UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

Lucas Goulart Vazquez

Detecção de falhas em painéis fotovoltaicos usando redes neurais

Sorocaba - SP
2023

Lucas Goulart Vazquez

Detecção de falhas em painéis fotovoltáicos usando redes neurais

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba (ICTS), Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), como parte dos requisitos para obtenção do grau de bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Galdenoro Botura Junior

Sorocaba - SP
2023

V393d

Vazquez, Lucas Goulart
Detecção de falhas em painéis fotovoltaicos usando redes neurais / Lucas Goulart Vazquez. -- Sorocaba, 2023
46 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Galdenoro Botura Junior

1. Inteligência artificial. 2. Redes neurais (Computação). 3. Energia Solar. 4. Visão por computador. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

DETECÇÃO DE FALHAS EM PAINÉIS FOTOVOLTAICOS USANDO REDES NEURAIS

LUCAS GOULART VAZQUEZ

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Profº. Dr. Everson Martins

Coordenador

BANCA EXAMINADORA:

Prof. Dr. GALDENORO BOTURA JR

Orientador/UNESP-Campus de Sorocaba

Prof. Dr. EDUARDO VERRI LIBERADO

UNESP-Campus de Sorocaba

Prof. Dr. ALYSSON FERNANDES MAZONI

UNICAMP

Dezembro de 2023

Agradecimentos

Aos meus pais, Emilio Vazquez Claro e Silvia Augusta do Carmo Goulart, por tornarem tudo isso possível. Ao professor Alysson Mazoni por não largar do meu pé. A Adriano Luiz Imenes Dias por não me deixar afogar em burocracia. Ao meu orientador professor Galdenoro Botura Junior por me ligar após meses de silencio e dar o ultimo empurrãozinho.

Resumo

A geração de energia em residências por meio de placas solares, também conhecida como energia solar fotovoltaica, é uma prática sustentável e renovável. Os principais benefícios estão relacionados ao fato de a energia solar fotovoltaica ser sustentável, renovável e abundante. A capacidade de geração de energia solar residencial varia, mas em média, painéis solares podem gerar eletricidade suficiente para abastecer milhões de lares. Com a diminuição dos custos ao longo dos anos, a energia solar tornou-se uma opção acessível para a população. O objetivo do projeto é desenvolver uma metodologia por meio de um conjunto de programas que permita detectar falhas na geração através de fotos dos painéis fotovoltaicos. Para tal fim, técnicas baseadas em processamento de sinais não são suficientes devido à grande variabilidade de cenários possíveis que cercam as falhas, ainda que as falhas sejam muito semelhantes em aparências. As técnicas usadas são baseadas em redes neurais de aprendizado profundo como as que são comumente usadas para detecção de características em imagens. Dada a dificuldade de obtenção de imagens marcadas e a natureza padronizada dos problemas, utilizou-se a técnica de geração automática de imagens por meio de um sistema de simulação tridimensional. Assim com as imagens geradas, puderam-se treinar redes neurais para detecção dessas falhas. Os resultados mostraram que houve uma exatidão de cerca de 98% na detecção dos painéis e 87% na detecção de falhas, quando se utilizou a metodologia aqui apresentada.

Palavras-chave: inteligência artificial; redes neurais (computação); energia solar; visão por computador.

Abstract

Generating energy in homes through solar panels, also known as photovoltaic solar energy, is a sustainable and renewable practice. The main benefits are related to the fact that solar energy is sustainable, renewable, and abundant. The capacity for residential solar energy generation varies, but on average, solar panels can generate enough electricity to power millions of homes. With decreasing costs over the years, solar energy has become an affordable option for the general population. This project's objective is to develop a methodology through a set of programs that allows the detection of faults in generation through photos of photovoltaic panels. For this purpose, techniques based on signal processing are not sufficient due to the significant variability of possible scenarios surrounding the faults, even if the faults appear very similar. The techniques used are based on deep learning neural networks, similar to those commonly used for feature detection in images. Given the difficulty of obtaining labeled images and the standardized nature of the problem, the technique of automatic image generation through a three-dimensional simulation system was used. With the generated images, neural networks were trained to detect these faults. The results showed an accuracy of about 98% in panel detection and 87% in fault detection when using the presented methodology.

Keywords: artificial intelligence; artificial neural networks; solar energy; computer vision.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

1.1 Descrição do problema

Os painéis para geração de energia fotovoltaica estão crescentemente em uso com propósito de micro-geração distribuída e também para fornecer energia elétrica em locais distantes de linhas de alta tensão, (CUCCHIELLA; D'ADAMO; GASTALDI, 2016; ZAHEDI, 2006).

Sua vida útil atingiu o ponto de compensarem seu alto custo de instalação. Uma dificuldade de sua manutenção quando instalada em locais de acesso difícil é a substituição de elementos defeituosos, (CARNEVALE; LOMBARDI; ZANCHI, 2014; RAUGEI; FRANKL, 2009).

As falhas na fabricação dos painéis solares podem resultar de processos inadequados, controle de qualidade deficiente ou materiais defeituosos, como variações na espessura da camada fotovoltaica, presença de impurezas e conexões mal soldadas. Esses problemas concretos têm o potencial de comprometer significativamente a eficiência do painel e reduzir sua vida útil.

Em um cenário de crescente demanda por energia solar (FAUZI et al., 2023), a compreensão e o aprimoramento dos processos de fabricação tornam-se cruciais para assegurar a confiabilidade e o desempenho dos sistemas fotovoltaicos. Paralelamente, a degradação natural das células fotovoltaicas é um fenômeno inevitável ao longo do tempo. A exposição constante à radiação solar, as variações climáticas e os ciclos de temperatura contribuem para a degradação progressiva dos materiais semicondutores, resultando na diminuição gradual da eficiência energética ao longo dos anos.

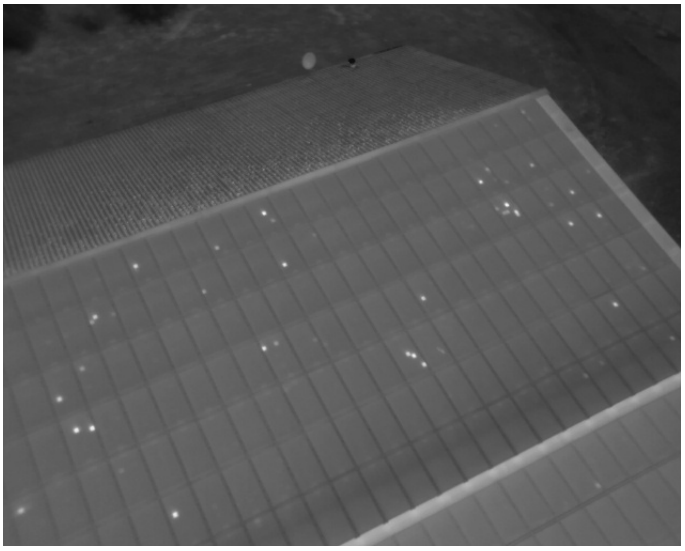
Em um contexto atual, onde a transição para fontes de energia mais sustentáveis é uma prioridade global (ARENT; WISE; GELMAN, 2011), compreender os mecanismos dessa degradação é essencial. Isso não apenas para desenvolver estratégias de manutenção eficazes, mas também para maximizar a vida útil dos painéis solares e promover sua contribuição contínua para o fornecimento de energia limpa.

Nesse cenário global, a superação desses desafios é fundamental para consolidar a posição da energia solar como uma fonte sustentável e viável no mix energético mundial (REIS, 2002; REIS; BOTURA JUNIOR; SILVEIRA, 2003). Ao enfrentar e resolver questões relacionadas à fabricação e degradação celular, a indústria solar poderá desempenhar um papel cada vez mais significativona transição para um futuro mais sustentável e ecologicamente equilibrado.

As falhas de desgastes são usualmente visíveis como marcas aproximadamente circulares mais claras. Essas marcas também ficam mais nítidas em imagens de espectro infravermelho, como na figura 1. Isso ocorre porque pontos de conversão ineficiente ou inexistente da luz solar devem aquecer mais. Ou seja, a ausência de efeito fotoelétrico que direcione a energia dos fótons

de luz faz que mais energia se converta em calor (HONG; PULA, 2022).

Figura 1 – Falhas visíveis em painel fotovoltaico usando foto de espectro infravermelho.



Fonte: (MAZONI, 2021).

O diagnóstico de falhas em painéis fotovoltaicos em instalações pouco acessíveis usa imagens obtidas com drones, (HENRY et al., 2020). Os drones podem ser equipados com câmeras infravermelhas para aproveitar o efeito mencionado, (ALSAFASFEH et al., 2018). Porém, as imagens obtidas por drones são observadas individualmente para o diagnóstico. Nesse sentido, este trabalho propõe técnicas que combinam redes neurais e informações do cinemáticas do drone para a localização de falhas.

O problema de localização de falhas poderia ser dividido em dois: o de localizar as falhas na imagem e o de localizar os painéis fotovoltaicos. O cruzamento dessas duas respostas permite localizar as falhas com uma localização aproximada dentro dos painéis, bem como garantir maior robustez ao método, pois falhas localizadas fora da área dos painéis não são falhas de fato.

1.2 Localização de falhas na imagem

Uma imagem gerada por drone geralmente é enviada sem fio para um computador pelo programas já usados para o controle de drones. Disponível assim para automação de um processo de análise imediatamente após a captura.

Uma imagem típica capturada por um drone com a presença de falhas apresentará o painel com pontos mais claros, como na figura 1 em que a imagem apresentada é de uma fotografia por infravermelho.

A localização de objetos extensos em imagens depende de um conjunto volumoso de imagens com esses objetos marcados. Ou seja, de uma entrada que contenha a posição na imagem do que se deseja localizar. Para o treinamento de redes neurais, é comum que isso seja na forma

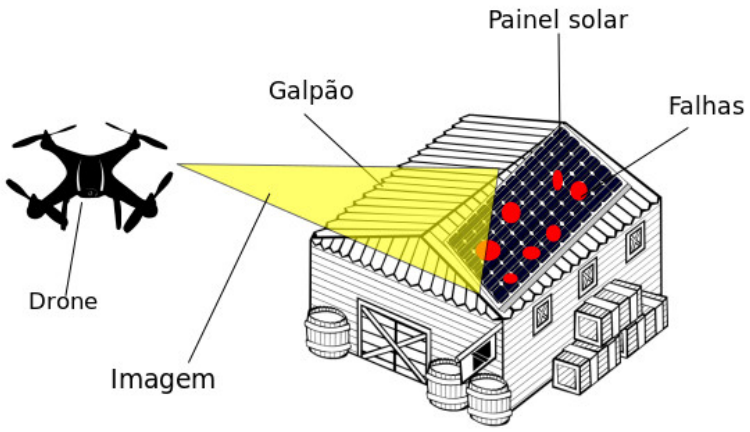
de uma máscara, isto é, de uma matriz com as mesmas dimensões em pixels que a imagem. Para cada posição da matriz há uma marcação de que aquele pixel representa um ponto do objeto a ser detectado ou não. Em alguns casos podem-se também definir pixels como pertencentes à fronteira do objeto. Se definem três possibilidades, pixel do painel, falha ou fundo.

A produção de imagens e de rótulos para o treinamento de uma rede neural, como em outros problemas, é custosa. Cada imagem precisaria ter seus pixels marcados manualmente para definir a presença das falhas. Trata-se de mais um contexto para a utilização de imagens sintetizadas em computação gráfica como entradas de treinamento usando o conceito de um mundo simulado, (MAZONI, 2021).

1.3 Localização no sistema do drone

Todos os drones possuem sistemas de navegação complexos que se baseiam em fusão de sensores, (DAPONTE et al., 2015). A tecnologia já tem maturidade permitindo conhecer de forma precisa a posição em coordenadas geo-estacionárias e outros parâmetros cinemáticos como altitude e inclinação, (SORBELLI et al., 2018). Um esquema de funcionamento do sistema é apresentado na figura 2.

Figura 2 – Funcionamento da detecção de falhas em uma situação prática.



Fonte: (MAZONI, 2021).

A posição sobre o solo e os parâmetros cinemáticos de altitude rotação e inclinação são

conhecidos com ordens de grandeza diferentes. A posição é obtida pelo sinal de GPS, e os outros parâmetros pelas leituras de acelerômetros e magnetômetros. A fusão de sensores é usada com filtros de Kalman para obter essas medidas e também suas derivadas como velocidades lineares e angulares, (DAPONTE et al., 2015).

Usando esses parâmetros, a direção de visada da câmera é resultado direto das medidas do drone. E portanto, artefatos localizados em uma imagem capturada pela câmera podem ter sua posição estimada em relação ao mundo. Ao detectar uma falha, através da câmera e localizá-la em termos da posição na imagem, pode-se obter a localização da falha no painel usando a separação proposta pela rede neural. Além disso, a localização do painel na imagem, permite a determinação de que painel se trata em um banco de dados que tenham localizações de paineis registradas.

A localização do painel identificado em uma imagem pode ser extrapolada para uma localização no mundo usando a projeção do sistema de coordenadas da câmera para o sistema de coordenadas no mundo pela rotação conhecida dos sensores do drone. A única ambiguidade restante é a distância até a câmera. Uma imagem apenas não pode diferenciar entre um objeto grande distante e um objeto pequeno próximo. Porém, tal ambiguidade é resolvida quando se supõe o conhecimento das dimensões dos painéis. Quando ajustadas as dimensões reais para as dimensões na imagem, o fator encontrado é a distância da câmera.

2 Referencial teórico

2.1 Redes neurais

O cérebro ou rede neural biológica é considerado o sistema mais bem organizado para processar informações de diferentes sentidos, como visão, audição, tato, paladar e olfato de maneira eficiente e inteligente. Um dos principais mecanismos de processamento de informações no cérebro humano é que as informações complicadas de alto nível são processadas por meio da colaboração, ou seja, das conexões (chamadas sinapses), de um grande número de elementos estruturalmente simples (chamados neurônios). Na aprendizagem de máquina, as redes neurais artificiais são uma família de modelos que imitam a elegância estrutural do sistema neural e aprendem padrões inerentes às observações (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.1 Redes neurais *feedforward*

Redes neurais *feedforward* (também conhecidas como perceptrons multicamadas) são um tipo de modelo de rede neural artificial que consiste em camadas de neurônios interconectados, onde a informação flui em uma única direção, da camada de entrada para a camada de saída (GOODFELLOW; BENGIO; COURVILLE, 2016). Isso é chamado de "feedforward" porque não há realimentação ou ciclos de retroalimentação na rede. Essas redes são amplamente usadas para tarefas de aprendizado supervisionado, como classificação e regressão.

Para ilustrar uma rede neural *feedforward*, considere uma rede simples com uma camada de entrada, uma camada oculta e uma camada de saída. Cada camada é composta por neurônios interconectados. A rede processa um vetor de entrada $\mathbf{X}$ e gera uma saída $\mathbf{Y}$. Cada conexão entre neurônios tem um peso associado.

A propagação de informações na rede pode ser representada por uma série de equações matemáticas. Primeiro, a saída de um neurônio em uma camada é calculada usando a função de ativação, que é frequentemente uma função não linear, como a função sigmoide (σ) ou a função de ativação ReLU (*Retificação Linear Unit*):

$$a_i = f\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right)$$

Onde:

- a_i é a saída do neurônio i na camada.
- w_{ij} é o peso da conexão entre o neurônio i e o neurônio j.

- x_j é a saída do neurônio j na camada anterior (para a camada de entrada, isso é o valor de entrada real).
- b_i é o termo de polarização (*bias*) do neurônio i.
- f é a função de ativação aplicada à soma ponderada.

A saída da camada de entrada é o vetor de entrada $\mathbf{X}$ e a saída da camada de saída é o vetor $\mathbf{Y}$. Para a camada oculta, a saída da camada anterior é usada como entrada.

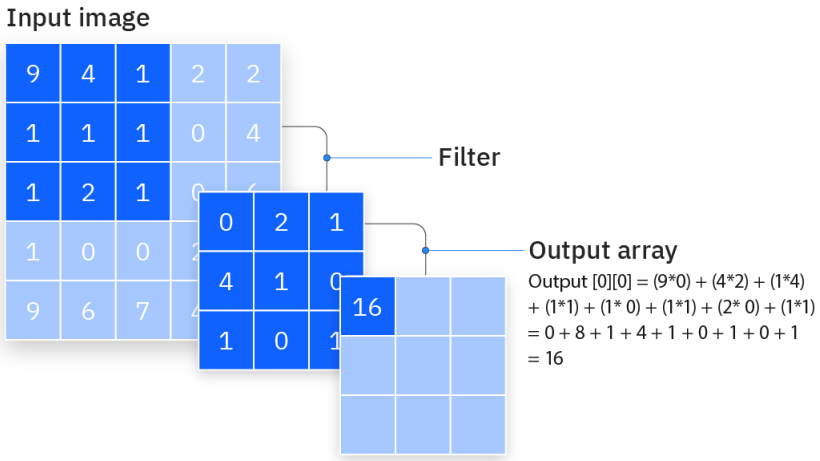
A propagação ocorre de camada em camada até que a saída final seja produzida. Para treinar a rede, é usado um algoritmo de retropropagação do erro (*backpropagation*) para ajustar os pesos das conexões, minimizando o erro entre a saída prevista e a saída desejada.

Em resumo, as redes neurais *feedforward* processam informações em uma única direção, passando pela rede de entrada para a saída, usando funções de ativação nos neurônios para introduzir não linearidades. Elas são capazes de aprender relações complexas nos dados, tornando-as valiosas para uma variedade de tarefas de aprendizado de máquina.

2.1.2 Redes Convolucionais

Existem algumas arquiteturas de redes que são utilizadas no campo de visão computacional, a mais comum e também utilizada nesse trabalho é conhecida como Rede Neural Convolucional (CNN). Essas redes são eficientes em processar imagens devido a utilização de uma topologia em estilo de grade como mostrado na Figura 3.

Figura 3 – Filtro (kernel) de uma CNN.



Fonte: (IBM, 2023)

Cada camada da CNN realiza a operação de convolução, pode-se imaginar que cada unidade da rede desliza sobre imagem de forma que podemos imaginar cada conjunto de unidades como uma janela.

Após o treinamento cada camada da CNN acaba se especializando em detectar características especificas da imagem (LI et al., 2020) – camadas iniciais detectam características mais básicas como cantos e formas geométricas, camadas mais profundas detectam características mais especificas e "alto nível". Como exemplo, vamos supor que estamos tentando determinar se uma imagem contém uma bicicleta. Você pode pensar na bicicleta como um conjunto de partes. Ela é composta por um quadro, guidão, rodas, pedais, etc. Cada parte individual da bicicleta compõe um padrão de nível inferior na rede neural, e a combinação de suas partes representa um padrão de nível superior, criando uma hierarquia de características dentro da CNN como mostrado na Figura 4. Em última análise, a camada convolucional converte a imagem em valores numéricos, permitindo que a rede neural interprete e extraia padrões relevantes (IBM, 2023).

Figura 4 – Hierarquia de características dentro da CNN.

Fonte: (IBM, 2023)

Podemos a cada camada reduzir a dimensionalidade da imagem ao controlar como cada janela é deslizada. Isso ajuda a reduzir a complexidade computacional para cada imagem de entrada, possibilitando redes mais profundas.

2.1.3 U-Net

O modelo U-Net é um modelo que foi desenvolvido por (RONNEBERGER; FISCHER; BROX, 2015a) para segmentação de imagens medicas com um pequeno numero de imagens de treinamento.

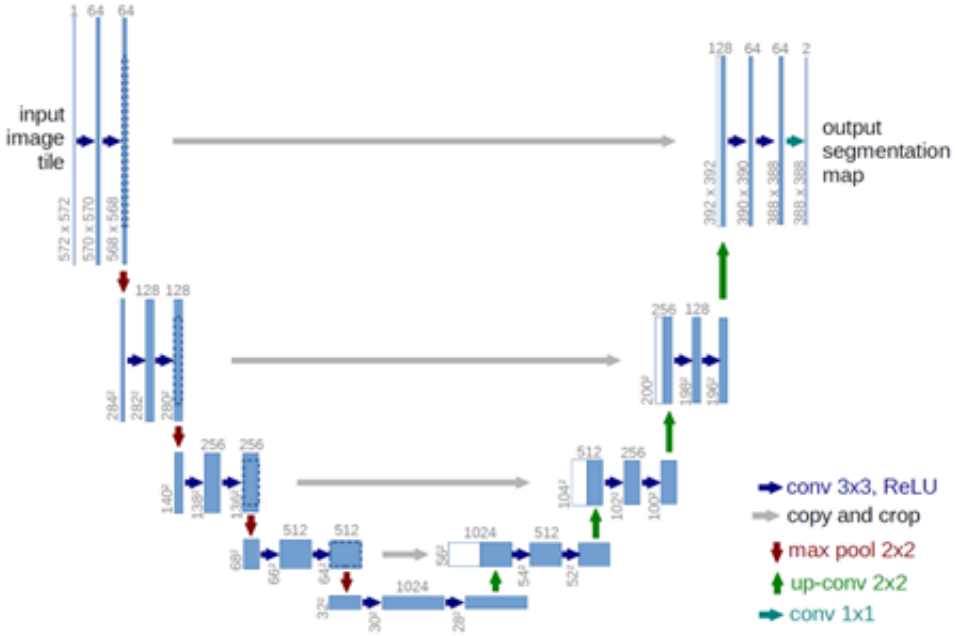
A arquitetura consiste em um caminho codificador e um caminho decodificador como pode ser observado na Figura 5. O caminho codificador é responsável por extrair características importantes da imagem e comprimi-las para um estado latente, cada camada codificadora reduz a dimensão espacial da imagem e aumenta o numero de canais. O decodificador parte do caminho latente e constrói o mapa de segmentação na saída, que tem o mesmo tamanho da imagem de

entrada, cada camada faz o processo reverso em comparação ao codificador, aumentando o tamanho da imagem e diminuindo o numero de canais.

Conexões "atalho" entre camadas do codificador e decodificador permitem a retenção de detalhes finos sobre a imagem de entrada, especialmente importante para uma segmentação precisa em pequenos objetos, como o de falhas em painéis solares. Apenas camadas convolucionais sem conter nenhuma camada linear são utilizadas, devido a isso é possível lidar com imagens de entrada de tamanho arbitrário, o que torna esse tipo de rede ideal para aplicações no mundo real onde o tamanho da imagem e resolução podem mudar.

A arquitetura U-Net tem sido utilizada em diversos campos onde segmentacao semântica é necessária, sendo alguns deles: analises pulmonar de raio-x (YAHYATABAR; JOUVET; CHERIET, 2020), imagens de satélite (ALSABHAN; ALOTAIBY et al., 2022), e carros autônomos (TRAN; LE, 2019).

Figura 5 – Arquitetura da rede U-Net.



Fonte: (IBM, 2023)

2.2 Entropia Cruzada (Cross Entropy)

A Entropia Cruzada é uma função de perda amplamente utilizada em tarefas de classificação (GORDON-RODRIGUEZ et al., 2020), incluindo segmentação semântica. Em termos simples, ela mede a diferença entre duas distribuições de probabilidade: a distribuição real (verdadeira) e a distribuição prevista pelo modelo. Matematicamente, a Entropia Cruzada $H(p, q)$ entre duas distribuições de probabilidade p e q é definida como:

$$H(p, q) = -\sum_{i} p(i) \log(q(i)) \tag{2.1}$$

onde $p(i)$ é a probabilidade real da classe i e $q(i)$ é a probabilidade prevista pelo modelo para a classe i.

2.2.1 Entropia Cruzada em Segmentação Semântica

Em tarefas de segmentação semântica, o objetivo é classificar cada pixel de uma imagem em uma das classes possíveis. Para uma imagem, a saída prevista é um mapa semântico onde cada pixel tem uma distribuição de probabilidade associada às classes possíveis. A Entropia Cruzada é aplicada pixel a pixel, comparando a distribuição de probabilidade prevista com a verdadeira anotação (geralmente codificada como um vetor one-hot).

2.2.2 Vantagens da Entropia Cruzada para Segmentação Semântica

1. **Penalização de Confiança Incorreta**: A Entropia Cruzada penaliza fortemente as previsões que são confiantes, mas erradas. Por exemplo, se um pixel pertence à classe A, mas o modelo prevê com alta confiança que pertence à classe B, o valor da entropia cruzada será alto. Isso é crucial para tarefas de segmentação semântica, pois garante que os erros graves sejam adequadamente penalizados.

2. **Suave para Previsões Corretas**: Para previsões corretas, especialmente aquelas feitas com alta confiança, a Entropia Cruzada oferece valores mais baixos, incentivando o modelo a fazer previsões corretas e confiantes.

3. **Diferenciável**: A função de Entropia Cruzada é diferenciável, o que é essencial para a otimização usando métodos baseados em gradientes, como o gradiente descendente.

4. **Adapta-se a Desbalanceamento de Classes**: A Entropia Cruzada pode ser facilmente combinada com ponderações de classe para lidar com desequilíbrios no conjunto de dados, uma situação comum em segmentação semântica, onde certas classes podem ser raras em comparação com outras.

5. **Consistência com Problemas de Classificação**: Como a segmentação semântica pode ser vista como um problema de classificação em nível de pixel, a utilização de Entropia Cruzada oferece uma abordagem intuitiva e consistente para medir o desempenho.

A Entropia Cruzada é uma escolha natural e eficaz como função de perda para tarefas de segmentação semântica devido à sua capacidade de lidar com previsões erradas e confiantes, sua diferenciabilidade e sua adaptabilidade a desequilíbrios de classes. Ela provou ser eficaz em muitos problemas práticos de segmentação, tornando-se o padrão de fato para muitos frameworks e aplicações.

2.3 Métricas

2.3.1 Precision (Precisão)

A precisão é a proporção de identificações positivas feitas corretamente. Em outras palavras, dentre todas as instâncias classificadas como positivas pelo modelo, quantas efetivamente são positivas (TERVEN et al., 2023).

$$\text{Precision} = \frac{\text{Verdadeiros Positivos (VP)}}{\text{Verdadeiros Positivos (VP)} + \text{Falsos Positivos (FP)}} \tag{2.2}$$

2.3.2 Recall (Sensibilidade ou Taxa de Verdadeiros Positivos)

Recall é a proporção de verdadeiros positivos que foram identificados corretamente. Indica, dentre todas as instâncias positivas reais, quantas foram corretamente classificadas pelo modelo (TERVEN et al., 2023).

$$\text{Recall} = \frac{\text{Verdadeiros Positivos (VP)}}{\text{Verdadeiros Positivos (VP)} + \text{Falsos Negativos (FN)}} \tag{2.3}$$

2.3.3 F1-Score

F1-Score é a média harmônica entre Precision e Recall. Em situações onde uma métrica (Precision ou Recall) é mais importante que a outra, a F1-Score pode não ser a melhor métrica. No entanto, quando ambas as métricas são igualmente importantes, a F1-Score é extremamente útil (TERVEN et al., 2023).

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

2.3.4 Análise Conjunta

Ao analisar modelos de classificação, é essencial considerar tanto a Precision quanto o Recall, em vez de confiar em apenas uma delas. Uma alta Precision com baixo Recall pode significar que o modelo é excessivamente cauteloso e perde muitas classificações positivas. Já um alto Recall com baixa Precision indica que o modelo tem muitos falsos positivos.

Em segmentação semântica, estamos interessados em classificar cada pixel de uma imagem em uma categoria específica. As métricas mencionadas acima também são relevantes neste contexto, mas têm algumas particularidades:

- **Precision**: Dos pixels classificados como pertencentes a uma determinada categoria, quantos efetivamente pertencem a essa categoria?

- **Recall**: Dos pixels reais de uma categoria, quantos foram corretamente classificados pelo modelo?
- **F1-Score**: Como as imagens podem ter milhões de pixels, pequenos erros podem levar a grandes diferenças na Precision e Recall. Portanto, o F1-Score pode ser uma métrica útil para ter uma visão geral do equilíbrio entre essas duas métricas.

No contexto da segmentação semântica, também é comum utilizar outras métricas, como a métrica Dice 2.3.5, que oferece uma perspectiva da sobreposição entre a previsão do modelo e a anotação real.

Ao avaliar modelos de segmentação semântica, é vital considerar o contexto do problema e as necessidades do projeto. Em algumas aplicações, pequenos erros de segmentação podem ser aceitáveis, enquanto em outras (por exemplo, aplicações médicas), um alto nível de precisão é essencial.

A métrica Dice, também conhecida como coeficiente de Dice, é uma medida estatística usada para calcular a similaridade entre dois conjuntos. É frequentemente usada em segmentação de imagens, especialmente em segmentação médica.

2.3.5 Dice

A fórmula do coeficiente Dice é dada por:

$$\text{Dice} = \frac{2 \times \text{Interseção}}{\text{União} + \text{Interseção}} \tag{2.5}$$

A métrica Dice varia de 0 a 1:

- Valor de 1: Indica perfeita concordância entre a predição e o ground truth.
- Valor de 0: Indica nenhuma concordância ou sobreposição entre a predição e o ground truth.

2.3.6 Dice vs Acurácia

Embora tanto a métrica Dice quanto a acurácia sejam usadas para medir o desempenho de um modelo, elas possuem diferenças cruciais:

1. **Desbalanceamento de Classes**: A acurácia pode ser enganosa em casos de desbalanceamento de classes. Em segmentação médica, o objeto de interesse (como um tumor) pode ocupar uma pequena porção da imagem. O coeficiente Dice, por outro lado, dá uma visão mais equilibrada.
2. **Foco na Sobreposição**: O coeficiente Dice foca especificamente na sobreposição entre a predição e o ground truth.

2.3.7 Vantagens do Dice em Segmentação Semântica

1. **Sensível a Pequenas Estruturas**: É sensível a pequenas estruturas de grande importância.
2. **Robustez ao Desbalanceamento de Classes**: Menos suscetível a ser enganada por desbalanceamento de classes.
3. **Mais Interpretável para Segmentação**: Mede a sobreposição direta entre a predição e o ground truth.
4. **Promove a Precisão e a Sensibilidade**: Leva em consideração tanto a precisão quanto a sensibilidade.

Em resumo, o coeficiente Dice é particularmente útil e informativo para tarefas de segmentação de imagens.

3 Materiais e métodos

3.1 Python

O Python (ROSSUM; DRAKE, 2009) emergiu como uma linguagem de programação central no contexto da inteligência artificial (IA) e do aprendizado de máquina (AM). Sua ascensão e consolidação neste campo podem ser atribuídas a várias características e vantagens intrínsecas à linguagem.

A sintaxe clara e legível do Python facilita a expressão de conceitos complexos de forma concisa. Tal legibilidade torna-se crucial quando pesquisadores e cientistas, que podem não ter uma base sólida em programação, buscam implementar algoritmos e modelos avançados. A natureza intuitiva do Python permite que ideias complexas sejam traduzidas em código de forma mais eficiente, sem a necessidade de extensas linhas de programação.

O vasto ecossistema de bibliotecas e frameworks do Python, dedicados a tarefas de IA e AM, é um de seus maiores atrativos. Ferramentas como TensorFlow (ABADI et al., 2016), PyTorch (PASZKE et al., 2019), Fastai (W3TECHS, 2017) e scikit-learn (PEDREGOSA et al., 2011) são instrumentais para diversas tarefas, desde a implementação de redes neurais profundas até análises estatísticas e processamento de dados.

A comunidade de desenvolvedores e pesquisadores em torno do Python é notavelmente ativa. Esta comunidade engajada não só contribui com uma variedade de recursos de aprendizado, mas também desempenha um papel vital na identificação e correção de problemas, garantindo que as bibliotecas e ferramentas estejam sempre evoluindo e mantendo-se atualizadas.

Do ponto de vista da integração, o Python exibe uma capacidade notável de interoperar com outras linguagens e plataformas. Isso permite que desenvolvedores combinem a simplicidade do Python com a eficiência e desempenho de linguagens de baixo nível, como C++.

Além disso, o Python tem recebido suporte substancial de grandes entidades tecnológicas, incluindo Google, Facebook e Microsoft. O investimento dessas corporações no desenvolvimento de ferramentas de IA específicas para Python não só endossa a linguagem, mas também promete suporte e inovação contínuos.

Em conclusão, o estabelecimento do Python no campo da inteligência artificial é uma consequência de suas características versáteis, um ecossistema robusto e o apoio contínuo da comunidade e da indústria. A combinação desses fatores tem solidificado sua posição como uma linguagem de escolha no panorama da IA e do AM.

3.2 PyTorch

PyTorch (PASZKE et al., 2019) é uma ferramenta para criação de redes neurais desenvolvida pelo facebook. Uma das características mais importantes desse pacote (e de qualquer pacote de redes neurais) é a habilidade de utilizar GPUs para acelerar a computação necessária para treinar redes profundas. GPUs são altamente eficazes em computação paralela, ideal para redes profundas. A biblioteca permite o movimento de dados entre o CPU e GPU de maneira fácil e eficiente, tornando a prototipagem e desenvolvimento de modelos mais simples. A biblioteca também possui um modulo de diferenciação automática que permite a computação de gradientes da rede de maneira eficaz, permitindo o uso de técnicas como gradiente descendente.

3.3 FastAI

Fastai (W3TECHS, 2017) é uma desenvolvida em cima da ferramenta PyTorch. A biblioteca pode ser considerada de alto nível e foi desenvolvida ambos para iniciantes e experientes na área de inteligência artificial. Ela possui blocos para construção de diversas redes neurais, incluindo visão computacional, processamento de linguagem natural e dados tabulares. A biblioteca FastAI permite que usuários treinem modelos estado da arte em poucas linhas de código. Por exemplo, encontramos o modelo U-Net e técnicas para transformação de imagens.

Uma das principais vantagens da Fastai é a sua vasta coleção de modelos pré-treinados, eles permitem experimentação imediata, sem a necessidade de treinar o modelo do zero. Estes modelos foram treinados em grandes conjuntos de dados, como ImageNet, e já capturaram características genéricas de imagens. A técnica de transferência de aprendizado permite que essas características sejam usadas como ponto de partida, necessitando apenas o ajuste fino para tarefas específicas. Isso leva a um treinamento mais rápido e a modelos mais precisos.

A biblioteca oferece uma gama de arquiteturas populares, como ResNets, VGGs e Densenets. Dependendo das especificidades e exigências do problema, os usuários podem escolher a arquitetura que melhor se adapta às suas necessidades.

3.4 W&B

Weights & Biases (W&B) (BIEWALD, 2020) é uma plataforma popular para o rastreamento e gerenciamento de experimentos em aprendizado de máquina. Com esse pacote é possível armazenar, visualizar e gerenciar todos os experimentos, modelos e conjuntos de dados em um único local. Isso facilita a organização e a recuperação de recursos quando necessário. Além disso, o W&B suporta o versionamento de artefatos, dessa maneira é possível salvar diferentes versões de um conjunto de dados ou modelo e rastrear as mudanças ao longo do tempo. É extremamente útil para reproduzir resultados e entender a evolução dos seus experimentos.

W&B oferece ferramentas de visualização integradas que permitem visualizar métricas de treinamento e comparar experimentos. Isso facilita a análise e a tomada de decisões baseadas em dados.

A biblioteca foi projetada para se integrar facilmente com uma variedade de frameworks de aprendizado de máquina, como, PyTorch e Fastai, facilitando a inclusão do rastreamento de experimentos. Através do uso da API é muito fácil baixar e usar artefatos, seja para treinamento, inferência ou análise. Com apenas algumas linhas de código, é possível iniciar execuções, registrar métricas e baixar conjuntos de dados ou modelos. Ao armazenar grandes conjuntos de dados e modelos no W&B, não é necessário se preocupar com o espaço de armazenamento local ou com o gerenciamento de backups. Os artefatos são armazenados de forma segura na nuvem e podem ser baixados conforme necessário.

Ao combinar o rastreamento de hiper-parâmetros, métricas, código e artefatos (conjuntos de dados e modelos) em um único local, o W&B facilita a reprodução de experimentos, o que é essencial para a pesquisa científica e o desenvolvimento robusto de modelos.

Em resumo, o Weights & Biases oferece uma plataforma unificada que facilita o rastreamento, o gerenciamento e a colaboração em projetos de aprendizado de máquina. A capacidade de armazenar e recuperar facilmente artefatos programaticamente é uma grande vantagem, especialmente em ambientes colaborativos e dinâmicos.

3.5 Jupyter

O Projeto Jupyter (KLUYVER et al., 2016), anteriormente conhecido como IPython Notebook, é uma iniciativa de código aberto que fornece uma plataforma de computação interativa rica em recursos. Ele permite que os usuários criem e compartilhem documentos contendo código ativo, equações, visualizações e texto explicativo. Esses documentos, conhecidos como Jupyter Notebooks, tornaram-se padrão na comunidade de ciência de dados devido à sua versatilidade e interatividade.

A natureza interativa dos Jupyter Notebooks é ideal para a análise de dados exploratória, permitindo a execução de código em células individuais e a visualização dos resultados imediatamente. Isso facilita a iteração rápida e a experimentação.

Os notebooks suportam uma variedade de bibliotecas de visualização, como Matplotlib, Seaborn e Plotly. Essa integração permite aos cientistas de dados visualizar seus dados de forma eficaz e direta no ambiente do notebook. Como os notebooks contêm código, saída e texto explicativo, eles fornecem um registro abrangente do processo de análise. Isso é essencial para revisões, colaborações e publicações. O ambiente é altamente personalizável e extensível, com uma ampla gama de extensões disponíveis que adicionam funcionalidades adicionais, como correção automática de código, formatação e suporte a markdown.

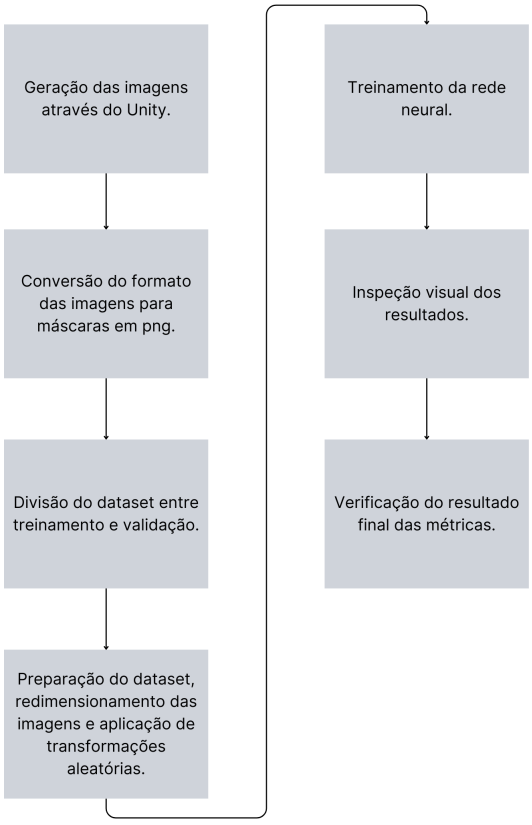
Para tarefas de aprendizado de máquina e ciência de dados, o processo geralmente envolve múltiplas fases, como limpeza de dados, análise exploratória, modelagem e validação. O projeto Jupyter fornece um ambiente unificado onde todas essas etapas podem ser realizadas de forma coesa, documentada e visual.

4 Desenvolvimento

No desenvolvimento deste trabalho optou-se pela escolha da linguagem de programação Python, devido ao seu extenso ecossistema de pacotes voltados para machine learning. A robustez e flexibilidade proporcionadas por bibliotecas como NumPy, PyTorch e scikit-learn ofereceram uma base sólida para a execução das tarefas complexas relacionadas à detecção de falhas em painéis solares. A vasta comunidade de desenvolvedores contribuíram significativamente para a implementação suave e rápida das técnicas propostas.

O projeto foi estrategicamente dividido em duas fases distintas, cada uma abordando aspectos cruciais do processo. A primeira etapa envolveu a geração de imagens por meio do software Unity, explorando a capacidade desse ambiente de simulação tridimensional para criar cenários representativos das condições reais de funcionamento dos painéis solares. A segunda fase concentrou-se no treinamento da rede neural, utilizando a biblioteca fastai. Essa escolha se mostrou particularmente vantajosa devido à sua abordagem de alto nível para deep learning, simplificando consideravelmente o processo de implementação e ajuste dos modelo. O diagrama de blocos da figura 6 mostra uma versão simplificada das etapas.

Figura 6 – Diagrama de blocos com as etapas do desenvolvimento.

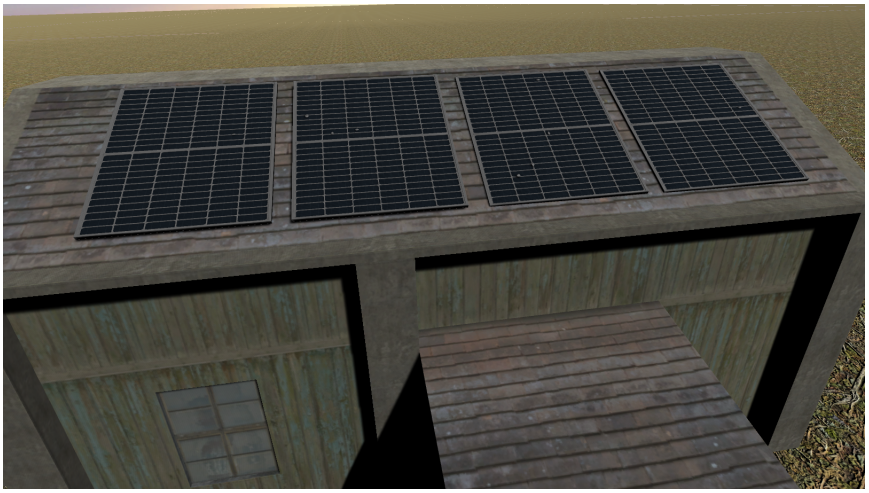


Fonte: Autoria própria (2023).

4.1 Produção de imagens em simulação

A detecção e localização de falhas por redes neurais para esse problema segue a proposta deste trabalho como apresentada em (MAZONI, 2021) sobre sistemas de simulação. Assim, usando a propriedade de generalidade que se pode obter com redes neurais com as variações de imagens apresentadas e inspirado pelo sucessos de outros trabalhos (LEVINE et al., 2018; OPENAI et al., 2018), as imagens e as falhas foram geradas automaticamente usando um sistema de simulação construído na plataforma Unity 3D. Um exemplo de imagem gerada é mostrado na figura 7.

Figura 7 – Exemplo de imagem gerada pelo programa de simulação (MAZONI, 2021).



Fonte: Autoria própria (2023).

O sistema de geração de imagens com falhas usa a seguinte sequência:

1 Gera-se um modelo de telhado de construção e com quatro painéis fotovoltaicos posicionados.

2 Gera-se uma trajetória de pontos e direções para a câmera virtual do drone que fotografa os painéis. Essas posições são ajustadas para representar diversos ângulos mas devem conter a imagem dos painéis totalmente ou na maior parte.

3 Geram-se modelos de falhas (pequenas esferas esbranquiçadas).

4 Sorteia-se um número de falhas de cada painel.

5 Para o número de falhas sorteado, geram-se valores de distorções de escala em dois eixos. Quando as falhas são afetadas desses valores ficam maiores ou menores dentro de uma faixa. Ao ficarem maiores ou menores, emula-se uma severidade maior das falhas.

6 As posições das falhas são sorteadas sobre coordenadas horizontal e vertical sobre cada painel com distribuição uniforme.

7 A câmera que visualiza o ambiente é posicionada em uma nova posição.

8 Uma imagem é capturada e salva em arquivo.

9 As coordenadas tridimensionais das falhas e dos cantos de cada painel são obtidas e salvas num arquivo. A posição da câmera, bem como sua inclinação são também registradas.

10 As coordenadas das falhas e dos cantos dos painéis são projetadas no plano de visão câmera que representa a captura da imagem e convertidas para pixels. Esses valores são também salvos em arquivo.

11 Repetem-se os passos de 4 a 10 até o fim das posições válidas para a câmera virtual.

Em uma sequência de simulação como a descrita, não há a necessidade de simulações físicas e portanto a escala de tempo pode ser desativada. Apenas se reposicionam falhas e câmera para obter mais imagens.

O código e detalhes completos de simulação estão disponíveis em um projeto hospedado publicamente e são também apresentados em (MAZONI, 2021).

Em (MAZONI, 2021), foram geradas 1000 imagens, sendo 700 usadas no treinamento da rede e 300 usadas como teste. Notificam-se resultados de aprendizado das falhas com sucesso de 82% da captura de falhas, perdendo-se as falhas de menores dimensões relativamente à imagem. Para as imagens localizadas a soma dos erros das coordenadas de falha corresponde a um equivalente de médio de 50 pixels.

4.2 Tratamento de arquivos para treinamento

As imagens geradas conforme descrito no capítulo 4 juntamente com as posições das falhas relativamente aos pixels das imagens são usadas para o treinamento. Tais arquivos são armazenados na plataforma online WandB para que sejam diretamente carregadas. Uma etapa inicial foi necessária: a conversão das coordenadas de falhas e sua severidade para um conjunto de máscaras. Ou seja, para cada imagem e a localização das falhas, produz-se uma imagem com o mesmo número de pixels porém marcados com 1, 2 ou 0 conforme representam um pixel que faz parte da área visual de uma falha, painel ou nenhum dos dois, respectivamente.

Além das falhas, para que seja possível também uma localização prática das falhas, é necessário localizar também os painéis nas imagens. Os dois tipos de detecção são realizados pela mesma rede neural, com diferentes saídas para falhas e para painéis. Para problemas bastante uniformes, a abordagem de ter a mesma rede neural detectando diversas características é vantajosa sobre diversas redes neurais dedicadas porque tais estruturas podem compartilhar conhecimento dos dois problemas em seus pesos, pois há uma correlação entre a presença de uma característica e outra, (HOWARD; GUGGER, 2020).

4.3 Critérios de aprendizagem e resultados

A estrutura utilizada para a rede neural é a U-Net, reconhecida pelo seu sucesso com a detecção de características de imagens biomédicas e extensivamente aplicada em outros problemas, (RONNEBERGER; FISCHER; BROX, 2015b; NAZEM et al., 2021; ANDERSSON; AHLSTRöM; KULLBERG, 2019).

A arquitetura da U-Net é combinada com uma rede do tipo encoder chamada Resnet34, que é frequentemente usada em combinação com a U-Net formando uma arquitetura combinada (WU; SHEN; HENGEL, 2019; LI et al., 2016; SAHA; ZHANG; SATAPATHY, 2021).

As estruturas padronizadas de redes neurais como as citadas são extensivamente usadas em competições de detecção e caracterização de conjuntos de dados, (YANG et al., 2018). São também muito usadas em aplicações industriais e de pesquisa (SAHA; ZHANG; SATAPATHY, 2021). Essas arquiteturas, apesar de muito profundas em termos de número de camadas, o que exigiria grande volume de dados para treinamento, permitem o uso do conceito de transferência de conhecimento, (TORREY; SHAVLIK, 2010), pelo qual uma rede pré-treinada com grande volume de parâmetros pode ser retreinada apenas em suas camadas finais com dados diferentes. Esses dados correspondem às imagens de aplicação do problema em questão. O raciocínio aplicado é de que as camadas mais próximas à entrada contém informação mais abstrata e menos relacionada à aplicação do resultado final, portanto possuem informações relacionadas a tarefas mais comuns, como detecção de cantos e contornos.

Como critério de otimização para a aprendizagem, foi usada a entropia cruzadas, que é mais apropriada para detecção de características em imagens, (ZHANG; SABUNCU, 2018).

4.4 Programação

4.4.1 Preparação do dataset.

Além dos já citados programas para geração de imagens em ambiente simulado, obtidas do trabalho (MAZONI, 2021), há um programa para converter os arquivos com as informações das posições de painéis e falhas para máscaras disponível em <https://github.com/lgvaz/solar-panel>.

Para o treinamento e teste de rede neural, o programa final desenvolvido para este trabalho contém as seguintes etapas:

- Coletar as imagens.
- As imagens originais são de dimensões (1614,908), foram recortadas para (908,908).
- Aplicação de transformações aleatórias que permitem gerar mais imagens e máscaras a partir das originais. Essas transformações são:

  - Rotações entre 0 e 360 graus.
  - Amplificações (zoom) com aumento de um fator até 1,1.
  - Rotação completa (180 graus) na vertical e na horizontal.

- As imagens finais são redimensionadas para as dimensões (640,640)

Com essas transformações, as imagens são usadas em um treinamento da estrutura de rede escolhida, o código está disponível em <https://github.com/lgvaz/solar-panel>.

De maneira mais detalhada, os seguintes blocos de código são fundamentais:

```
def data_from_wandb(artifact_name: str, run=None, save_dir=None):
    import wandb

    run = run or wandb.init(project="solar-panel",
      job_type="dataset-download")
    artifact = run.use_artifact(artifact_name,
      type="dataset")

    save_dir = Path(save_dir)
      if save_dir is not None else Path.home() / "data"
    save_dir = save_dir / run.project_name() /
      artifact.name.split(":")[0]
    save_dir.mkdir(exist_ok=True, parents=True)

    return Path(artifact.download(root=save_dir))
```

A função `data_from_wandb` é projetada para interagir com a plataforma Weights & Biases (W&B). Sua principal finalidade é baixar um artefato especificado, um conjunto de dados ou modelo, associado ao projeto `"solar-panel"`.

Em relação ao local de salvamento, se o usuário não especificar um diretório, a função usará um padrão: uma subpasta chamada "data"no diretório inicial do usuário. Os artefatos são então organizados nesse diretório por projeto e nome do artefato.

Após definir e preparar o local de salvamento, a função baixa o artefato para esse diretório e retorna o caminho do arquivo ou diretório baixado.

```
def mask_file_from_image_file(image_file):
    data_dir = image_file.parent.parent
    mask_name = image_file.with_suffix('.png').name
    return data_dir / 'masks' / mask_name
```

```
image_files = get_image_files(data_dir/'images')
mask_files = get_image_files(data_dir/'masks')
```

A função `mask_file_from_image_file` é projetada para trabalhar com conjuntos de dados de visão computacional, especialmente aqueles que têm imagens e suas correspondentes máscaras (geralmente usadas em tarefas de segmentação). A função recebe o caminho de um arquivo de imagem e retorna o caminho correspondente do arquivo de máscara associado a essa imagem.

A lógica da função é a seguinte:

1. A função identifica o diretório pai do arquivo da imagem, que é assumido ser o diretório principal do conjunto de dados.
2. Ela então constrói o nome do arquivo da máscara com base no nome do arquivo da imagem, mas com a extensão '.png'.
3. A função constrói o caminho completo para o arquivo da máscara, assumindo que as máscaras estão armazenadas em uma subpasta chamada 'masks' dentro do diretório principal do conjunto de dados.

As linhas adicionais de código após a função utilizam a biblioteca `fastai` para:

- Obter uma lista de todos os arquivos de imagem na subpasta 'images' do diretório de dados.
- Obter uma lista de todos os arquivos de máscara na subpasta 'masks' do diretório de dados.

Em resumo, o conjunto de códigos é uma ferramenta útil para trabalhar com conjuntos de dados de segmentação que têm imagens e suas respectivas máscaras organizadas em subpastas separadas.

```
splits = RandomSplitter(seed=42)(image_files)
splits = (splits[0], splits[1])
```

A função `RandomSplitter` da biblioteca `fastai` é usada para dividir um conjunto de dados em dois subconjuntos de forma aleatória: geralmente um conjunto de treinamento e um conjunto de validação.

1. Ele chama a função `RandomSplitter` com uma semente fixa (`seed=42`). A semente garante que a divisão aleatória seja reprodutível; toda vez que o código for executado com a mesma semente, a divisão será a mesma.
2. `RandomSplitter` é aplicado aos `image_files`, que é uma lista de caminhos de arquivos de imagem.

3. O resultado é uma tupla de dois arrays: o primeiro array contém os índices das imagens selecionadas para o conjunto de treinamento e o segundo array contém os índices para o conjunto de validação.

4. O código subsequente simplesmente reatribui a tupla `splits` de uma maneira mais direta, mas o resultado final permanece o mesmo: `splits` contém dois arrays de índices representando a divisão do conjunto de dados.

Em resumo, o código divide aleatoriamente a lista `image_files` em dois subconjuntos com base em índices, garantindo a reprodutibilidade da divisão graças à semente fixa.

```
image_pipeline = [PILImage.create]
mask_pipeline = [mask_file_from_image_file, PILMask.create]

ds = Datasets(image_files,
              [image_pipeline, mask_pipeline],
              splits=splits)
```

Esse segmento de código é responsável por configurar e criar um conjunto de dados de imagens e máscaras.

- **image_pipeline**: Esta é uma lista de funções para processar imagens. Ela contém apenas uma função, `PILImage.create`, que é responsável por criar uma imagem usando a biblioteca PIL a partir de um arquivo.

- **mask_pipeline**: Esta lista de funções processa máscaras associadas às imagens. Primeiro, `mask_file_from_image_file` determina o nome do arquivo da máscara com base no nome do arquivo da imagem. Depois, `PILMask.create` cria a máscara usando a biblioteca PIL.

- **ds**: Aqui, o conjunto de dados é criado usando a classe `Datasets` do 'fastai'. `image_files` é uma lista de caminhos de arquivos de imagem. As duas listas de funções (`image_pipeline` e `mask_pipeline`) são usadas para processar as imagens e máscaras, respectivamente. `splits` define como o conjunto de dados deve ser dividido em treinamento e validação.

Transformações after_item: Estas são as transformações que são aplicadas a cada item individualmente.

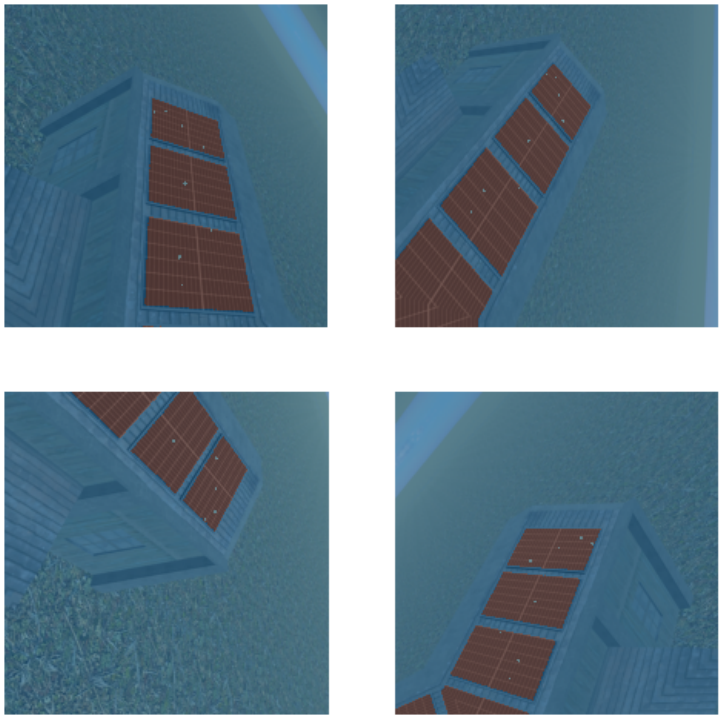
- `ToTensor()`: Converte a imagem em um tensor PyTorch.

- `Resize(908, 908)`: Redimensiona a imagem para 908x908 pixels.

Transformações after_batch: Estas são as transformações aplicadas a um lote (batch) inteiro de itens. São transformações que modificam a imagem de uma forma aleatoriamente especificada cada vez que são executadas, com o propósito de gerar imagens variadas e "aumentar" o dataset artificialmente. A Figura 8 traz exemplos dessa transformação sendo aplicada a mesma imagem múltiplas vezes.

- `IntToFloatTensor()`: Converte o tensor de inteiros para ponto flutuante e normaliza entre 0 e 1.
- `aug_transforms(...)`: Uma série de transformações de aumento de dados com as seguintes especificações:
  - `size=(640, 640)`: Redimensiona para 640x640 pixels.
  - `min_scale=0.75`: Limita o zoom out para 75% do tamanho original.
  - `flip_vert=True`: Permite inversões verticais.
  - `max_rotate=180`: Rotação aleatória até 180 graus.

Figura 8 – Transformação after batch aplicada na mesma imagem.



Fonte: Autoria própria (2023).

Criação do DataLoader:

- Tamanho do lote definido como 4.
- Aplica transformações `after_item` e `after_batch` especificadas.

4.4.2 Métrica

A classe `CategoryDice` define uma métrica Dice para uma categoria específica em uma tarefa de segmentação.

- **`__init__(self, category_id, category_name, axis=1)`**: Inicializa o objeto com o ID da categoria de interesse e seu nome. `axis` determina em qual dimensão a função argmax será aplicada para encontrar a categoria predita.
- **`reset(self)`**: Reseta as estatísticas acumuladas.
- **`_convert_to_binary(self, mask)`**: Esta é uma função auxiliar privada para converter a máscara de múltiplas categorias em uma máscara binária, onde a categoria de interesse é marcada como 1 e todas as outras como 0.
- **`accumulate(self, learn)`**: Acumula as estatísticas de interesse e união a partir das previsões e ground truth. A previsão (`pred`) é obtida pelo argmax ao longo do eixo especificado e então convertida para uma máscara binária. O cálculo do inter (intersecção) e union (união) é então realizado usando operações de tensor elementares.
- **`value`**: Calcula e retorna o coeficiente de Dice com base nas estatísticas acumuladas. Se `union` for 0 (para evitar divisão por zero), retorna `None`.
- **`name`**: Retorna o nome da métrica, que é baseado no nome da superclasse e no nome da categoria.

Em resumo, esta classe permite calcular o coeficiente de Dice para uma categoria específica em tarefas de segmentação. Isso é útil quando se tem múltiplas categorias em uma imagem e quer-se avaliar o desempenho do modelo em segmentar uma categoria específica.

- `foreground_acc`, `panel_acc` e `fault_acc`: São métricas de acurácia para diferentes classes, respectivamente fundo, painel e falha.
- `DiceMulti`: Calcula a média do coeficiente Dice de todas categorias.
- `CategoryDice(1, 'panel')` e `CategoryDice(2, 'fault')`: Estes são coeficientes Dice específicos para as classes painel e falha, respectivamente. A classe `CategoryDice` foi definida anteriormente e calcula o coeficiente Dice para uma categoria específica.

4.4.3 Definição da Função de Perda

- `CrossEntropyLossFlat(axis=1, weight=tensor([1.0, 1.0, 10.0]).cuda())`: Está definindo uma função de perda de entropia cruzada. O argumento `axis=1` indica o eixo ao

longo do qual a entropia cruzada é calculada. O argumento `weight` está dando diferentes pesos para as classes. A classe falha tem um peso de 10.0, o que indica que os erros nesta classe são considerados 10 vezes mais "graves"do que erros nas outras classes.

4.4.4 Inicialização do Modelo

- `unet_learner(dls, resnet34, n_out=3, metrics=metrics, cbs=cbs, loss_func=loss_`
  Está inicializando um aprendiz (learner) usando a arquitetura U-Net com uma backbone ResNet34. `dls` é o conjunto de dados carregado, `n_out=3` indica que há três classes de saída, e `cbs` são os callbacks.

4.4.5 Treinamento do Modelo

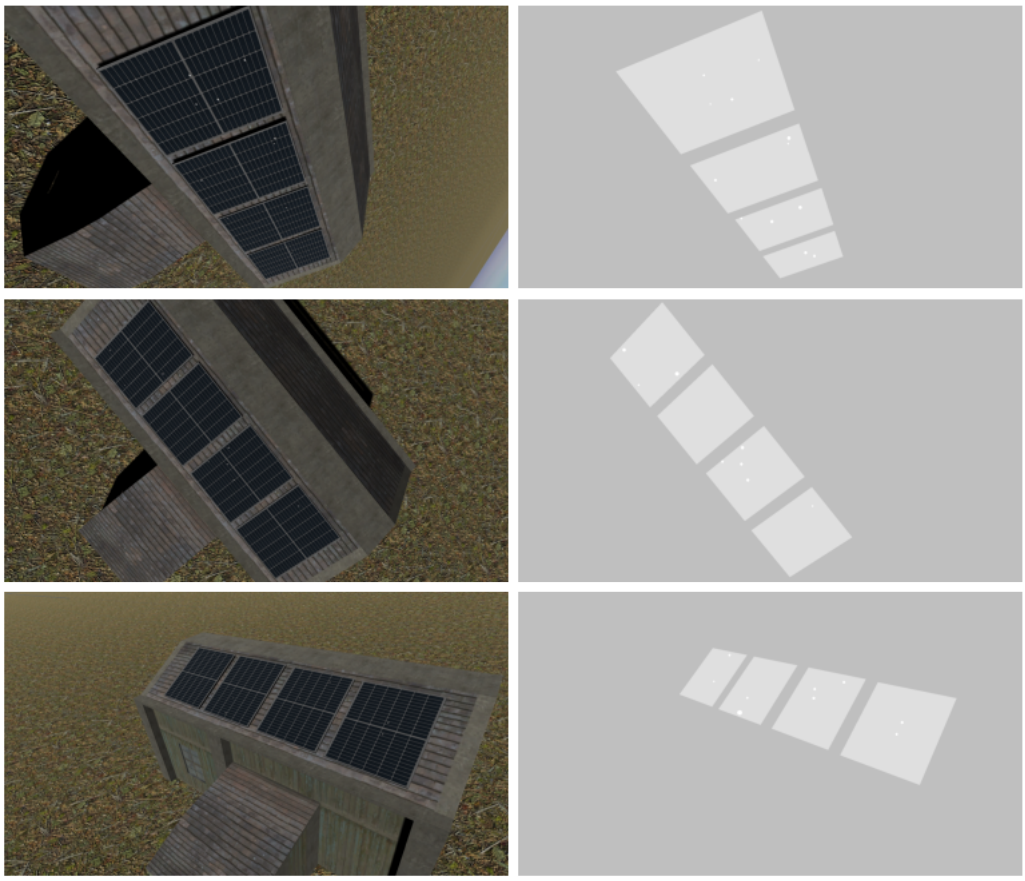
- `learn.fine_tune(10, 1e-3)`: Esta linha está treinando (ou afinando) o modelo. O modelo será treinado por 10 épocas com uma taxa de aprendizado que segue um cronograma cossenoidal com o valor máximo de 0.001 (1e-3).

Em resumo, este código está configurando e treinando um modelo U-Net para segmentação com um foco particular na classe falha, que é ponderada mais fortemente na função de perda.

5 Resultados e Discussões

A abordagem de usar redes neurais já conhecidas na literatura para se adaptar ao conjunto de imagens fornecidas foi bem sucedida. Como se pode ver na figura 9, as falhas foram localizadas com precisão, no lado esquerdo temos a imagem que é utilizada na entrada da rede. No lado direito, temos a saída da rede, que é uma imagem com valores [0, 1, 2] com a mesma dimensão da imagem de entrada. Os pixels mais escuros correspondem ao fundo, os mais claros a falha, e os intermediários ao painel. A precisão de localização de falhas (nos pixels de imagem) foi de 87% e a precisão da localização dos painéis de 98%.

Figura 9 – Imagem entrada do modelo (esquerda) e saída do modelo (direita).



Fonte: Autoria própria (2023).

5.1 Acurácia

A acurácia mede a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. A tabela 1 apresenta os resultados do treinamento.

1. **Acurácia total**: Representa a acurácia geral do modelo. A acurácia total está consistentemente alta, flutuando em torno de 98%, o que indica que o modelo tem um desempenho robusto.

2. **Acurácia Painel**: Representa a acurácia na identificação da categoria "Painel". Estes valores também são consistentemente altos, muito próximos à acurácia total, demonstrando um desempenho forte na identificação de painéis.

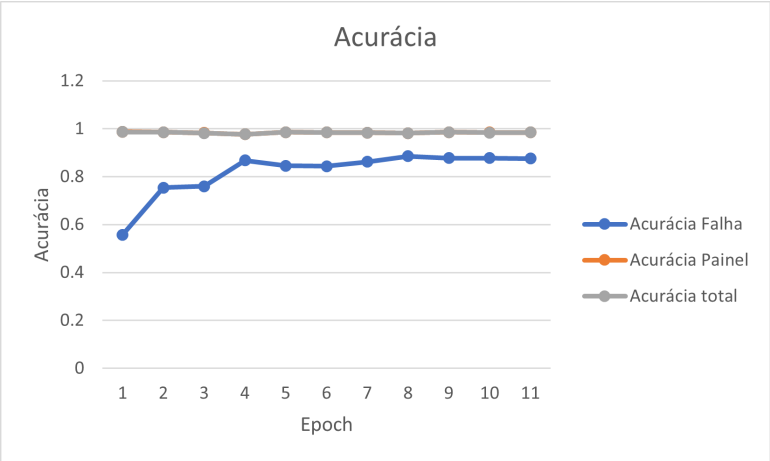
3. **Acurácia Falha**: Representa a acurácia na identificação da categoria "Falha". Esta métrica apresenta uma variação maior em comparação com as outras duas, com valores variando entre 55% e 87%. Isto sugere que a identificação de falhas é uma tarefa mais desafiadora para o modelo em comparação com a identificação de painéis.

Tabela 1 – Métricas de Acurácia

Epoch	Acurácia total	Acurácia Painel	Acurácia Falha
1	0.986384	0.987407	0.557684
2	0.985112	0.985654	0.753491
3	0.982172	0.98269	0.760136
4	0.976911	0.977157	0.86765
5	0.985761	0.986081	0.845751
6	0.98424	0.984562	0.843428
7	0.983501	0.983778	0.861862
8	0.981832	0.982047	0.885637
9	0.985098	0.98534	0.877751
10	0.984017	0.984258	0.877392
11	0.984377	0.984622	0.875599

Fonte: Autoria própria (2023)

Figura 10 – Grafico de precisão ao treinar a rede neural. Sendo o eixo X o numero de vezes que o dataset foi visto por completo, e Y a precisão.



Fonte: Autoria própria (2023)

5.2 Dice

O coeficiente Dice é uma métrica comum para avaliar o desempenho de modelos de segmentação. Valores mais próximos de 1 indicam melhor concordância entre as previsões do modelo e as verdadeiras anotações. A tabela 2 mostra os resultados do treinamento.

1. **Dice Multi**: Representa o coeficiente Dice médio para todas as categorias. Os valores flutuam em torno de 84-85%, indicando uma boa concordância geral entre as previsões do modelo e as anotações verdadeiras.

2. **Dice Painel**: Representa o coeficiente Dice para a categoria "Painel". Estes valores são consistentemente altos, quase chegando a 99% em alguns casos, o que demonstra um excelente desempenho do modelo na segmentação de painéis.

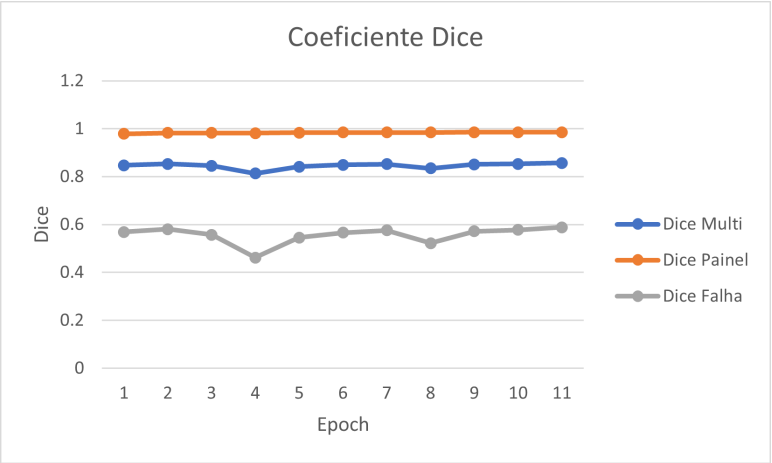
3. **Dice Falha**: Representa o coeficiente Dice para a categoria "Falha". Semelhante à "Acurácia Falha", esta métrica apresenta maior variabilidade, com valores variando entre 46% e 58%. Isso reitera que a segmentação de falhas é uma tarefa mais desafiadora para o modelo.

Tabela 2 – Dice

Epoch	Dice Multi	Dice Painel	Dice Falha
1	0.847154	0.979146	0.568611
2	0.852679	0.982564	0.580562
3	0.845045	0.98249	0.557699
4	0.812908	0.981984	0.46162
5	0.841746	0.983801	0.546039
6	0.848749	0.984467	0.566204
7	0.852121	0.984718	0.575989
8	0.834205	0.984504	0.522393
9	0.851067	0.985226	0.572156
10	0.853055	0.985389	0.577911
11	0.856616	0.985642	0.588278

Fonte: Autoria própria (2023)

Figura 11 – Gráfico do coeficiente Dice ao treinar a rede neural. Sendo o eixo X o numero de vezes que o dataset foi visto por completo, e Y o coeficiente.



Fonte: Autoria própria (2023)

5.3 Loss

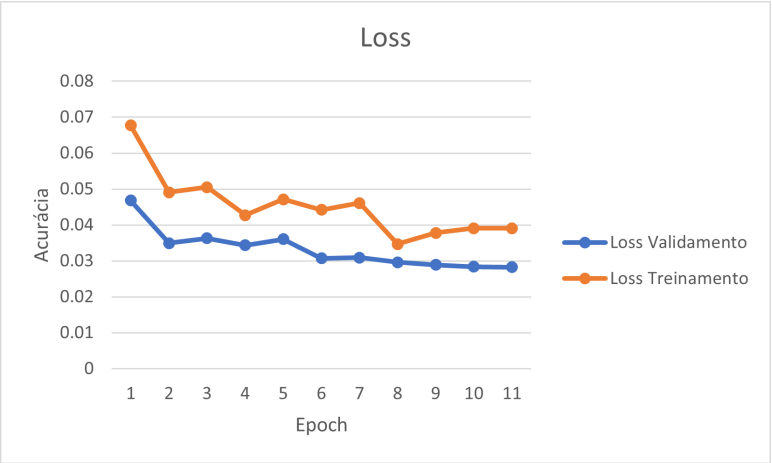
O "Loss Treinamento" e o "Loss Validamento" representam a medida de erro durante o treinamento e a validação, respectivamente. Valores mais baixos indicam um melhor desempenho do modelo. A tabela mostra que o loss de treinamento e validação tendem a diminuir ao longo do tempo, indicando uma melhoria no desempenho do modelo à medida que o treinamento avança. No entanto, a proximidade entre os valores de treinamento e validação sugere que o modelo não está superajustando significativamente aos dados de treinamento, o que é um bom sinal. A tabela 3 mostra os resultados do treinamento.

Tabela 3 – Loss

Epoch	**Loss Treinamento**	**Loss Validação**
1	0.067715	0.046786
2	0.049102	0.034924
3	0.050484	0.036289
4	0.042732	0.034339
5	0.047109	0.03601
6	0.044193	0.03069
7	0.04606	0.030891
8	0.034702	0.029592
9	0.037754	0.028892
10	0.039087	0.028352
11	0.039049	0.028272

Fonte: Autoria própria (2023)

Figura 12 – Grafico do erro (loss) ao treinar a rede neural. Sendo o eixo X o numero de vezes que o dataset foi visto por completo, e Y o erro.



Fonte: Autoria própria (2023)

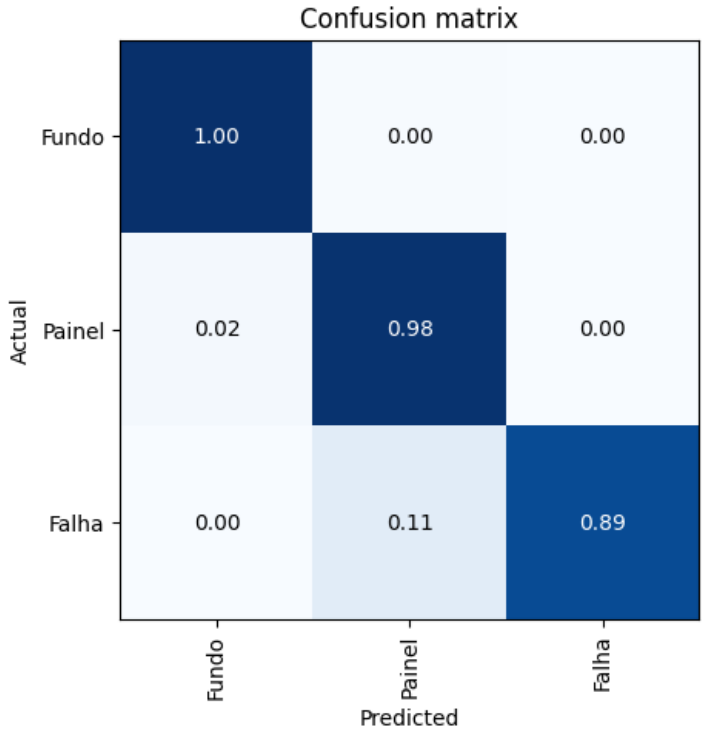
5.4 Matriz de Confusão

Matrizes de Confusão representam instrumentos analíticos fundamentais em avaliações de desempenho de modelos de classificação. Sua estrutura quadrada, com dimensões equivalentes ao número de classes presentes no problema, encapsula a relação entre as previsões do modelo e as verdadeiras classes dos dados. Os quatro elementos primordiais são definidos como segue: Verdadeiros Positivos (VP), indicando acurácia nas previsões positivas; Falsos Positivos (FP), evidenciando previsões incorretas de instâncias negativas; Verdadeiros Negativos (VN), refletindo acurácia nas previsões negativas; e Falsos Negativos (FN), denotando previsões equivocadas de instâncias positivas.

Dada a matriz de confusão fornecida pela figura 13, podemos notar que:

- **Primeira linha (fundo verdadeiro):**
  - Quase 100 (99.9%) das instâncias foram classificadas corretamente como fundo.
- **Segunda linha (painel verdadeiro):**
  - 2% das instâncias do painel foram incorretamente classificadas como fundo.
  - 98% das instâncias foram corretamente classificadas como painel.
- **Terceira linha (falha verdadeira):**
  - 11% das instâncias de falha foram incorretamente classificadas como painel.
  - 89% instâncias foram corretamente classificadas como falha.

Figura 13 – Matriz de confusão, normalizada por linha.



Fonte: Autoria própria (2023)

Métricas de Desempenho

$$\text{Precisão (fundo)} = \frac{76055}{76055 + 371} = 0.9951$$
$$\text{Recall (fundo)} = \frac{76055}{76055 + 318 + 3} = 0.9958$$
$$\text{F1-Score (fundo)} = 2 \times \frac{\text{Precisão (fundo)} \times \text{Recall (fundo)}}{\text{Precisão (fundo)} + \text{Recall (fundo)}} = 0.9955$$

$$\text{Precisão (painel)} = \frac{23120}{23120 + 318 + 6} = 0.9830$$
$$\text{Recall (painel)} = \frac{23120}{23120 + 371 + 79} = 0.9805$$
$$\text{F1-Score (painel)} = 2 \times \frac{\text{Precisão (painel)} \times \text{Recall (painel)}}{\text{Precisão (painel)} + \text{Recall (painel)}} = 0.9817$$

$$\text{Precisão (falha)} = \frac{1252}{1252 + 178} = 0.8755$$
$$\text{Recall (falha)} = \frac{1252}{1252 + 162} = 0.8885$$
$$\text{F1-Score (falha)} = 2 \times \frac{\text{Precisão (falha)} \times \text{Recall (falha)}}{\text{Precisão (falha)} + \text{Recall (falha)}} = 0.8820$$

6 Conclusão

Este estudo revelou que a abordagem adotada, baseada em redes neurais de aprendizado profundo, superou significativamente as limitações associadas às técnicas tradicionais de processamento de sinais para a detecção de falhas em sistemas de geração de energia solar. A utilização de uma metodologia que integra a geração automática de imagens por meio de um sistema de simulação tridimensional mostrou-se crucial para contornar desafios como a escassez de imagens marcadas e a padronização dos problemas. Os resultados obtidos demonstram a eficácia dessa abordagem, com uma impressionante taxa de precisão de cerca de 98% na detecção dos painéis solares e 87% na identificação de falhas específicas. Este estudo não apenas contribui para o avanço da detecção de falhas em sistemas fotovoltaicos, mas também destaca a relevância e o potencial das técnicas baseadas em aprendizado profundo para abordar desafios complexos e variáveis em contextos práticos, como a geração de energia solar em residências.

Para melhorias futuras, uma vez que estão definidas as coordenadas de tela das falhas e dos painéis, pode-se combinar a informação interna dos drones que obtém tais fotos para determinar a localização das falhas em relação aos seus painéis. Além disso, através de um cruzamento com um banco de dados de localização por GPS das placas, podem-se emitir alarmes de substituição ou de verificação de acordo com as inspeções visuais. Essas melhorais apontam para aproximar o estudo aqui realizado da aplicação.

É importante também que o sistema seja testado em aplicações práticas e que sejam absorvidas as variações entre as imagens simuladas e as reais. Ainda antes de testes com imagens reais também pode-se adaptar o sistema de simulação para que produza imagens com maior proximidade das imagens obtidas na realidade e também com maior variabilidade para garantir robustez dos resultados da rede conforme o treinamento.

Referências

ABADI, M. et al. Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. [S.l.: s.n.], 2016. p.265–283.

ALSABHAN, W.; ALOTAIBY, T. et al. Automatic building extraction on satellite images using unet and resnet50. *Computational Intelligence and Neuroscience*, Hindawi, v. 2022, 2022.

ALSAFASFEH, M. et al. Unsupervised fault detection and analysis for large photovoltaic systems using drones and machine vision. *Energies*, v. 11, n. 9, 2018.

ANDERSSON, J.; AHLSTRöM, H.; KULLBERG, J. Separation of water and fat signal in whole-body gradient echo scans using convolutional neural networks. *Magnetic Resonance in Medicine*, Wiley, v. 82, n. 3, p. 1177–1186, abr. 2019.

ARENT, D. J.; WISE, A.; GELMAN, R. The status and prospects of renewable energy for combating global warming. *Energy Economics*, Elsevier, v. 33, n. 4, p. 584–593, 2011.

BIEWALD, L. *Experiment Tracking with Weights and Biases*. [S. l.]: *Weights and Biases* 2020. Disponível em: https://www.wandb.com/. Acesso em: 18 set. 2023.

CARNEVALE, E.; LOMBARDI, L.; ZANCHI, L. Life cycle assessment of solar energy systems: Comparison of photovoltaic and water thermal heater at domestic scale. *Energy*, v. 77, p. 434 – 446, 2014. ISSN 0360-5442. Disponível em: http://www.sciencedirect.com/science/article/pii/S0360544214010846. Acesso em: 22 ago. 2023.

CUCCHIELLA, F.; D'ADAMO, I.; GASTALDI, M. Photovoltaic energy systems with battery storage for residential areas: an economic analysis. *Journal of Cleaner Production*, v. 131, p. 460 – 474, 2016. ISSN 0959-6526. Disponível em: http://www.sciencedirect.com/science/article/pii/S0959652616304462. Acesso em 19: set. 2023.

DAPONTE, P. et al. Metrology for drone and drone for metrology: Measurement systems on small civilian drones. In: *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*. [S.l.: s.n.], 2015. p. 306–311.

FAUZI, M. A. et al. Residential rooftop solar panel adoption behavior: Bibliometric analysis of the past and future trends. *Renewable Energy Focus*, Elsevier, 2023.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org. Acesso em: 30 ago. 2023.

GORDON-RODRIGUEZ, E. et al. Uses and abuses of the cross-entropy loss: Case studies in modern deep learning. PMLR, 2020.

HENRY, C. et al. Automatic detection system of deteriorated pv modules using drone with thermal camera. *Applied Sciences*, v. 20, n. 11, 2020.

HONG, Y.-Y.; PULA, R. A. Methods of photovoltaic fault detection and classification: A review. *Energy Reports*, v. 8, p. 5898–5929, 2022. ISSN 2352-4847. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352484722008022. Acesso em: 22 ago. 2023.

HOWARD, J.; GUGGER, S. *Deep Learning for Coders with fastai and PyTorch*. [S.l.]: O'Reilly Media, 2020.

IBM. *Convolutional neural networks*. [S. l.]: IBM 2023. https://www.ibm.com/topics/convolutional-neural-networks. Acesso em: 12 out. 2023.

KLUYVER, T. et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. [S.l.], 2016. p. 87 – 90.

LEVINE, S. et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research*, v. 37, p. 421–436, 2018. ISSN 17413176.

LI, S. et al. *Demystifying ResNet*. arXiv, 2016. Disponível em: https://arxiv.org/abs/1611.01186. Acesso em: 17 ago. 2023.

LI, Z. et al. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. 2020.

MAZONI, A. F. *Utilização das ferramentas da Inteligência Artificial em aplicações mecatrônicas – Estudo de casos*. 2021.

NAZEM, F. et al. 3d u-net: A voxel-based method in binding site prediction of protein structure. *Journal of Bioinformatics and Computational Biology*, World Scientific Pub Co Pte Lt, v. 19, n. 02, p. 2150006, abr. 2021. Disponível em: https://doi.org/10.1142/s0219720021500062. Acesso em: 29 set. 2023.

OPENAI et al. Learning dexterous in-hand manipulation. 8 2018. Disponível em: http://arxiv.org/abs/1808.00177. Acesso em: 2 out. 2023.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: WALLACH, H. et al. (Ed.). *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019. p. 8024–8035.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011.

RAUGEI, M.; FRANKL, P. Life cycle impacts and costs of photovoltaic systems: Current state of the art and future outlooks. *Energy*, v. 34, n. 3, p. 392 – 399, 2009. ISSN 0360-5442. WESC 2006 Advances in Energy Studies.

REIS, L. O. M. dos. Lógica fuzzy aplicada ao controle de um sistema híbrido de geração de energia elétrica: Eólica, fotovoltaica e biogás. *Doutorado em Engenharia Mecânica [Guaratinguetá] - Universidade Estadual Paulista Júlio de Mesquita Filho*, 2002.

REIS, L. O. M. dos; BOTURA JUNIOR, G.; SILVEIRA, J. L. Controle fuzzy aplicado a um sistemahíbrido de fontes renováveis de energia: Eólica, fotovoltaica e biogás. *Anais do Seminário Nacional de Produção e Transmissão de Energia Elétrica - XVII SNPTEE*, Uberlândia, 2003.

RONNEBERGER, O.; FISCHER, P.; BROX, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

RONNEBERGER, O.; FISCHER, P.; BROX, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv, 2015. Disponível em: https://arxiv.org/abs/1505.04597. Acesso em: 13 set. 2023.

ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

SAHA, A.; ZHANG, Y.-D.; SATAPATHY, S. C. Brain tumour segmentation with a muti-pathway ResNet based UNet. *Journal of Grid Computing*, Springer Science and Business Media LLC, v. 19, n. 4, out. 2021. Disponível em: https://doi.org/10.1007/s10723-021-09590-y. Acesso em: 17 set. 2023.

SORBELLI, F. B. et al. On the accuracy of localizing terrestrial objects using drones. In: *2018 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2018. p. 1–7.

TERVEN, J. et al. Loss functions and metrics in deep learning. a review. *arXiv preprint arXiv:2307.02694*, 2023.

TORREY, L.; SHAVLIK, J. Transfer learning. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. [S.l.]: IGI global, 2010. p. 242–264.

TRAN, L.-A.; LE, M.-H. Robust u-net-based road lane markings detection for autonomous driving. In: IEEE. *2019 International Conference on System Science and Engineering (ICSSE)*. [S.l.], 2019. p. 62–66.

W3TECHS. *Usage Statistics of Content Languages for Websites*. [S.l.] W3TECHS, 2017. Disponível em: http://w3techs.com/technologies/overview/content_language/all. Acesso em: 16 set. 2023.

WU, Z.; SHEN, C.; HENGEL, A. van den. Wider or deeper: Revisiting the ResNet model for visual recognition. *Pattern Recognition*, Elsevier BV, v. 90, p. 119–133, jun. 2019. Disponível em: https://doi.org/10.1016/j.patcog.2019.01.006. Acesso em: 16 set. 2023.

YAHYATABAR, M.; JOUVET, P.; CHERIET, F. Dense-unet: a light model for lung fields segmentation in chest x-ray images. In: IEEE. *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. [S.l.], 2020. p. 1242–1245.

YANG, X. et al. Deep learning for practical image recognition: Case study on kaggle competitions. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. [S.l.: s.n.], 2018. p. 923–931.

ZAHEDI, A. Solar photovoltaic (pv) energy; latest developments in the building integrated and hybrid pv systems. *Renewable Energy*, v. 31, n. 5, p. 711 – 718, 2006. ISSN 0960-1481. SOUTH/SOUTH. Disponível em: http://www.sciencedirect.com/science/article/pii/S0960148105001990. Acesso em: 19 set. 2023.

ZHANG, Z.; SABUNCU, M. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, v. 31, 2018.