; BENGIO; COURVILLE, 2016; CHOLLET, 2018).

Na Figura 6 é apresentada a estrutura típica de uma CNN. O processamento tem início na camada convolucional, responsável pela extração de características utilizando filtros sobre a imagem de entrada, gerando os chamados mapas de características (*feature maps*). Posteriormente, a camada de *pooling* reduz a dimensionalidade espacial dos mapas, diminuindo o custo

computacional e conferindo robustez a pequenas variações na posição dos objetos. A camada de achatamento (*flatten*) converte então os mapas multidimensionais em um vetor unidimensional, que serve de entrada para a camada totalmente conectada, onde ocorre a classificação final (ELGENDY, 2020; CHOLLET, 2018).

Figura 6 – Estrutura de uma CNN

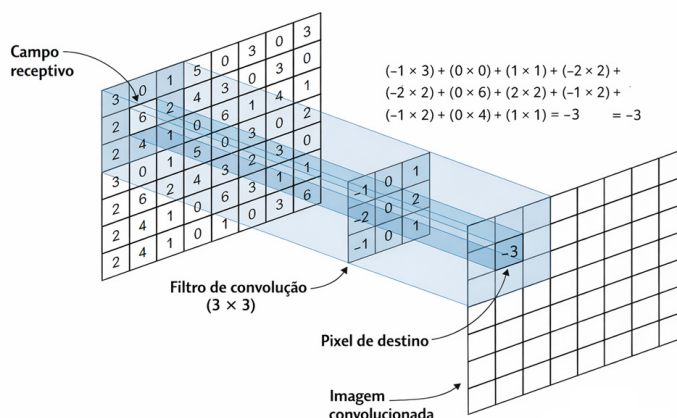


Fonte: Adaptado de: Peemen, Mesman e Corporaal (2011)

2.3.1 Camada de Convolução

A camada de convolução é o bloco principal de uma CNN, atuando como um mecanismo de busca de características que analisa a imagem de entrada e extrai informações relevantes para a identificação dos objetos presentes (ELGENDY, 2020). O funcionamento baseia-se na aplicação de filtros, denominados *kernels*, pequenas matrizes de pesos que são inicializadas aleatoriamente e aprendidas pela rede ao longo do treinamento, sem necessidade de definição manual (ELGENDY, 2020). A Figura 7 ilustra o processo, onde um filtro 3×3 desliza sobre a imagem de entrada gerando a imagem convoluída na camada seguinte.

Figura 7 – Filtro convolucional deslizando sobre a imagem de entrada

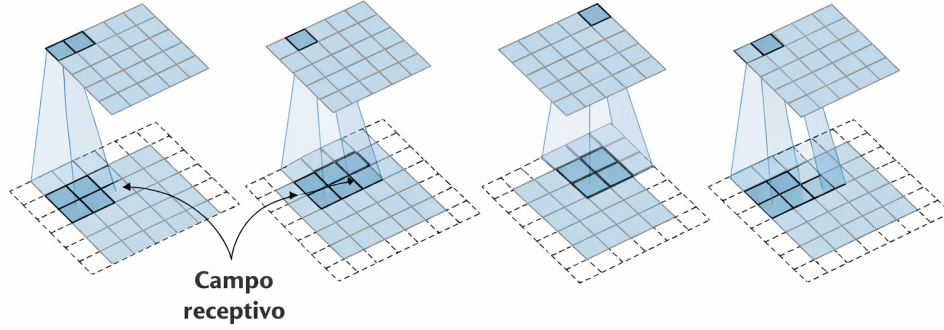


Fonte: Adaptado de: Elgendy (2020)

A região da imagem sobre a qual o filtro opera em cada posição é denominada campo receptivo (*receptive field*). O filtro percorre toda a extensão da imagem e, a cada posição, mul-

tiplica cada pixel do campo receptivo pelo valor correspondente no *kernel* e soma todos os resultados, gerando um único valor na saída. O conjunto de todos os valores forma o mapa de características (*feature map*), também chamado de mapa de ativação (ELGENDY, 2020). A Figura 8 representa o processo, evidenciando como o campo receptivo se desloca sobre a imagem original para construir a imagem convoluída.

Figura 8 – Campo receptivo do filtro convolucional



Fonte: Adaptado de: Elgendy (2020)

Como imagens digitais são discretas, a operação de convolução aplicada em redes neurais é expressa pela convolução discreta. Para uma imagem bidimensional I e um filtro bidimensional K , a operação é definida pelo somatório (SZELISKI, 2010):

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \quad (2.1)$$

O tamanho do mapa de características gerado na saída é determinado pelos hiperparâmetros da camada, sendo calculado por (LI *et al.*, 2020):

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \quad (2.2)$$

onde i é o tamanho da entrada, p é o *padding*, k é o tamanho do *kernel* e s é o *stride*.

Os *kernels* não são definidos manualmente, os valores são inicializados aleatoriamente e aprendidos pela rede durante o treinamento, o que permite que ela descubra automaticamente quais características são relevantes (ELGENDY, 2020). Diferentes filtros detectam diferentes tipos de características: nas camadas iniciais identificam elementos simples como bordas e linhas, enquanto nas camadas mais profundas capturam padrões mais complexos como formas e texturas (ELGENDY, 2020; ZHAO *et al.*, 2024). Cada filtro gera um mapa de características distinto, e o número de filtros em uma camada determina a profundidade da saída da camada (ELGENDY, 2020). O comportamento da operação é controlado por quatro hiperparâmetros: o número de filtros, o tamanho do *kernel*, geralmente 2×2 , 3×3 ou 5×5 , o *stride*, que define o passo de deslocamento do filtro, e o *padding*, que adiciona pixels nas bordas da entrada para evitar a redução progressiva do mapa de características (ELGENDY, 2020; ZHAO *et al.*, 2024).

2.3.2 Função de Ativação

A operação de convolução é linear, o que significa que, sem nenhuma transformação adicional, empilhar várias camadas convolucionais seria equivalente a aplicar uma única transformação linear, limitando a capacidade da rede de aprender padrões complexos (GOODFELLOW; BENGIO; COURVILLE, 2016). Para resolver o problema, após cada camada de convolução é aplicada uma função de ativação, cuja finalidade é introduzir não-linearidade ao modelo, permitindo que a rede aprenda relações mais eficientes entre as características extraídas (CHOLLET, 2018).

Dentre as funções de ativação existentes, como a Sigmoide e a Tangente Hiperbólica (Tanh), a Unidade Linear Retificada (*Rectified Linear Unit* ReLU) tornou-se o padrão nas CNNs modernas (GOODFELLOW; BENGIO; COURVILLE, 2016). A Sigmoide comprime os valores de entrada para o intervalo $(0, 1)$, enquanto a Tanh os mapeia para $(-1, 1)$. Ambas apresentam um problema crítico em redes profundas: quando o valor de entrada é muito grande ou muito pequeno, o gradiente das funções se torna próximo de zero, dificultando a atualização dos pesos durante o treinamento, fenômeno conhecido como desvanecimento do gradiente (*vanishing gradient*) (ZHAO *et al.*, 2024; GOODFELLOW; BENGIO; COURVILLE, 2016). A ReLU resolve o problema de forma eficiente, sendo definida pela equação:

$$f(x) = \max(0, x) \quad (2.3)$$

O funcionamento é direto: quando o valor de entrada x é maior ou igual a zero, a saída é o próprio valor, quando x é negativo, a saída é zero. Assim, a derivada da ReLU é constante e igual a 1 para valores positivos, o que evita o desvanecimento do gradiente e acelera significativamente a convergência do treinamento em comparação com a Sigmoide e a Tanh (ELGENDY, 2020; LI *et al.*, 2020). Além disso, a ReLU promove a esparsidade das ativações, como muitas saídas são zero, apenas os neurônios relevantes são ativados, o que reduz o custo computacional e melhora a capacidade de generalização da rede (CHOLLET, 2018).

Apesar das vantagens, a ReLU apresenta uma limitação conhecida como *Dead ReLU*: quando x é negativo, o gradiente é zero, o que significa que o erro retropropagado é multiplicado por zero e nenhuma informação é passada às camadas anteriores, podendo levar neurônios a não serem ativados durante o treinamento (ZHAO *et al.*, 2024). Para contornar o problema foram propostas variantes como a *Leaky ReLU*, que atribui um pequeno valor linear às entradas negativas em vez de zerá-las, estendendo assim o intervalo de ativação da função (LI *et al.*, 2020). Apesar das alternativas, a ReLU permanece como a função de ativação padrão nas CNNs devido à simplicidade e eficiência computacional (ELGENDY, 2020; GOODFELLOW; BENGIO; COURVILLE, 2016).

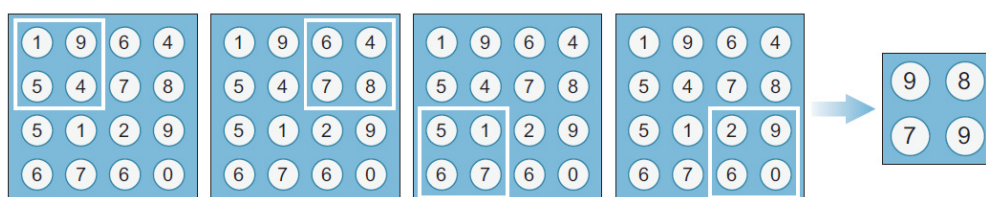
2.3.3 Camada de *pooling*

À medida que camadas convolucionais são empilhadas, o número de mapas de características cresce rapidamente, podendo conter dezenas ou centenas de filtros, cada um com os próprios pesos. Isso faz com que a dimensionalidade da rede aumente rapidamente, tornando o processo de aprendizado computacionalmente custoso. É no contexto que a camada de *pooling*, também chamada de subamostragem, se torna necessária (ELGENDY, 2020). O objetivo é reduzir o tamanho dos mapas de características produzidos pela camada convolucional, diminuindo o número de parâmetros repassados à camada seguinte e, consequentemente, reduzindo a complexidade computacional do modelo (ELGENDY, 2020). Além disso, a camada de *pooling* torna a rede mais robusta a pequenas variações de posição, escala e distorção dos objetos na imagem, conferindo ao modelo uma maior capacidade de generalização (ZHAO *et al.*, 2024). É uma prática comum adicionar uma camada de *pooling* após cada uma ou duas camadas convolucionais na arquitetura da CNN (ELGENDY, 2020).

Os dois métodos de *pooling* mais utilizados são o *Max Pooling* e o *Average Pooling* (ZHAO *et al.*, 2024). O *Max Pooling* funciona de forma similar aos *kernels* convolucionais: uma janela de tamanho fixo desliza sobre o mapa de características com um determinado *stride* e seleciona apenas o valor máximo de cada região percorrida, descartando os demais valores (ELGENDY, 2020). Já o *Average Pooling* calcula a média aritmética de todos os elementos da região coberta pela janela, retornando um valor que representa a resposta média daquela área (ZHAO *et al.*, 2024).

A Figura 9 ilustra o processo de *Max Pooling* com janela 2×2 e *stride* de 2, aplicado sobre um mapa de entrada 4×4 . A janela percorre o mapa em quatro posições distintas, selecionando o valor máximo de cada região: na primeira região superior esquerda, os valores 1, 9, 5 e 4 resultam no valor 9, na região superior direita, os valores 6, 4, 7 e 8 resultam no valor 8, na região inferior esquerda, os valores 5, 1, 6 e 7 resultam no valor 7, e na região inferior direita, os valores 2, 9, 6 e 0 resultam no valor 9. O mapa de saída resultante possui dimensão 2×2 , reduzindo a dimensionalidade espacial à metade enquanto preserva os valores mais representativos de cada região (ELGENDY, 2020).

Figura 9 – *Max Pooling* com janela 2×2 e *stride* 2



Fonte: Adaptado de: Elgendy (2020)

É importante destacar que, ao contrário das camadas convolucionais, a camada de *pooling* não possui pesos treináveis, ela apenas aplica uma função estatística sobre os valores

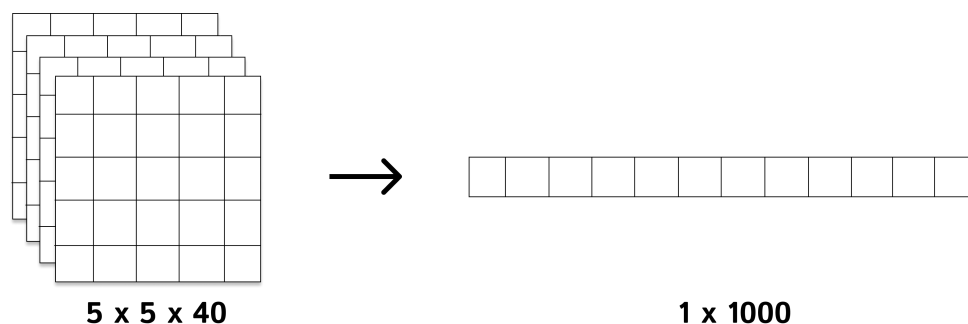
recebidos (ELGENDY, 2020). Quando o *pooling* é aplicado a todos os mapas de características de uma camada convolucional, a profundidade da saída é mantida, pois o filtro de *pooling* é aplicado individualmente a cada mapa, apenas as dimensões espaciais, largura e altura, são reduzidas (ELGENDY, 2020).

2.3.4 Camada de Achatamento

Após as múltiplas etapas de convolução e *pooling*, os mapas de características resultantes representam abstrações de alto nível da imagem original, organizadas em um tensor tridimensional com dimensões de largura, altura e profundidade. No entanto, as camadas totalmente conectadas que realizam a classificação final operam sobre vetores unidimensionais, e não sobre tensores multidimensionais (ELGENDY, 2020). A camada de achatamento (*Flatten*) é responsável por fazer a conversão: ela reorganiza todos os valores do tensor de características em um único vetor unidimensional, sem realizar nenhum cálculo ou aprendizado, é uma operação puramente estrutural (ELGENDY, 2020).

O processo pode ser entendido da seguinte forma: se os mapas de características ao final das camadas convolucionais e de *pooling* possuem dimensões $H \times W \times D$, onde H é a altura, W é a largura e D é a profundidade, ou seja, o número de mapas, a camada de achatamento transforma o tensor em um vetor de dimensão $(1, H \times W \times D)$ (ELGENDY, 2020). Por exemplo, um tensor de dimensões $5 \times 5 \times 40$ resulta em um vetor de 1.000 elementos, que será então repassado à camada totalmente conectada para classificação (ELGENDY, 2020). A Figura 10 ilustra o processo, mostrando como os mapas de características são reorganizados em um vetor antes de serem processados pelas camadas densas.

Figura 10 – Processo de achatamento dos mapas de características



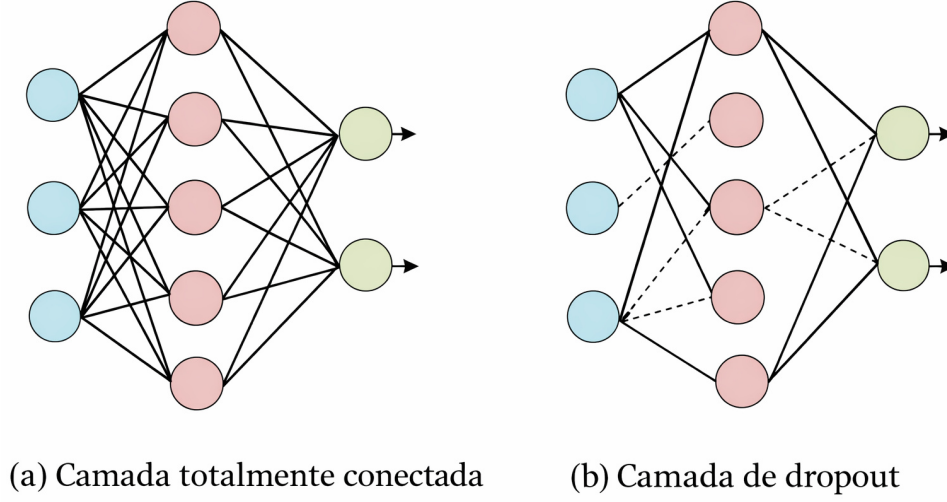
Fonte: O Autor (2026).

2.3.5 Camada Totalmente Conectada e Classificação

Após a etapa de achatamento, o vetor unidimensional resultante é repassado para uma ou mais camadas totalmente conectadas (do inglês *Fully Connected*), também denominadas camadas densas. Nas camadas, cada neurônio está conectado a todos os neurônios da camada anterior, e as conexões entre eles são ponderadas por pesos que representam a importância

relativa de cada entrada para a decisão final (ELGENDY, 2020). A Figura 11 ilustra a estrutura, onde cada nó de uma camada está ligado a todos os nós da camada seguinte. A função das camadas é integrar e combinar todas as características extraídas pelas camadas convolucionais e de *pooling* para realizar a classificação final da imagem (ZHAO *et al.*, 2024).

Figura 11 – Camada totalmente conectada e camada de *Dropout*



Fonte: Adaptado de: Zhao *et al.* (2024)

Internamente, cada neurônio da camada totalmente conectada recebe como entrada o vetor unidimensional proveniente da camada de achatamento, onde cada valor x_i é multiplicado pelo peso correspondente w_i , que indica a importância relativa. Em seguida, todos os produtos são somados e adiciona-se um valor b , denominado *bias*, que permite ajustar a saída da transformação e aumentar a flexibilidade do modelo (ELGENDY, 2020). A operação é expressa pela equação:

$$z = \sum_{i=1}^n w_i x_i + b \quad (2.4)$$

onde z é a entrada líquida do neurônio, w_i são os pesos, x_i são os valores de entrada e b é o *bias*. A camada concentra a maior parte dos parâmetros treináveis da CNN, podendo envolver milhões de pesos a serem ajustados, o que impacta diretamente na complexidade computacional do modelo (ELGENDY, 2020).

Na camada de saída, última camada totalmente conectada, o número de neurônios é igual ao número de classes do problema. Para classificação multiclasse, utiliza-se a função *Softmax*, que transforma os valores brutos produzidos pelos neurônios em uma distribuição de probabilidades, onde cada valor representa a probabilidade de a imagem pertencer a uma determinada classe (ZHAO *et al.*, 2024). A função *Softmax* é definida por:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (2.5)$$

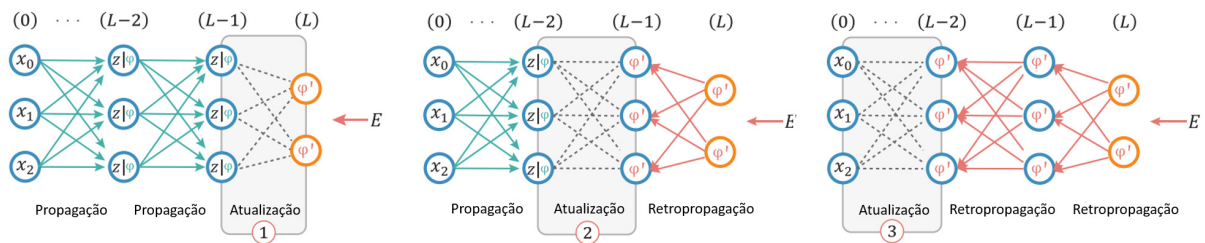
onde x_i é o valor do i -ésimo neurônio de saída e n é o número total de classes. A classe com maior probabilidade é adotada como a predição final da rede (ZHAO *et al.*, 2024). Para evitar *overfitting*, especialmente quando o modelo é treinado em conjuntos de dados pequenos, é comum aplicar *Dropout* nas camadas totalmente conectadas, técnica que desativa aleatoriamente uma fração dos neurônios durante o treinamento, forçando a rede a aprender representações mais robustas (ELGENDY, 2020; ZHAO *et al.*, 2024).

2.3.6 Treinamento e *Backpropagation*

O treinamento de uma CNN consiste em ajustar iterativamente os pesos de todos os filtros e camadas da rede de forma a minimizar o erro entre as predições do modelo e os rótulos reais do conjunto de dados (GOODFELLOW; BENGIO; COURVILLE, 2016). O processo é realizado pelo algoritmo de retropropagação do erro (*Backpropagation*), que opera em conjunto com o método do gradiente descendente para atualizar os parâmetros da rede (HAYKIN, 2009).

Cada iteração de treinamento envolve duas passagens principais. Na propagação direta (*forward pass*), a imagem de entrada percorre a rede camada por camada, desde a convolução até a camada de saída, produzindo uma predição. Em seguida, calcula-se o erro entre a predição obtida e o rótulo esperado por meio de uma função de custo, como a entropia cruzada (*cross-entropy*), amplamente utilizada em problemas de classificação (GOODFELLOW; BENGIO; COURVILLE, 2016). Na retropropagação (*backward pass*), o erro é propagado de volta pela rede, camada por camada, calculando o gradiente da função de custo em relação a cada peso por meio da regra da cadeia (HAYKIN, 2009). A Figura 12 ilustra o ciclo de treinamento com as três etapas: propagação direta, atualização e retropropagação.

Figura 12 – Ciclo de treinamento *Backpropagation*



Fonte: Adaptado de: Doremaele *et al.* (2024)

Com os gradientes calculados, os pesos são atualizados na direção que minimiza o erro, conforme a equação:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \frac{\partial E}{\partial w_{ij}} \quad (2.6)$$

onde $w_{ij}^{(t)}$ é o peso entre os neurônios i e j na iteração t , η é a taxa de aprendizado, E é a função de erro e $\frac{\partial E}{\partial w_{ij}}$ é o gradiente do erro em relação ao peso (HAYKIN, 2009). O processo é repetido ao longo de múltiplas épocas de treinamento até que a rede atinja um desempenho satisfatório (GOODFELLOW; BENGIO; COURVILLE, 2016).

A taxa de aprendizado η é um hiperparâmetro crítico: valores muito altos podem fazer o modelo divergir, ignorando o mínimo da função de custo, valores muito baixos tornam o treinamento excessivamente lento (ZHAO *et al.*, 2024). Para contornar os problemas, otimizadores adaptativos como o Adam e o AdamW são amplamente utilizados em CNNs modernas, pois ajustam automaticamente a taxa de aprendizado de forma individual para cada parâmetro ao longo do treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.7 Generalização e Aumento de Dados

Um dos principais desafios no treinamento de CNNs é garantir que o modelo não apenas aprenda os padrões presentes no conjunto de treinamento, mas também seja capaz de generalizar para dados novos, não vistos durante o treinamento (CHOLLET, 2018). A capacidade de generalização está diretamente relacionada à diversidade e ao volume dos dados disponíveis: quanto mais variadas forem as amostras apresentadas durante o treinamento, maior a chance de o modelo aprender representações que se apliquem a situações reais (GOODFELLOW; BENGIO; COURVILLE, 2016).

Quando o conjunto de dados é pequeno ou pouco diverso, o modelo tende a memorizar os exemplos de treinamento em vez de aprender os padrões subjacentes. No cenário, ocorre o fenômeno denominado *overfitting* (sobreajuste): o modelo apresenta alto desempenho nos dados de treino, mas falha ao ser aplicado a dados novos, pois capturou ruídos e variações específicas daquele conjunto como se fossem padrões reais (GOODFELLOW; BENGIO; COURVILLE, 2016). Na prática, isso se manifesta quando a acurácia no conjunto de treino continua crescendo enquanto a acurácia no conjunto de validação estagna ou diminui (CHOLLET, 2018).

Uma das abordagens mais eficazes para mitigar o *overfitting* e melhorar a capacidade de generalização é o *Data Augmentation* (aumento de dados), técnica que consiste em gerar novas instâncias de treinamento a partir das imagens existentes por meio de transformações aleatórias (CHOLLET, 2018). O objetivo é garantir que, durante o treinamento, o modelo nunca veja exatamente a mesma imagem duas vezes, expondo-o a uma maior variedade de aspectos dos dados (CHOLLET, 2018). O *Data Augmentation* é considerado uma técnica de regularização pois, ao apresentar ao modelo diversas variações de um mesmo objeto, reduz a dependência da forma original durante o aprendizado de características, tornando-o mais robusto quando testado em dados novos (ELGENDY, 2020).

As técnicas de aumento de dados mais comuns em visão computacional incluem inversão horizontal e vertical, rotação em diferentes ângulos, ajustes de brilho, contraste e saturação,

zoom, recortes parciais e distorções geométricas (ELGENDY, 2020; CHOLLET, 2018). É importante destacar que as transformações são aplicadas apenas durante o treinamento, na etapa de inferência, o modelo opera sobre as imagens originais sem qualquer modificação (CHOLLET, 2018).

2.3.8 Aprendizado por Transferência e *Fine-Tuning*

O treinamento de redes neurais profundas do zero exige grandes volumes de dados anotados e elevado custo computacional, o que pode ser inviável em cenários com *datasets* de tamanho limitado (ELGENDY, 2020). O Aprendizado por Transferência (*Transfer Learning*) é uma estratégia que permite reutilizar o conhecimento adquirido por um modelo treinado em um domínio de origem para acelerar e melhorar o aprendizado em um domínio de destino relacionado (CHOLLET, 2018).

Na prática, modelos de detecção de objetos como o YOLOv8 são disponibilizados com pesos pré-treinados no conjunto de dados *COCO* (*Common Objects in Context*), composto por mais de 330 mil imagens de 80 categorias de objetos do cotidiano (ELGENDY, 2020). Durante o pré-treinamento, as camadas convolucionais do modelo aprendem representações visuais genéricas e progressivamente mais abstratas: nas camadas iniciais são detectados elementos primitivos como bordas e texturas, enquanto nas camadas mais profundas formam-se representações de formas e padrões estruturais complexos (CHOLLET, 2018). O conhecimento visual é transferível entre domínios, pois bordas e texturas são características presentes em qualquer tipo de imagem, incluindo ortofotos de usinas fotovoltaicas.

O processo de *fine-tuning* consiste em inicializar o modelo com os pesos pré-treinados e continuar o treinamento com o *dataset* específico da tarefa de destino, permitindo que os pesos se ajustem às características do novo domínio (CHOLLET, 2018). A abordagem apresenta duas vantagens principais em relação ao treinamento do zero: a convergência ocorre de forma mais rápida, pois o modelo já parte de uma representação visual informativa, e o risco de *overfitting* é reduzido, pois as representações genéricas aprendidas no pré-treinamento funcionam como uma forma de regularização implícita (ELGENDY, 2020). As vantagens tornam o *fine-tuning* especialmente adequado para cenários com conjuntos de dados anotados de tamanho limitado, como é o caso de aplicações em domínios específicos como inspeção de usinas solares.

2.3.9 Detecção de Objetos

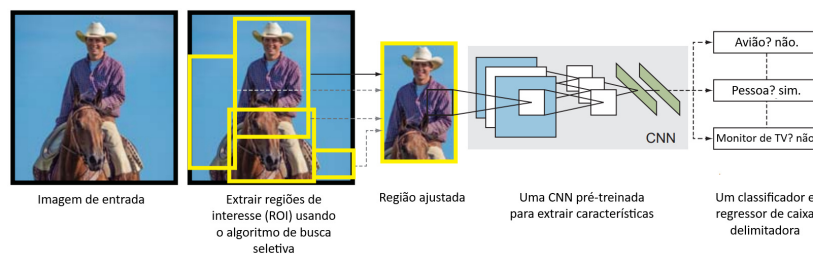
As CNNs, apresentadas nas seções anteriores, formam a base computacional para uma das tarefas mais relevantes da visão computacional: a detecção de objetos. Diferentemente da classificação de imagens, que atribui um rótulo à imagem como um todo sem indicar onde os objetos se encontram, a detecção de objetos tem como objetivo não apenas identificar a presença de objetos em uma imagem, mas também determinar a localização por meio de delimitadores

espaciais, geralmente representados por *bounding boxes* (ELGENDY, 2020). A técnica combina dois elementos essenciais: a classificação do objeto e a localização precisa dentro da imagem analisada.

É importante distinguir a detecção de objetos de outras tarefas correlatas. A classificação de imagens identifica um ou mais rótulos em uma imagem como um todo, sem indicar onde os objetos se encontram. Já a segmentação de imagens busca atribuir um rótulo a cada pixel da imagem, permitindo a separação precisa dos contornos dos objetos, sendo dividida em segmentação semântica e segmentação de instâncias. Assim, a detecção de objetos representa uma abordagem intermediária, sendo mais precisa que a simples classificação e menos detalhada que a segmentação pixel a pixel (ELGENDY, 2020).

Com o avanço das CNNs, diversos modelos foram propostos para a detecção de objetos. Os modelos baseados em duas etapas, como o *R-CNN (Regions with CNN features)* e as variantes *Fast R-CNN* e *Faster R-CNN*, realizam primeiro a seleção de regiões de interesse (ROIs) e, posteriormente, a classificação e ajuste das caixas delimitadoras (ELGENDY, 2020). A Figura 13 apresenta as etapas de detecção do modelo R-CNN. Os modelos apresentaram alto desempenho em termos de acurácia, porém com custo computacional elevado, o que os torna inadequados para aplicações que exigem resposta em tempo real.

Figura 13 – Resumo da arquitetura do modelo R-CNN



Fonte: Adaptado de: Elgendy

(2020)

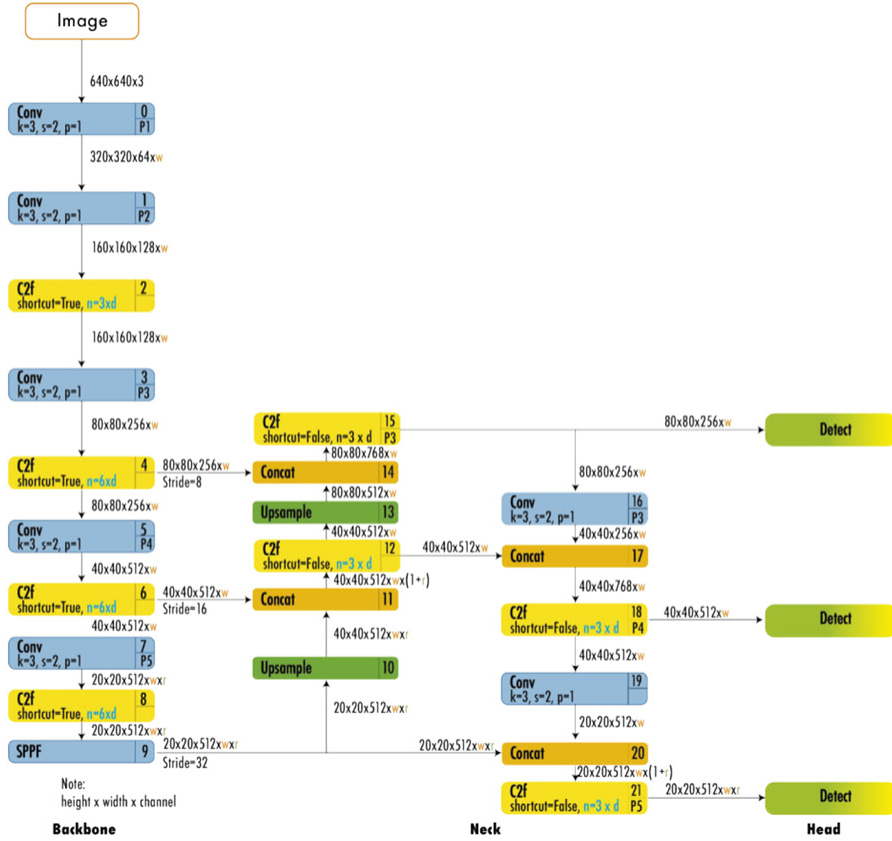
Buscando maior desempenho em tempo real, surgiram os detectores de uma etapa, como o *SSD (Single Shot MultiBox Detector)* e a família *YOLO (You Only Look Once)*. Os modelos realizam a detecção e a classificação em uma única etapa de inferência (REDMON *et al.*, 2016), o que os torna significativamente mais rápidos e adequados para aplicações que exigem resposta em tempo real, como inspeções por drones em usinas fotovoltaicas. A arquitetura YOLO, por conta da velocidade e precisão, tornou-se uma das abordagens mais adotadas na literatura para detecção de objetos em imagens aéreas (REDMON *et al.*, 2016; ELGENDY, 2020), sendo a base da solução desenvolvida no trabalho e detalhada na seção seguinte.

2.3.10 Arquitetura YOLOv8

O YOLO (*You Only Look Once*) é uma família de modelos de detecção de objetos que revolucionou a área ao reformular o problema de detecção como uma única regressão direta sobre a imagem, eliminando a necessidade de etapas separadas para proposta de regiões e classificação (REDMON *et al.*, 2016). Na versão original, a imagem de entrada é dividida em uma grade $S \times S$ células, onde cada célula é responsável por prever múltiplas *bounding boxes* com as respectivas pontuações de confiança e probabilidades de classe, tudo em uma única passagem pela rede (REDMON *et al.*, 2016). A abordagem unificada permitiu alcançar velocidades de inferência em tempo real, processando imagens a 45 quadros por segundo na versão base, e com menor número de erros de falsos positivos em relação a métodos anteriores como o Fast R-CNN (REDMON *et al.*, 2016).

O YOLOv8, lançado pela Ultralytics em janeiro de 2023, representa uma evolução significativa na família YOLO, introduzindo inovações arquiteturais que aprimoram tanto a precisão quanto a eficiência do modelo em relação às versões anteriores (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Ele está disponível em cinco variantes de escala *nano* (n), *small* (s), *medium* (m), *large* (l) e *extra-large* (x) e suporta múltiplas tarefas de visão computacional, como detecção de objetos, segmentação, estimativa de pose e classificação (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). A arquitetura do YOLOv8 é organizada em três componentes principais: o *Backbone*, o *Neck* e a *Head*, conforme ilustrado na Figura 14 (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

Figura 14 – Arquitetura completa do YOLOv8: *Backbone*, *Neck* e *Head*



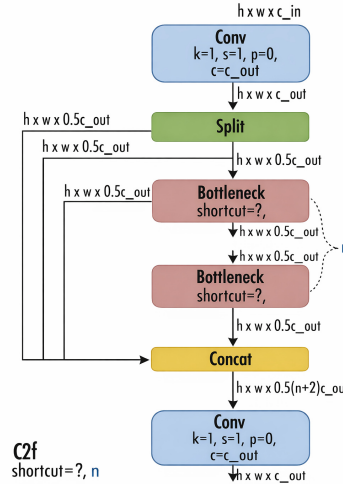
Fonte: Adaptado de: Terven, Córdoba-Esparza e Romero-González (2023)

O *Backbone* é a parte da rede responsável por receber a imagem de entrada e extrair progressivamente as características em diferentes níveis de abstração (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). O YOLOv8 utiliza uma versão modificada da rede CSP-Darknet53 como *backbone*, composta por blocos Conv, módulos C2f e um módulo SPPF ao final (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Os blocos Conv são camadas convolucionais com filtro 3×3 , seguidas de normalização em lote (*BatchNorm*) e função de ativação SiLU (*Sigmoid Linear Unit*), e são responsáveis por reduzir progressivamente as dimensões espaciais dos mapas de características à medida que a imagem avança pelas camadas (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

O módulo C2f (*Cross-Stage Partial with 2 convolutions*) é a principal inovação do YOLOv8 em relação ao YOLOv5, onde substituiu o módulo C3 (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Conforme ilustra a Figura 15, o módulo C2f recebe um mapa de características e o divide em duas partes por meio de uma operação *Split*: uma parte segue diretamente para a saída, enquanto a outra é processada por uma sequência de blocos *Bottleneck*, que aplicam duas convoluções com conexões residuais, extraindo características como formas e texturas (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Ao final, todas as saídas são agrupadas por uma operação de concatenação (*Concat*) e submetidas a uma última convo-

lução. A estrutura combina características de alto nível com informações contextuais, melhora o fluxo de gradientes durante o treinamento e permite aprendizado de representações mais ricas com menor custo computacional (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

Figura 15 – Módulos internos da arquitetura YOLOv8: C2f, Bottleneck, SPPF, Conv e cabeça desacoplada



Fonte: Adaptado de: Terven, Córdoba-Esparza e Romero-González (2023)

O módulo SPPF (*Spatial Pyramid Pooling Fast*), presente ao final do *Backbone*, aplica múltiplas operações de *MaxPool2d* em sequência sobre a mesma entrada e concatena os resultados, o que permite capturar características em diferentes escalas espaciais sem aumentar o custo computacional (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). A operação amplia o campo receptivo efetivo da rede, melhorando a capacidade do modelo de detectar objetos de diferentes tamanhos.

O *Neck* é o componente que conecta o *Backbone* à *Head* e tem como objetivo fundir os mapas de características extraídos em diferentes escalas (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Para isso, o YOLOv8 utiliza a estrutura PAN-FPN (*Path Aggregation Feature Pyramid Network*), que combina operações de *Upsample* e *Concat* para criar caminhos de informação em duas direções: de cima para baixo, propagando características semânticas de alto nível para camadas anteriores, e de baixo para cima, preservando informações espaciais detalhadas das camadas iniciais (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). A fusão multi-escala é essencial para que o modelo consiga detectar tanto objetos grandes quanto objetos pequenos com boa precisão.

A *Head* é a parte final da rede, responsável por gerar as predições de *bounding boxes*, classes e pontuações de confiança a partir dos mapas de características fornecidos pelo *Neck* (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Uma das mudanças mais importantes do YOLOv8 em relação às versões anteriores é a adoção de uma cabeça desacoplada (*decoupled head*): os ramos de classificação e de regressão das *bounding boxes* são processados de

forma independente, o que melhora a convergência e a precisão do modelo (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Além disso, o YOLOv8 adota uma abordagem *anchor-free*, eliminando a necessidade de âncoras predefinidas que precisavam ser configuradas manualmente em versões anteriores (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). A *Head* possui três blocos de detecção (*Detect*), cada um responsável por detectar objetos em uma escala diferente, pequena (P3), média (P4) e grande (P5), possibilitando a detecção de objetos de tamanhos variados em uma mesma imagem (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023). Para a função de perda, o YOLOv8 utiliza CIoU e DFL (*Distribution Focal Loss*) para regressão das caixas delimitadoras, e entropia cruzada binária para classificação (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

Após a inferência, as predições passam por um pós-processamento baseado na Supressão Não Máxima (do inglês *Non-Maximum Suppression*), que elimina *bounding boxes* redundantes com alta sobreposição, retraindo apenas as detecções mais representativas para cada objeto detectado (REDMON *et al.*, 2016).

2.3.11 Métricas de Avaliação de Desempenho

Para avaliar o desempenho de modelos de detecção de objetos, utilizam-se métricas padrão estabelecidas pelo desafio *Microsoft COCO* (do inglês *Common Objects in Context*) (ELGENDY, 2020). As métricas partem do conceito de *Intersection over Union* (IoU), que é a base para classificar cada predição do modelo como correta ou incorreta.

O IoU mede o grau de sobreposição entre a *bounding box* predita pelo modelo (B_{pred}) e a *bounding box* real anotada manualmente (B_{gt}), chamada de *ground truth*. É calculado pela razão entre a área de interseção e a área de união das duas caixas (ELGENDY, 2020):

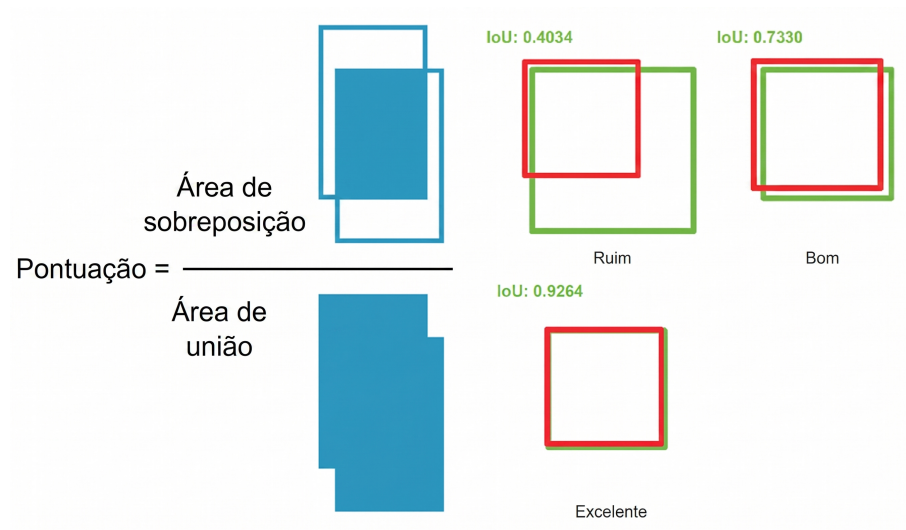
$$IoU = \frac{|B_{pred} \cap B_{gt}|}{|B_{pred} \cup B_{gt}|} \quad (2.7)$$

O valor do IoU varia de 0 a 1, onde 0 indica nenhuma sobreposição e 1 indica sobreposição perfeita. A Figura 16 ilustra o cálculo do IoU e exemplos de diferentes níveis de sobreposição.

O IoU é utilizado para classificar cada predição em uma das seguintes categorias, a partir de um limiar de sobreposição (*threshold*) definido (ELGENDY, 2020): a predição é considerada um Verdadeiro Positivo (TP) quando o IoU entre a caixa predita e a *ground truth* é maior ou igual ao limiar, ou seja, a detecção é correta, é classificada como Falso Positivo (FP) quando o modelo detecta um objeto mas o IoU é inferior ao limiar, indicando que a detecção não corresponde adequadamente a nenhum objeto real, e é registrada como Falso Negativo (FN) quando existe um objeto real na imagem que o modelo não detectou (ELGENDY, 2020).

A partir das classificações, a Precisão (*Precision*) indica a proporção de detecções cor-

Figura 16 – Cálculo do *Intersection over Union* (IoU) entre a caixa predita e a caixa real



Fonte: Adaptado de: Elgendy (2020)

retas em relação ao total de detecções realizadas, medindo o quanto o modelo pode ser confiado quando aponta a presença de um objeto (ELGENDY, 2020):

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.8)$$

Já a Revocação (*Recall*) mede a capacidade do modelo de encontrar todos os objetos reais presentes na imagem, um recall alto indica que o modelo deixa poucos objetos sem detectar (ELGENDY, 2020):

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.9)$$

A métrica global de desempenho padrão em detecção de objetos é a *Mean Average Precision* (mAP). Para calculá-la, plota-se a curva Precisão-Revocação (curva PR), que representa como a precisão varia à medida que a revocação aumenta ao variar o limiar de confiança das detecções. A *Average Precision* (AP) é calculada como a área sob a curva para cada classe, e o mAP é a média das APs de todas as classes (ELGENDY, 2020). O mAP@0.5 considera uma detecção correta quando o IoU é superior a 0,5, limiar padrão amplamente adotado na literatura. Já o mAP@0.5:0.95 calcula a média do mAP sob diferentes limiares de IoU, variando de 0,5 a 0,95 com passo de 0,05, sendo uma métrica mais rigorosa que penaliza previsões com baixa sobreposição geométrica mesmo quando localizam corretamente o objeto (ELGENDY, 2020).

2.4 PLANEJAMENTO DE ROTAS COM ALGORITMOS DE BUSCA EM GRAFOS

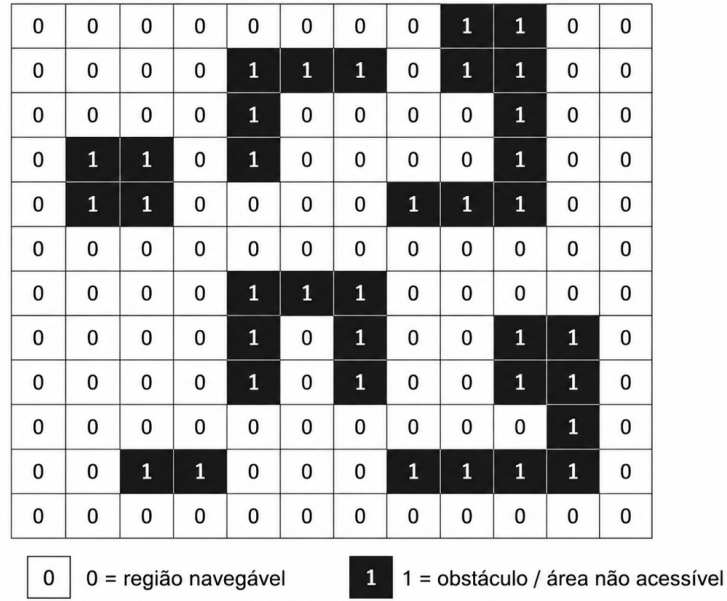
O planejamento de rotas é uma etapa fundamental em sistemas que necessitam navegar em ambientes estruturados, especialmente quando os ambientes possuem corredores bem definidos, obstáculos rígidos e conectividade previsível (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011). O objetivo é determinar um percurso viável e eficiente que atenda a um critério de deslocamento, seja a ligação entre um ponto de origem e um ponto de destino, seja a cobertura de um conjunto de vias, considerando as restrições físicas do espaço e minimizando alguma métrica de custo, como a distância total percorrida (LAVALLE, 2006). O tipo de solução é amplamente empregado em navegação robótica, inspeção automatizada e sistemas de transporte inteligentes.

Para que o planejamento de rotas possa ser realizado computacionalmente, o ambiente físico precisa ser convertido em uma representação matemática adequada para operações algorítmicas (LAVALLE, 2006). Um conceito fundamental no contexto é o de Espaço de Configuração (do inglês *Configuration Space*): a ideia é aumentar o tamanho de cada obstáculo pelo raio do agente que vai se deslocar. Com os obstáculos já expandidos do modo que, o agente pode ser tratado como um simples ponto, sem dimensões, o que simplifica bastante os cálculos (LAVALLE, 2006). Assim, como a folga do tamanho do agente já está embutida nos obstáculos, qualquer caminho calculado pelo espaço livre resultante garante que o agente real não vai colidir ao longo do trajeto. As duas abordagens mais comuns para representar o espaço computacionalmente são os mapas de ocupação binários e os grafos ponderados, descritas a seguir.

2.4.1 Mapas de Ocupação Binários

Uma abordagem amplamente utilizada para representar ambientes discretos é o mapa de ocupação binário (*Occupancy Grid Map*). Na representação, o ambiente é particionado em uma grade bidimensional de células de tamanho fixo, onde cada célula armazena um valor binário que indica o estado de ocupação (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011). Células com valor 0 representam regiões navegáveis, enquanto células com valor 1 representam obstáculos ou áreas não acessíveis. A Figura 17 ilustra a representação, na qual as células de valor 1, em preto, correspondem aos obstáculos, e as células de valor 0, em branco, à região navegável.

Figura 17 – Exemplo de mapa de ocupação binário.



Fonte: O Autor (2026).

A partir do mapa binário, é possível extrair uma representação mais compacta do espaço navegável por meio de um processo chamado esqueletização (do inglês *skeletonization*), que reduz as regiões navegáveis à linha medial, o eixo central dos corredores (GONZALEZ; WOODS, 2018). O resultado é um conjunto de pixels conectados que seguem o traçado das vias internas, preservando a conectividade global do ambiente e reduzindo significativamente a complexidade espacial. Sobre o esqueleto, constroem-se os nós e arestas de um grafo ponderado, onde cada nó representa uma posição relevante, como bifurcações, interseções ou extremidades, e cada aresta recebe um peso correspondente à distância física entre os nós conectados (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011). A estrutura de grafo reduz o espaço de busca em relação à matriz de ocupação completa, tornando o planejamento computacionalmente mais eficiente (LAVALLE, 2006).

Para que a máscara navegável represente com segurança os corredores disponíveis, é necessário considerar as dimensões físicas do agente navegante. A dilatação morfológica é uma operação de processamento de imagens que expande regiões binárias pelo contorno, adicionando pixels ao redor de cada objeto presente na imagem (GONZALEZ; WOODS, 2018). No contexto do planejamento de rotas, a operação é aplicada sobre a máscara de obstáculos com um elemento estruturante de raio r , expandindo as áreas proibidas por uma margem de segurança proporcional às dimensões do agente, conforme a equação:

$$M_{segura} = M_{obs} \oplus B \quad (2.10)$$

onde M_{obs} é a máscara binária de obstáculos, $\oplus$ denota a operação de dilatação e B é o elemento estruturante circular de raio r (GONZALEZ; WOODS, 2018). Após a dilatação, realiza-se

a inversão lógica da matriz para obter a máscara navegável M_{nav} , onde o valor 1 representa os corredores permitidos e o valor 0 as zonas de exclusão. O resultado é um mapa binário onde qualquer rota calculada sobre o esqueleto dos corredores está garantidamente afastada dos obstáculos por pelo menos r pixels, preservando a segurança do percurso (GONZALEZ; WOODS, 2018).

2.4.2 Representação em Grafo e Circuito de Euler

A partir do esqueleto extraído da máscara navegável, o espaço livre pode ser modelado como um grafo ponderado $G = (V, E)$, no qual os nós correspondem às junções e extremidades das vias e as arestas correspondem aos trechos que as conectam, ponderadas pelo comprimento (CORMEN *et al.*, 2009). Enquanto o algoritmo de Dijkstra responde a perguntas sobre o caminho mínimo entre dois nós, diversos problemas práticos exigem percorrer um conjunto de arestas, e não apenas alcançar um destino. Para tais casos, o conceito relevante é o de trilha euleriana.

Uma trilha euleriana é um percurso que atravessa cada aresta do grafo exatamente uma vez, e um circuito euleriano é uma trilha euleriana fechada, que retorna ao nó de partida (CORMEN *et al.*, 2009). A existência dos percursos é caracterizada pelo grau dos vértices, definido como o número de arestas incidentes a cada nó. Um grafo conexo admite um circuito euleriano se, e somente se, todos os vértices possuem grau par, e admite uma trilha euleriana aberta se, e somente se, exatamente dois vértices possuem grau ímpar (CORMEN *et al.*, 2009). O resultado, cuja origem remonta ao problema das pontes de *Königsberg*, fornece a base teórica para os problemas de percurso de arestas tratados nas seções seguintes: quando o grafo não satisfaz a condição de paridade, é necessário repetir algumas arestas para torná-lo euleriano antes de construir o percurso.

2.4.3 Planejamento de Cobertura

O planejamento de cobertura (do inglês *Coverage Path Planning (CPP)*), consiste em determinar um percurso que passe sobre todos os pontos de uma área ou de um conjunto de vias de interesse, evitando os obstáculos do ambiente (GALCERAN; CARRERAS, 2013). O problema difere do planejamento ponto a ponto, no qual se busca um único caminho entre uma origem e um destino, pois o objetivo passa a ser a varredura completa de uma região, com aplicações em robôs de limpeza, desminagem, agricultura automatizada e inspeção de estruturas (GALCERAN; CARRERAS, 2013).

Uma das abordagens clássicas para o problema é a decomposição celular, na qual a área livre é particionada em células mais simples que podem ser cobertas individualmente por movimentos regulares de vaivém, como na decomposição celular boustrophedon (CHOSSET, 2000). Após a cobertura de todas as células, o ambiente inteiro é coberto, e o problema se reduz a

encontrar um percurso que visite as células de acordo com as relações de adjacência (CHOSSET, 2000). Quando o ambiente é representado diretamente por um grafo cujas arestas correspondem às vias a percorrer, o problema de cobertura pode ser formulado como um problema de percurso de arestas, abordado a seguir (GALCERAN; CARRERAS, 2013).

2.4.4 Algoritmo de Dijkstra

Com o ambiente representado como um grafo ponderado $G = (V, E)$, uma operação fundamental no planejamento de rotas é a determinação do caminho de menor custo entre dois nós (LAVALLE, 2006). Além de constituir um problema de interesse em si, a operação serve de bloco básico para métodos de roteamento mais elaborados, nos quais o cálculo de distâncias mínimas entre pares de pontos é utilizado de forma repetida na construção de percursos mais complexos. Diversos algoritmos foram propostos para tal finalidade, diferindo principalmente na estratégia de busca adotada. O algoritmo A*, proposto por Hart, Nilsson e Raphael (1968), introduz uma função heurística $h(n)$ que estima o custo restante do nó atual até o destino, guiando a busca de forma mais eficiente em direção à solução. No entanto, a qualidade do resultado depende diretamente da precisão da estimativa (HART; NILSSON; RAPHAEL, 1968).

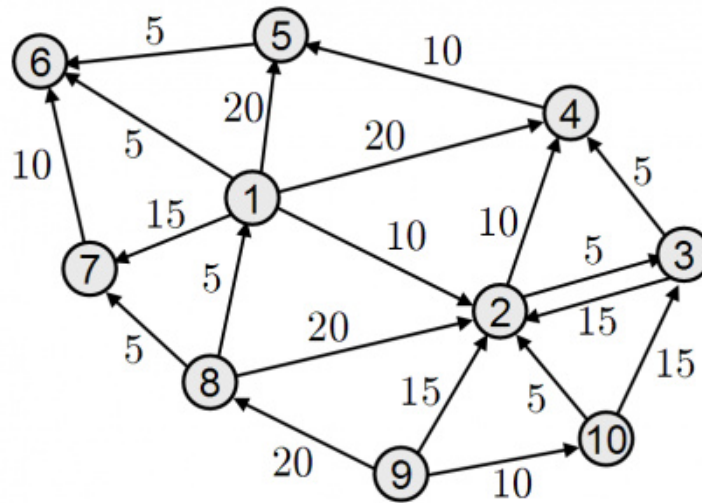
O algoritmo de Dijkstra, proposto por Dijkstra (1959), adota uma abordagem diferente: sem utilizar nenhuma função de estimativa, ele realiza uma busca sistemática garantindo que o caminho encontrado é sempre o de menor custo possível em grafos com pesos não negativos (CORMEN *et al.*, 2009). A característica o torna especialmente adequado para ambientes nos quais se deseja garantir a solução ótima independentemente da geometria do grafo.

O algoritmo mantém, para cada nó $v \in V$, o melhor custo conhecido $d(v)$ a partir do nó de origem. Inicializa-se $d(\text{origem}) = 0$ e $d(v) = \infty$ para todos os demais nós. Em cada iteração, seleciona-se, por meio de uma fila de prioridade, o nó u com menor custo atual $d(u)$ ainda não processado. Para cada vizinho v de u , realiza-se a operação de relaxamento:

$$d(v) \leftarrow \min\{d(v), d(u) + w(u, v)\} \quad (2.11)$$

onde $w(u, v)$ é o peso da aresta (u, v) . O processo se repete até que o nó de destino seja removido da fila (FEOFILOFF, 2020). A complexidade computacional do algoritmo, quando implementado com uma fila de prioridade, é de $O(|E| + |V| \log |V|)$, onde $|V|$ é o número de nós e $|E|$ é o número de arestas do grafo (CORMEN *et al.*, 2009). A complexidade evidencia a vantagem de aplicar o algoritmo sobre o grafo extraído do esqueleto em vez da matriz de ocupação completa: ao reduzir drasticamente o número de nós $|V|$, o custo computacional da busca diminui de forma proporcional, tornando o recálculo de rotas viável mesmo em ambientes de grande extensão. A Figura 18 ilustra um exemplo de grafo ponderado com o caminho de menor custo calculado pelo algoritmo.

Figura 18 – Exemplo de grafo ponderado com caminho calculado pelo algoritmo de Dijkstra



Fonte: Adaptado de: Feofiloff (2020)

2.4.5 Problema do Carteiro Chinês e Emparelhamento de Custo Mínimo

O Problema do Carteiro Chinês consiste em encontrar, em um grafo conexo e ponderado, o percurso fechado de menor comprimento total que atravesse todas as arestas pelo menos uma vez (EDMONDS; JOHNSON, 1973). Se o grafo já é euleriano, ou seja, se todos os vértices possuem grau par, o próprio circuito euleriano constitui a solução ótima, pois nenhuma aresta precisa ser repetida. Caso contrário, é necessário repetir o menor conjunto possível de arestas para igualar a paridade de todos os vértices.

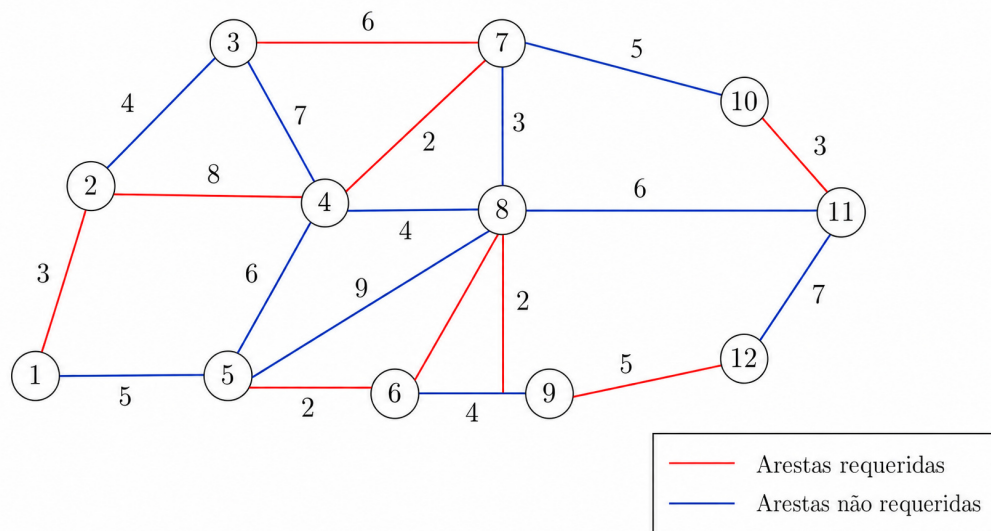
Edmonds e Johnson (1973) demonstraram que o problema pode ser resolvido em tempo polinomial. A estratégia consiste em identificar os vértices de grau ímpar, que ocorrem sempre em número par, e determinar entre eles um emparelhamento de custo mínimo. Um emparelhamento de custo mínimo é o pareamento dos vértices em duplas que minimiza a soma dos custos das ligações, e o custo de cada ligação é dado pela distância do caminho mínimo entre os dois vértices, obtida pelo algoritmo de Dijkstra (EDMONDS; JOHNSON, 1973). Os caminhos correspondentes ao emparelhamento escolhido são então duplicados no grafo, tornando pares os graus de todos os vértices, o que garante a existência de um circuito euleriano que percorre o grafo resultante (EDMONDS; JOHNSON, 1973).

2.4.6 Problema do Carteiro Rural

O Problema do Carteiro Rural é uma generalização do Problema do Carteiro Chinês na qual apenas um subconjunto das arestas, denominadas arestas obrigatórias, precisa ser percorrido, enquanto as demais arestas são opcionais e podem ser utilizadas apenas para conectar as obrigatórias e completar o percurso (EISELT; GENDREAU; LAPORTE, 1995). A formulação é adequada a cenários em que somente parte das vias do ambiente é de interesse para o deslo-

camento, e as demais servem apenas como ligações. Diferentemente do Problema do Carteiro Chinês, que possui solução exata em tempo polinomial, o Problema do Carteiro Rural pertence à classe dos problemas NP-difíceis, o que motiva o uso de algoritmos de aproximação (EISELT; GENDREAU; LAPORTE, 1995; FREDERICKSON, 1979). A Figura 19 ilustra uma instância do problema, na qual as arestas em vermelho correspondem às vias de percurso obrigatório e as arestas em azul correspondem aos trechos opcionais, utilizados apenas para viabilizar a conexão entre as obrigatórias.

Figura 19 – Exemplo de instância do Problema do Carteiro Rural



Fonte: O Autor (2026).

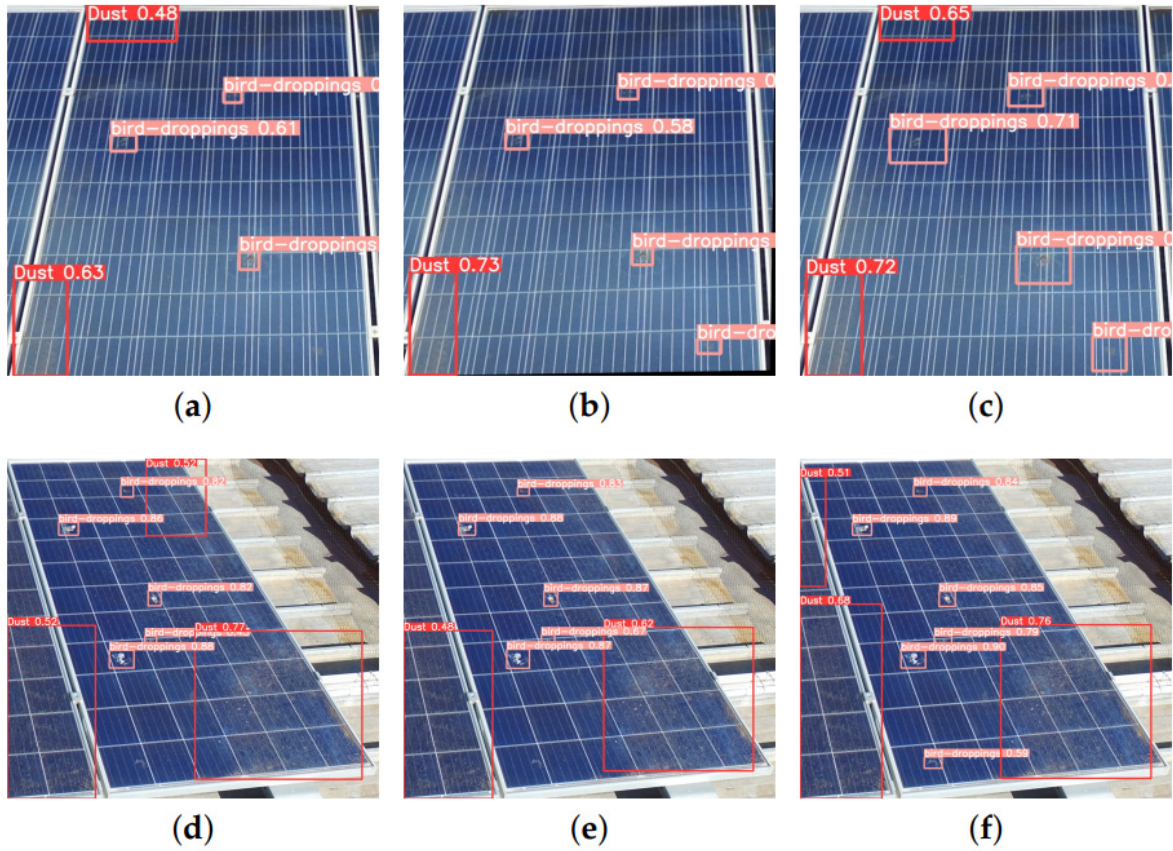
2.5 REVISÃO BIBLIOGRÁFICA

2.5.1 *Aerial Imaging-Based Soiling Detection System for Solar Photovoltaic Panel Cleanliness Inspection* (NAEEM *et al.*, 2025)

O trabalho propõe um método automático para detectar sujeira em módulos fotovoltaicos utilizando visão computacional e uma arquitetura baseada em YOLO aprimorada, denominada SDS-YOLO. O problema abordado é a perda de desempenho energético causada por poeira, sujeira orgânica e sombreamento sobre os módulos, que normalmente exige inspeções manuais demoradas e suscetíveis a falhas, tornando o processo ineficiente em usinas de grande porte (NAEEM *et al.*, 2025).

A abordagem metodológica envolve a criação de um banco de imagens contendo diferentes padrões de sujeira sobre painéis solares, seguido de etapas de pré-processamento, recorte e anotações manuais. O modelo SDS-YOLO é treinado para identificar regiões sujas ou degradadas nos módulos, combinando a detecção rápida do YOLO com um módulo de atenção convolucional (*Convolutional Block Attention Module* (CBAM)) e duas cabeças de detecção dedicadas, otimizadas para as classes de poeira e excrementos de pássaros (NAEEM *et al.*, 2025). A arquitetura aprimora a sensibilidade a pequenas variações de textura presentes na superfície dos módulos, fator essencial para distinguir sujeira leve de ruído visual. Os resultados demonstraram que o SDS-YOLO alcançou um aumento de 40,2% no mAP@50 e uma melhoria de 26,6% no F1-score para a classe de excrementos de pássaros em comparação com o YOLOv5 base, mantendo desempenho robusto em diferentes condições de luminosidade e ângulos de captura (NAEEM *et al.*, 2025). A Figura 20 apresenta exemplos de resultados de detecção de múltiplas classes de sujidade no subconjunto de teste.

Figura 20 – Resultados de detecção de múltiplas classes de sujidade: (a,d) YOLOv5; (b,e) HIC-YOLOv5; (c,f) outros modelos



Fonte: (NAEEM *et al.*, 2025)

A relevância do estudo para o presente trabalho está na validação do uso da arquitetura YOLO como técnica de detecção aplicada diretamente a módulos fotovoltaicos em imagens aéreas capturadas por VANTs. Embora o objetivo do TCC seja detectar e localizar mesas fotovoltaicas para geração de rotas de manutenção, e não identificar sujeira, ambos os estudos compartilham o mesmo contexto operacional: imagens aéreas de usinas solares de grande porte, anotação manual de regiões de interesse e avaliação por métricas COCO. O artigo reforça que arquiteturas baseadas em YOLO são eficazes no domínio, validando a escolha do YOLOv8 como arquitetura central na solução desenvolvida no presente trabalho.

2.5.2 YOLO-Based Photovoltaic Panel Detection: A Comparative Study (MOUSSAOUI *et al.*, 2025)

O trabalho avalia e compara a eficácia dos modelos YOLOv5n e YOLOv8 para a detecção de módulos fotovoltaicos em imagens aéreas, com o objetivo de identificar qual arquitetura apresenta melhor desempenho para a classe de problema (MOUSSAOUI *et al.*, 2025). O problema motivador é a necessidade de soluções automatizadas para monitoramento e inspeção de usinas

fotovoltaicas, onde a detecção precisa dos painéis em imagens aéreas é etapa fundamental para qualquer sistema de análise ou planejamento de manutenção.

A metodologia consistiu na aquisição de um conjunto de dados composto por 646 imagens aéreas de painéis fotovoltaicos com as respectivas *bounding box*. As imagens foram particionadas em 506 para treinamento e 140 para validação, com dois tipos de anotação gerados: formato COCO JSON e formato YOLO. Ambos os modelos foram treinados e avaliados sobre o conjunto de dados, com desempenho medido pela precisão das caixas delimitadoras (MOUSSAOUI *et al.*, 2025). Os resultados evidenciaram superioridade do YOLOv8 sobre o YOLOv5n, alcançando precisão de 96,7% contra 93,8% do modelo anterior. Os autores atribuem o ganho às inovações arquiteturais do YOLOv8, em particular ao módulo C2f, à cabeça desacoplada *anchor-free* e ao melhor fluxo de gradientes durante o treinamento, que resultam em representações de características mais ricas com menor custo computacional (MOUSSAOUI *et al.*, 2025).

A contribuição do estudo para o presente TCC é direta e objetiva: a comparação experimental entre YOLOv5 e YOLOv8 aplicada especificamente à detecção de painéis fotovoltaicos em imagens aéreas valida empiricamente a escolha do YOLOv8 como arquitetura de detecção no presente trabalho. Os resultados superiores obtidos pelo YOLOv8 em um cenário de domínio análogo, imagens aéreas de usinas fotovoltaicas, conjunto de dados anotado manualmente, avaliação por mAP, reforçam que a arquitetura adotada é adequada para a tarefa de identificação e localização das mesas fotovoltaicas nas imagens aéreas.

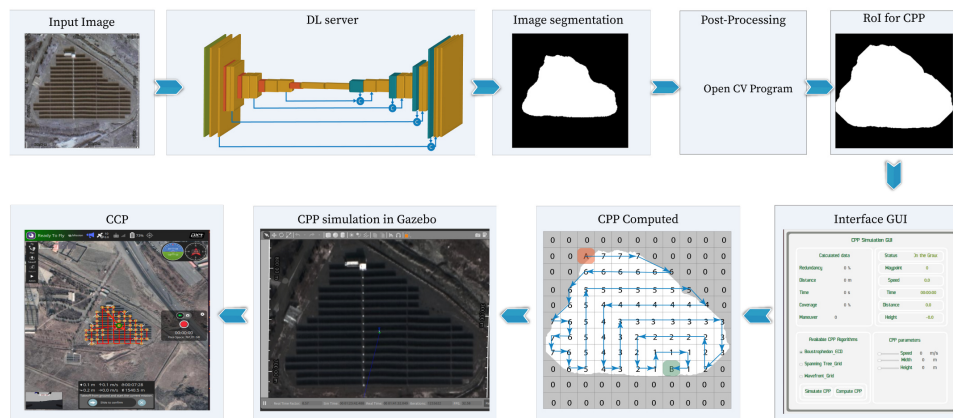
2.5.3 Coverage Path Planning with Semantic Segmentation for UAV in PV Plants (PÉREZ-GONZÁLEZ *et al.*, 2021)

O trabalho apresenta uma abordagem para o planejamento automático de rotas de inspeção em usinas fotovoltaicas utilizando VANTs, integrando segmentação semântica por *deep learning* com métodos de planejamento de cobertura de área (*Coverage Path Planning* (CPP)) (PÉREZ-GONZÁLEZ *et al.*, 2021). O problema central é que as rotas geradas manualmente por operadores em estações de controle em solo impõem restrições operacionais que dificultam a cobertura eficiente de toda a área da usina, tornando o processo de inspeção lento, custoso e altamente dependente de decisões humanas.

A metodologia proposta, ilustrada na Figura 21, é estruturada em três etapas principais. Na primeira, obtém-se uma imagem de satélite da usina fotovoltaica em altitude e limites predefinidos. Na segunda, a imagem é processada por um servidor de *deep learning* baseado em TensorFlow que aplica segmentação semântica para gerar uma máscara binária da região de interesse, os painéis fotovoltaicos e os corredores. Na terceira etapa, a máscara passa por funções de pós-processamento em OpenCV (operações morfológicas de erosão, dilatação e extração de contornos) para ajustar a região navegável, que é então submetida a um dos três métodos de CPP avaliados (PÉREZ-GONZÁLEZ *et al.*, 2021). As rotas calculadas são transmitidas ao VANT

via protocolo MAVLink para execução autônoma. Os três métodos de CPP avaliados foram o *Boustrophedon Exact Cellular Decomposition* (BECD), o *Grid-Based Spanning Tree Coverage* (GBSTC) e o *Grid-Based Wavefront Coverage* (GBWC), comparados em termos de comprimento do percurso, tempo de voo, número de manobras e consumo energético. Os experimentos foram conduzidos em três usinas fotovoltaicas reais e os resultados indicaram que o BECD e o GBWC apresentaram o melhor equilíbrio entre cobertura e eficiência energética, dependendo das dimensões da usina e da largura de cobertura adotada (PÉREZ-GONZÁLEZ *et al.*, 2021).

Figura 21 – Pipeline do método proposto por Pérez-González *et al.* (2021): imagem de entrada, segmentação semântica, máscara pós-processada e rota CPP calculada



Fonte: Pérez-González *et al.* (2021)

A relação do artigo com o presente trabalho é a mais direta entre os trabalhos revisados. Em ambos os casos, imagens aéreas de usinas fotovoltaicas são processadas por modelos de visão computacional para identificar as regiões de interesse, gerando uma representação navegável do ambiente sobre a qual se calcula uma rota de cobertura. A diferença reside na forma de identificar o ambiente e de formular a cobertura. Pérez-González *et al.* (2021) empregam segmentação semântica para obter uma máscara binária da área de interesse e aplicam métodos de cobertura baseados em decomposição celular para planejar o voo de um VANT sobre toda a superfície da usina. O presente trabalho, emprega detecção de objetos para localizar as fileiras de módulos, constrói a partir delas uma máscara navegável dos corredores e formula a inspeção como um Problema do Carteiro Rural, no qual os corredores entre as fileiras constituem as arestas de percurso obrigatório. Assim, enquanto o estudo de referência trata da cobertura de área por VANTs, o trabalho trata da cobertura dos corredores de circulação por equipes de manutenção em solo. O estudo de referência valida a viabilidade da integração entre análise de imagens aéreas e planejamento de cobertura em usinas fotovoltaicas, contribuindo para fundamentar tecnicamente a metodologia adotada no presente trabalho.

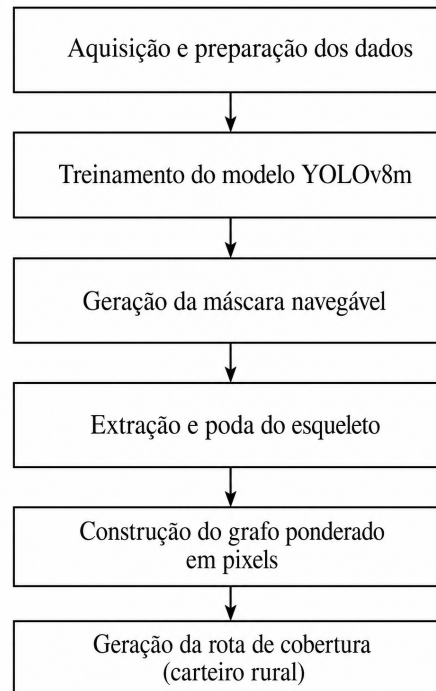
3 IMPLEMENTAÇÃO DESENVOLVIDA

As inspeções manuais em usinas fotovoltaicas de grande porte demandam elevado tempo, recursos humanos e expõem os trabalhadores a riscos operacionais, como a exposição prolongada ao sol e o deslocamento em áreas extensas sem roteiros previamente otimizados (PORTAL SOLAR, 2024c). A automação do processo, por meio da análise de imagens aéreas combinada ao planejamento computacional de percursos, representa uma alternativa consolidada na literatura para reduzir os custos operacionais (PÉREZ-GONZÁLEZ *et al.*, 2021; MOUSSAOUI *et al.*, 2025).

Este capítulo descreve a implementação desenvolvida para detectar automaticamente as mesas fotovoltaicas em imagens aéreas e gerar uma rota de cobertura que percorra os corredores de circulação entre as fileiras de módulos, voltada às equipes de manutenção em solo. A abordagem integra a arquitetura de detecção de objetos YOLOv8m, cuja eficácia para o domínio de imagens aéreas de usinas fotovoltaicas foi validada empiricamente por Moussaoui *et al.* (2025), com técnicas clássicas de processamento morfológico de imagens e algoritmos clássicos de otimização em grafos. O conjunto de dados utilizado é composto por 488 imagens aéreas obtidas na plataforma Roboflow, e as distâncias ao longo das rotas calculadas são expressas em pixels, dado que as imagens não dispõem de informações de *Ground Sample Distance* (GSD).

A Figura 22 apresenta o fluxograma da implementação, sintetizando as seis etapas que compõem o *pipeline*: aquisição e preparação dos dados, treinamento do modelo de detecção, geração da máscara navegável, extração do esqueleto dos corredores, construção do grafo ponderado em *pixels* e geração da rota de cobertura pelo Problema do Carteiro Rural. Cada etapa é detalhada nas seções seguintes.

Figura 22 – Fluxograma do *pipeline* de implementação



Fonte: O Autor (2026).

3.1 DATASET

3.1.1 Fonte e Composição das Imagens

O conjunto de dados utilizado neste trabalho é composto por 488 imagens aéreas de usinas fotovoltaicas obtidas na plataforma Roboflow (ROBOFLOW, 2025). O Roboflow é uma plataforma voltada ao desenvolvimento de sistemas de visão computacional que oferece ferramentas integradas para organização, anotação, pré-processamento e exportação de conjuntos de dados nos formatos exigidos pelos principais modelos de detecção de objetos. As imagens apresentam usinas fotovoltaicas de diferentes configurações geométricas, variações de iluminação e distintos tipos de cobertura de solo, características que conferem diversidade visual ao conjunto e são relevantes para a capacidade de generalização do modelo treinado.

3.1.2 Divisão do Conjunto de Dados

O conjunto foi particionado em três subconjuntos mutuamente exclusivos, seguindo a proporção de 80%, 10% e 10% para as etapas de treinamento, validação e teste, respectivamente, conforme recomendado pela documentação oficial da Ultralytics (JOCHER *et al.*, 2024). A distribuição resulta em 392 imagens destinadas ao treinamento, 48 à validação e 48 ao teste, conforme sintetizado na Tabela 1.

Tabela 1 – Divisão do conjunto de dados

Subconjunto	Quantidade	Proporção
Treinamento	392	80%
Validação	48	10%
Teste	48	10%
Total	488	100 %

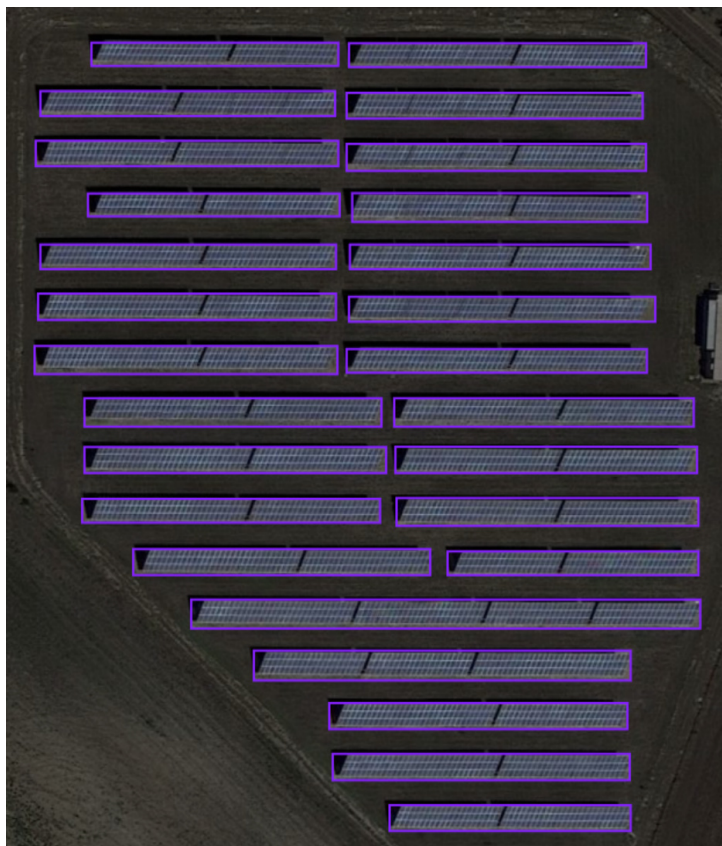
Fonte: O Autor (2026).

A proporção adotada maximiza o volume de amostras disponíveis para o ajuste dos pesos do modelo, fator relevante em cenários com conjuntos de dados de tamanho limitado, ao mesmo tempo em que reserva amostras suficientes para o monitoramento do treinamento e a avaliação final independente do desempenho.

3.1.3 Anotação das Imagens

As anotações foram realizadas manualmente na plataforma Roboflow, adotando uma única classe denominada mesa. Cada instância anotada corresponde a uma fileira completa de módulos fotovoltaicos, delimitada por uma única *bounding box* que engloba toda a estrutura de suporte, conforme ilustrado na Figura 23. A estratégia de anotação por fileira, em detrimento da anotação módulo a módulo, foi adotada por duas razões complementares. Primeiro, os módulos individuais possuem dimensões reduzidas em escala aérea, tornando suas *bounding boxes* próximas de quadrados e dificultando a distinção entre objetos adjacentes. Segundo, o objetivo do *pipeline* não é contar ou identificar módulos individualmente, mas delimitar as regiões ocupadas pelas mesas como obstáculos para a geração da máscara navegável. Para isso, a anotação por fileira é suficiente e geometricamente mais representativa da área real de cada obstáculo (NAEEM *et al.*, 2025; MOUSSAOUI *et al.*, 2025).

Figura 23 – Exemplo de anotação no formato *bouding box* por fileira de módulos fotovoltaicos



Fonte: O Autor (2026).

As anotações seguem o formato *bouding box* exigido pelo YOLOv8, representadas em arquivos .txt com as coordenadas normalizadas do centro (x, y), largura e altura de cada objeto em relação às dimensões da imagem (JOCHER *et al.*, 2024). O Roboflow exporta diretamente no formato, eliminando etapas de conversão manual (ROBOFLOW, 2025).

3.1.4 Pré-processamento e Aumento de Dados

O YOLOv8 realiza automaticamente o redimensionamento das imagens para 640×640 pixels durante o carregamento de cada *batch*, utilizando *letterboxing* para preservar as proporções originais sem distorção dos objetos anotados (JOCHER *et al.*, 2024). Por isso, nenhum pré-processamento de redimensionamento foi aplicado ao *dataset* antes da exportação pelo Roboflow.

O aumento de dados (*data augmentation*) foi integralmente gerenciado pelo *pipeline* de treinamento do YOLOv8, que aplica um conjunto abrangente de transformações dinamicamente a cada *batch*, sem necessidade de armazenar imagens aumentadas em disco (JOCHER *et al.*, 2024). As transformações ativas por padrão incluem o Mosaic, que combina quatro imagens em uma única amostra enriquecendo o contexto visual e forçando o modelo a lidar com objetos em diferentes escalas e posições, ajustes de matiz, saturação e brilho no espaço HSV

($hsv_h=0,015$, $hsv_s=0,7$, $hsv_v=0,4$), translação ($translate=0,1$), escala ($scale=0,5$) e inversão horizontal ($fliplr=0,5$) (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023; JOCHER *et al.*, 2024).

Adicionalmente, dois parâmetros desativados por padrão foram habilitados para adequar o *pipeline* ao domínio de imagens aéreas: a inversão vertical ($flipud=0,5$) e a rotação aleatória de até ± 15 ($degrees=15$). A inversão vertical justifica-se pelo fato de que a orientação da câmera em relação às fileiras de módulos pode variar conforme a trajetória de captura. A rotação justifica-se pelas variações de ângulo de voo, reproduzindo pequenas inclinações comuns em capturas aéreas por drones, consistente com o observado por Naeem *et al.* (2025), que demonstrou que pequenas rotações melhoram significativamente a robustez de modelos YOLO aplicados a imagens aéreas de usinas fotovoltaicas. Todos os parâmetros de *augmentation* foram configurados diretamente no comando de treinamento, sem modificação do *dataset* exportado pelo Roboflow.

3.2 TREINAMENTO DO MODELO

Esta seção descreve o processo de treinamento dos modelos de detecção, abrangendo o ambiente computacional utilizado, a estratégia de aprendizado por transferência, a configuração dos hiperparâmetros e as métricas adotadas para a avaliação de desempenho. A definição da variante de escala da arquitetura YOLOv8 empregada como modelo definitivo é apresentada no Capítulo 4, a partir de análise comparativa experimental.

3.2.1 Ambiente de Treinamento

O treinamento foi conduzido na plataforma Google Colaboratory (Colab), ambiente de execução baseado em nuvem que disponibiliza gratuitamente unidades de processamento gráfico (GPU) para tarefas de aprendizado profundo (JOCHER *et al.*, 2024). A GPU utilizada foi a NVIDIA T4, com 15 GB de memória de vídeo (VRAM) e 320 núcleos Tensor para cálculos matriciais, suficiente para acomodar os modelos avaliados com resolução de entrada de 640×640 pixels e batch size de 16 imagens. O gerenciamento do modelo, do *pipeline* de treinamento e da inferência foi realizado por meio da biblioteca Ultralytics (JOCHER *et al.*, 2024), que oferece uma interface unificada em Python para configuração, execução e avaliação do modelo.

3.2.2 Estratégia de Treinamento

O treinamento foi conduzido com base no paradigma de Transfer Learning (PAN; YANG, 2010), inicializando o modelo com pesos pré-treinados no conjunto de dados COCO (Common Objects in Context), composto por mais de 330 mil imagens de 80 categorias de objetos (ELGENDY, 2020). A estratégia permite que o modelo parta de representações visuais genéricas

já aprendidas, como bordas, texturas e formas, e as adapte ao domínio específico de imagens aéreas de usinas fotovoltaicas por meio do processo de fine-tuning (CHOLLET, 2018).

De acordo com (ELGENDY, 2020), o cenário de Transfer Learning aplicado neste trabalho corresponde ao Cenário 1: conjunto de dados pequeno e similaridade razoável com o domínio de origem. No cenário, a recomendação é utilizar a rede pré-treinada como extrator de características, com ajuste fino sobre os pesos de toda a rede, uma vez que as representações aprendidas no COCO são transferíveis para o domínio fotovoltaico. A abordagem acelera a convergência e reduz o risco de overfitting em comparação ao treinamento do zero (ELGENDY, 2020; CHOLLET, 2018), sendo especialmente adequada para conjuntos de dados de tamanho limitado como o utilizado neste trabalho (MOUSSAOUI *et al.*, 2025).

A arquitetura YOLOv8 é disponibilizada em cinco variantes de escala, que diferem entre si principalmente no número de parâmetros da rede, o que impacta diretamente a capacidade de aprendizado e o custo computacional do treinamento. Variantes menores, como o YOLOv8n (nano), são mais rápidas e indicadas para aplicações em tempo real ou dispositivos com recursos limitados, enquanto variantes maiores, como o YOLOv8x (extra large), oferecem maior acurácia ao custo de maior demanda de hardware (JOCHER *et al.*, 2024). A Tabela 2 apresenta as variantes disponíveis para a tarefa de detecção e seus respectivos números de parâmetros.

Tabela 2 – Variantes da arquitetura YOLOv8 para detecção e número de parâmetros.

Modelo	Entrada (px)	Parâmetros (M)
YOLOv8n	640	3,2
YOLOv8s	640	11,2
YOLOv8m	640	25,9
YOLOv8l	640	43,7
YOLOv8x	640	68,2

Fonte: Jocher *et al.* (2024).

Para este trabalho, foram treinadas e avaliadas as variantes *nano*, *medium* e *large*, com o objetivo de identificar experimentalmente aquela que oferece o melhor compromisso entre acurácia de detecção e custo computacional. A análise comparativa entre as três variantes é apresentada no Capítulo 4.

3.2.3 Configuração e Hiperparâmetros

O treinamento foi executado por 100 épocas, valor recomendado pela documentação da Ultralytics para conjuntos de dados de até 1.000 imagens (JOCHER *et al.*, 2024). Para monitorar e evitar o overfitting, foi utilizado o mecanismo de early stopping com paciência de 50 épocas, que interrompe automaticamente o treinamento caso o mAP de validação não apresente melhora ao longo das épocas monitoradas (JOCHER *et al.*, 2024). O modelo com melhor desempenho no conjunto de validação ao longo do treinamento é salvo automaticamente como best.pt e utilizado

na etapa de inferência. O parâmetro seed foi fixado em 42 para garantir a reprodutibilidade dos resultados.

O *batch size* foi configurado em 16 imagens por iteração, compatível com a memória disponível na GPU T4 para a combinação de modelo e resolução adotadas (JOCHER *et al.*, 2024). A resolução de entrada foi definida em 640×640 pixels (imgsz=640), valor padrão do YOLOv8 validado para o domínio de detecção de painéis fotovoltaicos (TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

Adotou-se o otimizador SGD (*Stochastic Gradient Descent*) com taxa de aprendizado inicial de 1×10^{-2} e momento de 0,937. Embora a configuração padrão do YOLOv8 utilize o modo automático de seleção de otimizador, que combina o Adam nas iterações iniciais e o SGD nas etapas seguintes, optou-se por fixar o SGD durante todo o treinamento por sua reconhecida capacidade de generalização em conjuntos de dados de porte reduzido (JOCHER *et al.*, 2024). A escolha foi posteriormente validada de forma experimental, conforme apresentado no Capítulo 4. Um agendador de taxa de aprendizado do tipo *cosine annealing* (cos_lr=True) decaiu a taxa progressivamente ao longo das épocas, favorecendo a estabilidade nas épocas finais do treinamento (JOCHER *et al.*, 2024).

Diferentemente da configuração padrão da biblioteca, na qual os aumentos de dados geométricos de inversão vertical e rotação encontram-se desativados, dois parâmetros adicionais foram habilitados para adequar o *pipeline* ao domínio de imagens aéreas: a inversão vertical (flipud=0,5) e a rotação aleatória de até ± 15 graus (degrees=15). A Tabela 3 sintetiza os principais hiperparâmetros utilizados no treinamento.

Tabela 3 – Hiperparâmetros de treinamento.

Parâmetro	Valor	Descrição
epochs	100	Número máximo de épocas
patience	50	Épocas sem melhora para early stopping
batch	16	Imagens por iteração
imgsz	640	Resolução de entrada (pixels)
optimizer	SGD	Otimizador
lr0	0,01	Taxa de aprendizado inicial
momentum	0,937	Momento do SGD
cos_lr	True	Agendador cosine annealing
flipud	0,5	Probabilidade de inversão vertical
degrees	15	Rotação aleatória máxima (graus)
seed	42	Semente para reprodutibilidade

Fonte: O Autor (2026).

3.2.4 Métricas de Avaliação

O desempenho dos modelos treinados foi avaliado sobre o subconjunto de teste, composto por 48 imagens não utilizadas durante o treinamento ou a validação. As métricas adotadas

seguem o protocolo padrão do Microsoft COCO (ELGENDY, 2020), já descritas no Capítulo 2: Precisão (Precision), Revocação (Recall), $mAP@0,5$ e $mAP@0,5:0,95$.

A Precisão mede a proporção de detecções corretas em relação ao total de detecções realizadas, indicando a confiabilidade do modelo ao apontar a presença de uma mesa fotovoltaica. A Revocação mede a capacidade do modelo de localizar todas as mesas presentes na imagem. O $mAP@0,5$ corresponde à precisão média considerando o limiar de IoU de 0,5, amplamente adotado na literatura, enquanto o $mAP@0,5:0,95$ calcula a média sobre dez limiares de IoU entre 0,5 e 0,95, com passo de 0,05, constituindo uma métrica mais exigente e representativa da precisão geométrica das detecções (ELGENDY, 2020; TERVEN; CÓRDOVA-ESPARZA; ROMERO-GONZÁLEZ, 2023).

3.3 GERAÇÃO DA MÁSCARA NAVEGÁVEL

A primeira etapa do planejamento de rota consiste em converter o resultado da detecção em uma representação do espaço por onde a equipe de manutenção pode circular. As *bounding boxes* produzidas pelo YOLOv8m delimitam as regiões ocupadas pelas mesas fotovoltaicas, que são tratadas como obstáculos. A partir delas, constrói-se uma máscara binária de obstáculos M_{obs} , na qual o valor 1 indica os *pixels* internos a alguma caixa detectada e o valor 0 indica o espaço livre. A Figura 24 ilustra o resultado da detecção para uma das usinas avaliadas, com as *bounding boxes* sobrepostas à imagem aérea.

Figura 24 – Resultado da detecção das fileiras de módulos fotovoltaicos pelo YOLOv8m, com as *bounding boxes* sobrepostas à imagem aérea da usina.



Fonte: O Autor (2026).

Como as imagens não dispõem de informação de *Ground Sample Distance* (GSD), todos os parâmetros geométricos desta etapa são definidos de forma adaptativa, em *pixels*, em função da escala aparente da usina em cada imagem. A escala é estimada pelo espaçamento mediano entre fileiras, denotado por *gap*, obtido a partir das caixas detectadas: ordenam-se os centros verticais das caixas, calcula-se a mediana das distâncias entre centros consecutivos, desconsiderando as distâncias muito pequenas associadas a caixas de uma mesma fileira, e subtrai-se a altura mediana das caixas, resultando na largura típica do corredor livre entre duas fileiras. Para evitar valores degenerados em imagens com poucas fileiras, adota-se um valor mínimo de 6 *pixels* para o *gap*.

A construção da máscara navegável combina duas restrições. A primeira garante uma margem de segurança em relação aos painéis. Aplicando a dilatação morfológica sobre M_{obs} com um elemento estruturante B_R de raio R e tomando o complemento do resultado, obtém-se a região de espaço livre afastada dos obstáculos por pelo menos R *pixels* (GONZALEZ; WOODS, 2018). O procedimento corresponde à construção do espaço de configuração descrita na Seção 2.4.1, que permite tratar a equipe como um ponto durante o planejamento (LAVALLE, 2006). O raio é definido por $R \approx 0,28 \cdot \text{gap}$, com mínimo de 2 *pixels*.

A segunda restrição confina a área navegável ao contorno da usina, evitando que a rota se estenda indefinidamente pelo terreno aberto ao redor das fileiras. Para isso, aplica-se um fechamento morfológico (*closing*) sobre M_{obs} com um elemento estruturante B_K de raio K elevado, o que une as fileiras vizinhas em uma única região sólida que aproxima o contorno côncavo ocupado pela usina (GONZALEZ; WOODS, 2018). A região é então dilatada por um elemento B_{M_d} de raio M_d , acrescentando uma margem que mantém disponíveis os corredores de perímetro. Os parâmetros são definidos por $K \approx 2,5 \cdot \text{gap}$, com mínimo de 25 *pixels*, e $M_d \approx 0,8 \cdot \text{gap}$, com mínimo de 10 *pixels*.

A máscara navegável final M_{nav} é obtida pela interseção entre a região de espaço livre e o contorno da usina, conforme a Equação 3.1:

$$M_{nav} = \overline{(M_{obs} \oplus B_R)} \wedge [(M_{obs} \bullet B_K) \oplus B_{M_d}] \quad (3.1)$$

onde $\oplus$ denota a dilatação morfológica, $\bullet$ o fechamento morfológico, $\overline{(\cdot)}$ o complemento lógico e $\wedge$ a interseção lógica (GONZALEZ; WOODS, 2018). O resultado é um mapa binário no qual o valor 1 representa os corredores percorráveis, preservando simultaneamente o afastamento de segurança dos painéis e o confinamento à área da usina. Quando a máscara resulta em regiões desconexas, a seleção da componente principal é tratada na etapa de construção do grafo, descrita na Seção 3.5. A Figura 25 ilustra a máscara navegável obtida para uma das usinas avaliadas.

Figura 25 – Máscara navegável, sobreposta à imagem da usina.



Fonte: O Autor (2026).

3.4 EXTRAÇÃO E PODA DO ESQUELETO

A máscara navegável obtida na etapa anterior descreve toda a área percorrível como uma região de espessura variável, o que ainda não constitui uma representação adequada para o planejamento da rota. Para reduzir a região às linhas centrais dos corredores, aplica-se a esqueletização da máscara navegável, processo que extrai o eixo medial das regiões livres, conforme descrito na Seção 2.4.1. O resultado é um conjunto de *pixels* conectados, de espessura unitária, que segue o traçado central de cada corredor e preserva a conectividade global do ambiente (GONZALEZ; WOODS, 2018). A esqueletização foi realizada com a função *skeletonize* da biblioteca *scikit-image*.

A esqueletização de máscaras obtidas a partir de detecções reais tende a produzir ramos terminais espúrios, conhecidos como *spurs*, pequenos segmentos que partem do eixo central em direção às bordas da região navegável e que não correspondem a corredores efetivos. Os ramos decorrem de irregularidades no contorno da máscara e, se mantidos, introduzem no grafo nós e arestas sem significado prático. Para eliminá-los, aplica-se uma etapa de poda sobre o esqueleto.

A poda baseia-se no grau de cada *pixel* do esqueleto, definido como o número de vizinhos pertencentes ao esqueleto em uma vizinhança de oito conexões. A partir do critério, distinguem-se três tipos de *pixels*: as pontas, de grau 1, que correspondem às extremidades dos ramos; os *pixels* de passagem, de grau 2, que compõem o interior dos corredores; e as junções, de grau igual ou superior a 3, que correspondem aos cruzamentos entre corredores. O procedimento percorre o esqueleto a partir de cada ponta, acompanhando o ramo *pixel a pixel*. Caso o

ramo alcance uma junção antes de atingir o comprimento limite, ele é classificado como espúrio e removido; caso contrário, é preservado por representar um corredor legítimo. O comprimento limite é definido de forma adaptativa por $L \approx 1,3 \cdot gap$, de modo que apenas ramos mais curtos que a distância típica entre fileiras sejam descartados. O processo é repetido iterativamente, pois a remoção de um ramo pode expor novas pontas, até que nenhuma alteração adicional ocorra.

A Figura 26 apresenta o esqueleto resultante após a poda, sobreposto à imagem da usina, evidenciando as linhas centrais dos corredores que servirão de base para a construção do grafo descrita na Seção 3.5.

Figura 26 – Esqueleto extraído após a poda, sobreposto à imagem da usina.



Fonte: O Autor (2026).

3.5 CONSTRUÇÃO DO GRAFO DE CORREDORES

O esqueleto podado descreve as linhas centrais dos corredores como um conjunto de *pixels* conectados, mas ainda em uma representação matricial. Para viabilizar a aplicação dos algoritmos de otimização, o esqueleto é convertido em um grafo ponderado $G = (V, E)$, no qual os nós representam pontos notáveis das vias e as arestas representam os trechos de corredor que os conectam, conforme descrito na Seção 2.4.1 (SIEGWART; NOURBAKHS; SCARAMUZZA, 2011).

A definição dos nós baseia-se novamente no grau de cada *pixel* do esqueleto. Os *pixels* de grau diferente de 2, ou seja, as pontas, de grau 1, e as junções, de grau igual ou superior a 3, são identificados como nós, enquanto os *pixels* de grau 2 constituem o interior das arestas. Como uma única junção pode ocupar mais de um *pixel* adjacente, os *pixels* de nó vizinhos entre

si são agrupados em um mesmo nó por meio da rotulação de componentes conexos, evitando a criação de nós redundantes em um mesmo cruzamento.

A partir de cada nó, percorre-se o esqueleto ao longo das sequências de *pixels* de grau 2 até alcançar outro nó, definindo assim uma aresta. A cada aresta associa-se o seu comprimento, calculado como a soma das distâncias euclidianas entre *pixels* consecutivos do trecho, de modo que deslocamentos diagonais contribuam com $\sqrt{2}$ e deslocamentos ortogonais com 1. O comprimento, expresso em *pixels*, é o peso da aresta. A sequência de *pixels* de cada trecho também é armazenada, para permitir a posterior reconstrução da polilinha da rota sobre a imagem. O grafo é representado como um multigrafo, uma vez que dois nós podem estar ligados por mais de um corredor distinto.

Por fim, seleciona-se a maior componente conexa do grafo, descartando-se eventuais fragmentos isolados que possam surgir quando a detecção não captura uma região contígua da usina. A escolha garante que a rota seja calculada sobre uma estrutura conexa, condição necessária para os algoritmos das etapas seguintes. A restrição à maior componente constitui uma limitação conhecida da abordagem, discutida no Capítulo 4, pois seções fragmentadas da usina podem ser excluídas do percurso. A Figura 27 apresenta o grafo de corredores extraído para uma das usinas avaliadas, com os nós e as arestas sobrepostos à imagem.

Figura 27 – Grafo de corredores extraído, com nós e arestas sobrepostos à imagem da usina.



Fonte: O Autor (2026).

3.6 GERAÇÃO DA ROTA DE COBERTURA PELO CARTEIRO RURAL

A etapa final formula o planejamento da inspeção como um Problema do Carteiro Rural, descrito na Seção 2.4.6, no qual apenas os corredores entre fileiras precisam ser percorridos, enquanto os demais trechos servem somente para conectar os corredores e completar o percurso (EISELT; GENDREAU; LAPORTE, 1995). Adota-se a variante de rota aberta, na qual o percurso entra pela borda da usina e termina em um ponto distinto do de partida, sem retornar ao início, o que reduz o deslocamento total da equipe de manutenção.

O primeiro passo é classificar cada aresta do grafo como corredor obrigatório ou como trecho conector. A classificação utiliza um teste geométrico simples: para o *pixel* central de cada aresta, verifica-se a presença de painéis tanto acima quanto abaixo, dentro de uma janela vertical proporcional ao espaçamento entre fileiras, de aproximadamente $1,8 \cdot gap$. Quando há painel dos dois lados, a aresta é interpretada como um corredor situado entre duas fileiras e marcada como obrigatória; caso contrário, é tratada como trecho conector ou de perímetro, de uso opcional. A soma dos comprimentos das arestas obrigatórias define o comprimento total de corredores a percorrer, denotado por L_{req} , utilizado na avaliação de eficiência apresentada no Capítulo 4.

Em seguida, os componentes formados pelas arestas obrigatórias são interligados. Enquanto houver mais de um componente, identificam-se os dois componentes mais próximos e adiciona-se ao grafo o caminho de menor custo que os conecta, repetindo-se o processo até que todas as arestas obrigatórias pertençam a um único componente conexo. As distâncias e os caminhos mínimos utilizados na fase são obtidos pelo algoritmo de Dijkstra, descrito na Seção 2.4.4 (DIJKSTRA, 1959).

A correção de paridade é então adaptada para produzir uma rota aberta. Em vez de emparelhar todos os nós de grau ímpar, o que fecharia o percurso, preservam-se dois nós de grau ímpar para servir de início e de fim da rota. O nó de início é escolhido entre os nós de grau ímpar situados próximos à borda da usina; para cada candidato a início, os demais nós ímpares são pareados por um emparelhamento de custo mínimo (EDMONDS; JOHNSON, 1973), restando um nó sem par, que se torna o ponto final do percurso. Adota-se o candidato a início que minimiza o custo total do emparelhamento. Os caminhos correspondentes são adicionados ao grafo, de modo que apenas os nós de início e de fim permaneçam com grau ímpar.

Com exatamente dois nós de grau ímpar, o grafo resultante admite uma trilha euleriana aberta entre eles, conforme a condição apresentada na Seção 2.4.2 (CORMEN *et al.*, 2009). A partir do nó de início, extrai-se a trilha, e a rota final é reconstruída concatenando-se as sequências de *pixels* das arestas percorridas, formando uma polilinha contínua sobre a imagem, do ponto de entrada ao ponto final, sem retorno ao início. No caso particular em que o conjunto conexo de corredores já é euleriano, ou seja, não possui nós de grau ímpar, o percurso reduz-se a um circuito euleriano fechado, iniciado no nó mais próximo do canto superior esquerdo da região.

Cabe registrar que a classificação entre corredor e conector adota um critério geométrico simples, que pode falhar em arranjos atípicos de fileiras, e que a heurística de Frederickson fornece uma solução aproximada, e não necessariamente ótima, para o Problema do Carteiro Rural (FREDERICKSON, 1979). As características são retomadas na análise de limitações do Capítulo 4. A Figura 28 apresenta a rota de cobertura gerada para uma das usinas avaliadas, com os pontos de início e de fim destacados.

Figura 28 – Rota de cobertura gerada, com ponto de início (círculo verde) e ponto de fim (quadrado vermelho) destacados.



Fonte: O Autor (2026).

4 TESTES E RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos em cada etapa do pipeline desenvolvido, desde a detecção das mesas fotovoltaicas até a geração da rota de manutenção. A Seção 4.1 aborda os resultados do treinamento e da detecção, incluindo a comparação experimental entre otimizadores e entre variantes de escala da arquitetura YOLOv8, bem como a análise das métricas obtidas. As seções subsequentes apresentam os resultados das etapas de geração da máscara navegável, extração do esqueleto, construção do grafo e cálculo da rota otimizada, encerrando com uma análise integrada do funcionamento do pipeline completo.

4.1 RESULTADOS DO TREINAMENTO E DETECÇÃO

Esta seção apresenta os resultados experimentais do treinamento e da avaliação dos modelos de detecção. Foram conduzidos dois conjuntos de experimentos comparativos: o primeiro avaliando o impacto do otimizador no desempenho do modelo, e o segundo, comparando diferentes variantes de escala da arquitetura YOLOv8. Todas as métricas reportadas foram obtidas sobre o conjunto de teste, mantido isolado durante o treinamento e a validação.

4.1.1 Comparação entre Otimizadores

O primeiro experimento avaliou o efeito da escolha do otimizador sobre o desempenho do modelo YOLOv8m, mantendo todos os demais hiperparâmetros constantes. Comparou-se o otimizador SGD com o AdamW, ajustando-se a taxa de aprendizado inicial conforme recomendado para cada algoritmo: 0,01 para o SGD e 0,001 para o AdamW. Os resultados são apresentados na Tabela 4.

Tabela 4 – Comparação de desempenho entre os otimizadores SGD e AdamW (YOLOv8m).

Métrica	SGD	AdamW
Precision	0,9088	0,8760
Recall	0,8028	0,7973
mAP@0,5	0,8136	0,8035
mAP@0,5:0,95	0,4921	0,4748

Fonte: O Autor (2026).

O SGD apresentou desempenho superior ao AdamW em todas as quatro métricas avaliadas. A diferença mais expressiva ocorreu na precisão, com vantagem de 3,3 pontos percentuais para o SGD. Embora as diferenças nas demais métricas tenham sido menores, a superioridade do SGD foi consistente em todos os indicadores, conferindo confiabilidade ao resultado. O comportamento é coerente com a literatura, que aponta melhor capacidade de generalização

dos otimizadores baseados em gradiente descendente estocástico em relação aos otimizadores adaptativos, particularmente em cenários com conjuntos de dados de porte reduzido. Com base no resultado, o SGD foi adotado como otimizador definitivo nos experimentos subsequentes.

4.1.2 Comparação entre Variantes de Escala

O segundo experimento comparou três variantes de escala da arquitetura YOLOv8: nano (n), medium (m) e large (l). O objetivo foi avaliar a relação entre a capacidade de representação de cada variante e o desempenho na tarefa de detecção, identificando o melhor compromisso entre acurácia e custo computacional. Todas as variantes foram treinadas com os mesmos hiperparâmetros e sobre o mesmo conjunto de dados. Os resultados são apresentados na Tabela 5.

Tabela 5 – Comparação de desempenho entre as variantes YOLOv8n, YOLOv8m e YOLOv8l.

Métrica	YOLOv8n	YOLOv8m	YOLOv8l
Precision	0,8216	0,9088	0,9113
Recall	0,7729	0,8028	0,8136
mAP@0,5	0,7562	0,8136	0,8156
mAP@0,5:0,95	0,4358	0,4921	0,4977

Fonte: O Autor (2026).

A variante nano apresentou o desempenho mais baixo em todas as métricas, resultado esperado dado seu número significativamente reduzido de parâmetros, que limita sua capacidade de aprender representações complexas. As variantes medium e large apresentaram desempenho substancialmente superior ao nano e próximos entre si. A variante large obteve os melhores resultados em todas as métricas, porém com margem reduzida em relação à medium: a diferença no mAP@0,5 foi de apenas 0,2 pontos percentuais, e no mAP@0,5:0,95 de 0,56 pontos percentuais.

Considerando que a variante large possui aproximadamente 70% mais parâmetros que a medium, com consequente aumento no custo computacional, no consumo de memória e no tempo de treinamento, o ganho marginal de desempenho não justifica sua adoção no contexto deste trabalho. A variante medium foi, portanto, selecionada como modelo definitivo, por oferecer o melhor equilíbrio entre acurácia e eficiência computacional. O modelo final alcançou Precisão de 0,9088, indicando que aproximadamente 91% das detecções realizadas correspondem a mesas fotovoltaicas reais, e mAP@0,5 de 0,8136, demonstrando desempenho adequado para a tarefa de localização das fileiras de módulos.

4.1.3 Análise das Métricas e Limitações

Observa-se uma diferença expressiva entre o mAP@0,5 (0,8136) e o mAP@0,5:0,95 (0,4921) do modelo selecionado. A diferença é atribuída a três fatores principais relacionados

às características do conjunto de dados e da tarefa.

Primeiro, a estratégia de anotação adotada, na qual cada fileira de mesas é delimitada por uma única bounding box, gera objetos longos e estreitos cuja localização geométrica precisa é intrinsecamente mais difícil. Pequenos deslocamentos na predição reduzem significativamente o valor de IoU nos limiares mais elevados, penalizando o $mAP@0,5:0,95$ mesmo quando a detecção é visualmente correta.

Segundo, o conjunto de dados apresenta heterogeneidade considerável quanto a resolução, altitude de captura e tamanho aparente das mesas entre as diferentes imagens, provenientes de usinas distintas. A variabilidade, embora benéfica para a capacidade de generalização, dificulta a convergência do modelo para predições geometricamente precisas em todas as condições. Cabe destacar que o cenário difere substancialmente de aplicações em que as imagens são padronizadas e capturadas sob condições controladas, nas quais valores mais elevados de $mAP@0,5:0,95$ são mais facilmente alcançados.

Terceiro, o volume reduzido de dados, com aproximadamente 490 imagens, limita a capacidade de generalização do modelo, refletindo-se especialmente no recall e nas métricas mais exigentes. A limitação decorre da dificuldade inerente à obtenção de imagens aéreas de usinas fotovoltaicas de grande porte devidamente anotadas, dado que se trata de um domínio especializado com disponibilidade restrita de conjuntos de dados públicos.

As limitações representam a principal oportunidade de melhoria identificada. A ampliação do conjunto de dados, com maior diversidade de usinas e condições de captura, tende a beneficiar diretamente o desempenho do modelo, sobretudo nas métricas de recall e $mAP@0,5:0,95$, e constitui direção relevante para trabalhos futuros.

No trabalho de Moussaoui *et al.* (2025), a arquitetura YOLOv8 aplicada à detecção de painéis fotovoltaicos em imagens aéreas alcançou precisão de 96,7%. O modelo treinado no presente trabalho obteve precisão de 90,9% e $mAP@0,5$ de 0,81, valores da mesma ordem de grandeza. A diferença é coerente com as diferenças metodológicas: a anotação por fileira completa, geometricamente mais desafiadora que a anotação por painel individual, e a maior heterogeneidade do conjunto de dados, composto por usinas de resoluções e altitudes distintas, conforme discutido nos parágrafos anteriores. Considerados os fatores, o desempenho obtido é adequado para a tarefa de localização das fileiras e coerente com o relatado na literatura para detectores da família YOLO no mesmo domínio.

4.2 CRITÉRIOS DE AVALIAÇÃO DA ROTA

A rota gerada é avaliada segundo três critérios complementares. O primeiro é a completude da cobertura, que mede a fração das fileiras detectadas efetivamente atendidas pela rota. Uma fileira é considerada atendida quando o percurso passa rente a pelo menos metade de sua extensão, em um de seus lados, dentro de uma faixa de tolerância proporcional ao espaçamento

entre fileiras.

O segundo critério é a eficiência do percurso, expressa pela razão entre a distância total da rota e o comprimento total dos corredores obrigatórios L_{req} . Como o percurso inclui trechos de conexão entre corredores, a razão é superior a um, e valores mais próximos de um indicam menor deslocamento improdutivo.

O terceiro critério é a comparação com o Problema do Carteiro Chinês, descrito na Seção 2.4.5, que percorre todas as arestas do grafo, incluindo o perímetro, e não apenas os corredores obrigatórios. A solução do carteiro chinês, obtida de forma exata (EDMONDS; JOHNSON, 1973), fornece uma rota de referência que cobre toda a estrutura de vias. A economia de deslocamento da rota do carteiro rural em relação a referência quantifica o ganho obtido ao percorrer apenas os corredores necessários.

4.3 RESULTADOS DA GERAÇÃO DA ROTA

Para ilustrar o funcionamento do pipeline completo, apresenta-se em detalhe o resultado obtido para uma usina representativa, cujas etapas intermediárias de geração da máscara navegável, extração do esqueleto e construção do grafo foram apresentadas no Capítulo 3. Na usina, o modelo de detecção localizou 90 fileiras de módulos, a partir das quais se construiu o grafo de corredores e se gerou a rota de cobertura. A rota resultante, apresentada na Figura 28, atendeu 88 das 90 fileiras, o que corresponde a uma cobertura de 98%, com um comprimento total equivalente a 1,14 vez o comprimento dos corredores obrigatórios. Em relação ao Problema do Carteiro Chinês, que percorreria também o perímetro completo da usina, a rota do carteiro rural apresentou uma economia de deslocamento de 21,6%.

Para verificar a generalização do método além do exemplo, o pipeline foi aplicado a um conjunto de usinas reais de diferentes escalas e formatos. A Tabela 6 apresenta os resultados para cinco usinas representativas, ordenadas pelo número de fileiras, abrangendo desde plantas pequenas, com algumas dezenas de fileiras, até uma planta de grande porte com mais de duzentas fileiras.

Tabela 6 – Resultados da geração da rota em usinas representativas.

Usina	Fileiras atendidas	Cobertura	Corredores (px)	Razão	Economia
A	27/28	96%	6722	1,175	28,0%
B	34/35	97%	9736	1,243	21,2%
C	40/40	100%	11561	1,163	20,6%
D	88/90	98%	19247	1,138	21,6%
E	231/237	97%	28746	1,174	25,7%
Média	—	97,6%	—	1,179	23,4%

Fonte: O Autor (2026).

Em todas as usinas avaliadas, a cobertura manteve-se entre 96% e 100%, indicando que

a rota percorre de forma consistente os corredores adjacentes às mesas fotovoltaicas. A razão de eficiência permaneceu próxima de um, refletindo o baixo deslocamento improdutivo da rota aberta, e a economia em relação ao carteiro chinês foi positiva em todos os casos, confirmando a vantagem de restringir o percurso aos corredores efetivamente necessários. O comportamento manteve-se estável da menor à maior planta avaliada, o que indica que a abordagem é aplicável a usinas de diferentes dimensões.

A formulação adotada para a rota encontra respaldo na literatura de roteamento em partes. O Problema do Carteiro Rural é um modelo consolidado para percorrer apenas um subconjunto de arestas obrigatórias a custo mínimo (EISELT; GENDREAU; LAPORTE, 1995), e formulações da família do carteiro têm sido aplicadas ao planejamento de cobertura para inspeção de infraestrutura, nas quais restringir o percurso às arestas obrigatórias reduz o custo total (MATLEKOVIC; SCHNEIDER-KAMP, 2023). No domínio específico das usinas fotovoltaicas, contudo, o planejamento de rotas de manutenção tem sido tratado predominantemente como roteamento de pontos, por meio do Problema do Caixeiro Viajante e suas variantes, ou como cobertura de área por VANT (PÉREZ-GONZÁLEZ *et al.*, 2021), enquanto a inspeção terrestre dos corredores tem sido abordada por planejamento ponto a ponto baseado em amostragem (CONG, 2023). Na literatura consultada, não foi identificado trabalho que quantifique a economia de deslocamento de uma rota de cobertura de corredores em relação ao Problema do Carteiro Chinês em usinas fotovoltaicas. Assim, a economia média de 23,4% obtida constitui uma contribuição própria deste trabalho, cuja adequação é avaliada por critérios internos, a cobertura entre 96% e 100% e a razão de eficiência próxima da unidade, na ausência de métrica diretamente equivalente na literatura correlata.

4.4 LIMITAÇÕES

A abordagem apresenta limitações que delimitam seu escopo de aplicação e indicam direções de aprimoramento. A completude da cobertura depende da qualidade das etapas que antecedem o roteamento, em especial da detecção das fileiras e da extração do grafo de corredores. Em plantas com fileiras muito densamente compactadas, ou com a área dividida em blocos fisicamente separados, a construção do grafo a partir do esqueleto pode capturar apenas parte da estrutura interna de corredores, o que reduz a cobertura da rota nos cenários. O aprimoramento da etapa de extração, de modo a tratar de forma robusta arranjos fragmentados e o roteamento de múltiplos blocos, constitui a principal direção de trabalho futuro.

Outras limitações dizem respeito a aspectos específicos do método. A classificação entre corredor obrigatório e trecho conector baseia-se em um teste geométrico simples, que pode não distinguir corretamente as vias em arranjos atípicos de fileiras. A heurística de Frederickson fornece uma solução aproximada, e não necessariamente ótima, para o Problema do Carteiro Rural. Por fim, a ausência de informação de *Ground Sample Distance* (GSD) faz com que todas as distâncias sejam expressas em *pixels*, o que impede a conversão direta dos resultados para

unidades métricas e constitui outra direção de trabalho futuro.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver um sistema baseado em visão computacional e algoritmos clássicos de otimização capaz de detectar automaticamente as fileiras de módulos fotovoltaicos em imagens aéreas e gerar uma rota de cobertura para o deslocamento de equipes de manutenção pelos corredores das usinas. A motivação partiu dos custos e dos riscos associados às inspeções manuais em usinas de grande porte, que demandam tempo, recursos humanos e expõem os trabalhadores a condições adversas.

Para a etapa de detecção, foi construído um conjunto de dados com 488 imagens aéreas de usinas fotovoltaicas obtidas na plataforma *Roboflow*, anotadas com uma única classe, correspondente à fileira de módulos. O modelo de detecção foi treinado por meio de aprendizado por transferência a partir da arquitetura YOLOv8m, com o otimizador *Stochastic Gradient Descent*, que apresentou desempenho superior ao *AdamW* nos experimentos comparativos. A variante *medium* foi adotada como modelo definitivo por oferecer o melhor compromisso entre acurácia e custo computacional frente às variantes *nano* e *large*. O modelo final alcançou precisão de aproximadamente 0,91 e *mAP@0,5* de 0,81 sobre o conjunto de teste, desempenho considerado adequado para a tarefa de localização das fileiras.

A partir das detecções, foi desenvolvido um pipeline de planejamento de rota composto pela geração de uma máscara navegável confinada ao contorno da usina, pela extração do esqueleto dos corredores, pela construção de um grafo ponderado em *pixels* e pela formulação da inspeção como um Problema do Carteiro Rural, resolvido pela heurística de Frederickson com a geração de uma rota aberta. Nas usinas avaliadas, a cobertura dos corredores ficou entre 96% e 100%, e a rota gerada apresentou economia de deslocamento superior a 23%, em média, em relação à formulação do Problema do Carteiro Chinês, que percorreria também o perímetro completo da planta. O comportamento manteve-se estável em usinas de diferentes dimensões, desde plantas pequenas até uma planta de grande porte com mais de duzentas fileiras.

Apesar dos resultados favoráveis, o sistema apresenta limitações. A completude da cobertura depende da qualidade das etapas anteriores ao roteamento, e em plantas com fileiras muito densamente compactadas ou com a área dividida em blocos fisicamente separados a extração do grafo de corredores pode capturar apenas parte da estrutura interna, reduzindo a cobertura. Além disso, a ausência de informação de *Ground Sample Distance* (GSD) faz com que as distâncias sejam expressas em *pixels*, sem conversão direta para unidades métricas. As limitações não comprometem a viabilidade da abordagem nas usinas de layout regular, mas delimitam seu escopo de aplicação e orientam os aprimoramentos futuros.

De modo geral, o trabalho demonstrou que a integração entre detecção de objetos por aprendizado profundo e algoritmos clássicos de otimização em grafos constitui uma alternativa

viável para automatizar o planejamento de inspeções em usinas fotovoltaicas, oferecendo uma base que pode ser estendida em aplicações futuras de operação e manutenção.

5.1 TRABALHOS FUTUROS

Como sugestões para trabalhos futuros, têm-se:

- Aprimorar a etapa de extração do grafo de corredores, de modo a tratar de forma robusta arranjos de fileiras densamente compactados e usinas divididas em múltiplos blocos, incluindo o roteamento de componentes fisicamente separadas;
- Ampliar e diversificar o conjunto de dados de treinamento, com maior variedade de usinas e de condições de captura, a fim de melhorar a capacidade de generalização do modelo de detecção em imagens fora da distribuição de treino;
- Incorporar a informação de *Ground Sample Distance* (GSD) das imagens, permitindo expressar as distâncias e a rota em unidades métricas;
- Avaliar o treinamento com resolução de entrada superior à utilizada neste trabalho, mediante a disponibilidade de hardware com maior capacidade de memória, investigando seu efeito sobre a precisão geométrica das detecções.

REFERÊNCIAS

ABSOLAR. **O que é energia solar fotovoltaica?** 2024. Acesso em: 19 maio 2025. Disponível em: <<https://www.absolar.org.br/mercado/o-que-e-energia-solar-fotovoltaica/>>.

_____. **Infográfico - Evolução da Fonte Solar Fotovoltaica no Brasil.** 2025. Acesso em: 15 maio 2025. Disponível em: <<https://www.absolar.org.br/mercado/infografico/>>.

CHOLLET, F. **Deep Learning with Python.** Manning Publications, 2018. Acesso em: 06 jun. 2025. Disponível em: <<https://www.manning.com/books/deep-learning-with-python>>.

CHOSSET, H. Coverage of known spaces: the boustrophedon cellular decomposition. **Autonomous Robots**, v. 9, n. 3, p. 247–253, 2000.

CONG, W. A path planning algorithm of inspection robots for solar power plants based on improved RRT*. **Electronics**, v. 12, n. 21, p. 4455, 2023.

CORMEN, T. H. *et al.* **Introduction to Algorithms.** 3. ed. Cambridge, MA: MIT press, 2009.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische mathematik**, Springer, v. 1, n. 1, p. 269–271, 1959.

DOREMAELE, E. V. V. *et al.* Hardware implementation of backpropagation using progressive gradient descent for in situ training of multilayer neural networks. **Science Advances**, v. 10, 2024.

EDMONDS, J.; JOHNSON, E. L. Matching, Euler tours and the Chinese postman. **Mathematical Programming**, v. 5, n. 1, p. 88–124, 1973.

EISELT, H. A.; GENDREAU, M.; LAPORTE, G. Arc routing problems, part II: The rural postman problem. **Operations Research**, v. 43, n. 3, p. 399–414, 1995.

ELGENDY, M. **Deep Learning for Vision Systems.** Manning Publications, 2020. Acesso em: 05 jun. 2025. Disponível em: <<https://www.manning.com/books/deep-learning-for-vision-systems/>>.

FEOFILOFF, P. **Algoritmos para grafos: Dijkstra.** 2020. Acesso em: 04 nov. 2025. Disponível em: <https://www.ime.usp.br/~{ }pf/algoritmos/_para/_grafos/aulas/dijkstra.ht>.

FREDERICKSON, G. N. Approximation algorithms for some postman problems. **Journal of the ACM**, v. 26, n. 3, p. 538–554, 1979.

GALCERAN, E.; CARRERAS, M. A survey on coverage path planning for robotics. **Robotics and Autonomous Systems**, v. 61, n. 12, p. 1258–1276, 2013.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing.** 4. ed. New York: Pearson, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning.** MIT Press, 2016. Acesso em: 02 jun. 2025. Disponível em: <<https://www.deeplearningbook.org/>>.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. [S.l.]: Pearson Education, 2009. Acesso em: 5 jun. 2025.

JOCHER, G. *et al.* **Ultralytics YOLO**. 2024. Software library. Disponível em: <<https://github.com/ultralytics/ultralytics>>.

KIATREUNGWATTANA, K.; VANGEET, O.; STOLTENBERG, B. **Facility-Scale Solar Photovoltaic Guidebook: Bureau of Reclamation**. Golden, CO, 2016. Acesso em: 16 maio 2025. Disponível em: <<https://www.nrel.gov/publications/>>.

LAVALLE, S. M. **Planning Algorithms**. Cambridge, UK: Cambridge University Press, 2006.

LI, Z. *et al.* A survey of convolutional neural networks: Analysis, applications, and prospects. **arXiv preprint arXiv:2004.00346**, 2020.

MATLEKOVIC, L.; SCHNEIDER-KAMP, P. Constraint programming approach to coverage-path planning for autonomous multi-uav infrastructure inspection. **Drones**, v. 7, n. 9, p. 563, 2023.

MOUSSAOUI, C. *et al.* YOLO-based photovoltaic panel detection: A comparative study. In: **Smart Computing and Control Renewable Energy Systems. IC-AIRES 2024**. [S.l.]: Springer, 2025. (Lecture Notes in Networks and Systems, v. 1238).

NAEEM, U. *et al.* Aerial imaging-based soiling detection system for solar photovoltaic panel cleanliness inspection. **Sensors**, MDPI, v. 25, n. 738, 2025.

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010.

PEEMEN, M.; MESMAN, B.; CORPORAAL, H. Efficiency optimization of trainable feature extractors for a consumer platform. In: **Advanced Concepts for Intelligent Vision Systems**. Berlin, Heidelberg: Springer, 2011. (Lecture Notes in Computer Science, v. 6915), p. 396–407.

PÉREZ-GONZÁLEZ, A. *et al.* Coverage path planning with semantic segmentation for UAV in PV plants. **Applied Sciences**, v. 11, n. 24, p. 12093, 2021.

PORTAL ENERGIA. **Medidas de painéis solares fotovoltaicos: dimensões e características**. 2024. Acesso em: 01 nov. 2025. Disponível em: <<https://www.portal-energia.com/medidas-paineis-solares-fotovoltaicos-147346>>.

PORTAL SOLAR. **Energia solar foi a fonte que mais cresceu no Brasil em 2024**. Portal Solar, 2024. Acesso em: 6 maio 2025. Disponível em: <<https://www.portalsolar.com.br/noticias/operacao-e-expansao/oem/energia-solar-foi-a-fonte-que-mais-cresceu-no-brasil-em-2024>>.

_____. **Geração Distribuída de Energia: o que é e como funciona**. 2024. Acesso em: 19 maio 2025. Disponível em: <<https://www.portalsolar.com.br/geracao-distribuida-de-energia.html>>.

_____. **Tudo sobre a manutenção do painel solar**. 2024. Acesso em: 01 nov. 2025. Disponível em: <<https://www.portalsolar.com.br/tudo-sobre-a-manutencao-do-painel-solar>>.

_____. **Usina solar: o que é, tipos, como funciona, vantagens e desvantagens.** 2025. Acesso em: 28 maio 2025. Disponível em: <<https://www.portalsolar.com.br/usina-solar.html>>.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 779–788.

ROBOFLOW. **Roboflow Documentation.** 2025. Acesso em: 03 nov. 2025. Disponível em: <<https://docs.roboflow.com>>.

SIEGWART, R.; NOURBAKHSI, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots.** 2. ed. Cambridge, MA: MIT press, 2011.

SOLARVOLT ENERGIA. **Geração Distribuída e Geração Centralizada de Energia: Entenda as Diferenças.** 2024. Acesso em: 21 maio 2025. Disponível em: <<https://www.solarvolt.com.br/geracao-distribuida-e-geracao-centralizada-energia/>>.

_____. **Quais são os principais componentes de uma usina fotovoltaica?** 2024. Acesso em: 21 maio 2025. Disponível em: <<https://solarvolt.com.br/componentes-usina-fotovoltaica/>>.

SZELISKI, R. **Computer Vision: Algorithms and Applications.** 1st. ed. London: Springer, 2010. (Texts in Computer Science). ISBN 978-1848829343.

TERVEN, J.; CÓRDOVA-ESPARZA, D.-M.; ROMERO-GONZÁLEZ, J.-A. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. **Machine Learning and Knowledge Extraction**, v. 5, p. 1680–1716, 2023.

VILLALVA, M. G.; GAZOLI, J. R. **Energia Solar Fotovoltaica: Conceitos e Aplicações.** 1. ed. São Paulo: Érica, 2012.

WEG. **Energia Solar: Como Ela é Convertida em Eletricidade.** 2025. Acesso em: 11 abr. 2025. Disponível em: <<https://www.weg.net/solar/blog/energia-solar-como-ela-e-convertida-em-eletricidade/>>.

WORLD ECONOMIC FORUM. **Global renewable energy capacity grew by a record-breaking 15.1% in 2024, IRENA reports.** World Economic Forum, 2025. Acesso em: 6 maio 2025. Disponível em: <<https://www.weforum.org/stories/2025/04/renewable-energy-transition-wind-solar-power-2024>>.

ZHAO, X. *et al.* A review of convolutional neural networks in computer vision. **Artificial Intelligence Review**, Springer, v. 57, n. 99, 2024.