

João Pedro Augusto Costa

**Redes Neurais Convolucionais  
Aplicadas a Detecção de Defeitos  
em Redes de Energia  
ao Longo de Linhas Ferroviárias**

São Luis - Maranhão

2020

João Pedro Augusto Costa

**Redes Neurais Convolucionais  
Aplicadas a Detecção de Defeitos  
em Redes de Energia  
ao Longo de Linhas Ferroviárias**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como parte das para obtenção do título de Mestre em Engenharia de Computação e Sistemas.

Universidade Estadual do Maranhão – UEMA

Engenharia de Computação e Sistemas

Programa de Pós-Graduação

Orientador: Prof. Dr. Omar Andres Carmona Cortes

São Luis - Maranhão

2020

Costa, João Pedro Augusto.

Redes neurais convolucionais aplicadas à detecção de defeitos em redes de energia ao longo de linhas ferroviárias / João Pedro Augusto Costa. – São Luís, 2020.

57 p.

Dissertação (Mestrado) – Curso de Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2020.

Orientador: Prof. Omar Andres Carmona Cortes.

1.Redes neurais convolucionais. 2.Aprendizado profundo. 3.Redes elétricas. I.Título

CDU: 004.8.032.26:621.311.1

João Pedro Augusto Costa

**Redes Neurais Convolucionais  
Aplicadas a Detecção de Defeitos  
em Redes de Energia  
ao Longo de Linhas Ferroviárias**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como parte das para obtenção do título de Mestre em Engenharia de Computação e Sistemas.

Trabalho aprovado. São Luis - Maranhão, 31 de julho de 2020:



---

**Dr. Omar Andres Carmona Cortes**

Instituto Federal de Educação, Ciência e  
Tecnologia do Maranhão  
Orientador



---

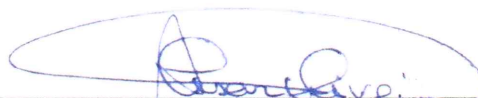
**Me. Antonio Fernando Lavareda  
Jacob Junior**

Universidade Estadual do Maranhão  
Convidado 1



---

**Dr. Ginalber Luiz de Oliveira Serra**  
Instituto Federal de Educação, Ciência e  
Tecnologia do Maranhão  
Convidado 2



---

**Dr. Alexandre César Muniz de  
Oliveira**

Universidade Federal do Maranhão  
Convidado 3



*Este trabalho é dedicado aos meus pais, Pedro Augusto Sobrinho e Luisa Pereira Costa, que sempre deram para meu irmão e eu tudo aquilo que o dinheiro não pode comprar.*

# Agradecimentos

Aos meus pais e meu irmão, por todos os sacrifícios e por me mostrarem o verdadeiro significado do que é ser uma família.

A minhas tias e tios, em especial a Irene Pereira Costa (*in memoriam*) e Joana Pereira Costa, por tudo.

Aos meus amigos do IFMA, da UEMA, da VALE S.A e da vida. Vocês tornam a rotina muito mais agradável.

Ao meu orientador, pela paciência e por sempre me incentivar a dar passos maiores.

*“Faça as coisas o mais simples  
que você puder, porém não se  
restringa às mais simples..”  
(Albert Einstein)*

# Resumo

Este trabalho apresenta a utilização de uma Rede Neural Convolutacional (CNN) chamada YOLO com objetivo de detectar falhas em componentes de redes de distribuição de energia ao longo de uma ferrovia. A ideia principal é acelerar o processo de inspeção que atualmente é realizado de forma manual. Os cenários de falha foram simulados em um laboratório que contém todas as estruturas que podem ser encontradas em redes de distribuição reais ao longo de ferrovias. Adicionalmente, a CNN pode detectar três tipos de falha: cabo fora do isolador, cabo fora do espaçador, e isolador sem anel de amarração. O laboratório possibilitou a obtenção do *dataset* utilizado para treinar e testar a CNN. Nesse contexto, se obteve um *dataset* composto de 708 imagens e respectivas anotações que foram utilizadas no treinamento da rede neural. Considerando que CNNs são redes neurais convolucionais, a etapa de treinamento requer computações envolvendo um conjunto grande de parâmetros, demandando o uso de computação paralela. Uma GPU (General Purpose Graphical Processing Unit) foi utilizada para executar as etapas de treino e teste em quantidades de tempo factíveis. Os resultados mostram que a arquitetura de CNN alcançou uma precisão de 98% e um MAP de 96.58%.

**Palavras-chave:** redes neurais convolucionais. aprendizado profundo. redes elétricas.

# Abstract

This work presents a Convolutional Neural Network (CNN) called YOLO for detecting failures in components of power lines along a railway. The main idea is to speed up the inspection process that currently is made manually. The failure scenarios have been simulated in a laboratory containing all the structures that can be found in real-world power lines along railways. Moreover, we can detect three kinds of fails: cable out of isolator, cable out of spacer, and isolator with no ring. The laboratory gave us the possibility of obtaining the dataset for training and test of the CNN. In this context, we got a dataset composed of 708 images with annotations that have been used for training the neural network. As CNNs are deep neural networks, training them requires dealing with a massive number of parameters, demanding the use of parallel computing. Thus, we used a GPGPU (General Purpose Graphical Processing Unit) to make the training and testing steps in a factible amount of time. Results have shown that the final CNN could reach a precision of 98% and a MAP of 96.58%.

**Keywords:** convolutional neural networks. railway. deep learning. eletric networks.

# Lista de ilustrações

Figura 1 – Ferrovia . . . . .	19
Figura 2 – Esquema Sistema Elétrico . . . . .	21
Figura 3 – Eventos de Energia 2019 . . . . .	22
Figura 4 – Planta Baixa Laboratório . . . . .	23
Figura 5 – Estrutura PTA1 . . . . .	24
Figura 6 – Estrutura N1TR . . . . .	24
Figura 7 – Estrutura NHFHA . . . . .	24
Figura 8 – Estrutura N4 . . . . .	24
Figura 9 – Estrutura N3-TSB . . . . .	24
Figura 10 – Estrutura ADA . . . . .	24
Figura 11 – Estrutura SAT . . . . .	24
Figura 12 – Estrutura SDAI-FU . . . . .	24
Figura 13 – Estrutura ITA-PFL . . . . .	25
Figura 14 – Estrutura FLAD . . . . .	25
Figura 15 – Estrutura ITA-FL . . . . .	25
Figura 16 – Estrutura CE3 . . . . .	25
Figura 17 – Estrutura CE4-FAH . . . . .	25
Figura 18 – Estrutura CE1-A . . . . .	25
Figura 19 – Estrutura CE2 . . . . .	25
Figura 20 – Estrutura CE4 . . . . .	25
Figura 21 – Estrutura CETR . . . . .	26
Figura 22 – Estrutura N3-PR . . . . .	26
Figura 23 – Estrutura N2-TR-FU . . . . .	26
Figura 24 – Neurônio Artificial . . . . .	28
Figura 25 – Rede Perceptron . . . . .	29
Figura 26 – Rede Perceptron Multicamadas . . . . .	30
Figura 27 – Convolutional Neural Network . . . . .	31
Figura 28 – Operação de Convolução . . . . .	32
Figura 29 – Operação de Max <i>Pooling</i> . . . . .	34
Figura 30 – Rede <i>Multilayer Perceptron</i> . . . . .	36
Figura 31 – Sistema de Detecção YOLO . . . . .	38
Figura 32 – Arquitetura YOLO . . . . .	40
Figura 33 – Fluxo de pré-processamento . . . . .	43
Figura 34 – Cabo fora do Isolador . . . . .	44
Figura 35 – Cabo fora do Espaçador . . . . .	45
Figura 36 – Isolador sem Anel de Amarração . . . . .	46

Figura 37 – Iterações VS Erro . . . . .	47
Figura 38 – Detecção Cabo Fora do Isolador . . . . .	49
Figura 39 – Detecção Cabo Fora do Espaçador . . . . .	50
Figura 40 – Detecção Isolador sem Anel de Amarração . . . . .	51

# Lista de tabelas

Tabela 1 – Composição das Estruturas . . . . .	26
Tabela 2 – Parâmetros YOLO . . . . .	45
Tabela 3 – Resultados YOLO . . . . .	48



# Lista de abreviaturas e siglas

AP	<i>Average Precision</i>
CNN	<i>Convolutional Neural Network</i>
GPU	<i>Graphic Processing Unit</i>
IoU	<i>Intersection over Union</i>
mAP	<i>Mean Average Precision</i>
MLP	<i>Multilayer Perceptron</i>

# Lista de símbolos

$\Lambda$	Lambda
$\Gamma$	Letra grega Gama
$\zeta$	Letra grega minúscula zeta
$\in$	Símbolo matemático Pertence

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
1.1	Trabalhos Correlatos	16
1.2	Justificativa	17
1.3	Objetivos	17
1.3.1	Objetivo Geral	17
1.3.2	Objetivos Específicos	17
1.4	Estrutura	18
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>19</b>
2.1	Redes de Distribuição de Energia	20
2.2	Laboratório de Energia	22
2.3	Considerações Finais	27
<b>3</b>	<b>REDES NEURAIIS CONVOLUCIONAIS</b>	<b>28</b>
3.1	Redes Neurais	28
3.2	Redes Neurais Convolucionais	29
3.3	Camada de Convolução	31
3.4	Camada de <i>Pooling</i>	34
3.5	Camada Totalmente Conectada	35
3.6	Arquitetura de <i>CNN</i> Utilizada	37
3.7	<b>YOLO V3</b>	<b>37</b>
3.7.1	Detecção	38
3.7.2	Arquitetura da YOLO	39
3.8	Considerações Finais	41
<b>4</b>	<b>EXPERIMENTOS COMPUTACIONAIS</b>	<b>42</b>
4.1	Fluxo de Trabalho	42
4.2	Experimentos	43
<b>5</b>	<b>CONCLUSÃO</b>	<b>52</b>
	<b>REFERÊNCIAS</b>	<b>54</b>

# 1 Introdução

Ferrovias são uma excelente alternativa para o transporte de diversos tipos de materiais, tais como combustíveis, grãos e minerais, dentre eles minério de ferro e el (PINHEIRO; MIRANDA; OLIVEIRA, ). Em uma ferrovia que tem como objetivo logística, os materiais transportados geralmente são levados de um ponto inicial a um ponto final, que no caso do transporte de minerais, se trata do traslado **Mina - Porto**.

Uma ferrovia é composta de uma série de componentes elétricos e eletrônicos distribuídos por toda a sua extensão, componentes estes que são responsáveis pela sinalização da via e geram uma demanda de alimentação. Esta demanda acarreta necessidade de distribuição de redes de energia ao longo da ferrovia.

Considerando que as redes de energia acompanham a ferrovia através de toda a sua extensão, é fácil inferir que está sujeita variados tipos de condições de falha, tais quais a linha férrea. Tal afirmação é verdadeira, sendo ambas as estruturas suscetíveis a falhas de seus componentes internos, além de vandalismo e condições climáticas.

Os equipamentos pertencentes a linha férrea possuem além de manutenção corretiva, um plano de manutenção preventivo, em que cada equipamento recebe manutenção cíclica dentro de um determinado período de tempo, ao contrário dos componentes das redes de energia, em que o processo preventivo é realizado através de inspeção visual, sendo este procedimento sujeito a falhas humanas, de equipamento e com uma taxa de precisão baixa, dada a extensão da rede e o caráter tridimensional dos elementos de uma rede de energia.

Considerando as dificuldades oferecidas pela inspeção visual em redes de energia, se fazem necessários meios para automatizar essa inspeção, que possam identificar defeitos utilizando padrões conhecidos nos variados tipos de rede elétrica ao longo de uma ferrovia. Para a tarefa de classificação de imagens, as redes neurais convolucionais (*Convolutional Neural Networks*) ou simplesmente *CNNs* vem obtendo resultados expressivos, como em Krizhevsky, Sutskever e Hinton (2012a) e Ciresan, Meier e Schmidhuber (2012).

Redes Neurais Convolucionais podem ser utilizadas como algoritmos de classificação supervisionada (NASRABADI, 2007), o que de forma resumida representa um aproximador que permita a descoberta de uma função  $f(\vec{x}) = y$ , na qual  $y$  faz parte de um conjunto fechado, tal qual { DEFEITO, NORMAL } e  $\vec{x}$  seja um conjunto de características dos dados utilizados para realizar a predição, tais quais as posições dos componentes de uma rede de energia em uma imagem.

Algoritmos de classificação supervisionada utilizam exemplos para realizarem inferências sobre os dados, ou seja, um classificador necessita de amostras do que se deseja

classificar de modo a construir um bom modelo. Considerando o cenário de uma rede elétrica, simular os possíveis defeitos na estrutura de seus componentes se torna inviável, partindo do ponto que se trata de uma estrutura vital, que não pode ser paralisada, portanto a dependência de defeitos em campo inviabilizaria a construção de uma base de anomalias de curto a médio espaço de tempo. Nesse ponto se destaca um ambiente primordial para a construção desse trabalho, o **Laboratório de Energia**, um ambiente localizado na VALE S.A, que possui as estruturas das redes de energia utilizadas neste trabalho e permitem a simulação dos defeitos em todos os componentes de uma rede de energia.

Nesse contexto, o trabalho utiliza redes neurais convolucionais para identificar defeitos em redes de energia. Foi utilizado um laboratório de energia como ambiente de simulação, que permitiu a obtenção de imagens de variados defeitos, o que obriga a arquitetura da rede seja mais generalista, possibilitando que o modelo seja utilizado como ferramenta de inspeção em redes ao longo da estrada de ferro.

## 1.1 Trabalhos Correlatos

O uso de redes neurais convolucionais em problemas de classificação de imagens não é uma técnica recente, trazendo trabalhos conhecidos desde a década de 90, como [LeCun et al. \(1998\)](#), que utiliza uma rede neural convolucional para classificar caracteres. Na época, a dimensão das imagens utilizadas era limitada pelo *hardware* existente, cujo desenvolvimento recente, principalmente das *GPUs*, permitiu a criação de arquitetura mais elaboradas, como em [Krizhevsky, Sutskever e Hinton \(2012b\)](#) e [Simonyan e Zisserman \(2014\)](#).

No que tange aos trabalhos que envolvem aprendizado de máquina e redes de distribuição de energia, destacam-se aqueles que utilizam tais técnicas em cima de dados históricos para prevenção de falhas, como em [Rudin et al. \(2012\)](#), extraem conhecimento automaticamente para otimizar as operações da rede, tal qual [Mocanu, Nguyen e Gibescu \(2018\)](#) e até reduzir a quantidade de energia a ser armazenada, como em [He \(2017\)](#).

No que tange ao uso de redes neurais convolucionais e redes de distribuição de energia, foi encontrado apenas um trabalho recente. No projeto de Zillman ([SILVA VINICIUS FERREIRA VIDAL, 2018](#)) são utilizadas duas arquiteturas de redes neurais profundas com o objetivo de identificar postes ao longo de uma ferrovia.

Com base nas informações dessa seção, pode-se concluir que existe um vasto conjunto de trabalhos envolvendo algoritmos de aprendizado de máquina e redes de distribuição de energia. A quantidade das produções diminui quando se trata especificamente de redes neurais convolucionais, o que proporciona menos base para comparação, porém possibilita um estudo com proposta original.

## 1.2 Justificativa

Uma rede de distribuição que serve equipamentos ao longo de uma estrada de ferro, torna sua inspeção uma tarefa complexa, considerando a variedade de componentes e a extensão da rede. Uma solução computacional se torna interessante para esse tipo de inspeção, considerando que existem meios que agilizam a obtenção das imagens da rede de distribuição, como câmeras que registram imagens panorâmicas e *drones*.

Assim, uma alternativa para esse tipo de problema é uma solução baseada em aprendizado de máquina, que exige que existam exemplos ou instâncias para que se possa treinar o algoritmo ou modelo. Em situações comuns, isso inviabiliza a construção de um modelo generalista, considerando que é inviável obter uma base de imagens de defeitos ampla em um sistema de produção. O cenário desse trabalho é possibilitado pela disponibilidade de um laboratório de energia, um ambiente que possui as estruturas necessárias para simular variados tipos de defeitos nos componentes da rede, como postes, transformadores, cruzetas e chaves faça.

Considerando a disponibilidade dos recursos para aquisição das imagens, simulação dos defeitos e treinamentos de redes neurais convolucionais, tem-se todos os requisitos para desenvolver uma ferramenta utilizando redes neurais convolucionais que irá auxiliar nas inspeções ao longo de toda a ferrovia de forma ágil e confiável, diminuindo o possível erro da inspeção humana que pode ser influenciada pela qualidade de visão do avaliador, do cansaço e outros fatores que podem afetar a avaliação.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

O objetivo geral deste trabalho consiste em estruturar uma arquitetura de Rede Neural Convolucional que seja capaz de identificar os defeitos mais comuns nos componentes da rede de distribuição de energia.

### 1.3.2 Objetivos Específicos

Os objetivos específicos consistem em:

- Criar um *dataset* rotulado de componentes de uma rede de distribuição de energia.
- Avaliar o desempenho de uma *CNN* utilizando uma GPU (*General Purpose Graphical Processing Unit*).
- Identificar mudanças na arquitetura que melhorem o desempenho para objetos de menor porte em uma imagem.

## 1.4 Estrutura

O Capítulo 2 deste trabalho apresenta uma introdução as redes de distribuição de energia e sua importância para o funcionamento da ferrovia. Este capítulo apresenta uma espécie de resumo para melhor entendimento do trabalho, sendo responsável por apresentar conceitos básicos sobre ferrovias e sistemas de distribuição de energia, além do laboratório de energia, sua estrutura e os tipos de componentes contidos no mesmo.

O Capítulo 3 apresenta uma visão geral sobre redes neurais convolucionais e suas camadas internas, servindo de base para entender como esse tipo de algoritmo de aprendizado funciona, permitindo melhor entendimento de como a rede neural consegue identificar os defeitos. Ainda no Capítulo 3, a Seção 3.6 é responsável por apresentar arquiteturas clássicas, parâmetros utilizados na literatura e por fim justificar a escolha do modelo utilizado, além da metodologia aplicada.

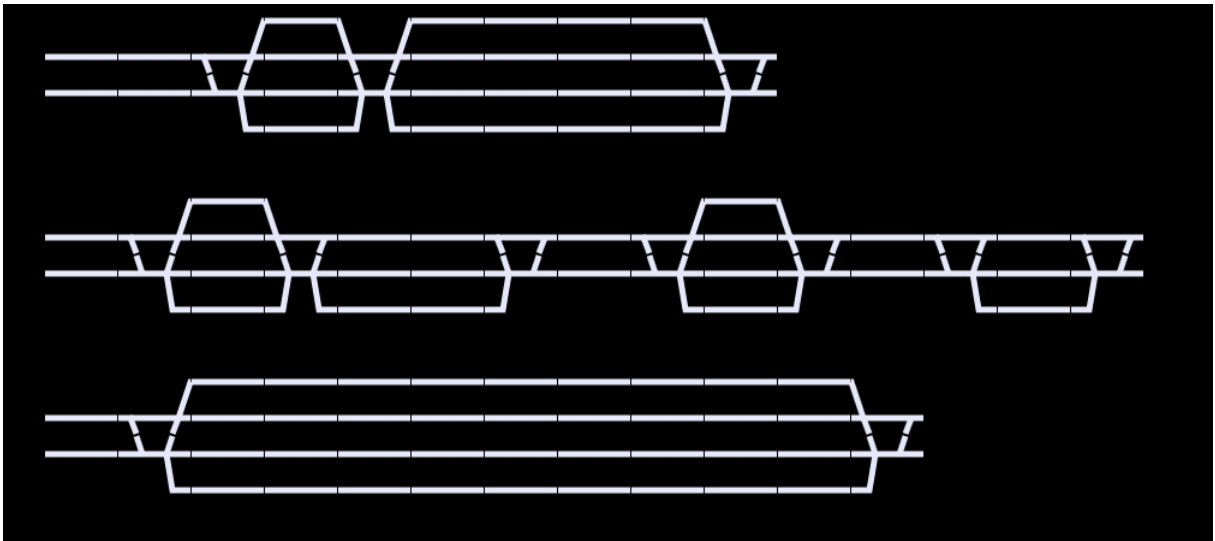
O Capítulo 4 apresenta os experimentos executados e o desempenho da arquitetura escolhida, de forma a verificar a eficácia do modelo que servirá como núcleo da aplicação de inspeção automática de redes de distribuição de energia.

O Capítulo 5 apresenta as conclusões a respeito dos resultados obtidos, além de apresentar possíveis pontos de melhoria e trabalhos futuros.

## 2 Referencial Teórico

Uma ferrovia ou sistema ferroviário é um sistema de transporte composto de uma ou mais linhas de circulação compostas por segmentos, que podem conter estruturas mais simples ou mais sofisticadas, conforme Figura 1. Do ponto de vista estrutural, cada segmento é formado por trilhos, que são as estruturas físicas pelas quais ocorre a circulação.

Figura 1 – Ferrovia



Fonte: Acervo do autor.

Na maior parte das ferrovias, os trens representam a maior porcentagem dos veículos que circulam sobre as linhas, sendo possível também a circulação de bondes e carros de manutenção. Cada trem é composto por uma ou mais locomotivas e um conjunto de vagões. A locomotiva é o componente do trem que possui tração, geralmente guiada por um maquinista, enquanto os vagões são comandados pela locomotiva.

Quanto a classificação de trens, pode-se destacar dois tipos:

- **Trens Cargueiros:** São responsáveis por trabalho de logística, geralmente transportando um produto de sua origem para um destino. Trens cargueiros apresentam o melhor custo benefício na relação *Volume Transportado \* Combustível Utilizado*. Produtos comumente transportados são grãos, como soja, além de produtos derivados de mineração, como minério de ferro, níquel e manganês (TERRESTRES, 2012).
- **Trens de Passageiros:** São responsáveis pelo transporte de passageiros e cargas pessoais, realizando paradas em estações definidas ao longo do trajeto. Um trem de passageiros pode realizar um trajeto urbano ou um trajeto interurbano, no segundo caso podendo também oferecer serviços de frete de pequenos volumes.



Em uma ferrovia típica, vários trens circulam na via ao mesmo tempo, em velocidades e sentidos diferentes. Considerando quantidade de movimentações de trens, cruzamentos, além de manutenções na via, é possível deduzir que nenhuma ação é dada ao acaso. Todos os detalhes na operação de uma ferrovia são planejados, desde manutenções preventivas até as rotas seguidas pelos trens. A obtenção de dados a respeito da localização de cada trem, os segmentos pelos quais ele passou, as movimentações dos segmentos que foram aplicadas para que o mesmo chegasse, além de restrições de velocidade são obtidas por equipamentos que formam o **Sistema de Sinalização** da ferrovia.

Além do sistema de sinalização, existem equipamentos eletroeletrônicos cuja utilização oferece melhoria e segurança operacional, tal quais cancelas em cruzamentos, equipamentos para detecção de *Hot Box* (GALLAGHER; PELINO, 1959) e *Hot Wheel*<sup>1</sup> (GALLAGHER, 1995). Esses equipamentos formam o **Sistema de Automação** da ferrovia.

Os equipamentos eletroeletrônicos de sinalização e automação possuem uma característica em comum, necessitam de uma fonte de alimentação. Essa alimentação geralmente parte de uma subestação e chega aos equipamentos por meio de redes de distribuição. As redes de distribuição de energia estão presentes ao longo de toda a ferrovia, o que por si demonstra sua importância, porém aumenta radicalmente o número de pontos de falha.

Considerando o cenário explicado nos parágrafos anteriores, a seção a seguir apresenta uma visão geral a respeito de redes de distribuição.

## 2.1 Redes de Distribuição de Energia

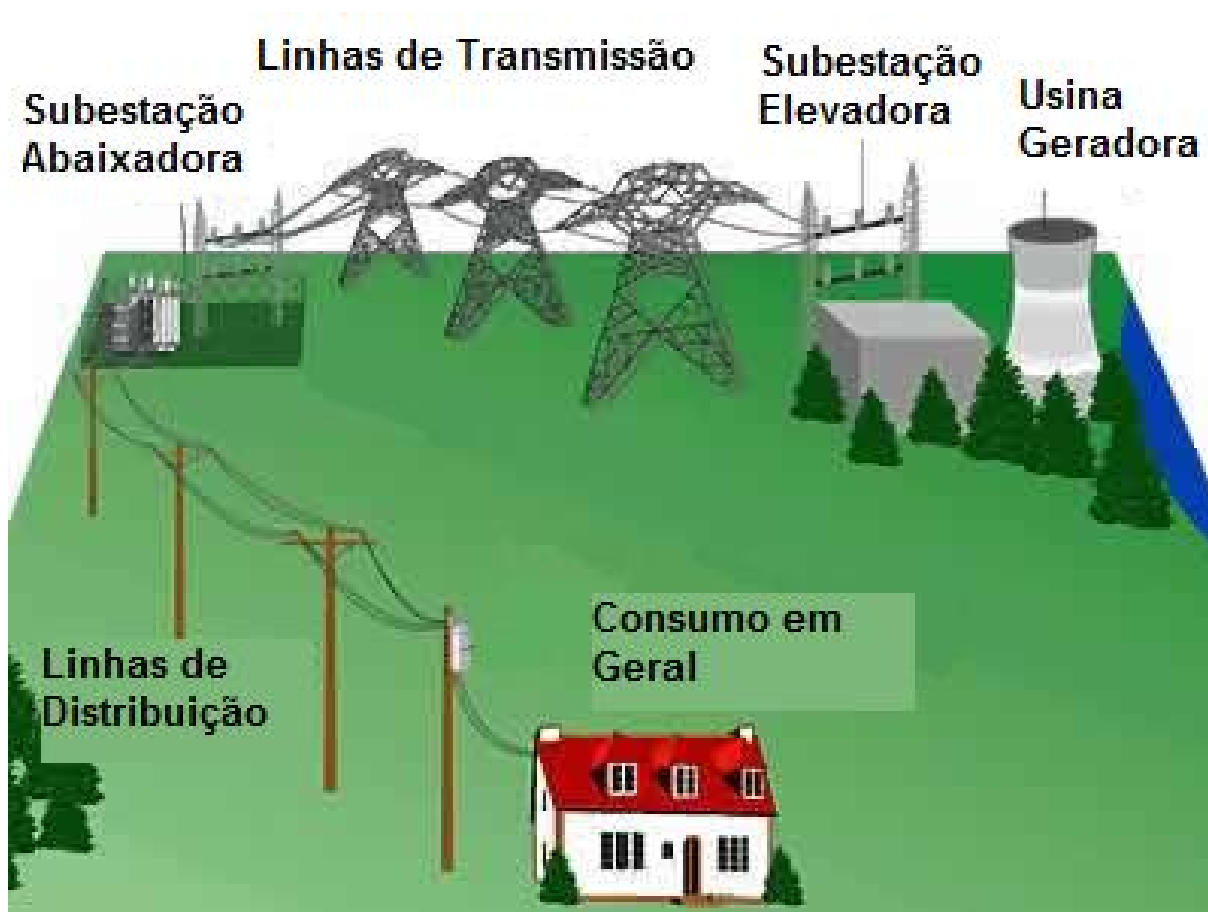
Um sistema elétrico é uma estrutura complexa que tem como função fornecer energia elétrica as unidades consumidoras, tais como residências, empresas e no contexto deste trabalho, equipamentos ao longo de uma ferrovia. Dentro de um sistema elétrico de potência, podem-se identificar três grandes blocos:

- **Geração:** Responsável por converter alguma forma de energia em energia elétrica, como usinas hidrelétricas e termelétricas por exemplo.
- **Transmissão:** Responsável por transmitir energia elétrica dos centros de transmissão as unidades consumidoras.
- **Distribuição:** Responsável por distribuir a energia elétrica do sistema de transmissão aos consumidores de todos os portes, como residências e equipamentos eletrônicos.

<sup>1</sup> *Hot Box Hot Wheel* são equipamentos utilizados para detecção de temperaturas elevadas em rolamentos e rodeiros, respectivamente.

Dentro do bloco de transmissão, pode-se identificar um componente importante para o andamento desse trabalho, a **Subestação de Distribuição** ou subestação abaixadora de tensão. A subestação de distribuição é responsável pela transformação da tensão de transmissão para a de distribuição, conforme Figura 2.

Figura 2 – Esquema Sistema Elétrico



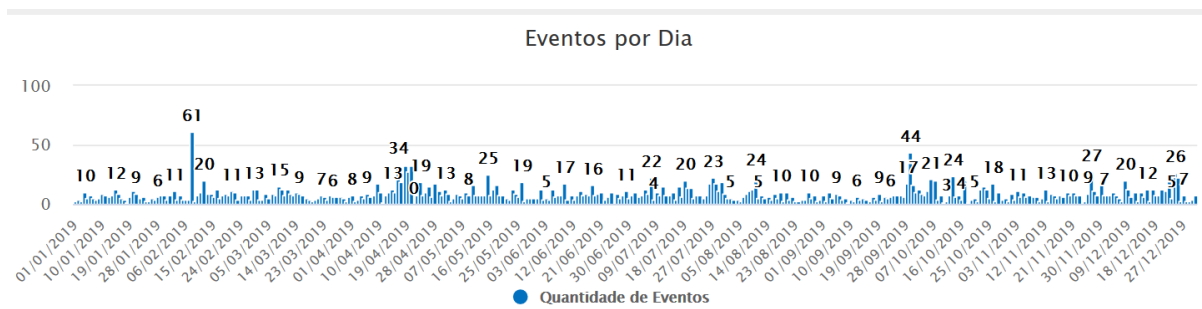
Fonte: (MUZY, 2012).

Dentro de uma rede de distribuição existem vários componentes, tais como postes, transformadores, chaves faca, chaves fusíveis e religadores. Todos estes componentes estão sujeitos a falhas em função de surtos, quedas ou oscilação de tensão. No caso de uma rede elétrica distribuída ao longo de uma ferrovia, as linhas de distribuição ainda estão sujeitas a erosão do solo, que pode prejudicar a estrutura dos postes, devido ao locais em que os mesmos devem ser posicionados, além de vandalismo, com objetivo de prejudicar as operações ou furtar componentes, tais como cabos e baterias.

A Rede de distribuição de energia de uma ferrovia se encontra distribuída por muitos quilômetros, o que aumenta os possíveis pontos de falha e dificulta a atividade de inspeção. A distribuição dos 3100 defeitos relacionados a energia elétrica na ferrovia objeto de estudo deste trabalho no ano de 2019 é apresentada na Figura 3.

Dado o alto número de defeitos na rede, se faz necessária uma rotina de inspeção

Figura 3 – Eventos de Energia 2019



Fonte: Acervo do autor.

com intuito de identificar possíveis problemas em sua estrutura e evitar futuras falhas devido ao estado dos componentes. As inspeções ao longo da rede são realizadas de forma manual, em que um empregado utiliza um binóculo para observar os componentes e detectar possíveis anomalias. Esse método de inspeção possui alguns pontos problemáticos, que incentivam a busca por uma solução automatizada, dentre eles podem-se destacar:

- **Interpretação do Especialista:** Mesmo a interpretação do especialista mais experiente está sujeita as condições do ambiente, como iluminação e posição dos postes, além do ângulo de captação das imagens, assim como qualidade do binóculo utilizado.
- **Monotonia:** Devido a extensão das linhas de distribuição, a atividade de inspeção pode se tornar exaustiva, o que pode prejudicar a avaliação do responsável pela inspeção.

Dados os pontos problemáticos, pode-se considerar uma solução computacional adequada para resolver esses problemas, levando em conta os avanços em aprendizado de máquina e visão computacional, como em [Redmon e Farhadi \(2018\)](#) e [Liu et al. \(2016\)](#). A problemática se dá pela falta de dados relativos aos problemas apresentados, em que o registro de imagens relativas ao defeito não faz parte do procedimento. Considerando que as linhas são energizadas, a simulação de defeitos nos componentes das mesmas se torna inviável, o que torna trabalhos como esse difíceis de se encontrar na literatura.

Esse trabalho se torna viável por contar com um ambiente onde é possível realizar simulações de defeitos nos componentes de uma rede de distribuição, trata-se do laboratório de energia, que será apresentado na próxima seção.

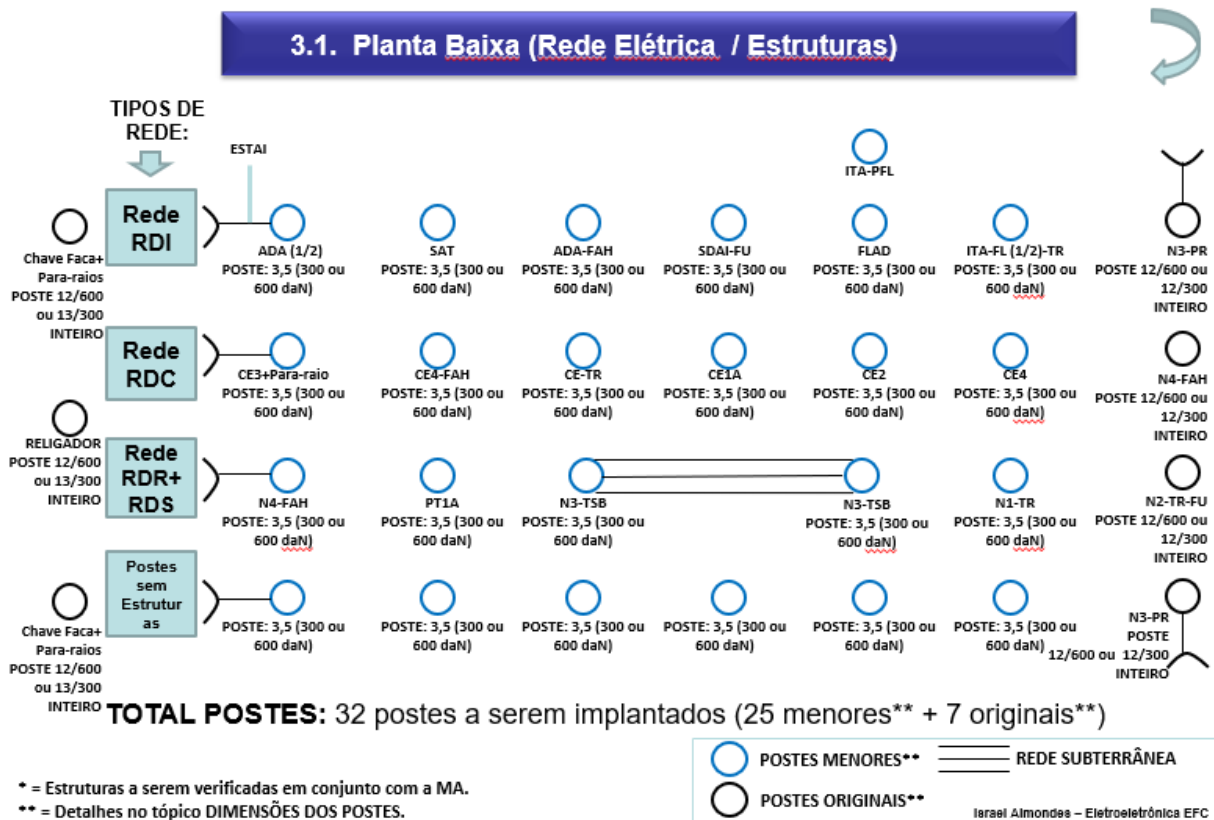
## 2.2 Laboratório de Energia

O laboratório de energia é uma estrutura projetada para fornecer treinamentos de manutenção nos componentes da rede de distribuição de energia elétrica. O ambiente

foi projetado para conter diversas estruturas que permitam habilitar empregados na manutenção de variados tipos de redes de distribuição ao longo da ferrovia.

A planta baixa do laboratório é exibida na Figura 4, onde é possível visualizar os diferentes componentes do laboratório de forma geral. Uma característica importante desse ambiente é que o mesmo encontra-se desenergizado, o que possibilita a manipulação de seus componentes para maior entendimento por parte daqueles que estão sendo treinados. Essa possibilidade de movimentação de seus componentes garante uma oportunidade interessante, a simulação de defeitos, que faz do laboratório uma peça fundamental neste trabalho.

Figura 4 – Planta Baixa Laboratório



Fonte: Acervo do autor.

As estruturas principais do laboratório são apresentadas nas Figuras de 5 a 23, enquanto a Tabela 1 apresenta seus respectivos componentes. Nas Figuras 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 e 21 é possível observar um detalhe importante: A altura do poste se encontra reduzida, o que facilita a manipulação de seus componentes, tornando os treinamentos mais interativos.

Além de desgaste natural por tempo de utilização, todos esses componentes estão sujeitos a desastres naturais, tais quais deslizamentos de terra, descargas elétricas atmosféricas, além de ações externas, como vandalismo, o que reforça a importância de uma rotina ágil de inspeções nos componentes da rede elétrica.

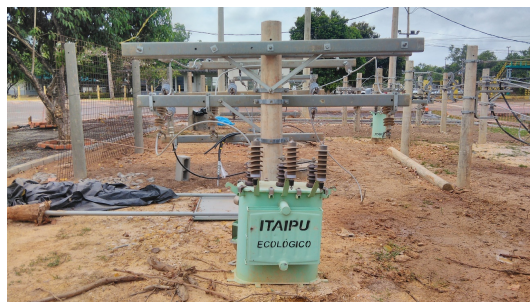


Figura 5 – Estrutura PTA1



Fonte: Acervo do autor.

Figura 6 – Estrutura N1TR



Fonte: Acervo do autor.

Figura 7 – Estrutura NHFHA



Fonte: Acervo do autor.

Figura 8 – Estrutura N4



Fonte: Acervo do autor.

Figura 9 – Estrutura N3-TSB



Fonte: Acervo do autor.

Figura 10 – Estrutura ADA



Fonte: Acervo do autor.

Figura 11 – Estrutura SAT



Fonte: Acervo do autor.

Figura 12 – Estrutura SDAI-FU



Fonte: Acervo do autor.



Figura 13 – Estrutura ITA-PFL



Fonte: Acervo do autor.

Figura 14 – Estrutura FLAD



Fonte: Acervo do autor.

Figura 15 – Estrutura ITA-FL



Fonte: Acervo do autor.

Figura 16 – Estrutura CE3



Fonte: Acervo do autor.

Figura 17 – Estrutura CE4-FAH



Fonte: Acervo do autor.

Figura 18 – Estrutura CE1-A



Fonte: Acervo do autor.

Figura 19 – Estrutura CE2



Fonte: Acervo do autor.

Figura 20 – Estrutura CE4



Fonte: Acervo do autor.

Figura 21 – Estrutura CETR



Fonte: Acervo do autor.

Figura 22 – Estrutura N3-PR



Fonte: Acervo do autor.

Figura 23 – Estrutura N2-TR-FU



Fonte: Acervo do autor.

Tabela 1 – Composição das Estruturas

<b>Estrutura</b>	<b>Componentes</b>
PTA1	Poste, Cinta e Isolador
N1TR	Poste, Cinta, Cruzeta, Isolador e Protetor de Bucha
NHFHA	Poste, Cinta e Cruzeta
N4	Poste, Isolador, Cinta e Cruzeta
N3-TSB	Poste, Chave Faca, Cinta e Cruzeta
ADA	Poste, Cruzeta e Isolador
SAT	Poste, Chave Faca, Cinta e Cruzeta
SDAI-FU	Poste, Chave Faca e Cruzeta
ITA-PFL	Poste, Cinta e Cruzeta
FLAD	Poste e Cinta
ITA-FL	Poste, Chave Faca, Cruzeta e Isolador
CE3	Poste, Cinta, Isolador e Mão Francesa
CE4-FAH	Poste, Cruzeta e Isolador
CE1-A	Poste, Cinta e Espaçador
CE2	Poste, Cinta e Isolador
CE4	Poste, Cinta, Isolador e Mão Francesa
CETR	Poste, Cruzeta, Protetor de Bucha e Transformador
N3-PR	Poste, Cruzeta e Isolador
N2-TR-FU	Poste, Cruzeta, Isolador e Protetor de Bucha

## 2.3 Considerações Finais

Neste capítulo foram apresentados conceitos essenciais para o entendimento deste trabalho. Foram explanados, o uso de equipamentos eletroeletrônicos nos sistemas de sinalização e automação, foram apresentados também os conceitos de ferrovia, segmentos, locomotivas e tipos básicos de trens.

Além disso, foram apresentados os conceitos básicos quanto a estrutura de uma rede de distribuição e foi apresentada uma visão geral quanto ao laboratório de energia, que foi utilizado neste trabalho. Em seguida, o Capítulo 3 inicia a parte técnica da pesquisa, apresentando uma introdução as redes neurais convolucionais, que são utilizadas nesse trabalho.



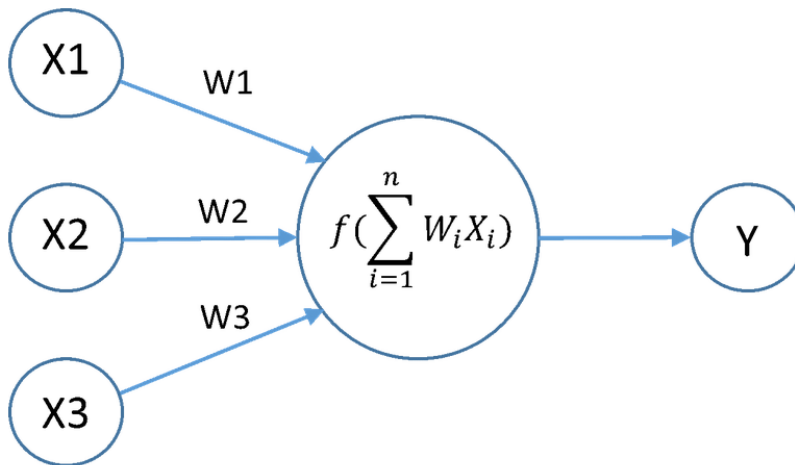
## 3 Redes Neurais Convolucionais

### 3.1 Redes Neurais

Uma Rede Neural pode ser definida como um conjunto de unidades de processamento conectadas por ligações direcionadas (RUSSELL; NORVIG, 2016). Cada unidade de processamento dessa estrutura recebe a nomenclatura de neurônio, enquanto cada ligação entre quaisquer neurônios  $i$  e  $j$  é denominada conexão, servindo para propagar a ativação entre  $i$  e  $j$ . Cada conexão possui um peso  $w_{ij}$  associado, que determina a força da conexão entre os dois neurônios.

Um neurônio é responsável por receber uma entrada, realizar uma computação e passar o resultado dessa computação como saída. Um modelo simplificado de neurônio pode ser visualizado na Figura 24.

Figura 24 – Neurônio Artificial



Fonte: Acervo do autor.

Nas redes neurais artificiais clássicas, os neurônios realizam a soma ponderada de suas entradas com as respectivas conexões, conforme Equação 3.1, a seguir aplicam uma função de ativação  $g$ , conforme Equação 3.2, obtendo assim a saída.

$$in_j = \sum_{i=0}^n w_{ij} x_i \quad (3.1)$$

$$a_j = g\left(\sum_{i=0}^n w_{ij} x_i\right) \quad (3.2)$$

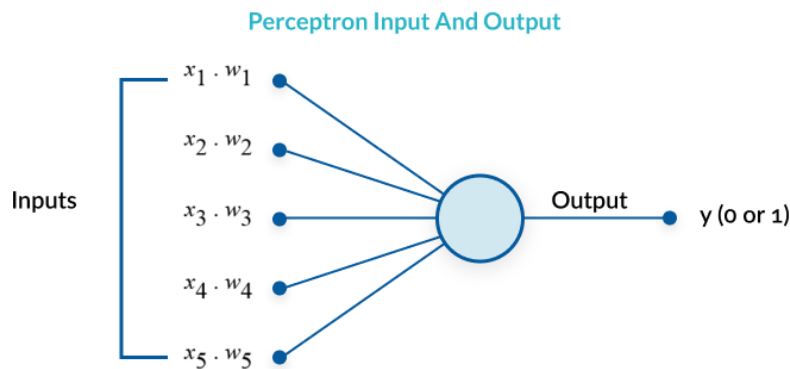
onde  $in_j$  é a entrada do neurônio  $j$ ,  $n$  é a quantidade de conexões entre a camada anterior e o neurônio  $j$  e  $w_{ij}$  representa a multiplicação entre o elemento  $i$  da camada anterior e o neurônio  $j$ .

A função de ativação  $g$  tipicamente é uma função de limiar rígido ou logística, o que garante que uma rede neural pode representar funções não lineares.

As redes neurais artificiais clássicas geralmente estão dispostas em camadas, de forma que neurônios da uma camada  $n$  obrigatoriamente recebam entradas apenas a partir de unidades da camada  $n - 1$  (RUSSELL; NORVIG, 2016).

Dentro do conjunto das redes neurais com alimentação para frente (SVOZIL; KVASNICKA; POSPICHAL, 1997), pode-se destacar as redes da camada única ou **perceptron**, cujas unidades que se conectam as entradas, se conectam diretamente a(s) unidade(s) de saída (Figura 25); Enquanto a variação mais conhecida se trata da **perceptron multicamadas** (Figura 26) ou *Multilayer Perceptron (MLP)*.

Figura 25 – Rede Perceptron



Fonte: (BEILE, 2017).

De forma a finalizar a seção, pode-se definir as diferenças básicas causadas pelas diferentes topologias entre os tipos de rede:

- Perceptron:

Não resolve problemas com funções não linearmente separáveis;

Aprendizado pela regra:  $w_i \leftarrow w_i + \alpha(y - h_w(X)) * X_i$ ;

- Perceptron Multicamadas:

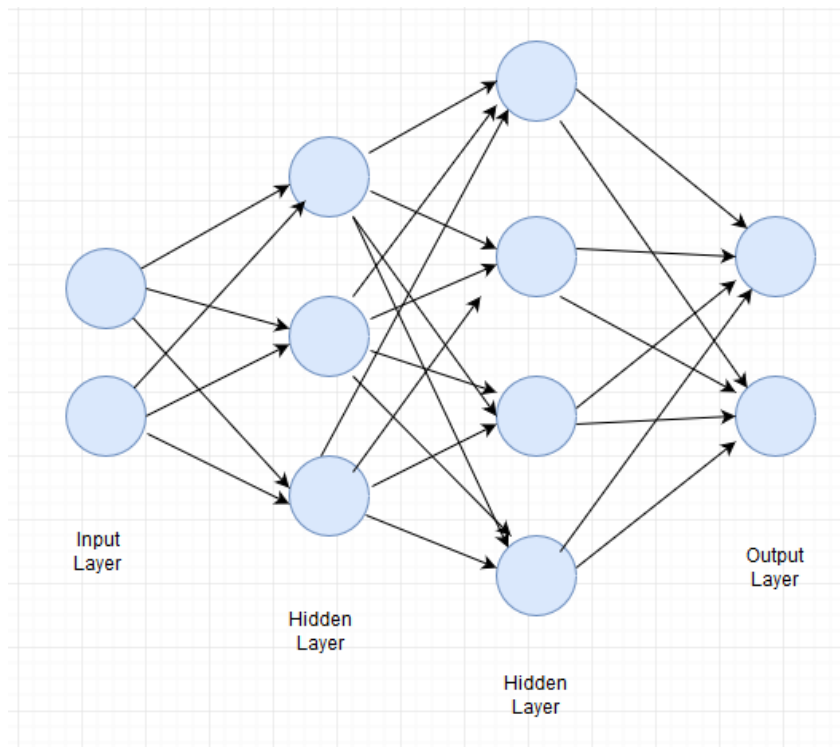
Resolve problemas com funções contínuas ou descontínuas (utilizando duas camadas escondidas);

Aprendizado pela regra:  $w_{i,j} \leftarrow w_{i,j} + \alpha * a_i * \delta_j$ ;

## 3.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais (*CNNs*) são similares às amplamente exploradas redes *Multilayer Perceptron (MLP)* (GARDNER; DORLING, 1998), sendo formadas por blocos

Figura 26 – Rede Perceptron Multicamadas



Fonte: Acervo do autor.

de neurônios, além de possuírem *biases* e pesos, podendo estes últimos serem aprendidos através de técnicas como *Backpropagation* ou suas variantes. Assim como nas *MLP*, cada neurônio ou unidade da *CNN* recebe algumas entradas, aplica um produto escalar entre os pesos do neurônio e os valores de entrada, podendo ainda aplicar uma função de ativação no resultado do produto (KARPATY, 2016). A Rede Neural Convolucional permanece como um aproximador, tendo como objetivo minimizar uma função de perda, além de possuir camadas completamente conectadas cujo objetivo normalmente é realizar a escolha de uma dentre  $n$  classes.

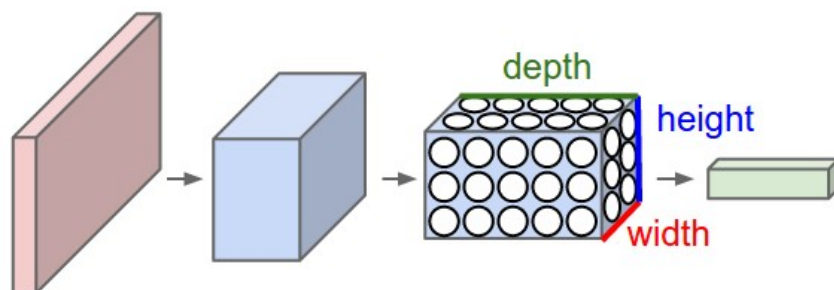
Considerando que as *CNNs* possuem diversas características em comum com as redes convencionais ou totalmente conectadas, pode-se partir para a principal assunção nas Redes Neurais Convolucionais, que se trata do tipo de entrada. As *CNNs* consideram explicitamente que sua entrada se trata de uma imagem, ou seja, esperam um tensor no formato  $W * H * C$ , onde  $W$  representa a quantidade de pixels no eixo horizontal (largura da imagem),  $H$  a mesma medida no eixo vertical (altura da imagem) e  $C$  representa quantidade de canais contidos na imagem (profundidade).

Dada a assunção a respeito das entradas de uma *CNN*, deve-se explicar sua vantagem em relação a uma rede neural convencional, que é a falta de escala em relação ao número de atributos. Considerando que uma rede neural totalmente conectada possui um peso  $w_i$  para cada entrada da rede, uma imagem de dimensão  $32 * 32 * 3$ , resultaria em 3072 pesos somente na primeira camada, que seriam multiplicados por mais  $h$  neurônios de uma possível camada escondida subsequente, o que deixa claro que essa abordagem é

inviável quando se trabalha com imagens de maior dimensão e arquiteturas de redes mais complexas.

Assim, em se tratando das redes convolucionais, cada neurônio é conectado com apenas uma parte da camada anterior (Figura 27), geralmente reduzindo as dimensões da imagem de entrada, fazendo com que o número de parâmetros seja menor, diminuindo assim o *overfitting*<sup>1</sup> (TETKO; LIVINGSTONE; LUIK, 1995).

Figura 27 – Convolutional Neural Network



Fonte: Acervo do autor.

Uma *CNN* pode também ser definida pelo empilhamento de três tipos principais de camadas: **Camada de Convolução**, **Camada de *Pooling*** e **Camada Totalmente Conectada**. Cada uma dessas camadas é discutida nas próximas seções, de forma a facilitar o entendimento de uma Rede Neural Convolucional como um todo.

### 3.3 Camada de Convolução

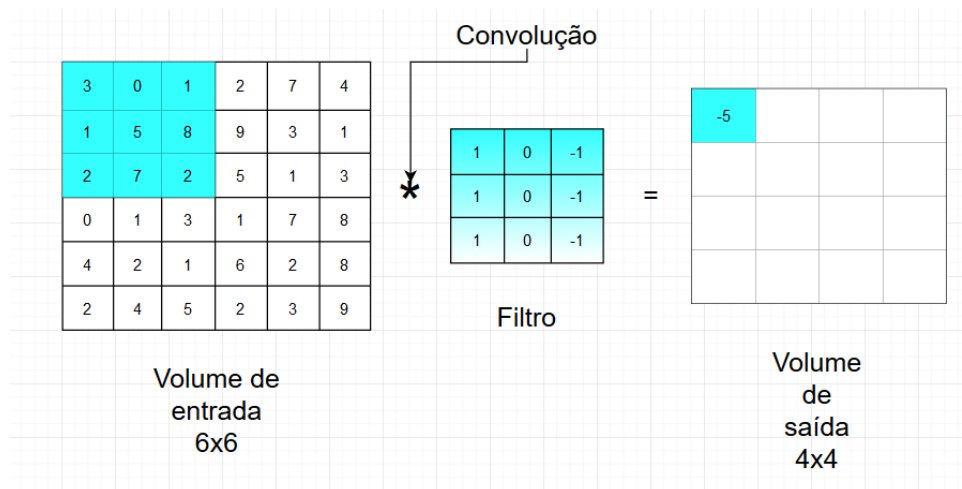
O operador de convolução é o principal componente de uma rede neural convolucional. Essa operação consiste na aplicação de filtros ou *kernels* nas imagens de entrada, de forma a extrair determinadas características das mesmas. Os parâmetros de uma camada de convolução consistem nos valores desses filtros, sendo esses, valores aprendíveis.

A operação de uma camada de convolução consiste em deslizar ou convolver cada filtro sobre a extensão do volume de entrada e realizar um produto escalar para cada posição dessa janela, conforme visto na Figura 28. A medida que essa janela desliza, o resultado de cada produto escalar é armazenado em um volume de saída, sendo esse volume um mapa de ativação de 2 dimensões que representa a resposta da aplicação do filtro em cada posição do volume entrada (KARPATHY, 2016).

Os filtros são partes essenciais da camada de convolução, sendo que cada um deles é ativado quando identificam algum tipo de padrão visual em sua entrada, seja ele uma borda ou até mesmo um objeto inteiro. Cada filtro produz seu próprio mapa de ativação, sendo que a saída padrão de uma camada de convolução é dada pelo empilhamento desses

<sup>1</sup> *Overfitting* se trata da superadaptação de um modelo em relação aos dados de treino, o que faz com que os resultados do mesmo em novos dados sejam ruins.

Figura 28 – Operação de Convolução



Fonte: Acervo do autor.

filtros. É importante notar que a medida  $C$  ou quantidade de canais da entrada deve ser igual a profundidade do filtro, ou seja, para uma imagem  $32 * 32 * 3$ , se o objetivo é aplicar um filtro  $5 * 5$ , cada neurônio na camada de convolução deverá ter profundidade 3, ou seja de dimensão  $5 * 5 * 3$ .

Além dos parâmetros que podem ser aprendidos, como os valores contidos nos filtros, a camada de convolução também possui hiperparâmetros, ou seja, parâmetros que não são aprendidos pela *CNN* e devem ser definidos durante a etapa de treino. Os hiperparâmetros da camada de convolução são **Profundidade da Saída**, que representa o tamanho da pilha de mapas de ativação ou quantidade de filtros; **Stride** ou Passo, que representa quantos *pixels* serão deslizados ou pulados a cada passo da convolução, quanto maior o *stride*, menor o volume de saída; O **Padding** apresenta um complemento ou moldura ao redor da imagem original. Esse hiperparâmetro permite que se controle o tamanho do volume de saída, sendo comumente utilizado para preservar a altura e a largura de um volume de entrada após uma convolução.

Considerando um volume de entrada em que  $W = H$ , ou seja, uma imagem quadrada,  $F$  representa o tamanho do filtro aplicado,  $S$  é o *stride* utilizado e o *padding* é dado por  $P$ , a dimensão do volume de saída é dada pela Equação 3.3.

$$\frac{W - F + 2P}{S} + 1 \quad (3.3)$$

Com relação ao *Padding*, sua propriedade de preservar o tamanho de um tensor de entrada é mais utilizada em operações que desejem manter características próximas a bordas da imagem original. Observando a Equação 3.3, pode-se inferir que quando se opta pela não utilização dessa técnica, há uma redução na dimensão de saída em comparação com o volume de entrada, o que pode ser inviável quando se trabalha com uma arquitetura

de muitas camadas. Esse problema pode ser evitado com o uso de **Same Convolutions**, que permite manter a dimensão de saída igual a de entrada. A fórmula aplicada para garantir essa igualdade de dimensões é dada por  $P = \frac{F-1}{2}$ , mantendo  $S = 1$ .

Os Hiperparâmetros de *Padding* e *Stride* possuem restrições mutuas que devem ser respeitadas para que o tamanho do volume de saída seja válido. Considerando a Equação 3.3, com  $W = 10$ ,  $P = 0$  e  $F = 3$ , não é possível utilizar  $S = 2$ , considerando que  $\frac{10-3+0}{2} + 1$  não é um número inteiro, obrigando a arquitetura a trabalhar com  $P - 1$  ou descartar parte das características de entrada com  $\lfloor \frac{W-F+2P}{S} + 1 \rfloor$  (KARPATHY, 2016).

Podem-se definir os parâmetros da camada de convolução como os pesos de cada filtro utilizado. Considerando que *CNNs* geralmente trabalham com volumes de 3 dimensões, o número de pesos de cada camada se dá por  $W * H * C$ , o que apresenta um número de parâmetros elevado em relação a redes neurais totalmente conectadas. Considerando uma camada que espera uma imagem de dimensões  $227 * 227 * 3$ , com  $F = 11$ ,  $S = 4$ ,  $P = 0$  e  $C = 96$ , conforme Krizhevsky, Sutskever e Hinton (2012b), obtêm-se uma saída de dimensão  $55 * 55 * 96$  e 290.400 parâmetros, o que pode ser reduzido utilizando **Compartilhamento de Parâmetros** (SHOKRI; SHMATIKOV, 2015).

O Compartilhamento de Parâmetros consiste de reutilizar valores de filtros capazes de detectar características em determinadas partes da imagem em outros pontos, diminuindo a quantidade de parâmetros, além de possibilitar que a rede localize elementos em qualquer local da imagem de entrada. Essa técnica utiliza os mesmos pesos em todos os neurônios de cada filtro. Seguindo o exemplo do paragrafo anterior, tem-se os mesmos  $55 * 55 * 96$  neurônios conectados com uma região de dimensão  $11 * 11 * 3$  do volume de entrada, o que resulta em 105.705.600, considerando o *bias*. Utilizando compartilhamento de parâmetros, a camada de entrada utilizaria 96 conjuntos únicos de peso, um para cada filtro, resultando em  $96 * 11 * 11 * 3$  ou 34.944 parâmetros, considerando o *bias* de cada filtro.

Para finalizar a seção, apresenta-se um resumo da Camada de Convolução segundo Karpathy (2016):

- Recebe como entrada um volume no formato  $W_1 * H_1 * D_1$ .
- Possui quatro hiperparâmetros:
  - Profundidade  $K$  ou Quantidade de Filtros.
  - Dimensão dos Filtros  $F$  (largura x altura).
  - *Stride*  $S$  ou Passo.
  - *Padding*  $P$ .
- Possui como saída um volume de dimensão  $W_2 * H_2 * D_2$ , onde:

$$\begin{aligned}
 - W_2 &= \frac{(W_1 - F + 2P)}{S} + 1 \\
 - H_2 &= \frac{(W_1 - F + 2P)}{S} + 1 \\
 - D_2 &= K
 \end{aligned}$$

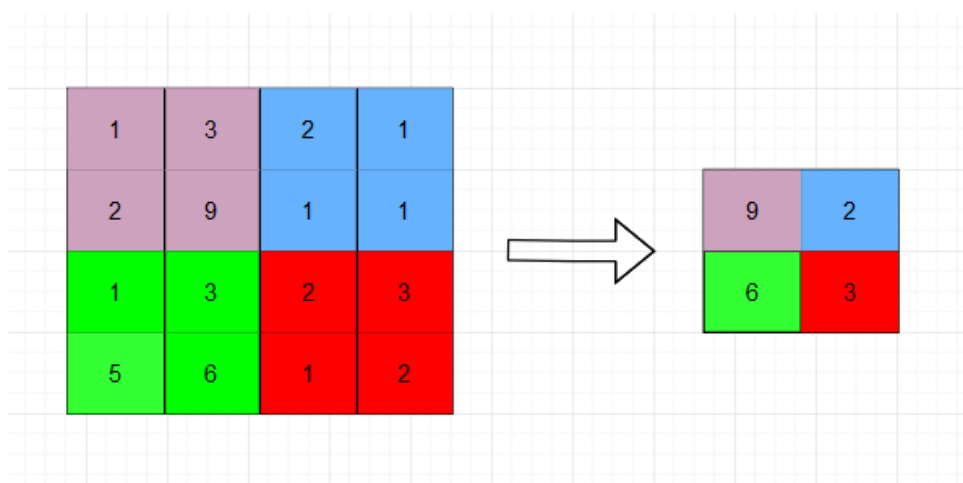
- Cada Unidade de  $K$  pode ser chamado de Fatia.
- Parâmetros comuns na camada de convolução são  $F = 3$ ,  $S = 1$ ,  $P = 1$ .

### 3.4 Camada de *Pooling*

A Camada de *Pooling* reduz a dimensão dos volumes de entrada, principalmente por dois motivos: O primeiro é diminuir a quantidade de parâmetros, consequentemente reduzindo a quantidade de operações necessárias a serem computadas pela rede; O segundo é reduzir o *overfitting* (HAWKINS, 2004), o que torna a capacidade de generalização da rede maior quando testada em novos dados.

As camadas de *Pooling* geralmente são posicionadas após as camadas de convolução dentro de uma *CNN*, sendo responsáveis por diminuir a dimensão de cada fatia do volume de entrada de forma independente. Cada região local em cada fatia do volume de entrada é reduzida utilizando uma função de redução, geralmente a de *MAX*, conforme apresentado na Figura 29, onde  $K = 1$ .

Figura 29 – Operação de Max *Pooling*



Fonte: Acervo do autor.

Quando utiliza a função de *MAX*, essa camada geralmente é chamada de *Max-Pooling*, possuindo variantes, como *Average-Pooling* (BOUREAU; PONCE; LECUN, 2010), cujo função é *AVG* ou média.

A operação de *Pooling* é comumente utilizada com *Stride* de 2 e filtros  $2 * 2$ , reduzindo assim a altura ( $W$ ) e a largura ( $H$ ) do volume de entrada pela metade. A

profundidade  $D$  não é modificada, sendo proporcional a profundidade do volume de entrada. Existem algumas variações para essa operação, como *Overlapping Pooling* e *General Pooling* (SCHERER; MÜLLER; BEHNKE, 2010).

Para finalizar a seção sobre essa camada, apresentam-se suas principais características, algumas delas comuns à camada de convolução:

- Recebe como entrada um volume no formato  $W_1 * H_1 * D_1$ .
- Não possui parâmetros aprendíveis, apenas hiperparâmetros:
  - Dimensão dos Filtros  $F$
  - *Stride*  $S$  ou Passo.
- Possui como saída um volume de dimensão  $W_2 * H_2 * D_2$ , onde:
  - $W_2 = \frac{(W_1 - F + 2P)}{S} + 1$
  - $H_2 = \frac{(H_1 - F + 2P)}{S} + 1$
  - $D_2 = D_1$
- Geralmente não se utiliza *Padding*.

### 3.5 Camada Totalmente Conectada

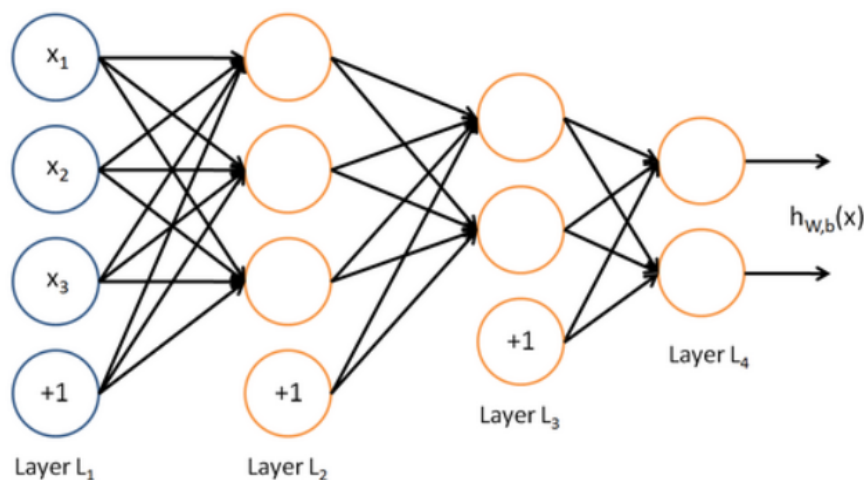
Camadas totalmente conectadas funcionam conforme as conexões de *MLP*, em que cada neurônio pertencente a uma camada é conectado com todos os neurônios da camada anterior (GARDNER; DORLING, 1998), incluindo o *bias*, conforme Figura 30. Pode-se definir a saída de um neurônio dessa camada pela aplicação de uma função de ativação em um produto escalar do vetor ou volume de entrada pelo vetor de pesos das conexões da camada anterior ligadas a unidade adicionado ao *bias*, conforme Equação 3.4.

$$y_1 = \left[ (w_{11} * x_1) + (w_{12} * x_2) + (w_{13} * x_3) \right] + b_1 \quad (3.4)$$

Considerando as características da camada totalmente conectada, consegue-se perceber que a mesma possui algumas semelhanças com a camada de convolução, ambas recebem um volume de entrada e computam um volume de saída; os parâmetros de ambas são aprendidos via *backpropagation* ou um de seus variantes; ambas as camadas produzem um valor de saída através da computação de produtos escalares.

De acordo com Karpathy (2016), duas propriedades são interessantes para observar que camadas totalmente conectadas e camadas convolucionais são intercambiáveis:



Figura 30 – Rede *Multilayer Perceptron*

Fonte: Acervo do autor.

- Do ponto de vista estrutural, uma matriz que representa uma operação de convolução, pode ser substituída por uma matriz resultante de uma operação escalar entre pesos e entradas de uma *MLP* regular, sendo que neste último caso, a maior parte dos elementos da matriz será formada por zeros, exceto as posições referentes as conexões com a parte do volume referente a janela do filtro.
- Qualquer camada totalmente conectada pode ser substituída por uma camada de convolução em que a profundidade dos volumes de entrada e de saída sejam iguais.

Essa possibilidade de conversão entre as camadas totalmente conectadas e as camadas de convolução é útil, principalmente quando se trata de agilizar o treinamento da rede, pois as camadas convolucionais permitem o compartilhamento de parâmetros, descrito na Seção 3.3. Utilizando um exemplo da arquitetura *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b), que tem como três últimas camadas, dois blocos totalmente conectados com 4096 neurônios e uma camada com 1000 unidades que apresenta a probabilidade para cada classe, podem-se substituir essas unidades por duas camadas de convolução de dimensões  $1 * 1 * 4096$  e uma de  $1 * 1 * 1000$ , permitindo que o produto escalar entre a entrada dessas camadas e os pesos se dê de forma ágil, utilizando a operação de convolução.

Para finalizar a seção, pode-se definir que a Camada Totalmente Conectada é utilizada em várias arquiteturas de *CNNs* (LECUN et al., 1998), (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b) e (SZEGEDY et al., 2015), além de ser possível converter a mesma em uma camada convolucional. Devido ao tempo requerido para treinamento de arquiteturas mais profundas, alguns trabalhos utilizam pesos já treinados nas camadas de convolução e treinam apenas as camadas totalmente conectadas, como em (ESTEVA et al., 2017).

## 3.6 Arquitetura de CNN Utilizada

Como mencionado previamente, este trabalho tem como objetivo identificar variados tipos de defeitos em uma extensa rede de distribuição de energia elétrica. Por objetivo final projetou-se substituir as inspeções com uso de binóculos por análises automatizadas utilizando redes neurais convolucionais. A aquisição das imagens será realizada por *drones* e analisada pela *CNN*, que será responsável por detectar as falhas.

Considerando todo o contexto até agora, tal como extensão da rede e possibilidades de falha de detecção por monotonia e até repetição de padrões na atividade de inspeção, apresenta-se ainda uma limitação de armazenamento, pois os arquivos gerados pela câmera do *drone* são de alta resolução, o que exige um grande repositório para armazenar seu histórico. Outro requisito não funcional é a velocidade na detecção dos defeitos, o que agiliza o processo de manutenção.

Junto aos requisitos levantados nos parágrafos anteriores, agrega-se a quantidade de elementos em uma rede de distribuição. Uma estrutura de rede agrupa vários componentes de tamanho variável, que por vezes são formados por outros componentes, o que gera a necessidade de uma arquitetura que consiga detectar várias características em uma mesma região da imagem.

Agregando requisitos funcionais e não funcionais, se faz uma escolha natural por detectores rápidos e que utilizam o conceitos de ancoras variadas, podendo assim detectar objetos de tamanhos variados. No momento do desenvolvimento deste trabalho, a rede YOLO V3 (REDMON; FARHADI, 2018) é o estado da arte em detecção de objetos, apresentando a melhor relação entre velocidade e *MAP* no conhecido *COCO Dataset*. A seguir, é apresentado um resumo a respeito YOLO V3.

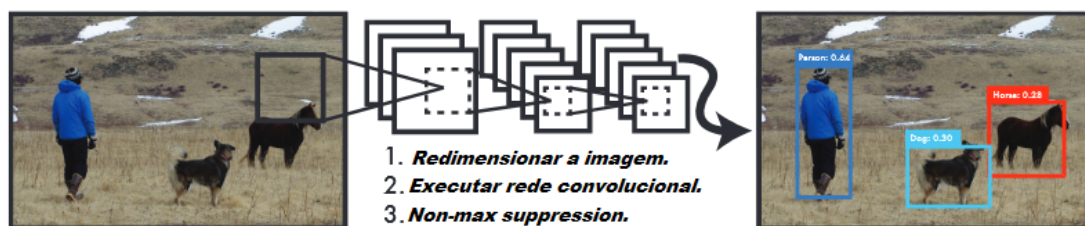
## 3.7 YOLO V3

A arquitetura utilizada na construção da rede YOLO V3 traz um novo paradigma em relação aos modelos anteriormente utilizados em detecção de objetos. Enquanto modelos como R-CNN (GIRSHICK, 2015) consistiam de um conjunto de processos, como Métodos de Proposição de Regiões, Classificadores e Pós Processadores, YOLO é uma abordagem baseada em regressão que tem como saída *bounding boxes* e as probabilidades de pertencimento a cada classe.

A figura 31 apresenta o fluxo de execução da rede, que é iniciado com o redimensionamento da imagem, passa pela execução da *CNN* e finaliza com o uso de *Non-max suppression* para excluir detecções duplicadas.

Considerando que o processo de detecção na YOLO consiste de um fluxo único, sua velocidade de detecção pode ser justificada pela possibilidade de realizar otimizações

Figura 31 – Sistema de Detecção YOLO



Fonte:([REDMON et al., 2016](#)).

do início ao fim do fluxo ([REDMON et al., 2016](#)), o que é mais custoso em modelos com *pipelines* complexos. A arquitetura padrão da YOLO é executada a 45 *fps* (*frames* por segundo), enquanto uma versão mais rápida porém com menos acurácia pode rodar a 150 *fps* em uma GeForce<sup>©</sup> Titan X.

Além de ser uma arquitetura extremamente rápida, a YOLO V3 destaca-se por considerar o contexto global de uma imagem para realizar predições. Essa abordagem faz com que a arquitetura seja menos suscetível a erros de identificação de objetos no plano de fundo da imagem.

Levando em conta o que já foi dito a respeito da arquitetura da YOLO V3, cabe dizer também que o modelo aprende representações altamente generalizáveis dos objetos, o que faz com que o mesmo seja menos sujeito a *overfitting*.

Enquanto a YOLO V3 se mostra mais rápida e menos sujeita a identificar objetos de forma incorreta no plano de fundo, pode-se destacar negativamente que sua falta de sensibilidade quanto a objetos no plano de fundo da imagem pode ocasionar baixa precisão nas coordenadas de alguns objetos, especialmente os pequenos ([REDMON; FARHADI, 2018](#)).

Após apresentada a visão geral a respeito da arquitetura utilizada, o procedimento de detecção de objetos é apresentado a seguir.

### 3.7.1 Detecção

Conforme apresentado anteriormente neste capítulo, a arquitetura da YOLO V3 utiliza todo o contexto da imagem para realizar suas predições, ou *bounding boxes*. O Processo de detecção é detalhado a seguir:

- Cada imagem é dividida em uma grade de tamanho  $S \times S$ .

Se o centro do objeto pertence a uma célula da grade, aquela célula é responsável por sua identificação.

- Cada célula prediz  $\beta$  *bounding boxes* e sua respectiva taxa de confiança <sup>2</sup>.

A taxa de confiança é calculada por  $Pr(\text{objeto}) * IOU_{\text{verdadeira}}^{\text{predicao}}$ , o seja, se a célula não possui objeto, a taxa de confiança será zero, do contrário  $IOU_{\text{verdadeira}}^{\text{predicao}}$ .

- Cada *bounding box* possui 5 predições:  $x$ ,  $y$ ,  $w$ ,  $h$  e a taxa de confiança.

O valores  $x$  e  $y$  são as coordenadas do centro do objeto com relação as fronteiras da célula, enquanto  $w$  e  $h$  representam a largura e altura total do objeto (que podem ultrapassar os limites da célula).

- Cada célula da grade também prediz  $C$  probabilidades condicionais de classe, que são dadas por  $Pr(\text{Classe}_i|\text{objeto})$ .

Para identificar a probabilidade de cada predição pertencer a uma classe, é feita a multiplicação da taxa de confiança de cada *bounding box* pela probabilidade condicional de cada classe, conforme:  $Pr(\text{Classe}_i|\text{objeto}) * Pr(\text{objeto}) * IOU_{\text{verdadeira}}^{\text{predicao}} = Pr(\text{classe}_i) * IOU_{\text{verdadeira}}^{\text{predicao}}$ .

- O final do processo de detecção consiste de um tensor de tamanho  $S \times S \times (\beta * 5 + C)$ .

Levando em conta o que foi explorado nessa seção, se consegue entender como funciona o processo de detecção, que tem como entrada uma imagem e como saída um vetor que contém a probabilidade de um objeto existir naquela célula, suas coordenadas e a probabilidade de pertencer a cada classe dada no conjunto de dados de treino. A seguir a arquitetura completa da YOLO será detalhada.

### 3.7.2 Arquitetura da YOLO

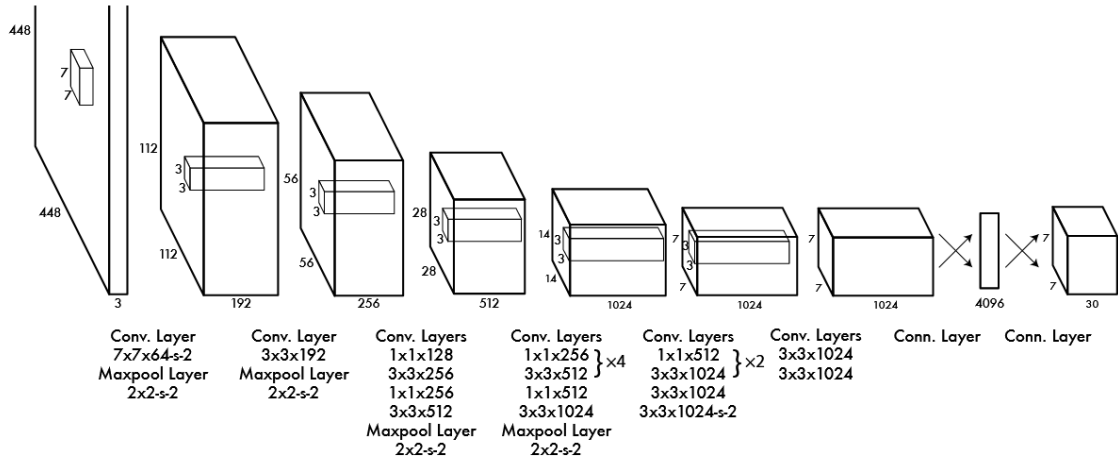
A arquitetura completa da YOLO pode ser vista na Figura 32. O Princípio básico consiste nas camadas exploradas na Seção 3.3. As camadas de convolução são responsáveis por extrair características das imagens, enquanto as camadas totalmente conectadas predizem as probabilidades condicionais das classes e as coordenadas dos objetos.

A rede YOLO possui 24 camadas de convolução, seguidas por duas camadas totalmente conectadas. As camadas de convolução geralmente são pré-treinadas em *datasets* com uma grande variedade de objetos, o que proporciona maior capacidade de generalização em tarefas específicas.

A arquitetura padrão da rede YOLO normaliza a altura e a largura de cada *bounding box* pelas respectivas medidas da imagem, o que faz com que as medidas  $w$  e  $h$  sempre se encontrem entre 0 e 1, assim como  $x$  e  $y$ , levando em conta que o centro de um objeto sempre irá pertencer a uma célula.

<sup>2</sup> Em detecção de objetos, pode-se definir *IOU* ou *Intersection over Union* por  $\frac{AI}{AU}$ , sendo  $AI$ , área de interseção e  $AU$ , área de união entre duas imagens.

Figura 32 – Arquitetura YOLO



Fonte:(REDMON et al., 2016).

Em relação as funções de ativação, a última camada possui ativação linear, conforme Equação 3.5, enquanto as outras camadas utilizam a variação *Leaky ReLu*, conforme Equação 3.6.

$$\theta(x) = cx \tag{3.5}$$

$$\theta(x) = \begin{cases} x, & \text{se } x > 0 \\ 0, 1x, & \text{senão.} \end{cases} \tag{3.6}$$

O erro a ser minimizado é o erro quadrático médio, calculado em sua forma básica conforme Equação 3.7. Essa métrica possibilita mais velocidade ao ser otimizada via método do gradiente, porém causa um aumento do gradiente, considerando que alguns *bouding boxes* não possuem objetos, possuindo probabilidades de classes iguais a 0. Outro problema dessa métrica é que por tratar o tensor de saída de forma igual, erros de localização e classificação possuem o mesmo peso na otimização.

$$\sum_{i=1}^n (x_i^{predito} - x_i^{real})^2 \tag{3.7}$$

De forma a inibir os problemas causados pela minimização do erro quadrático médio, são utilizados dos parâmetros,  $\lambda_{coord} = 5$  e  $\lambda_{noobj} = 0,5$ , que aumentam o erro para predições de localização e diminuem o erro para predições de classes (REDMON et al., 2016).

Por fim, pode-se definir a função a ser minimizada na arquitetura YOLO por:

$$\begin{aligned}
& \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{I}_{ij}^{obj} [(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2] \\
& + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{I}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\bar{w}_i})^2 + (\sqrt{h_i} - \sqrt{\bar{h}_i})^2] \\
& \quad + \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{I}_{ij}^{obj} (C_i - \bar{C}_i)^2 \\
& \quad + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{I}_{ij}^{noobj} (C_i - \bar{C}_i)^2 \\
& \quad + \sum_{i=0}^{S^2} \mathbb{I}_i^{obj} \sum_{c \in Classes} (p(i) - \bar{p}_i(c))^2
\end{aligned} \tag{3.8}$$

, onde  $\mathbb{I}_i^{obj}$  representa a presença de um objeto na célula  $i$  e  $\mathbb{I}_{ij}^{obj}$  representa a presença de um objeto na célula  $i$ , sendo o *bounding box*  $j$  responsável por sua detecção.

### 3.8 Considerações Finais

Neste capítulo foram apresentados conceitos essenciais para entendimento das *CNNs*; Desde o funcionamento das redes neurais clássicas até os blocos principais das redes neurais convolucionais. As *CNNs* compõem as arquiteturas das técnicas no estado da arte para detecção de objetos e classificação de imagens, como em [Redmon e Farhadi \(2018\)](#) e [Liu et al. \(2016\)](#). Com a intuição do funcionamento de cada bloco da rede é possível compreender a composição de arquiteturas mais elaboradas e profundas.

Ponto essencial discutido neste capítulo, foi arquitetura base da rede YOLO V3, que é utilizada neste trabalho. O funcionamento da rede foi detalhado, assim como os tipos de camadas utilizados e seus parâmetros mais comuns. Com a leitura deste capítulo é possível entender como o problema de detecção de objetos foi modelado como regressão, as adaptações que tiveram que ser feitas e o porquê o mesmo permite otimização de forma mais simples.

Levando em consideração o que foi apresentado neste capítulo, é possível partir para os experimentos computacionais, onde a arquitetura aqui explanada serviu de base para a detecção de defeitos em componentes de uma rede de distribuição de energia.

## 4 Experimentos Computacionais

### 4.1 Fluxo de Trabalho

No fluxo de treinamento de um problema de detecção de objetos é importante pontuar que o *dataset* consiste do conjunto formado pelas imagens e pelas anotações que representam as coordenadas dos objetos a serem detectados.

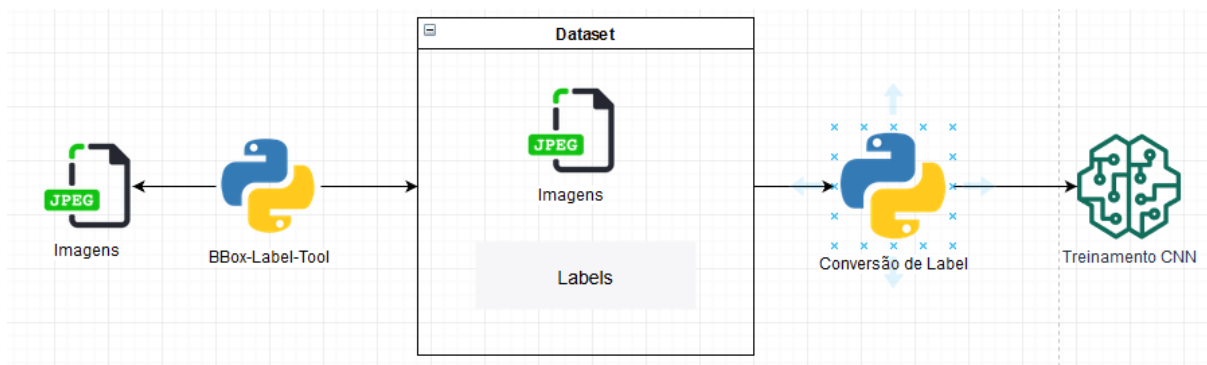
Os arquivos de anotações utilizados pelos diversos algoritmos de detecção de objetos devem possuir informações suficientes para que se identifique a classe do objeto e sua posição. O formato esperado para cada algoritmo é um detalhe de implementação, que no caso da YOLO V3 é dado pelo padrão:  $c \ x \ y \ w \ h$ , com  $c$  representando a classe do objeto (defeito neste trabalho),  $x$  e  $y$  sendo as coordenadas de centro e  $w$  e  $h$  os valores de largura e altura do objeto a ser detectado, respectivamente.

A ferramenta utilizada para criar as anotações relativas as imagens do *dataset* foi a *BBOX-Label-Tool* ([BBOX-LABEL-TOOL, 2019](#)). As anotações geradas pela ferramenta são definidas por  $x_{min} \ x_{max} \ y_{min} \ y_{max} \ nome\_da\_classe$ , sendo necessária uma conversão para adequação entre os formatos de anotações, de acordo com as regras da Equação 4.1.

$$\begin{cases} x = \frac{x_{min} + x_{max}}{2} * \frac{1}{w} \\ y = \frac{y_{min} + y_{max}}{2} * \frac{1}{h} \\ w = (x_{max} - x_{min}) * \frac{1}{w} \\ h = (y_{max} - y_{min}) * \frac{1}{h} \end{cases} \quad (4.1)$$

O fluxo final criado para possibilitar o treinamento da arquitetura YOLO V3 pode ser visualizado na Figura 33. As imagens obtidas foram rotuladas com o uso da ferramenta *BBOX-Label-Tool*, que dá origem a um primeiro conjunto de imagens e anotações. As anotações foram convertidas para o formato esperado pela YOLO, compondo o *dataset* final que foi utilizado no treinamento do modelo.

Figura 33 – Fluxo de pré-processamento



Fonte:Acervo do Autor.

## 4.2 Experimentos

De forma a avaliar a qualidade de detecção defeitos da YOLO V3 em componentes da rede de distribuição de energia, foram mapeadas as anomalias mais recorrentes encontradas nas linhas de distribuição. Os princípios utilizados para esta escolha foram a possibilidade de identificar estes desvios visualmente, além da frequência com que são encontrados. Com base nessas premissa, foram escolhidos:

1. Cabo fora do Isolador.
2. Cabo fora do Espaçador.
3. Isolador sem Anel de Amarração.

Para melhor compreensão, exemplos destes defeitos podem ser visualizados nas Figuras 34, 35, e 36.

Utilizando a estrutura do Laboratório de Energia, foram simulados os três tipos de defeitos mapeados. Com o registro de diferentes ângulos e distâncias dos defeitos, foi construído um *dataset* contendo 708 imagens, que apresentam exemplos dos defeitos mapeados neste trabalho. Deste conjunto, 600 imagens foram separadas para treinamento, enquanto 108 foram utilizadas na etapa de teste, de forma a verificar a precisão da YOLO V3 em novas imagens de entrada.

Para cada imagem do conjunto, foi criado um arquivo de anotação contendo as informações necessárias para as etapas de treinamento e teste da *CNN*. Esse arquivo contém um número inteiro correspondente a classe do defeito contido naquela imagem, além dos valores  $x$ ,  $y$ ,  $w$  e  $h$ , que representam respectivamente as coordenadas, largura e altura da área onde se encontra o defeito rotulado.

Todos os testes foram executados em um computador com 16GB de RAM, um processador Pentium G4560 com 4 threads, sendo dois núcleos reais e dois simulados,



Figura 34 – Cabo fora do Isolador



Fonte:Acervo do Autor.

além de uma *GPU* GeForce<sup>®</sup> GTX 1050TI. A implementação foi desenvolvida em C++, utilizando o framework Darknet (REDMON; FARHADI, 2019), enquanto as aplicações utilizadas no pré-processamento (redimensionamento, criação de rótulos, renomeação de imagens) foram desenvolvidas em Python e C#.

Os parâmetros utilizados para o treinamento da YOLO V3 podem ser encontrados na Tabela 2. De forma resumida, o critério de parada do treinamento é dado pela ocorrência de 6000 iterações, os pesos aprendíveis (filtros, neurônios simples) das camadas da *CNN* são atualizados a cada 64 imagens processadas, a *GPU* GeForce<sup>®</sup> GTX 1050TI utilizada processa  $batch/subdivisions = 1$  imagens por vez, além de que o fator de aprendizado é multiplicado por 0,1 após as iterações 4800 e 5400.

O andamento etapa de treinamento foi acompanhado pela variação do erro quadrático médio (série azul) em função do número de iterações, conforme o gráfico da Figura 37. Analisando o gráfico, se pode observar que há uma possível convergência aproximadamente a partir da iteração 4200.

Figura 35 – Cabo fora do Espaçador



Fonte:Acervo do Autor.

Tabela 2 – Parâmetros YOLO

Parâmetro	Valor
<i>Max Batches</i>	6000
<i>Batch</i>	64
<i>Subdivisions</i>	64
<i>Steps</i>	[4800,5400]
<i>Scales</i>	[0, 1,0, 1]

Outra métrica acompanhada na etapa de treinamento é o *Mean Average Precision* (série vermelha) (*MAP*) (YUE et al., 2007). A métrica *MAP* é calculada pela médias da *Average Precisions* (*AP's*) de cada classe de objetos a serem detectados.

A Equação 4.2 representa a formulação do cálculo da *AP* para cada classe de objetos que se deseja reconhecer durante o treinamento. Onde *p* é a métrica de precisão e *r* representa o recall, definidos respectivamente pelas Equações 4.3 e 4.4, com *TP* sendo os verdadeiros positivos, *FP* os falsos positivos e *FN* os falsos negativos (COSTA et al., 2019).

Figura 36 – Isolador sem Anel de Amarração



Fonte:Acervo do Autor.

Esta métrica vem sendo bem utilizada em problemas de detecção de objetos ([GIRSHICK et al., 2014](#)), ([WANG et al., 2013](#)), pois leva em consideração tanto o desempenho do classificador em relação as suas próprias detecções, quanto aos resultados esperados para as detecções.

$$AP = \int_0^1 p(r)dr \quad (4.2)$$

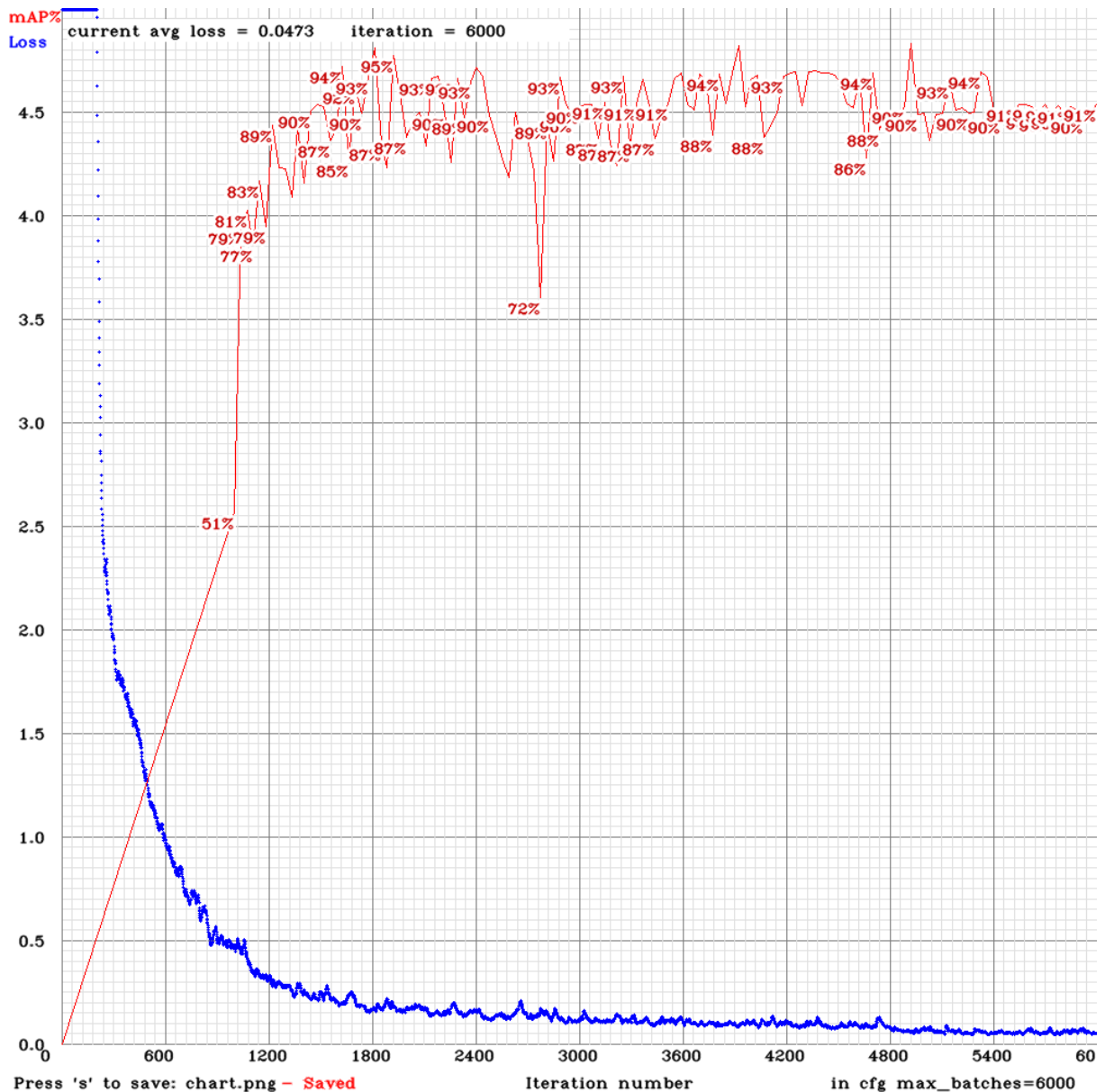
$$p = \frac{TP}{TP + FP} \quad (4.3)$$

$$r = \frac{TP}{TP + FN} \quad (4.4)$$

Realizando a análise da *MAP* do gráfico da etapa de treinamento, se pode observar que há uma região de boa estabilidade e bons valores de *MAP* no intervalo aproximado de iterações [3600, 4500]. A partir da iteração 5400, se observa uma pequena queda no



Figura 37 – Iterações VS Erro



Fonte:Acervo do Autor.

valor do *MAP*, porém acompanhada de uma estabilidade que pode indicar convergência da *CNN*.

De forma a verificar o desempenho da *YOLO V3* treinada, a arquitetura deve ser testada em dados novos, ou seja, que não foram utilizados na etapa de treinamento. As 108 imagens de teste foram avaliadas em três versões do modelo, gerados com os pesos da *CNN* com 2000 (Versão 1), 4000 (Versão 2) e 6000 (Versão 3) iterações. Essa técnica é utilizada para obter o modelo com maior *MAP*, considerando que modelos de aprendizado de máquina treinados por muitas gerações, podem vir a ter seus resultados afetados por *overfitting* (HAWKINS, 2004).

Além de avaliar versões da *CNN* com diferentes parâmetros, a *YOLO V3* também

foi comparada a uma arquitetura mais simples de CNN. Para fins de comparação, essa arquitetura será chamada de **Tiny**. A arquitetura Tiny é composta por 6 tuplas de camadas (convolução, *max pooling*), 6 camadas de convolução e um volume de saída para predição dos *bounding boxes*.

Tabela 3 – Resultados YOLO

-	Versão 1	Versão 2	Versão 3	Tiny
<i>TP</i>	95	<b>103</b>	100	80
<i>FP</i>	4	<b>2</b>	5	8
<i>FN</i>	13	<b>5</b>	8	28
Precisão	96%	<b>98%</b>	95%	91%
Recall	88%	<b>95%</b>	93%	74%
<i>AP</i> cabo fora do espaçador	89,45%	<b>100%</b>	90,12%	86,40%
<i>AP</i> cabo fora do isolador	<b>90,91%</b>	<b>90,91%</b>	89,36%	90,68%
<i>AP</i> isolador sem anel de amarração	97,31%	<b>98,84%</b>	90,68%	70,65%
Média <i>IoU</i>	71,49%	<b>76,03%</b>	75,29%	66,87%
<b><i>MAP</i></b>	<b>92,56%</b>	<b>96,58%</b>	<b>90,05%</b>	<b>82,57%</b>

A Tabela 3 apresenta em detalhes os resultados obtidos por cada versão do modelo. Analisando os resultados, se pode verificar que a Versão 2 da CNN obteve o melhor desempenho, enquanto os piores resultados ficam por conta da arquitetura simplificada **Tiny**, o que comprova melhor desempenho da arquitetura *YOLO*. Apresentando maior precisão e recall, pode-se concluir que as detecções realizadas pelo conjunto de pesos da **Versão 2** foram mais precisas quanto a classes dos respectivos objetos. O valor da métrica *IoU* também apresentou a maior taxa na Versão 2, implicando que as detecções desta configuração apresentam boa correspondência entre área original do objeto e área detectada.

Por fim, as figuras 38 a 39 apresentam exemplos de detecções realizadas pela *CNN* no conjunto de imagens de teste.

Figura 38 – Detecção Cabo Fora do Isolador



Fonte:Acervo do Autor.

Figura 39 – Detecção Cabo Fora do Espaçador



Fonte:Acervo do Autor.



Figura 40 – Detecção Isolador sem Anel de Amarração



Fonte:Acervo do Autor.



## 5 Conclusão

Neste trabalho foi utilizada uma rede neural convolucional para detecção de defeitos em componentes de uma rede de distribuição de energia elétrica. A arquitetura base utilizada foi a YOLO V3, uma *CNN* que apresenta uma boa relação entre velocidade e precisão em suas detecções.

A arquitetura foi testada em um conjunto de dados contendo imagens de diversos componentes da rede, como postes, transformadores, cruzetas e isoladores. O objetivo definido foi a identificação de três classes de defeitos:

1. Cabo fora do Isolador.
2. Cabo fora do Espaçador.
3. Isolador sem anel de amarração.

Utilizando o ambiente do Laboratório de Energia, foram obtidas imagens simulando os três tipos de defeitos mapeados. Dessas imagens, 708 foram rotuladas, criando um conjunto de imagens e anotações que formam o *dataset* utilizado neste trabalho.

O modelo foi treinado utilizando o *dataset* gerado, obtendo precisão de 98% e *MAP* de 96,58% em fase de teste. Os resultados obtidos são consideráveis quando se leva em conta o contexto do trabalho:

- Os componentes mapeados estão em uma escala pequena em relação a imagem total em uma inspeção com uso de drone;
- Os componentes de uma rede de distribuição geralmente estão dispostos em postes junto a outros componentes, os que deforma suas características físicas;
- As imagens de inspeções possuem ruído natural causado pelo plano de fundo da paisagem.

Como contribuições do trabalho, podem ser destacadas o desenvolvimento de um modelo para detecção dos tipos de defeitos mais comuns na rede de distribuição objeto de estudo do trabalho, além da criação e disponibilização de um *dataset* contendo os três tipos de defeitos identificados pela *CNN* (COSTA; CORTES; ALMONDES, ).

É importante destacar que o fluxo do trabalho é flexível, podendo ser aplicável em qualquer rede de distribuição. Existe também a possibilidade de adicionar novos tipos de defeitos, ao custo de realizar novo treinamento na arquitetura.

---

As próximas etapas consistem em testar modificações na arquitetura utilizando meta-heurísticas, categorizar mais modos de falha, além de avaliar o desempenho da *CNN* em plataformas embarcadas.

# Referências

- BBOX-LABEL-TOOL. 2019. Disponível em: <<https://github.com/puzzledqs/BBox-Label-Tool>>. Citado na página 42.
- BEILE, J. 2017. (Introduction to Perceptron: Neural Network). Citado na página 29.
- BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 111–118. Citado na página 34.
- CIRESAN, D. C.; MEIER, U.; SCHMIDHUBER, J. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012. Disponível em: <<http://arxiv.org/abs/1202.2745>>. Citado na página 15.
- COSTA, J. P.; CORTES, O.; ALMONDES, J. d. I. *Defects in Electricity Distribution Components*. <<http://doi.org/10.5281/zenodo.3972451>>. Citado na página 52.
- COSTA, J. P. A. et al. Deep neural networks applied to the dynamic helper system in a gpgpu. In: SPRINGER. *International Conference on Artificial Intelligence and Soft Computing*. [S.l.], 2019. p. 29–38. Citado na página 45.
- ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, Nature Publishing Group, v. 542, n. 7639, p. 115, 2017. Citado na página 36.
- GALLAGHER, C. A. *Infrared hot bearing and hot wheel detector*. [S.l.]: Google Patents, 1995. US Patent 5,448,072. Citado na página 20.
- GALLAGHER, C. A.; PELINO, W. M. *Hot-box detector*. [S.l.]: Google Patents, 1959. US Patent 2,880,309. Citado na página 20.
- GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998. Citado 2 vezes nas páginas 29 e 35.
- GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 37.
- GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587. Citado na página 46.
- HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, ACS Publications, v. 44, n. 1, p. 1–12, 2004. Citado 2 vezes nas páginas 34 e 47.
- HE, W. Load forecasting via deep neural networks. *Procedia Computer Science*, Elsevier BV, v. 122, p. 308–314, 2017. Disponível em: <<https://doi.org/10.1016/j.procs.2017.11.374>>. Citado na página 16.

- KARPATHY, A. Cs231n convolutional neural networks for visual recognition. *Neural networks*, v. 1, 2016. Citado 4 vezes nas páginas 30, 31, 33 e 35.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>>. Citado na página 15.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado 3 vezes nas páginas 16, 33 e 36.
- LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado 2 vezes nas páginas 16 e 36.
- LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37. Citado 2 vezes nas páginas 22 e 41.
- MOCANU, E.; NGUYEN, P. H.; GIBESCU, M. Deep learning for power system data analysis. In: *Big Data Application in Power Systems*. Elsevier, 2018. p. 125–158. Disponível em: <<https://doi.org/10.1016/b978-0-12-811968-6.00007-3>>. Citado na página 16.
- MUZY, G. L. C. de O. *Subestações Elétricas*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2012. Citado na página 21.
- NASRABADI, N. M. Pattern recognition and machine learning. *Journal of electronic imaging*, International Society for Optics and Photonics, v. 16, n. 4, p. 049901, 2007. Citado na página 15.
- NG, A. *Convolutional Neural Networks*. 2019. Disponível em: <<https://www.coursera.org/learn/convolutional-neural-networks>>. Nenhuma citação no texto.
- PINHEIRO, E.; MIRANDA, E.; OLIVEIRA, A. *On the train timetabling problem and heuristics*. [S.l.]: SBPO. Citado na página 15.
- 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*. [S.l.: s.n.], 2016. p. 779–788. Citado 2 vezes nas páginas 38 e 40.
- REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018. Citado 4 vezes nas páginas 22, 37, 38 e 41.
- REDMON, J.; FARHADI, A. *Yolo v3 and Yolo v2 for Windows and Linux*. 2019. Disponível em: <<https://github.com/AlexeyAB/darknet>>. Citado na página 44.
- RUDIN, C. et al. Machine learning for the new york city power grid. *IEEE transactions on pattern analysis and machine intelligence*, v. 34, n. 2, p. 328, 2012. Citado na página 16.
- RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Malaysia; Pearson Education Limited, 2016. Citado 2 vezes nas páginas 28 e 29.

- SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: *Artificial Neural Networks–ICANN 2010*. [S.l.]: Springer, 2010. p. 92–101. Citado na página 35.
- SHOKRI, R.; SHMATIKOV, V. Privacy-preserving deep learning. In: ACM. *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. [S.l.], 2015. p. 1310–1321. Citado na página 33.
- SILVA VINICIUS FERREIRA VIDAL, M. F. d. S. M. F. d. S. A. L. d. C. A. S. C. L. d. M. H. Luiz Augusto Zillmann da. Reconhecimento de componentes em linhas ferreas utilizando redes neurais convolucionais. *Congresso Brasileiro de Automatica*, 2018. Citado na página 16.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 16.
- SVOZIL, D.; KVASNICKA, V.; POSPICHAL, J. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, Elsevier, v. 39, n. 1, p. 43–62, 1997. Citado na página 29.
- SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9. Citado na página 36.
- TERRESTRES, A. N. de T. *Evolução do Transporte Ferroviário de Cargas*. 2012. <<http://portal.antt.gov.br/index.php/content/view/4751/Ferroviaria.html>>. Acessado: 10/2019. Citado na página 19.
- TETKO, I. V.; LIVINGSTONE, D. J.; LUIK, A. I. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of chemical information and computer sciences*, ACS Publications, v. 35, n. 5, p. 826–833, 1995. Citado na página 31.
- WANG, X. et al. Regionlets for generic object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2013. p. 17–24. Citado na página 46.
- YUE, Y. et al. A support vector method for optimizing average precision. In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.: s.n.], 2007. p. 271–278. Citado na página 45.