



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Eduardo Camilo do Canto

**Software com inteligência artificial para detecção automática de manchas na
imagem de câmeras durante processo produtivo**

Florianópolis
2022

Eduardo Camilo do Canto

**Software com inteligência artificial para detecção automática de manchas na
imagem de câmeras durante processo produtivo**

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.

Orientador: Prof. Marcelo De Lellis Costa de Oliveira,
Dr.

Supervisor: Luiz Henrique Oro do Nascimento, Eng.

Florianópolis
2022

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.

Orientações em:

<http://portalbu.ufsc.br/ficha>

Eduardo Camilo do Canto

**Software com inteligência artificial para detecção automática de manchas na
imagem de câmeras durante processo produtivo**

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 24 de Março de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Orientador
UFSC/CTC/DAS

Luiz Henrique Oro do Nascimento, Eng.
Supervisor
Empresa Intelbras

Prof. xxxx, Dr.
Avaliador
Instituição xxxx

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado a todos aqueles que me apoiaram em meus sonhos e sempre estiveram ao meu lado.

AGRADECIMENTOS

Primeiramente quero agradecer à minha família, em especial ao meu pai, Jefferson Furmanski do Canto, por ser peça fundamental na minha escolha de cursar Engenharia de Controle e Automação, e ao meu tio, Leandro Camilo, por me providenciar estadia na cidade de Florianópolis.

Agradeço a minha noiva, Gabriela de Souza D'avila, por me encorajar a realizar os meus sonhos e sempre estar ao meu lado vivendo-os comigo.

Meus agradecimentos aos meus colegas de Captação de Imagens, na Intelbras, por acreditarem no projeto e por acompanharem e contribuírem com o seu desenvolvimento.

Por fim, agradeço ao meu orientador Marcelo De Lellis Costa de Oliveira por me acompanhar neste trabalho e pela parceria e auxílio também em outros projetos que realizamos juntos durante a minha graduação.

RESUMO

A presença de manchas escuras na imagem é um problema comum entre os fabricantes de câmeras de segurança e ocorre, principalmente, pela presença de partículas entre o sensor de imagem e a lente, ocasionando manchas com perfil acinzentado e circular na imagem. Visando contornar esse tipo de problema, durante a linha de produção ocorre, em mais de um momento, a verificação da presença dessas manchas, a qual é realizada visualmente pelo operador, com uma taxa de erro de 15%. Caso a detecção não seja devidamente realizada na etapa inicial da produção, ela é realizada em etapas posteriores, porém a um custo 5 vezes maior. Diante desse cenário, foi proposto um *software*, a ser implementado em uma etapa inicial da linha de produção, que, através de técnicas de visão computacional e do uso de redes neurais artificiais, é capaz de identificar, de forma automática, a presença de manchas escuras na imagem. Após testes realizados a uma acurácia de 100%, o algoritmo foi implementado na linha de produção e apresentou erro nulo durante a validação. Devido ao alto volume de produção da fábrica onde a solução foi aplicada, calcula-se que o projeto possa gerar uma economia anual para a empresa de mais de R\$ 300.000,00.

Palavras-chave: Câmera de segurança. Detecção de mancha. Software. Redes Neurais.

ABSTRACT

The presence of dark spots in the image is a common problem among security camera manufacturers and occurs mainly due to the presence of particles between the image sensor and the lens, causing spots in the image with a grayish and circular profile. In order to get around this type of problem, the presence of these spots is checked more than once during the production line, which is performed visually by the operator, with an error rate of 15%. If detection is not properly performed in the initial stage of production, it is performed in later stages, but at a cost that can be 5 times higher. Given this scenario, a software was proposed, to be implemented in the initial stage of the production line, which, through computer vision techniques and the use of artificial neural networks, allows the automatic identification of the presence of dark spots in the image. After tests performed at an accuracy of 100%, the algorithm was implemented in the production line and presented a null error during validation. Due to the high production volume of the factory where the solution was applied, it is estimated that the project can generate annual savings for the company of more than R\$300,000.00.

Keywords: Security camera. Stain detection. Software. Neural Networks.

LISTA DE FIGURAS

Figura 1 – Câmera analógica - VHD 1220 D G6	12
Figura 2 – Imagem de câmera com o defeito da mancha escura em formato circular	13
Figura 3 – Imagem de câmera com o defeito da mancha escura em formato alongado	13
Figura 4 – Esquemático de um sistema de CFTV	18
Figura 5 – Câmera analógica do tipo <i>Bullet</i> - VHD 3230 B G6	19
Figura 6 – Câmera analógica do tipo <i>Dome</i> - VHD 1220 D G6 BLACK	19
Figura 7 – Câmera analógica do tipo <i>Speed Dome</i> - VHD 5225 SD IR	20
Figura 8 – Componentes fundamentais de uma câmera de segurança analógica	20
Figura 9 – Relação entre a espessura da lente, campo, e distância de visão	21
Figura 10 – Placa principal de uma câmera de segurança analógica com o sensor de imagem ao centro.	22
Figura 11 – Esquemático do funcionamento do teste de mancha.	25
Figura 12 – Componentes de um neurônio.	26
Figura 13 – Exemplo de topologia de uma Rede Neural.	27
Figura 14 – Topologia <i>Feed Forward</i>	28
Figura 15 – Topologia Recorrente	28
Figura 16 – Topologia Residual	28
Figura 17 – Componentes de uma CNN	29
Figura 18 – Arquitetura de uma ResNet34	30
Figura 19 – Matriz de confusão	33
Figura 20 – Esquemático do algoritmo proposto.	35
Figura 21 – Requisitos funcionais e não-funcionais do sistema.	36
Figura 22 – Diagrama de sequência <i>ClassifyImages</i>	38
Figura 23 – Diagrama de sequência <i>StopInspection</i>	39
Figura 24 – Diagrama de sequência <i>ShowIndicators</i>	39
Figura 25 – Diagrama de classes do software	40
Figura 26 – Imagem com mancha.	41
Figura 27 – Imagem sem mancha.	42
Figura 28 – Imagem com fundo não-branco classificada como inconclusiva	42
Figura 29 – Imagem com elemento crítico classificada como inconclusiva	43
Figura 30 – Imagem com elemento não-crítico classificada como normal	43
Figura 31 – Imagem do DVR sem câmera classificada como troca	44
Figura 32 – Comparativo de acurácia entre ResNet18 e ResNet34 durante o treinamento.	46

Figura 33 – Comparativo de erros entre ResNet18 e ResNet34 durante treinamento.	46
Figura 34 – Treinamento com diferentes resoluções de imagem.	47
Figura 35 – Ajuste fino com diferentes resoluções imagens.	47
Figura 36 – Interface do software desenvolvido.	49
Figura 37 – Estrutura de dados do arquivo XML.	52
Figura 38 – Funcionalidade da geração de relatórios.	54
Figura 39 – Interface do software implementado na fábrica.	55
Figura 40 – Programa da geração de relatórios implementado na fábrica.	55
Figura 41 – Resultados do treinamento do modelo em ResNet18 em forma gráfica.	57
Figura 42 – Resultados do ajuste fino do modelo em ResNet18 em forma gráfica.	58
Figura 43 – Matriz de confusão do conjunto de validação.	59
Figura 44 – Projeto de mecanismo para automação dos testes na montagem final.	62

LISTA DE TABELAS

Tabela 1 – Índices de mancha nas diferentes etapas da produção.	15
Tabela 2 – Tempos para retrabalho de câmeras com mancha.	15
Tabela 3 – Resultados do treinamento do modelo em ResNet18 em forma numérica.	57
Tabela 4 – Resultados do ajuste fino do modelo em ResNet18 em forma numérica. .	58

SUMÁRIO

1	INTRODUÇÃO	12
1.1	PROBLEMATIZAÇÃO	13
1.2	OBJETIVOS	15
1.3	A EMPRESA	16
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	CÂMERAS DE SEGURANÇA ANALÓGICAS	18
2.2	PRODUÇÃO DE CÂMERAS DE SEGURANÇA	23
2.3	TESTE DE MANCHA ESCURA	24
2.4	REDES NEURAIS	25
3	MODELAGEM DO SOFTWARE	34
3.1	SOLUÇÃO PROPOSTA	34
3.2	REQUISITOS DE PROJETO	35
3.3	DIAGRAMAS	37
4	DESENVOLVIMENTO	41
4.1	ALGORITMO DE PREDIÇÃO DE MANCHAS	41
4.2	IMPLEMENTAÇÃO DO SOFTWARE	48
4.3	IMPLEMENTAÇÃO NA FÁBRICA	54
5	RESULTADOS	57
5.1	RESULTADOS DA PREDIÇÃO DE MANCHAS	57
5.2	RESULTADOS DA IMPLEMENTAÇÃO NA FÁBRICA	59
6	CONSIDERAÇÕES FINAIS	61
6.1	IMPLICAÇÕES DO PROJETO	61
6.2	PERSPECTIVAS FUTURAS	62
6.3	CONCLUSÃO	62
	REFERÊNCIAS	64

1 INTRODUÇÃO

No ano de 2021, o mercado de câmeras de segurança foi avaliado em US\$ 26,41 bilhões (WIRE, 2020). Segundo os especialistas, a projeção de valor de mercado desse segmento para 2025 é de US\$ 39,13 bilhões, desempenhando uma taxa de crescimento anual de 8,17% (MARKETS, 2020). Com a relevância desse setor tão em alta, o mercado de câmeras de segurança tem se tornado mais competitivo, com as grandes companhias buscando, cada vez mais, a minimização do seu custo de produção, seja pela obtenção de matéria-prima mais barata ou pelo uso da tecnologia na otimização do processo produtivo.

Uma câmera de segurança (Figura 1) é composta, essencialmente, por uma lente e uma placa principal, a qual é constituída por um sensor de imagem, um processador, um módulo de alimentação e um módulo de saída de vídeo. Opcionalmente, uma câmera pode possuir uma placa LED (*Light-Emitting Diode*), utilizada para visualização noturna.

Figura 1 – Câmera analógica - VHD 1220 D G6



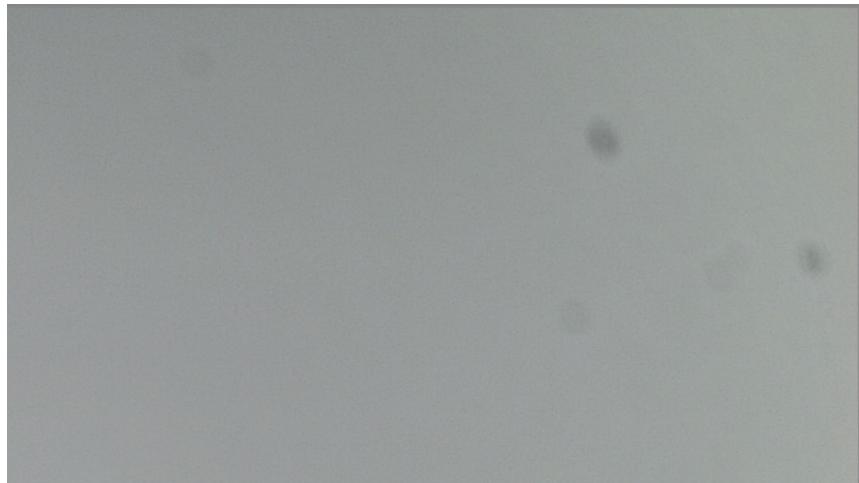
Fonte: www.intelbras.com.br

Durante cada uma das etapas do processo produtivo, a câmera pode estar sujeita a falhas que comprometem a sua integridade, seja por defeito na matéria-prima ou por erro na manufatura. Em particular, um dos problemas mais comuns ocorridos durante a produção de câmeras de segurança é a presença de manchas escuras na imagem.

1.1 PROBLEMATIZAÇÃO

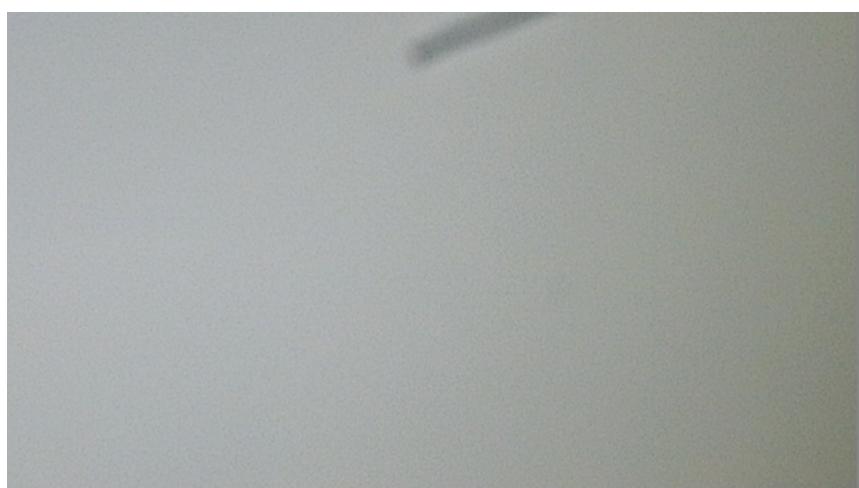
O problema da mancha escura na imagem resulta em marcas em escala de cinza com formatos normalmente arredondados (Figuras 2 e 3), visíveis quando a lente está montada sobre o sensor de imagem.

Figura 2 – Imagem de câmera com o defeito da mancha escura em formato circular



Fonte: Autoria própria

Figura 3 – Imagem de câmera com o defeito da mancha escura em formato alongado



Fonte: Autoria própria

A origem do problema da mancha está relacionada à presença de partículas de poeira no ambiente, as quais se depositam sobre o sensor de imagem tanto durante a confecção do sensor quanto na montagem da lente na placa principal. Por vezes, as manchas são ocasionadas por resíduos do próprio pano utilizado para a limpeza. Além

disso, existe um fenômeno chamado de mancha de transporte, onde partículas que estavam depositadas na cavidade interna da lente podem cair sobre o sensor durante o transporte da câmera.

De maneira simplificada, o processo produtivo pode ser dividido em três etapas. Na primeira etapa é realizada a montagem dos componentes eletrônicos sobre a placa principal nua. Na segunda etapa o sensor é limpo, a lente é fixada sobre ele e, na sequência, o conjunto placa-lente segue para o ajuste de foco, finalizando com a verificação de manchas na imagem. Finalmente, o conjunto é levado para a montagem final, onde é montado dentro do case e novamente testado para verificar se há alguma irregularidade. Nas etapas 2 e 3, caso alguma mancha escura seja detectada, a câmera é separada para limpeza do sensor.

Segundo um levantamento realizado pelo autor deste trabalho, obtido através da análise 500 câmeras, estima-se que 20% dos conjuntos placa-lente que chegam à etapa 2 apresentam mancha. Além disso, visto que, em média, 3% das câmeras que deixam a etapa 2 ainda possuem manchas, pode-se calcular uma acurácia média do operador de 85% em reprovar câmeras com mancha, aprovando, erroneamente, 15% de câmeras com mancha. Na etapa 3, calculou-se um índice de reprovação médio de 4,2% por mancha, resultando em uma estimativa do número de manchas adquiridas durante o transporte calculada pela expressão

$$M_t = \frac{0,042 - 0,03}{1 - 0,03} = 1,24\%. \quad (1)$$

Dessa forma, aproximadamente 4,2% de uma produção diária de 15 mil câmeras são retornadas ao processo para limpeza por causa do problema da mancha escura.

Finalmente, as câmeras separadas para a limpeza do sensor passam por um processo chamado de retrabalho. O tempo médio de retrabalho de um conjunto placa-lente com mancha na segunda etapa de verificação é de, aproximadamente, 30 segundos. Já o tempo de retrabalho da câmera montada na terceira etapa é de cerca de 2,5 minutos. Observa-se que a detecção tardia acarreta em mais tempo de retrabalho, visto que é necessário realizar a desmontagem da câmera por completo. Consequentemente, quanto maior o tempo de retrabalho, maiores são os custos adicionais que poderiam ser evitados, caso a detecção fosse realizada antes.

Apesar dos altos índices de detecção de manchas, devido às múltiplas etapas em que a verificação é feita, um número baixíssimo de câmeras é enviada para o consumidor final com o defeito. Além disso, por depender de um fundo claro para visualização, por vezes as manchas não são perceptíveis em um cenário real de aplicação. Entretanto, mesmo não se caracterizando como um problema de qualidade ao consumidor final, a detecção tardia do defeito possui impacto considerável no tempo de produção e, consequentemente, no seu custo, acarretando em um desperdício anual na casa de centenas de milhares de reais para a empresa.

1.2 OBJETIVOS

Diante da problematização apresentada, o objetivo geral deste projeto é detectar o defeito da mancha escura logo no início do processo, onde o retrabalho é realizado mais rapidamente, acarretando uma diminuição dos custos de produção. Além disso, os objetivos específicos são:

- Analisar uma câmera em menos de 1 segundo e com assertividade acima de 99%;
- Automatizar o controle da produção através do armazenamento de dados em tempo real;
- Agregar mais confiabilidade ao processo e reduzir a subjetividade da identificação de manchas.
- Possibilitar um futuro processo de automatização completa dos teste, com participação mínima do operador.

Considerando-se a produção média de 15 mil câmeras do tipo analógica ao dia, uma redução de 2 minutos no tempo de retrabalho de cada câmera e uma redução de 3% para 0,2% no número de câmeras que saem da etapa 2 com mancha, teria-se uma diminuição de 14 horas por dia de tempo dedicado a retrabalho, conforme as Tabelas 1 e 2. Utilizando-se a estimativa do custo-hora de retrabalho de R\$ 73,60, que considera tanto mão de obra quanto matéria-prima, e levando em conta que a fábrica funciona de forma ininterrupta, espera-se uma economia anual para a empresa de, aproximadamente, R\$ 376 mil.

Tabela 1 – Índices de mancha nas diferentes etapas da produção.

Índice de manchas atual na entrada da 2 ^a etapa	20,0%
Índice de manchas atual na saída da 2 ^a etapa	3,0%
Índice de manchas previsto na saída da 2 ^a etapa	0,2%
Estimativa do índice de manchas por transporte	1,2%
Índice de manchas atual na 3 ^a etapa	4,2%
Índice de manchas previsto na 3 ^a etapa	1,4%

Tabela 2 – Tempos para retrabalho de câmeras com mancha.

Tempo de retrabalho na 2 ^a etapa	30 segundos
Tempo de retrabalho na 3 ^a etapa	150 segundos

1.3 A EMPRESA

A Intelbras é uma empresa brasileira com sede na cidade de São José, Santa Catarina. Fundada no ano de 1976, a empresa é fabricante e distribuidora de produtos eletro-eletrônicos e digitais, gerando com soluções para residências, empresas, condomínios, transportes e projetos especiais.

Atualmente, a empresa é composta por cinco unidades de negócio, cujos principais produtos e soluções são descritos a seguir:

- **Segurança Eletrônica:** Câmeras analógicas, câmeras IP, câmeras Wi-Fi, acessórios para câmeras, videoporteiros, alarmes e cercas elétricas;
- **Comunicação:** Antenas e conversores de TV, centrais telefônicas, telefones, *webcams*, *headsets*, softwares e serviços de *cloud*.
- **Redes:** Modems e roteadores, adaptadores *wireless*, placas de rede, *switches*, softwares e cabos para rede.
- **Controle de acesso:** Fechaduras digitais, interfones, automatizadores e controladores de acesso corporativo;
- **Energia:** Baterias, *nobreaks*, protetores eletrônicos, fontes e dispositivos de automação residencial;
- **Energia Solar:** Luminárias solares, painéis fotovoltaicos, inversores e geradores.

No ano de 2021, a Intelbras detinha 44% de participação de mercado no segmento de segurança eletrônica, 32% no segmento de comunicação, 23% em redes, 22% no setor de controle de acesso, 9,3% em energia e 1,3% de participação em energia solar (VIEIRA, 2021).

Na data de elaboração desse trabalho, a Intelbras possuía cinco unidades de operação no Brasil: a Unidade Matriz e as Filiais São José, Manaus, Minas Gerais e Nordeste. Além disso, a empresa conta com uma unidade na China, que emprega cerca de 60 colaboradores para atuar na área de pesquisas e trazer inovação.

A unidade matriz abriga os setores administrativo, financeiro, de marketing e de pesquisa e desenvolvimento (P&D) de produto. Por sua vez, as filiais são compostas por unidades fabris e centros de distribuição, onde são produzidos e estocados os produtos comercializados pela empresa. Responsáveis pela maior parcela do faturamento da empresa, as câmeras de segurança da Intelbras são produzidas na fábrica de Manaus.

ESTRUTURA

O presente trabalho está organizado em capítulos da seguinte maneira:

- O capítulo 2 aborda o princípio de funcionamento de câmeras de segurança do tipo analógicas e os processos de teste de qualidade em ambiente fabril, com destaque para o teste de mancha. O capítulo é finalizado com a fundamentação teórica de Redes Neurais;
- O capítulo 3 apresenta os requisitos do projeto e os fluxogramas, modelos e diagramas desenvolvidos durante a fase de modelagem do software;
- O capítulo 4 aborda o desenvolvimento do algoritmo de análise de imagens e do software, apontando as decisões tomadas frente aos requisitos e apresentando em detalhes a implementação da solução;
- No capítulo 5 são apresentados os resultados obtidos após a implementação do projeto na unidade fabril;
- Finalmente, o capítulo 6 resume todo o trabalho desenvolvido durante o Projeto de Fim de Curso e discorre acerca de possíveis melhorias no projeto e trabalho futuros.

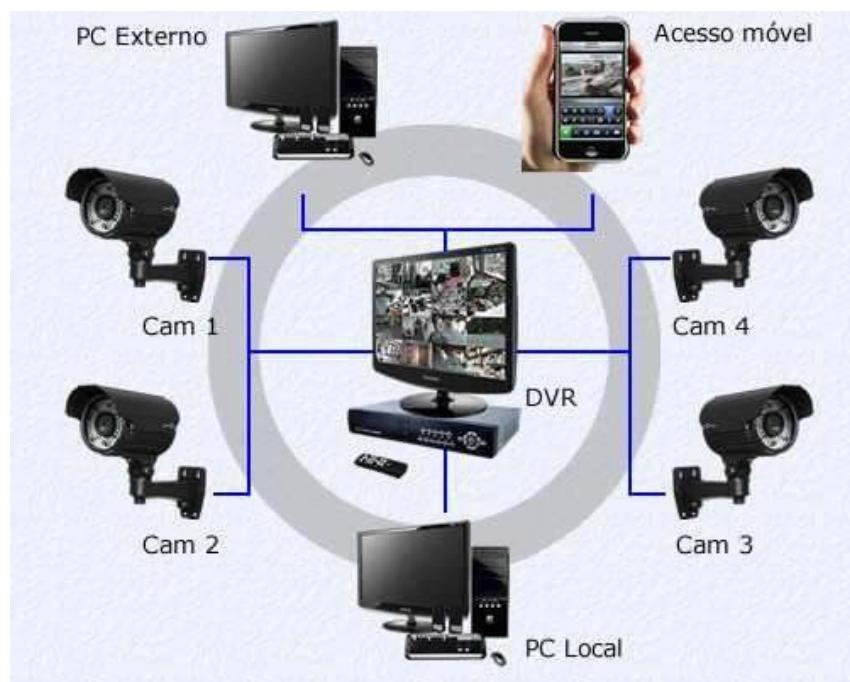
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será abordado aspectos teóricos e as principais tecnologias associadas ao projeto, iniciando com a apresentação dos principais componentes e descrição do funcionamento de uma câmera de segurança analógica; partindo para o detalhamento do processo produtivo das câmeras, abordando os testes realizados durante o processo produtivo, com ênfase no teste de mancha escura; e finalizando com a formalização de Redes Neurais, abordagem de Inteligência Artificial utilizada para a resolução da problematização proposta.

2.1 CÂMERAS DE SEGURANÇA ANALÓGICAS

Câmeras analógicas são dispositivos responsáveis pela captação de imagens e compõem o chamado Circuito Fechado de Televisão (CFTV), que é um sistema composto por câmeras, cabeamento coaxial, *Digital Video Recorder* (DVR) e monitor (PAESSLER, 2021), dispostos conforme a Figura 4. Apesar de outras tecnologias como a IP e Wi-Fi apresentarem maior flexibilidade na instalação, as câmeras analógicas são as mais utilizadas principalmente por conta do custo-benefício.

Figura 4 – Esquemático de um sistema de CFTV



Fonte: www.conforuge.com.br

A tecnologia CFTV foi desenvolvida por cientistas germânicos, em 1942, inicialmente para o monitoramento do lançamento de foguetes. Mais tarde, foi utilizada por

cientistas americanos nos testes com bombas atômicas. Hoje em dia, as principais aplicações de câmeras analógicas em sistemas de CFTV são detecção de atividades suspeitas, desastres e monitoramento de trânsito.

Existem três principais tipos de câmeras analógicas no mercado: câmeras *Bullet* (Figura 5), câmeras *Dome* (Figura 6) e câmeras *Speed Dome* (Figura 7). As primeiras se caracterizam por terem ângulo de abertura menor e serem utilizadas em ambientes externos. Já as segundas, possuem maior ângulo de abertura e normalmente são utilizadas para monitoramento de ambientes internos. Por fim, as últimas caracterizam-se por serem mais robustas e por permitirem ao usuário o seu controle de movimento. As câmeras costumam também serem classificadas pela sua resolução de vídeo, sendo que as principais são *HD* (720p), *Full HD* (1080p) e *4K* (2160p).

Figura 5 – Câmera analógica do tipo *Bullet* - VHD 3230 B G6



Fonte: www.intelbras.com.br

Figura 6 – Câmera analógica do tipo *Dome* - VHD 1220 D G6 BLACK



Fonte: www.intelbras.com.br

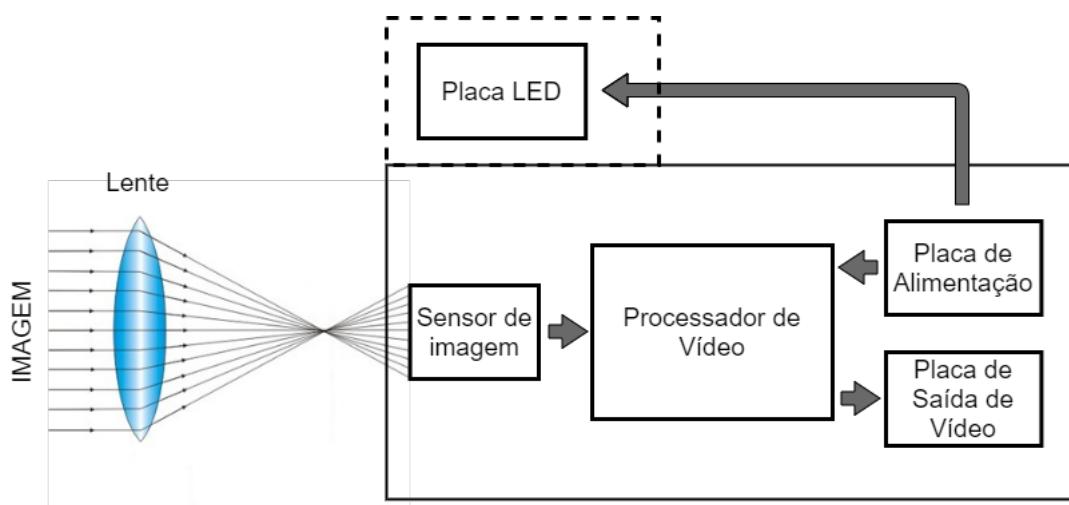
Figura 7 – Câmera analógica do tipo *Speed Dome* - VHD 5225 SD IR



Fonte: www.intelbras.com.br

Apesar da existência de variados tipos de câmeras com características únicas, a maioria delas compartilha as estruturas que serão apresentadas a seguir. A Figura 8 mostra os principais componentes de uma câmera de vídeo.

Figura 8 – Componentes fundamentais de uma câmera de segurança analógica

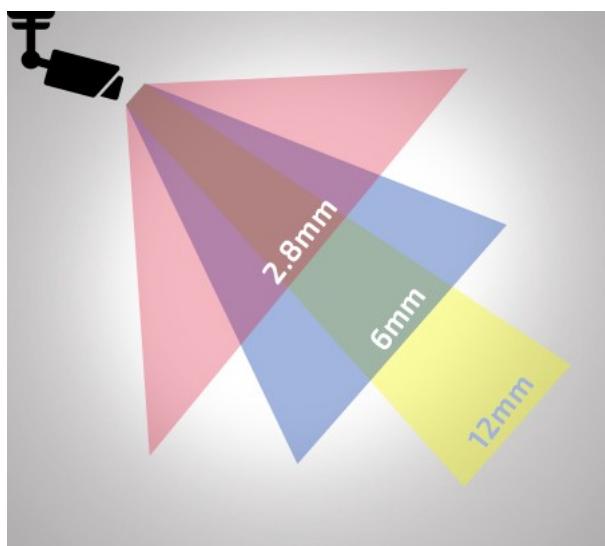


Fonte: Autoria própria

Lentes

A captura da imagem é realizada através de uma lente do tipo convergente utilizando o princípio da refração (BRITANNICA, 2021), a qual visa direcionar os raios luminosos capturados no ambiente para o sensor de imagem. O principal fator que influencia a refração dos sinais luminosos na lente é a sua espessura. Quanto menos espessa for a lente, maior será o seu ângulo de abertura e maior será o campo de visão da câmera, conforme a Figura 9. Do mesmo modo, lentes com maior espessura implicam em um menor campo de visão para a câmera.

Figura 9 – Relação entre a espessura da lente, campo, e distância de visão



Fonte: Torre - Sistemas Inteligentes de Segurança.

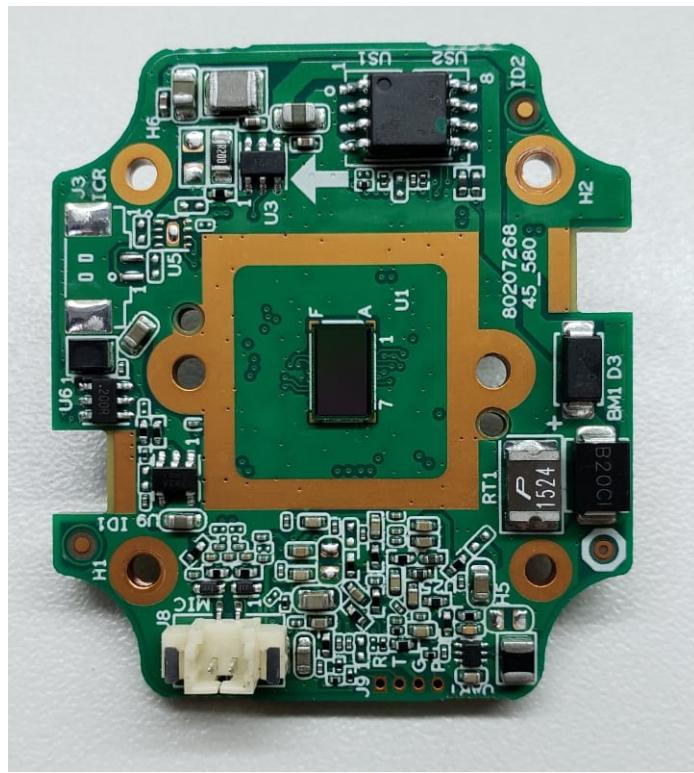
Os valores mais comuns de espessura de lentes encontradas em câmeras são de 2,8mm e 3,6mm. Normalmente, câmeras com lente menos espessas são utilizadas para o monitoramento de ambientes interno, enquanto que as mais espessas costumam ser utilizadas em produtos instalados em ambientes externos.

Sensor de imagem

O sensor de captura de imagem (Figura 10) é responsável por receber os sinais da lente e transformar as informações luminosas em informações elétricas. Os raios de luz são captados através da lente, que fica posicionada sobre o sensor. No mercado, os sensores estão disponíveis em diversos tamanhos, os quais estão diretamente relacionados com a resolução máxima suportada pela imagem da câmera.

Os sensores do tipo CMOS (*Complementary Metal Oxide Semiconductor*), os mais utilizados nas câmeras de segurança da Intelbras, são dispositivos digitais forma-

Figura 10 – Placa principal de uma câmera de segurança analógica com o sensor de imagem ao centro.



Fonte: Autoria própria

dos por uma malha de pixels fotosensíveis, onde a carga de cada pixel é convertida para um sinal de tensão e multiplexada por linha e coluna (WORLDWIDE, 2021).

Processador de Vídeo

O processador de vídeo, também chamado de ISP (*Image Signal Processor*) e DSP (*Digital Signal Processor*), é responsável por traduzir a informação analógica recebida do sensor de imagem e digitalizá-la através de conversores A/D (NAZNEEN, 2020). Além disso, controla outros circuitos da câmera, como o do acionamento dos LEDs.

Placa de alimentação

Visto que uma câmera possui circuitos internos que operam em diferentes tensões, a placa de alimentação, composta, basicamente, por reguladores de tensão, resistores e capacitores, é responsável pelo fornecimento adequado do nível de tensão para cada circuito da câmera.

Placa LED

A função da placa LED (*Light-Emitting Diode*) é possibilitar que a câmera consiga capturar imagens mesmo sem qualquer iluminação presente no ambiente. Existem dois tipos de LED comumente acoplados nas câmeras: LED IR (infravermelho) e o LED branco.

O princípio de funcionamento de uma câmera com LED IR consiste na emissão de luz infravermelha e captação dos raios refletidos no ambiente (SECURITY, 2018). A imagem formada será em preto e branco. Já câmeras com LED branco possuem um sensor de imagem de alta sensibilidade, utilizando a luz branca do LED para iluminar o ambiente e produzir imagens coloridas mesmo em breu total.

A ativação do LED ocorre quando a câmera detecta baixa luminosidade no ambiente, a qual é feita com um fototransistor ou por *software*. De forma que não ocorram sucessivas ativações e desativações por conta do ruído na medição da luminosidade do ambiente, é definido uma histerese e utilizado um controle ON-OFF para coordenar o acionamento do LED.

Placa de Saída de Vídeo

Após o tratamento do sinal pela câmera, a imagem capturada será enviada a um equipamento para visualização, ocorrendo a transformação de sinais elétricos em luminosos. Entretanto, existem padrões seguidos para a correta realização deste procedimento.

A forma com que monitores e televisores fazem a leitura das imagens é através de uma varredura percorrendo cada pixel, linha por linha, onde, ao final do processo, tem-se um quadro. De forma que a comunicação seja efetiva, o sinal elétrico de cada quadro emitido pela câmera deve estar em um formato pré-estabelecido. Esse sinal é composto por três informações, que aparecem nessa ordem: sincronismo horizontal, *burst* de cor e o sinal da imagem, propriamente (KAPATKER, 2020).

2.2 PRODUÇÃO DE CÂMERAS DE SEGURANÇA

Em geral, tanto câmeras analógicas quanto IP e Wi-Fi possuem etapas de montagem semelhantes, sendo que o fator determinante para a definição das etapas utilizadas é o regime de importação.

O regime de importação da matéria-prima das câmeras pode ser CKD (*Completely Knocked Down*) ou SKD (*Semi Knocked Down*). O primeiro caracteriza-se pelo envio do produto completamente desmontado, ou seja, o fornecedor envia os componentes eletrônicos (resistores, capacitores, indutores, processadores...) e componentes mecânicos (parafusos, lente, case...) e o produto é submetido ao processo completo de

montagem. Já o segundo, consiste na aquisição de partes pré-montadas da câmera, como a placa principal com os componentes eletrônicos já inclusos, por exemplo.

O processo produtivo de câmeras de segurança pode ser dividido, para fins de organização, em três etapas, conforme abordado, simplificadamente, na Seção 1.1:

1. **Montagem dos componentes eletrônicos na placa:** Nesta etapa, com a ajuda de um maquinário específico, os componentes eletrônicos são agregados à placa principal nua. Ao fim deste processo tem-se uma placa eletrônica com sensor de imagem, processador de vídeo, circuitos de alimentação e de saída de vídeo. Esse etapa não é realizada em produtos com importação do tipo SKD.
2. **Montagem da lente:** Consiste na limpeza do sensor de imagem e fixação da lente na placa principal. Na sequência, o foco da lente é ajustado e o conjunto placa principal-lente é submetido a testes, como o teste de mancha escura e estouro de branco.
3. **Montagem final:** Alocação do conjunto placa principal com lente dentro do case da câmera, seguido da agregação da placa LED e fixação das estruturas internas. Posteriormente são realizados outros testes, como o teste do LED, e a câmera é embalada.

O detalhamento dos principais testes realizados durante as etapas produtivas descritas, com exceção do teste de mancha escura, que será abordado na Seção 2.3, são:

- **Teste de foco:** Consiste em verificar o foco da câmera para que o operador realize o ajuste manual na lente. A verificação do foco é realizada com o auxílio de um software, que indica o quanto desfocada está a câmera, visando auxiliar o operador durante o ajuste.
- **Teste de estouro de branco:** Consiste em apontar a câmera para uma fonte de luz e analisar a presença de manchas rosadas na imagem, causadas por imperfeições na lente. O teste é realizado de forma integralmente manual.
- **Teste do LED:** Consiste em testar o acionamento do LED submetendo a câmera à baixa luminosidade, momento em que o LED é ligado. A realização do teste é feita de forma manual, colocando-se a câmera em uma caixa escura e observando o acionamento ou não do LED.

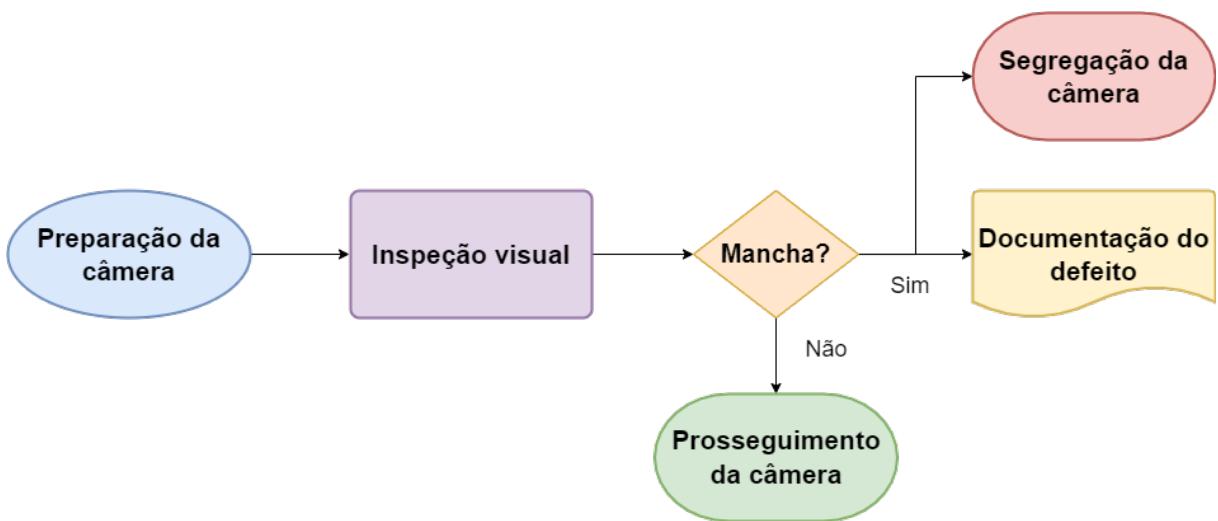
2.3 TESTE DE MANCHA ESCURA

O teste de mancha escura consiste na identificação de manchas circulares (e por vezes cilíndricas) em tons de cinza através da observação da imagem capturada

pela câmera em um monitor de vídeo. Durante o teste, a câmera é apontada para um fundo branco, de forma a melhorar a visualização das manchas, que facilmente podem passar despercebidas em determinados cenários.

O teste tem duração média de 5 segundos, tendo como resultado a aprovação, caso nenhuma mancha seja detectada, ou reprovação, caso alguma mancha seja detectada pelo operador. Todas as câmeras que apresentam mancha na imagem são separadas para limpeza do sensor, ficando sob responsabilidade do operador contabilizar o número de câmeras com o defeito a cada hora, conforme a Figura 11.

Figura 11 – Esquemático do funcionamento do teste de mancha.



Fonte: Autoria própria

Realizado nas etapas 2 (montagem da lente) e 3 (montagem final), o teste de mancha escura é feito integralmente de forma manual, ficando a critério do operador decidir se há ou não mancha com base na sua interpretação da imagem. Conforme apresentado na Seção 1.1, a acurácia do operador na realização do teste é de, aproximadamente, 85%.

2.4 REDES NEURAIS

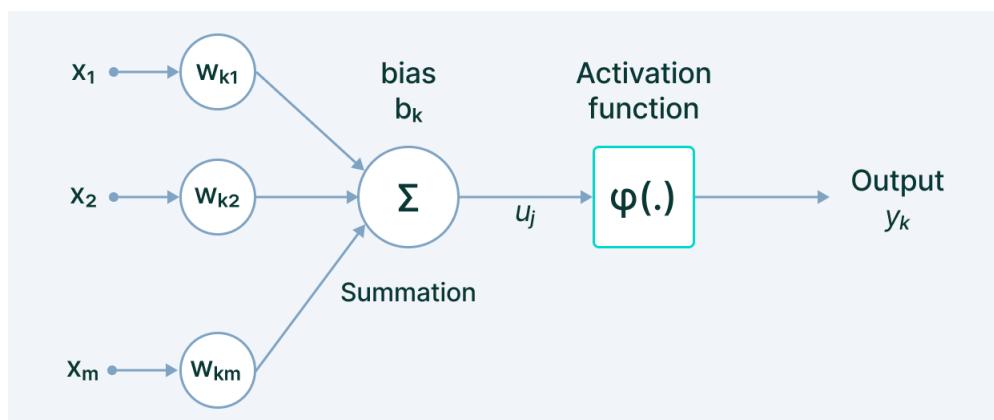
As Redes Neurais foram propostas pela primeira vez no ano de 1943 pelo neurofisiologista Warren McCulloch e o matemático Walter Pitts, os quais desenvolveram circuitos elétricos que, segundo eles, simulariam o comportamento dos neurônios (STRACHNYI, 2019). Em 1949, Donald Hebb propôs um esquema de aprendizado para atualizar as conexões neurais (HEBB, 1949), que seria utilizado por Kruskal-Rosenblatt, em 1958, para a construção de uma rede neural implementada em um IBM® 704 (LEFKOWITZ, 2019), a qual ele mesmo descreveu como “a primeira máquina capaz

de ter uma ideia original". Desde então, um número crescente de estudos na área contribuiu para o desenvolvimento das chamadas Redes Neurais Artificiais, tornando-se uma linha de ferramentas poderosas da Inteligência Artificial. Atualmente, devido aos avanços computacionais, as Redes Neurais são amplamente utilizadas na resolução de problemas complexos baseados em dados (BAHETI, 2021).

Arquitetura de Redes Neurais

Basicamente, os componentes da arquitetura de uma Rede Neural são um conjunto de neurônios (Figura 12), os quais são compostos por entradas, pesos, *bias*, função somatório e função de ativação. Sua inspiração é o cérebro humano e, quando treinada, atua imitando a forma como os neurônios biológicos enviam sinais uns para os outros.

Figura 12 – Componentes de um neurônio.



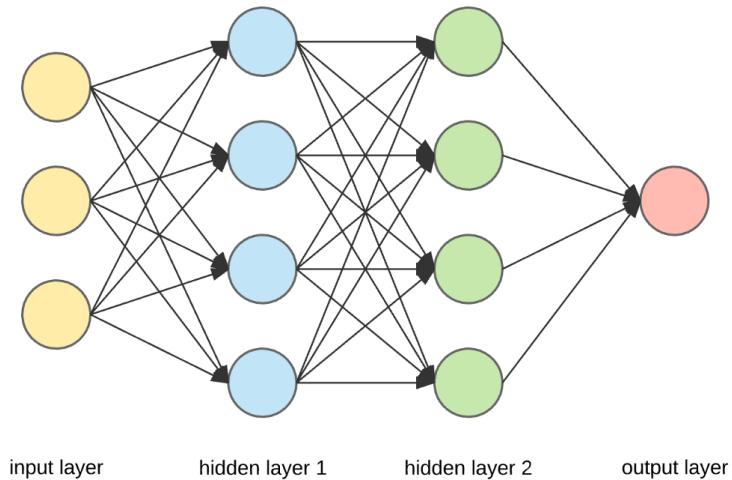
Fonte: www.v7labs.com.

Os sinais de entrada são as percepções do neurônio do meio em que ele está inserido, podendo ser do meio externo, ou de outro neurônio, quando em uma composição de rede. A intensidade com que uma entrada afeta o neurônio é ponderada através dos pesos (também chamados de parâmetros), de forma a dar mais importância aos atributos desejados. Uma entrada em particular é o *bias*, o qual atua como uma entrada de valor 1 e está presente em todos os neurônios artificiais, com o objetivo de transladar a função de ativação. Exemplos de entradas de um neurônio ou de uma Rede Neural são os valores numéricos de cada *pixel* de uma imagem.

Um neurônio artificial é ainda composto por duas outras estruturas: a função de somatório e a função de ativação. A primeira caracteriza-se por combinar múltiplos valores de entrada e convertê-los em um único sinal de saída. Já função de ativação serve para introduzir não-linearidade ao sistema, visto que, sem ela, o neurônio seria apenas uma combinação linear dos valores de entrada.

A combinação de neurônios em coluna forma o que é chamado de camada, a qual pode ser de entrada, oculta ou de saída. A primeira caracteriza-se por receber os dados ou sinais de fontes externas, sendo que o número de neurônios presente nesta camada é determinado pelo número de atributos ou entradas de interesse. Já as camadas ocultas são responsáveis por realizarem todos os cálculos e extrair os recursos dos dados. Por sua vez, a camada de saída recebe os valores obtidos nas camadas ocultas para calcular o valor final de saída. Finalmente, a união de múltiplas camadas de neurônios artificiais forma o que se chama de uma Rede Neural Artificial (Figura 13), ou, de forma simplificada, apenas Rede Neural.

Figura 13 – Exemplo de topologia de uma Rede Neural.

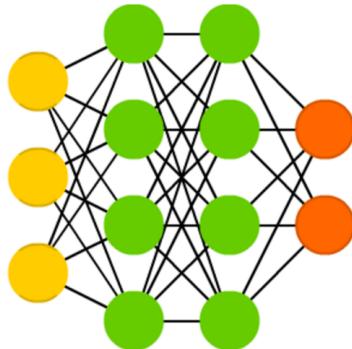


Fonte: www.v7labs.com.

Existe, na literatura, uma grande variedade de topologias, de forma que o fator determinante na escolha da que melhor se aplica é o tipo do problema que se quer resolver. Algumas topologias bem conhecidas (TCH, 2017) são:

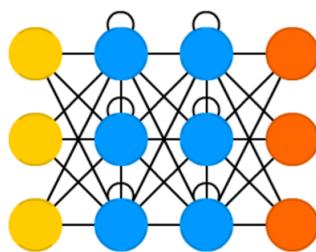
- **Feed Forward:** Podem possuir uma ou mais camadas ocultas, sendo que todos os neurônios estão totalmente conectados (Figura 14). O fluxo de ativação é da entrada para a saída, sem retroalimentação. Aplicadas em problemas de classificação de padrões, identificação de sistemas e otimizações.
- **Recorrente:** Caracterizam-se por possuírem realimentações, fazendo com que as informações persistam (Figura 15). Dessa forma, as informações passadas também são levadas em conta, imitando a memória humana. Devido à essa característica, costumam ser aplicadas em contextos de previsão de séries temporais e controle de processos.

Figura 14 – Topologia Feed Forward



Fonte: www.towardsdatascience.com.

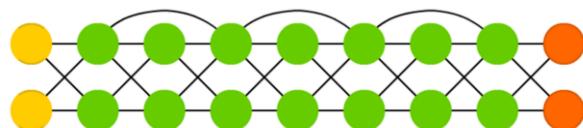
Figura 15 – Topologia Recorrente



Fonte: www.towardsdatascience.com.

- **Residual:** Funcionam de forma similar às Redes *Feed Forward*, entretanto, sua estratégia de pular conexões (*skip connections*) permite a utilização de um grande número de camadas profundas, minimizando problemas numéricos de gradiente e com rápido treinamento e processamento. São amplamente utilizadas no processamento de imagens.

Figura 16 – Topologia Residual



Fonte: www.towardsdatascience.com.

Na área de classificação de imagens, a topologia de rede mais utilizada é a Convolucional Profunda. Essa arquitetura diferencia-se das demais por utilizar técnicas

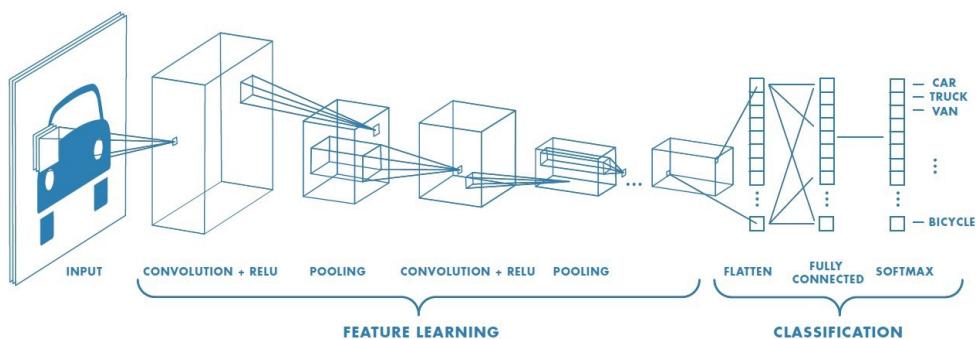
de convolução, permitindo a classificação de imagens utilizando bem menos parâmetros que uma rede do tipo *Feed Forward* e, consequentemente, menos recursos computacionais.

Redes Neurais Convolucionais

Redes Neurais Convolucionais, também chamadas de CNNs (*Convolutional Neural Networks*), foram inicialmente introduzidas na década de 80 por Yann LeCun, um pesquisador em ciência da computação. Em 1998, o pesquisador publicou um artigo introduzindo a LeNet-5, uma CNN capaz de reconhecer *caracteres* (LECUN, 1998). O marco das Redes Neurais Convolucionais ocorreu em 2012, quando uma rede chamada de AlexNet (Krizhevsky, 2012) venceu a *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), uma competição anual renomada de classificação de imagens, atingindo um taxa de erro de 15,3%, contra 26,2% do segundo colocado.

Uma Rede Neural Convolucional é um algoritmo de aprendizado profundo capaz de receber uma imagem como entrada, extrair características de interesse e identificar recursos (SAHA, 2018). Conforme a Figura 17, sua topologia caracteriza-se pela presença de camadas pré-treinadas compostas de matrizes de convolução, as quais atuam como filtros especializados para realçar características de interesse, como bordas e cores; camadas pré-treinadas de *pooling*, que atuam de forma a reduzir a resolução de uma camada convolucional para reduzir a quantidade necessária de processamento; e camadas convencionais de *Feed Forward* para classificação das imagens submetidas às camadas anteriores.

Figura 17 – Componentes de uma CNN



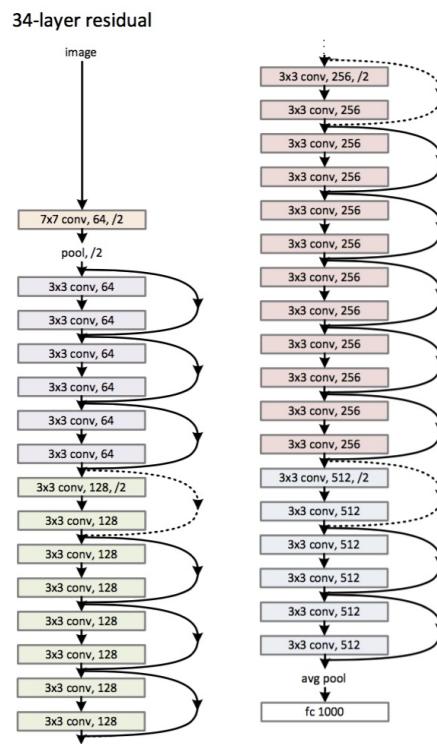
Fonte: [www.towardsdatascience.com.](http://www.towardsdatascience.com/)

Apresentada pela primeira vez em 2015 na ILSVRC e vencendo seus concorrentes com uma taxa de erro de 3,6%, a Rede Neural Residual (ResNet) destacou-se por utilizar uma arquitetura chamada de *identity shortcut connection* (KAIMING HE et al., 2015). Essa técnica, conforme já descrito anteriormente, consiste em pular conexões entre a camadas, de forma a diminuir a carga de processamento e permitir a inclusão

de mais camadas profundas. Dessa forma, a ResNet foi capaz de obter um desempenho melhor que outras redes mesmo com uma quantidade menor de parâmetros. A título de curiosidade, a AlexNet possui um número de 62 milhões de parâmetros para serem treinados, enquanto que uma ResNet de 154 camadas (ResNet154), 60,3 milhões.

A Figura 18 mostra a arquitetura em alto nível de uma ResNet34. Outras variantes da ResNet com 18, 50, 101 e 154 camadas diferem-se entre si apenas pelo número de camadas de convolução.

Figura 18 – Arquitetura de uma ResNet34



Fonte: *Artigo Deep Residual Learning for Image Recognition*

Treinamento de Redes Neurais

Para que seja possível a utilização das Redes Neurais em qualquer aplicação, é imprescindível que seja realizado a etapa de treinamento da rede, a qual consiste no ajuste dos pesos utilizando-se uma parcela do conjunto de dados existentes, chamada de conjunto de treino. Em geral, o objetivo do treinamento de uma Rede Neural é minimizar o erro de predição da saída do sistema, de forma que ele seja capaz de resolver um problema, ou uma classe específica de problemas, com a saída o mais próxima possível do valor esperado. Porém, em alguns casos, o problema nem mesmo

é conhecido, ficando sob a responsabilidade da Rede Neural identificar padrões entre os dados e fazer inferências sobre eles.

O processo de treinamento de uma Rede Neural é chamado de aprendizado e pode acontecer de três formas distintas: supervisionado, não-supervisionado ou por reforço (SATHYA, 2013). O aprendizado supervisionado conta com a presença de um agente externo, o qual classifica os exemplos para o treinamento em classes, de forma que a rede possa avaliar a eficácia do seu aprendizado. Já no aprendizado não-supervisionado, os dados não são classificados previamente e a própria rede encontra padrões e categoriza os dados, de forma heurística. Finalmente, no aprendizado por reforço a Rede Neural aprende com o próprio ambiente por meio de tentativa e erro, em um sistema de penalização e recompensa estabelecidos por um agente.

A abordagem de aprendizado supervisionado caracteriza-se por possuir duas classes de algoritmos: regressão e classificação (PYKES, 2021). No primeiro, o resultado será um valor numérico real e contínuo, quantitativo, como, por exemplo, a previsão do valor de um imóvel. Já no algoritmo de classificação, o resultado será um valor discreto, com o objetivo de determinar a qual categoria pré-definida o objeto em estudo se enquadra. Alguns exemplos de problemas de classificação são determinar se um email é ou não *spam* e identificar se uma imagem possui conteúdo adulto.

Nesse contexto de treinamento de Redes Neurais, encontra-se, dentre outros, os seguintes termos, chamados de hiperparâmetros (RADHAKRISHNAN, 2017):

- **Taxa de aprendizagem:** Define o quanto rápido a rede atualiza os seus parâmetros. Valores baixos tendem a diminuir a velocidade do treinamento, mas garantem convergência suave. Por outro lado, valores muito altos aceleram o treinamento, mas podem não convergir.
- **Batch size:** Número de amostras utilizadas em cada atualização de pesos. Um *Batch size* alto implica em maior tempo de treinamento, maior custo computacional e generalização mais pobre da solução, mas em maior garantia de se encontrar o ótimo global. Já a utilização de um *Batch size* baixo permite treinamentos mais rápidos e com menor custo computacional, mas não garante solução ótima (SHEN, 2018).
- **Número de épocas:** Número de vezes que a rede acessa o conjunto de dados de treino, determinando o número de atualizações dos pesos. Um alto número de épocas pode levar a rede a um *overfitting*, onde o modelo apresenta alta especialização sobre o conjunto de teste e possui generalização pobre. Já um baixo número de épocas pode levar a rede a um *underfitting*, onde o modelo apresenta desempenho ruim já no próprio treinamento, ou seja, não conseguiu aprender o suficiente sobre os dados (BROWNLEE, 2019).

Visto que não há melhores ou piores valores de hiperparâmetros e arquiteturas de rede, durante a etapa de treinamento deve-se testar diferentes combinações de hiperparâmetros e topologias, de forma a obter-se o melhor ajuste possível. Uma rede bem ajustada é aquela que consegue fazer inferências acuradas e precisas sobre um conjunto de dados.

Em particular, as CNNs permitem realizar um ajuste fino dos pesos “descongelando” as camadas pré-treinadas de convolução e *pooling*. Essa técnica é utilizada após o treinamento das camadas de *Feed Forward*, de forma a se obter resultados ainda melhores na rede.

Métricas de avaliação de Redes Neurais

As métricas de avaliação são utilizadas para mensurar o desempenho de uma rede durante e após o treinamento e são fundamentais para identificar se o modelo está ou não bem ajustado. Através da análise desses dados, pode-se verificar, por exemplo, se a topologia escolhida da rede é adequada, se o modelo apresenta *underfitting* ou *overfitting* e avaliar a taxa de aprendizagem.

Durante o treinamento de Redes Neurais, costuma-se utilizar o erro de predição para verificar se a arquitetura utilizada possui complexidade suficiente para modelar o problema proposto e fazer previsões corretas. É comum a comparação do erro do conjunto de treino e erro do conjunto de validação, de forma que baixo erro no primeiro conjunto e alto erro no segundo caracteriza *overfitting*, e baixo erro nos dois conjuntos indica *underfitting*, ambas situações a serem evitadas. Outro uso dessa métrica consiste na avaliação da diminuição do erro ao longo do tempo, onde um decaimento lento ou rápido demais pode significar baixo ou alto valor de taxa de aprendizagem.

Uma métrica de avaliação também bastante utilizada em Redes Neurais é a acurácia, que consiste em mensurar o índice de previsões realizadas corretamente pelo modelo. A acurácia (A) pode ser obtida através da seguinte relação:

$$A = \frac{NPC}{NTP} \quad (2)$$

onde NPC é o número de previsões corretas do modelo e NTP é o número total de previsões realizadas.

Uma ferramenta bem popular que permite uma boa visualização da acurácia de uma Rede Neural de múltiplas classes é a matriz de confusão (Figura 19). Ela consiste em uma tabela cuja dimensão é determinada pelo número de classes do problema. Nessa matriz, os valores contidos na sua diagonal principal são os números de previsões corretas do modelo em cada classe, ao passo que os valores fora da diagonal principal representam o número de previsões erroneamente realizadas em

cada classe. A matriz de confusão é bem útil para se analisar em quais classes o modelo, possivelmente, tem mais dificuldade em realizar previsões corretas.

Figura 19 – Matriz de confusão

		Valor Preditivo	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: www.diegonogare.net.

Uma Rede Neural que faz previsões com 100% de acurácia sobre um conjunto de dados apresentará uma matriz de confusão que contém zeros em todas células fora da diagonal principal. Por outro lado, um modelo com 0% de acurácia apresentará matriz de confusão com diagonal principal igual a zero.

3 MODELAGEM DO SOFTWARE

Neste capítulo serão apresentados os primeiros esboços sobre a solução proposta para o problema, bem como a formalização dos requisitos do projeto, agrupados em funcionais e não-funcionais, desenvolvidos durante a etapa de modelagem. O capítulo é finalizado com a exposição dos diagramas de sequência e do diagrama de classes, os quais foram essenciais para o desenvolvimento do software e implementação do projeto.

3.1 SOLUÇÃO PROPOSTA

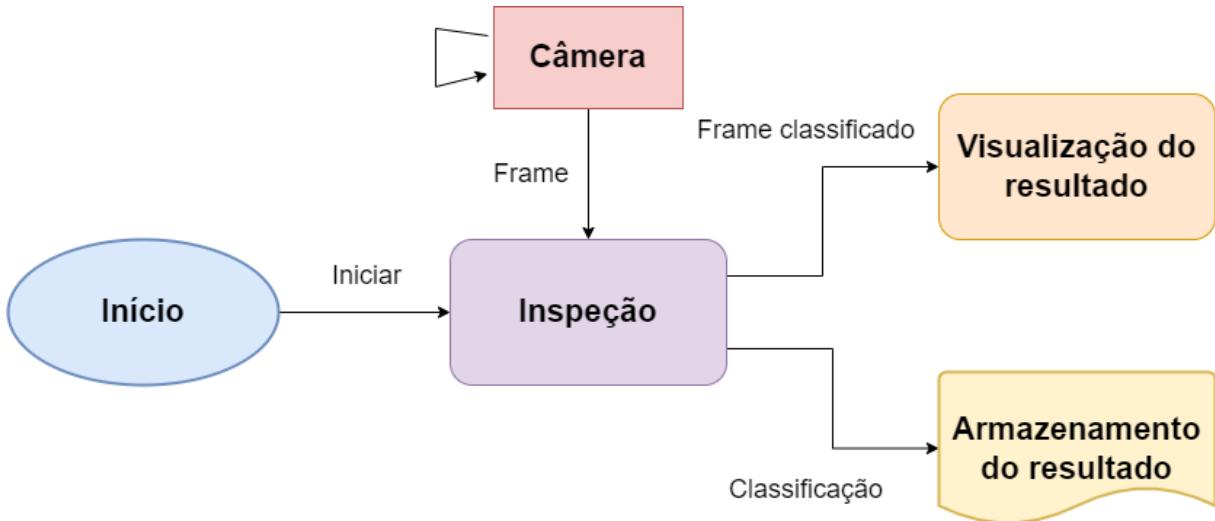
Diante do problema de detecção tardia do defeito da mancha escura em câmeras de segurança, descrito na Seção 1.1, o qual acarreta no aumento médio do tempo de produção e no desperdício de dinheiro, a solução proposta tem como objetivo principal a detecção de manchas, de maneira altamente acurada, logo nas etapas iniciais do processo de montagem da câmera.

Visto que o defeito da mancha ocorre já na primeira etapa de produção das câmeras (Seção 2.2) e que a visualização das manchas na imagem só é possível com a lente acoplada na placa principal, o local escolhido para a aplicação da solução proposta foi na segunda etapa da produção, logo após a montagem da lente e do ajuste de foco, no mesmo setor onde o teste manual é realizado. Neste local já se encontram um computador, um DVR e um monitor, utilizados para ajuste de foco e realização de outros testes, além de um mecanismo para apoio das câmeras.

Levando-se em conta os recursos disponíveis, mas não descartando a possibilidade de adquirir recursos novos, caso fossem necessários, a solução proposta consiste em um software com um algoritmo treinado para identificar as manchas de forma automática, através da imagem da própria câmera. Para tal, o sistema fará a aquisição das imagens em tempo real através da saída HDMI (*High-Definition Multi-media Interface*) do DVR com o auxílio de uma placa de captura USB (*Universal Serial Bus*). A imagem, capturada com a câmera apontada para um fundo branco, para melhor visualização de manchas, é então transmitida para o computador, que conterá o algoritmo que fará o processamento dos dados, conforme a Figura 20.

Dessa forma, o teste automático de mancha proposto passará a substituir o teste realizado de forma manual pelo operador, ficando a cargo deste apenas posicionar a câmera no mecanismo e aguardar a finalização do teste. Em seguida, retira-se a câmera e se dá a ela a finalidade adequada de acordo com o resultado, que poderá ser aprovado ou reprovado.

Figura 20 – Esquemático do algoritmo proposto.



Fonte: Autoria própria

3.2 REQUISITOS DE PROJETO

Com base nos objetivos apontados e levando-se em conta a solução proposta, pode-se elencar os requisitos funcionais e não-funcionais do software do projeto. Os primeiros consistem nas utilidades que o sistema deve ter, enquanto que os segundos compreendem as restrições sobre essas utilidades. Os requisitos podem ainda ser classificados como ocultos, quando são internos ao sistema e inacessíveis ao usuário.

O primeiro requisito funcional do sistema é a capacidade de adquirir imagens da própria câmera submetida ao teste de mancha. O processo de aquisição deverá ser rápido, não ultrapassando o tempo de 100 ms, de forma que a aplicação possa ser executada próxima do tempo real. Outra restrição é que uma nova imagem deverá ser obtida automaticamente assim que a imagem atual terminar de ser analisada, de forma que o usuário não necessite realizar nenhum comando de ativação dessa funcionalidade.

O segundo requisito funcional é o processamento da imagem para identificar se há ou não manchas na imagem, onde aquelas com manchas serão consideradas reprovadas. A análise deverá ser feita em menos de um segundo e com acurácia de pelo menos 99%. O resultado de cada câmera, bem como as informações de horário de análise e tempo despendido, deverão ser armazenadas em um arquivo de registro.

O próximo requisito do sistema é a disponibilização das imagens classificadas ao operador, juntamente com o resultado, em tempo real. A visualização do resultado será por meio de texto (APROVADO ou REPROVADO) e por meio de cores, de forma que regiões com mancha sejam sinalizadas para o operador através de um quadrado

vermelho.

Finalmente, o último requisito funcional do sistema é a geração de relatórios ao usuário com base na leitura dos arquivos de registro obtidos durante as análises. Mediante requisição do usuário, deverá ser disponibilizado o índice de câmeras com mancha no período em questão, bem como, tempo médio de análise e outras informações julgados relevantes para análise do processo.

A Figura 21 mostra todos os requisitos funcionais e não funcionais do sistemas abordados em detalhe.

Figura 21 – Requisitos funcionais e não-funcionais do sistema.

Nome: F1 Detectar manchas escuras na imagem		Oculto (X)				
Descrição: O sistema deve analisar as imagens obtidas, tentando identificar a presença de manchas escuras.						
Requisitos Não Funcionais						
Nome	Restrição	Categoria	Desejável	Permanente		
NF1.1 Tempo de análise	O tempo de análise deverá ser menor do que 1 segundo para cada imagem.	Temporal	()	(X)		
NF1.2 Acurácia do sistema	A taxa de acerto do sistema na identificação da presença de mancha deverá ser superior à 99%.	Performance	()	(X)		
NF1.3 Início da classificação	O processo de detecção será iniciado logo após a aquisição de uma imagem.	Especificação	()	(X)		
NF1.4 Armazenamento da imagem	Mediante opção do operador, as imagens classificadas serão armazenadas para posterior consulta e validação.	Especificação	(X)	()		
Nome: F2 Obter imagens da câmera		Oculto (X)				
Descrição: O sistema deve obter as imagens da câmera em análise.						
Requisitos Não Funcionais						
Nome	Restrição	Categoria	Desejável	Permanente		
NF2.1 Intervalo de obtenção	O sistema obterá uma nova imagem assim que a imagem anterior tiver sido classificada.	Especificação	()	(X)		
Nome: F3 Apresentar imagens com avaliação		Oculto ()				
Descrição: O sistema deve apresentar as imagens obtidas para visualização do usuário.						
Requisitos Não Funcionais						
Nome	Restrição	Categoria	Desejável	Permanente		
NF3.1 Forma de visualização	A classificação da imagem será disponibilizada em forma textual e visual em cor verde (sem mancha) ou vermelho (com mancha), logo após a classificação.	Interface	()	(X)		
Nome: F4 Gerar relatórios		Oculto ()				
Descrição: Disponibilizar a informação da quantidade de câmeras com mancha no período selecionado.						
Requisitos Não Funcionais						
Nome	Restrição	Categoria	Desejável	Permanente		
NF4.1 Disponibilização do relatório	Relatórios disponibilizados mediante requisição.	Especificação	()	(X)		

Fonte: Autoria própria

3.3 DIAGRAMAS

Após a finalização da formalização dos requisitos do sistema, uma boa prática de programação é a elaboração dos diagramas de sequência e do diagrama de classes, os quais serão utilizados na etapa de implementação do software e, possivelmente no futuro, como um guia para a implementação de novas funcionalidades e manutenção do código. A ferramenta utilizada para a elaboração dos diagramas foi o software *Astah Professional*.

Diagramas de sequência

Os diagramas de sequência são utilizados para explicitar a sequência de eventos ocorridos após cada ação do usuário. Sua utilidade é identificar as rotinas que deverão ser implementadas no sistema, bem como a sequência em que elas deverão ocorrer.

A Figura 22 mostra o diagrama *ClassifyImages*, o qual modela o processo de classificação das imagens. Inicialmente, o sistema aguardará a ação do operador para iniciar a operação, que se dará através de um botão em uma interface. Em seguida, o sistema fará a obtenção da imagem da câmera, seguido da classificação da imagem em com mancha, ou sem mancha. Os resultados serão processados e o resultado será exibido de forma visual e textual ao operador. O sistema fará essas ações repetidamente até que o operador sinalize o fim da operação.

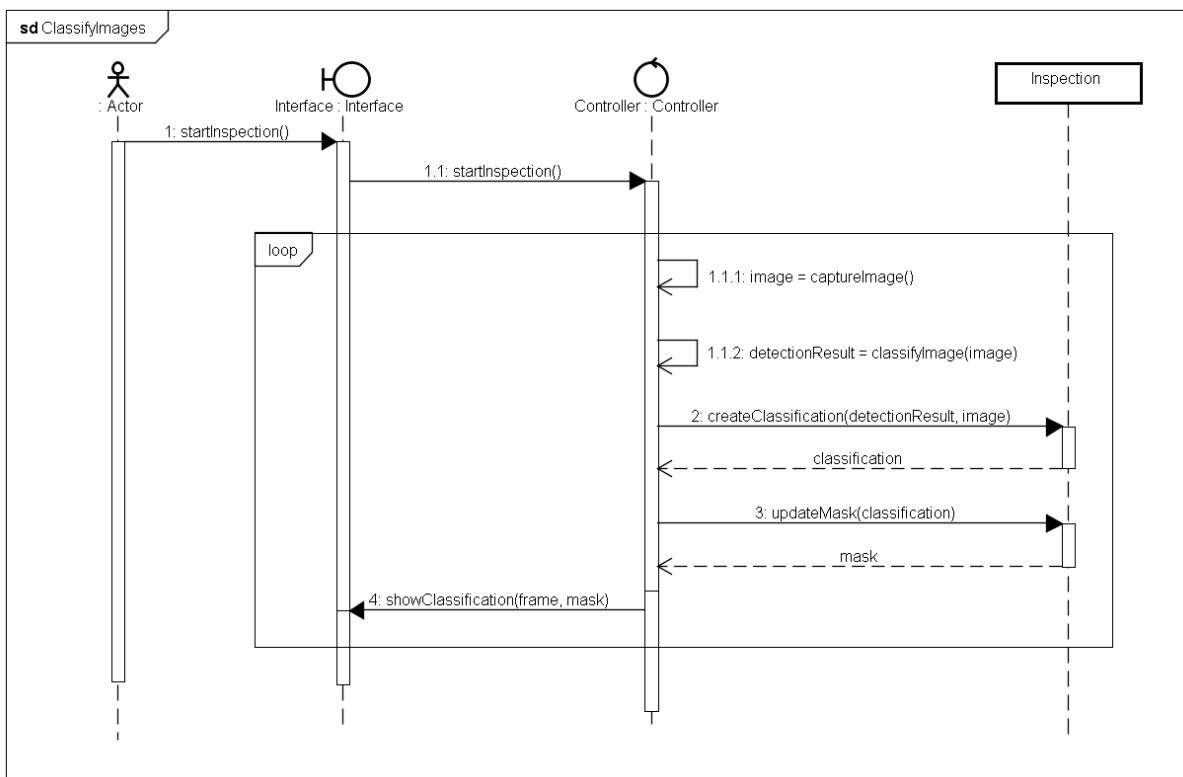
A Figura 23 mostra o diagrama *StopInspection*, responsável pela modelagem do término do processo de inspeção. Através de um botão na interface, o operador sinalizará o fim da inspeção. O sistema, então, salvará todas as informações relevantes obtidas da classificação das imagens para posterior consulta.

Finalmente, o diagrama *ShowIndicators* (Figura 24) modela a geração de relatórios. Após o comando do usuário, o sistema disponibilizará os arquivos de dados para que o usuário selecione de qual deles deseja obter informações. Em seguida, o sistema fará o processamento do arquivo e disponibilizará informações e estatísticas para o operador.

Diagrama de classes

O diagrama de classes consiste no mapeamento da estrutura completa do sistema, visando modelar classes, atributos, operações e suas relações, com base nos requisitos do projeto e nos diagramas de sequência. Futuramente, toda a arquitetura do software será desenvolvida com base no diagrama de classes.

O sistema será composto por três classes principais: *MainWindow*, *Controller* e *Inspection*. A primeira conterá todos os elementos da interface gráfica do programa, como botões, caixas de texto e imagens. Já a segunda classe será responsável pelo

Figura 22 – Diagrama de sequência *ClassifyImages*

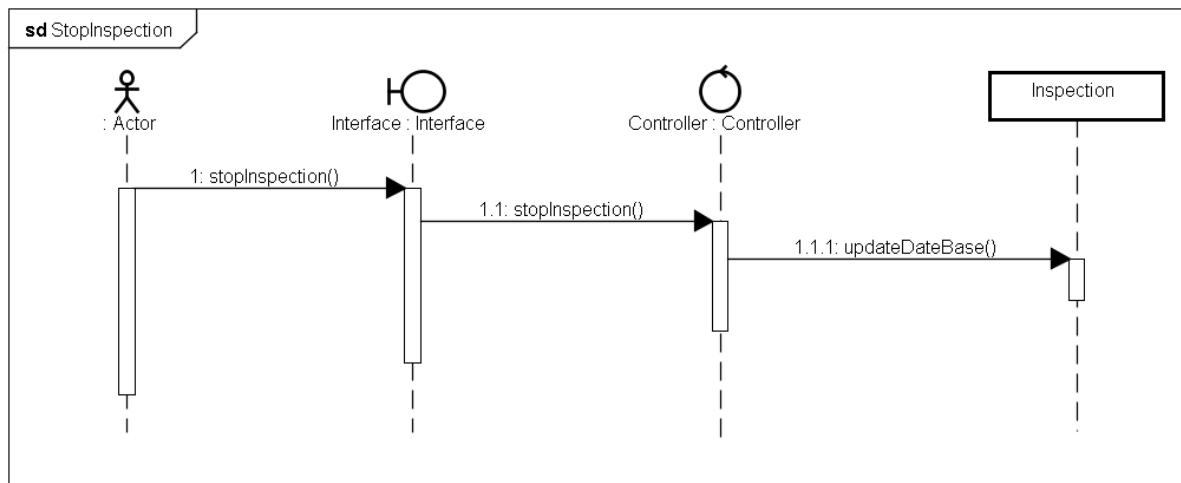
Fonte: Autoria própria

processamento das imagens e tratamento das informações. Por fim, a última classe fará o armazenamento de toda a informação do sistema em tempo de execução.

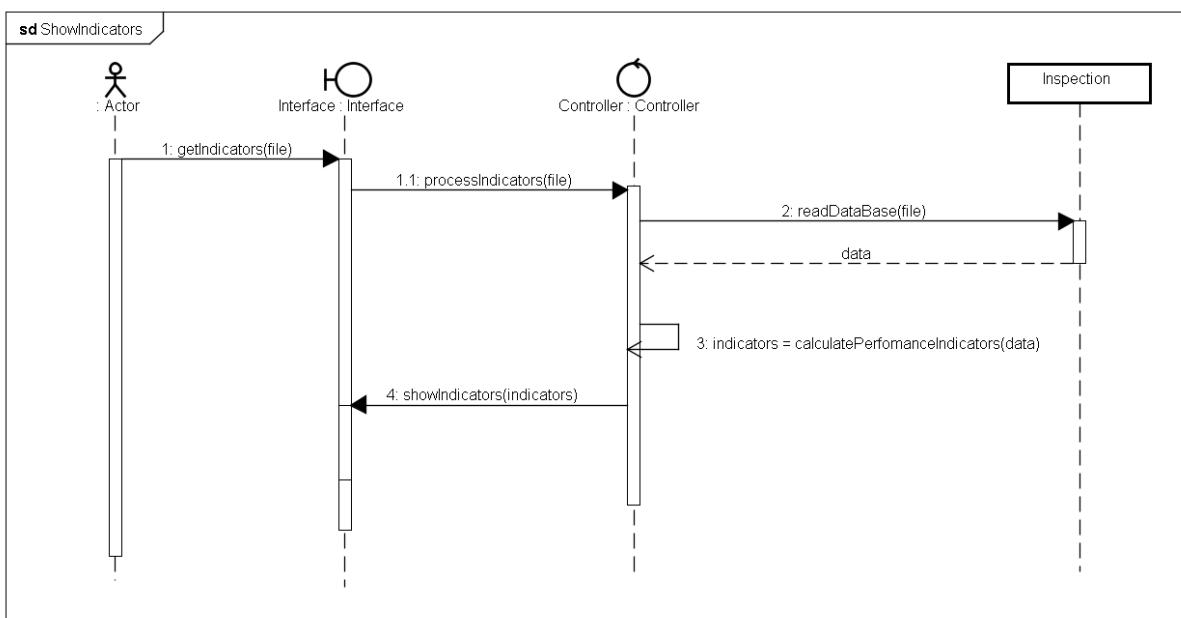
A Figura 25 mostra o diagrama desenvolvido para o sistema. Observa-se a classe *Controller*, representada por um símbolo de controle pelo programa *Astah*; a classe *Inspection*; além de outras classes auxiliares. A classe *MainWindow* é omitida nessa representação.

O software funcionará com uma instância de cada classe principal, sendo que a classe *Inspection* será composta por objetos das classes *Mask* e *Classification*. Essa última contém uma lista de quatro instâncias de *LocalClassification*, conforme será explicado no Capítulo 4. Ainda, o programa contará com objetos das classes auxiliares *ImageTools*, *DetectionResult* e *Location*.

- **Inspection:** Conterá as informações do modelo da câmera, número de pedido, dia e hora de início da operação, o modelo do algoritmo de classificação das imagens e a imagem atual em análise. Além disso, contará com métodos para obter, criar e atualizar objetos de classes relacionadas, bem como, para gerar um arquivo de relatório e interpretá-lo.
- **Classification:** Conterá as informações da análise de uma imagem, como tempo

Figura 23 – Diagrama de sequência *StopInspection*

Fonte: Autoria própria

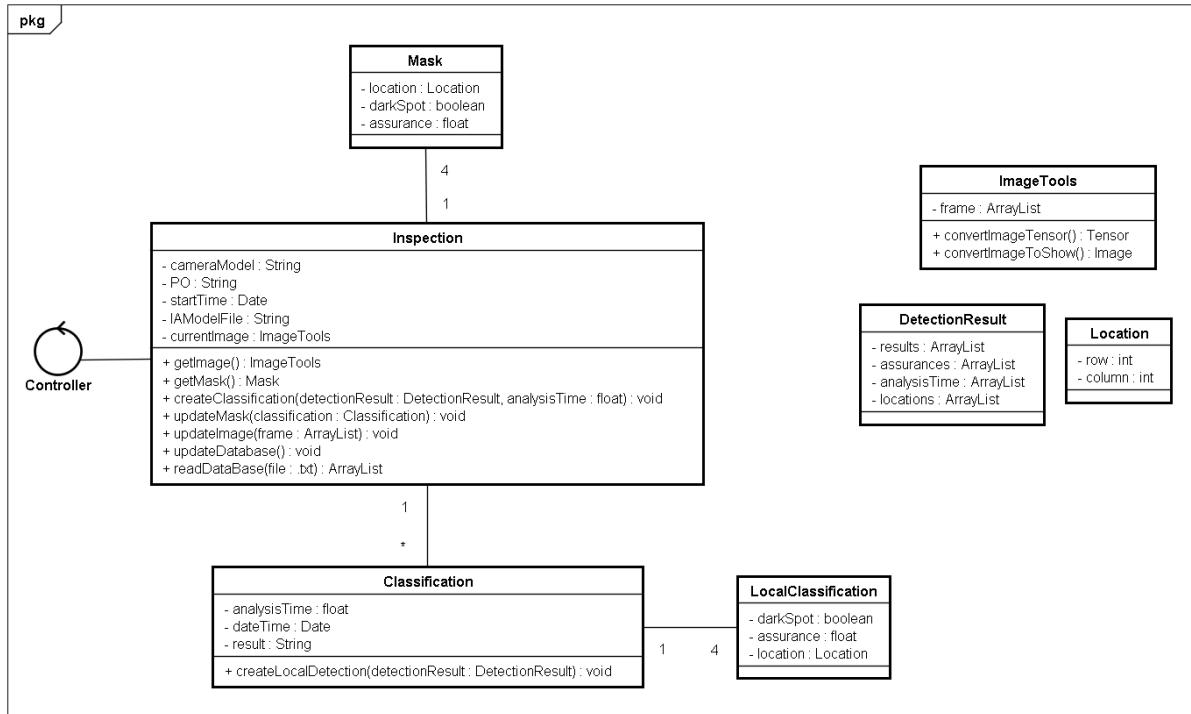
Figura 24 – Diagrama de sequência *ShowIndicators*

Fonte: Autoria própria

total despendido, horário da classificação e resultado. Ainda, conterá um método para criação de detecções locais.

- **LocalClassification:** Carrega informação do resultado do teste de mancha, certeza do algoritmo de predição e localização da região na imagem.

Figura 25 – Diagrama de classes do software



Fonte: Autoria própria

- **Mask:** Obtido através de cada classificação local. Contém os atributos da região de localização na imagem, resultado da classificação e certeza de predição; e será responsável por sobrepor a imagem analisada na forma de bordas coloridas, com cores diferentes para cada resultado.
- **ImageTools:** Classe auxiliar para armazenamento da imagem em sua forma padrão, contendo métodos para conversão em formato para predição e para exibição.
- **DetectionResult:** Classe auxiliar para agrupamento das informações adquiridos pelo modelo de predição logo após a finalização do processo do classificação. Origina as instâncias de *Classification* e *LocalClassification*.
- **Location:** Classe auxiliar para representação das regiões da imagem na forma de coordenadas.

4 DESENVOLVIMENTO

Nesse capítulo será abordado a implementação do sistema, o qual consiste em um algoritmo de predição capaz de realizar a detecção de manchas em imagens, e em um software, responsável pela aquisição, pré-processamento de imagem e apresentação do resultado para o usuário.

4.1 ALGORITMO DE PREDIÇÃO DE MANCHAS

A solução escolhida para a análise das imagens e verificação da presença de manchas foi o uso de uma rede neural convolucional com topologia residual, a *ResNet*. O motivo da escolha se deu por que essa arquitetura é amplamente utilizada em imagens e possibilita um rápido processamento com bom desempenho, um dos requisitos apontados do projeto. Além disso, optou-se por uma abordagem de aprendizado supervisionado, escolhendo-se o algoritmo de classificação, visto que o objetivo será classificar imagens como contendo ou não contendo o defeito da mancha escura.

O processo de definição das classes do problema, obtenção do banco de dados, pré-processamento das imagens, parametrizações realizadas no treinamento, resultados, bem como as decisões de projeto tomadas, serão descritos nessa seção.

Definição das classes

Visando classificar as câmeras como contendo ou não mancha, o modelo seguirá as classes chamadas **mancha** e **normal**. Uma imagem categorizada na classe mancha é aquela que, sobre um fundo branco, apresentar pelo menos uma mancha, conforme a Figura 26. Por outro lado, uma imagem considerada normal é aquela que, sobre um fundo branco, não apresentar nenhum mancha, conforme a Figura 27.

Figura 26 – Imagem com mancha.



Fonte: Autoria própria

Figura 27 – Imagem sem mancha.



Fonte: Autoria própria

Visto que o fundo branco é um pré-requisito para a realização do teste, já que outros elementos na imagem poderiam ocultar possíveis manchas, desenvolveu-se uma terceira classe chamada **inconclusivo**. Imagens caracterizadas como inconclusivas são aquelas que possuem qualquer outro elemento que não seja o fundo branco (Figura 28), inclusive mensagens de aviso opacas do DVR ou informações de data e hora (Figura 29). As exceções são o ponteiro do mouse e pequenas mensagens com fundo não-opaco (Figura 30), visto que não prejudicam a visualização de manchas e costumam aparecer continuamente na tela.

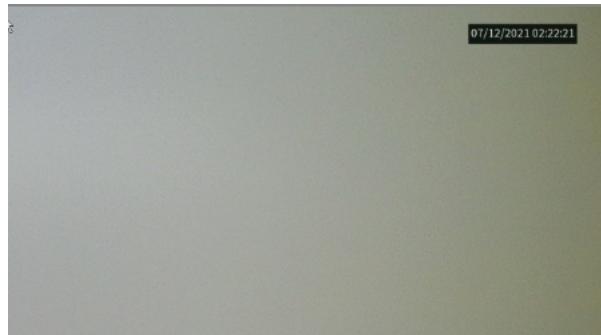
Figura 28 – Imagem com fundo não-branco classificada como **inconclusiva**.



Fonte: Autoria própria

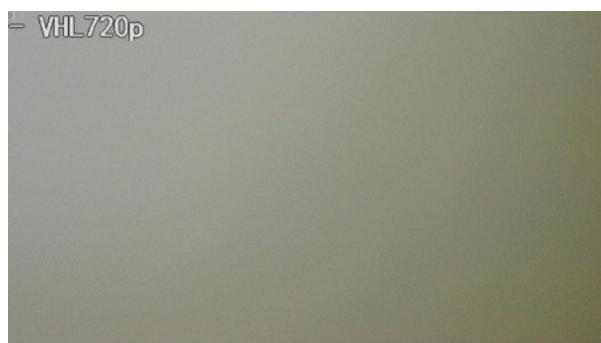
Além das três classes descritas, uma quarta e última classe com o nome de **troca** foi utilizada. Esta classe visa auxiliar a geração de relatórios do sistema, servindo para identificar a troca de uma câmera e, em seguida, contabilizar apenas um resultado **mancha** ou **normal** por câmera. Imagens com classificação **troca** são imagens padrão do DVR de câmera desconectada, apresentando um fundo preto com a logo da Intelbras, conforme a Figura 31.

Figura 29 – Imagem com elemento crítico classificada como **inconclusiva**.



Fonte: Autoria própria

Figura 30 – Imagem com elemento não-crítico classificada como **normal**.



Fonte: Autoria própria

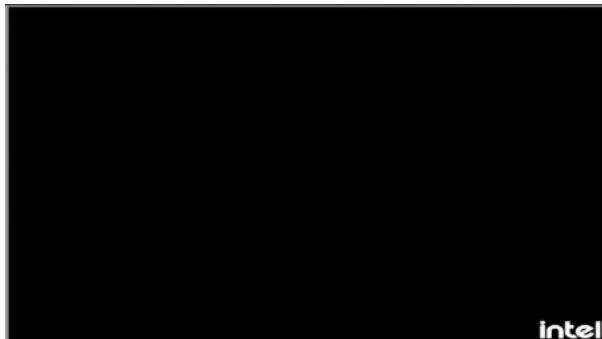
Obtenção do banco de dados de imagens

As imagens utilizadas para o treinamento da rede neural foram obtidas através de diversos modelos de câmeras, visando indicar ao modelo que os parâmetros de imagens de diferentes câmeras não devem ser levados em conta para análise de mancha. Com essa premissa, imagens das quatro classes (**mancha**, **normal**, **inconclusivo** e **troca**) foram obtidas através de uma placa de captura e salvas no formato *.jpg*.

Durante a aquisição de imagens, buscou-se incluir nas imagens de todas as classes elementos ou parâmetros que, por decisão de projeto, deveriam ser ignorados pelo algoritmo de predição no cálculo do resultado, como o *mouse*, mensagens de aviso do DVR com fundo não-opaco, luminosidade do ambiente e proximidade da câmera com a carta branca. Além disso, em particular para a classe **inconclusivo**, tomou-se o cuidado de capturar cenários não-brancos com vários elementos distintos.

Visando evitar que o algoritmo se comporte de forma tendenciosa durante o ajuste dos pesos no treinamento da rede, buscou-se obter o mesmo número de imagens de cada classe, com exceção da classe **troca**, visto que suas amostras são muito

Figura 31 – Imagem do DVR sem câmera classificada como **troca**.



Fonte: Autoria própria

semelhantes. Dessa forma, o número de imagens capturadas durante essa etapa foi de, aproximadamente, 150 imagens.

Pré-processamento das imagens

O pré-processamento das imagens capturadas na etapa anterior consiste no redimensionamento das imagens para uma resolução menor, divisão de uma imagem em quatro partes de mesmo tamanho e espelhamento de cada uma das partes.

Através de testes, foi constatado que o tempo de processamento do algoritmo é diretamente proporcional à resolução da imagem. Dessa forma, a resolução da imagem foi diminuída para que esse tempo fosse menor, mas sem que a acurácia da predição fosse prejudicada. Assim, o valor escolhido para o redimensionamento das imagens, utilizadas tanto no treinamento quanto em futuras aplicações do algoritmo, foi de 896x504 pixels.

Fazendo-se uso dos conceitos de *data augmentation* (GANDHI, 2021), optou-se, por decisão de projeto, dividir as imagens do banco de dados em quatro partes, de forma que o conjunto de treino fosse substancialmente aumentado. Além disso, a decisão vai ao encontro do requisito de projeto de apontar a região de detecção da mancha. Dessa forma, o tamanho das imagens utilizadas no treinamento do modelo (e futuramente também na aplicação) foi de 448x252 pixels.

Por fim, de forma a ensinar o algoritmo que a localização geográfica da mancha não é relevante, cada imagem foi espelhada na vertical, na horizontal e de ambas as formas. Assim, o banco de dados foi aumentando novamente em quatro vezes, resultando em um total de 1500 mil imagens, sendo 468 imagens com mancha, 468 imagens sem mancha, 468 imagens **inconclusivas** e 96 imagens classificadas como **troca**.

Treinamento do modelo

A etapa de treinamento do modelo foi realizada com todas as imagens do banco de dados, sendo 80% do conjunto dedicado, de fato, para o treinamento da rede neural e 20% do conjunto utilizado para validação.

Durante o treinamento, foram testadas duas redes neurais residuais, a *ResNet34* e a *ResNet18*. A primeira caracteriza-se por ser mais poderosa, ao passo que exige maior poder computacional e mais tempo de processamento. Já a *Resnet18* é uma rede composta por menos camadas e, portanto, é menos poderosa. Porém, a sua menor quantidade de parâmetros permite treinamentos e predições mais rápidas, exigindo menor poder computacional. Optou-se por treinar ambas as redes e, posteriormente, comparar o seu desempenho.

Por fim, faz-se necessário o ajuste dos hiperparâmetros para a realização do treinamento. Dentre os valores testados de *batch size*, os melhores resultados foram obtidos com um *batch size* de tamanho 12. Constatou-se, também, que o treinamento da rede neural apresentou melhor desempenho quando utilizado uma taxa de aprendizagem variante (SMITH, 2015), eliminando-se, também, a necessidade de ajuste deste hiperparâmetro. Por último, o número de épocas foi ajustado para que não houvesse *overfitting* nem *underfitting*, ou seja, para que o erro do conjunto de treino e de validação fossem semelhantes e baixos.

Como última etapa do treinamento, um ajuste fino foi realizado de forma a obter-se resultados melhores. Por isso, as camadas iniciais pré-treinadas das *ResNets* foram descongeladas e submetidas também ao ajuste de parâmetros do treinamento.

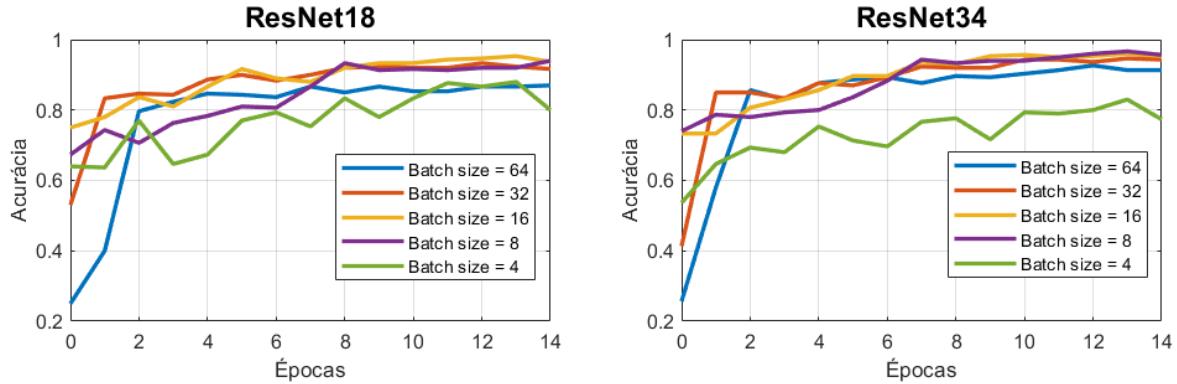
Estudo dos hiperparâmetros

A escolha do melhor conjunto de hiperparâmetros para a resolução do problema proposto, bem como da arquitetura da rede neural, se deu através da realização de diversos treinamentos onde variou-se o *batch size*, o número de camadas da rede e o número de épocas. Em seguida, foi realizado um estudo para identificar até que ponto a resolução da imagem poderia ser diminuída sem que houvesse perdas significativas na acurácia do modelo, de forma a, futuramente, otimizar o tempo de execução do algoritmo.

A Figura 32 mostra a evolução da acurácia do conjunto de validação ao longo das épocas em um comparativo de treinamentos realizados com *batch size* 1, 8, 16, 32 e 64, aplicados em redes com arquitetura de 18 camadas (*ResNet18*) e 34 camadas (*ResNet34*), respectivamente.

Com base nos resultados obtidos, observa-se que a faixa de valores de *batch size* de 8 a 16 leva o sistema a um valor final de acurácia mais alto. Além disso, conclui-se que a *ResNet34*, em geral, apresentou valores de acurácia levemente maiores,

Figura 32 – Comparativo de acurácia entre ResNet18 e ResNet34 durante o treinamento.

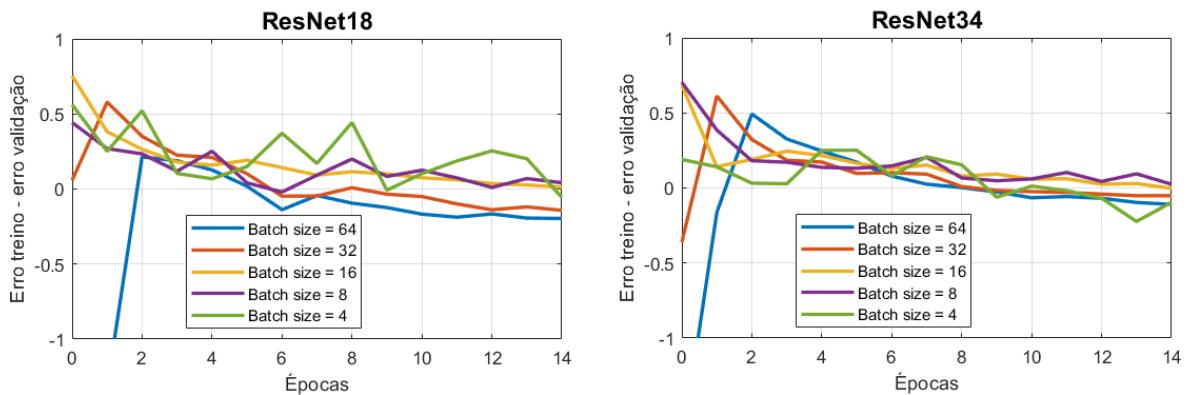


Fonte: Autoria própria

apesar de muito similares aos da *ResNet18*.

A Figura 33 mostra a diferença entre o erro do conjunto de treino e o erro do conjunto de validação, de forma que valores positivos remetem a um erro maior do modelo sobre conjunto de testes do que do conjunto de validação.

Figura 33 – Comparativo de erros entre ResNet18 e ResNet34 durante treinamento.

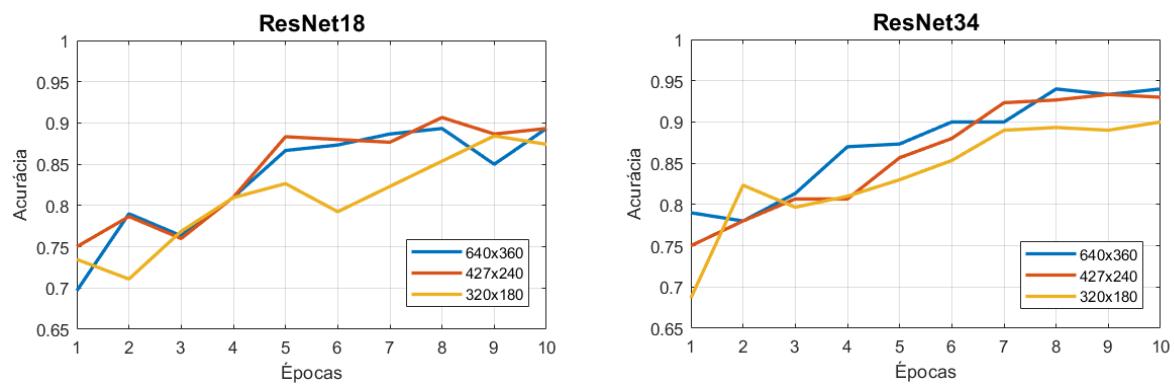


Fonte: Autoria própria

Analizando-se os gráficos, observa-se que para valores de *batch size* na faixa de 4 a 16 nos testes sobre a *ResNet18* a diferença dos erros é próxima zero, indicando que não há *overfitting* no modelo. Já para valores maiores de *batch size*, o erro do conjunto de validação é maior que o de treino e o modelo apresenta indícios de *overfitting*. Analisando-se os resultados da *ResNet34*, observa-se uma menor variação das diferenças dos erros entre os diferentes valores de *batch size*, bem como, uma tendência menor de *overfitting*.

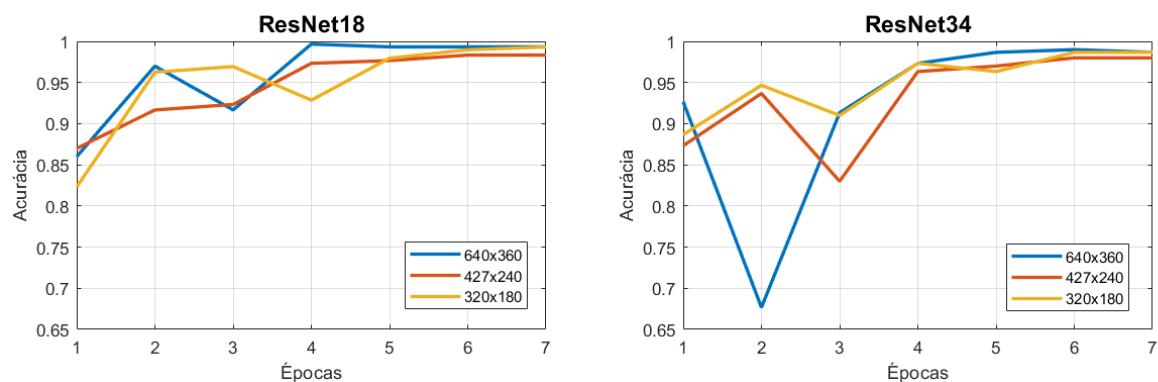
A Figura 34 mostra a evolução da acurácia do conjunto de validação ao longo das épocas em um comparativo de treinamentos realizados com as imagens do banco de dados com resolução 640x360, 427x240 e 320x180 pixels, aplicados na *ResNet18* e na *ResNet34*, respectivamente, com *batch size* igual a 12. Na sequência, a Figura 35 mostra os resultados de cada aplicação após o treinamento das camadas de convolução (ajuste fino).

Figura 34 – Treinamento com diferentes resoluções de imagem.



Fonte: Autoria própria

Figura 35 – Ajuste fino com diferentes resoluções imagens.



Fonte: Autoria própria

Observa-se, pelos gráficos, que a *ResNet34* obtém um melhor desempenho no treinamento sem ajuste fino devido ao seu maior número de camadas de convolução. Entretanto, durante o ajuste dos pesos das camadas de convolução no ajuste fino, a *ResNet18* alcança um desempenho muito similar à *ResNet34*, indicando que a rede de 18 camadas possui capacidade suficiente para classificar, com bom desempenho, as imagens. Além disso, observa-se um melhora util de desempenho de ambas as redes ao se utilizar resoluções mais altas nas imagens de entrada.

Com base nos dados apresentados, optou-se pela utilização da *ResNet18*, visto ter apresentado capacidade suficiente e resultados tão bons quanto a *ResNet34*. Além disso, a escolha da rede com menor número de camadas contribui para um menor tempo de processamento do algoritmo e na prevenção de *overfitting*. Além disso, foi com base nos estudos realizados que as decisões de projeto, já citadas nas seções anteriores, foram tomadas:

- Resolução aplicada às imagens tanto no treinamento quanto na aplicação de 896x504 pixels, processada em 4 partes com 448x252 pixels;
- Valor de *batch size* igual a 12, por apresentar melhor desempenho;
- Taxa de aprendizagem variante, por apresentar melhor desempenho;
- Número de épocas de treinamento igual a 10 e 7 épocas extras para o ajuste fino, para evitar *overfitting*;

4.2 IMPLEMENTAÇÃO DO SOFTWARE

A linguagem de programação utilizada para o desenvolvimento do software foi *Python*, por permitir desenvolvimento rápido, ser compatível com diferentes plataformas e por possuir uma ampla gama de pacotes disponíveis para muitas aplicações, além de ser a linguagem utilizada para a criação do modelo de predição.

As principais bibliotecas utilizadas no código e suas aplicações foram:

- ***PyQt5***: Desenvolvimento da interface gráfica;
- ***openCV***: Pré-processamento de imagens;
- ***fastai***: Realização de predições a partir de um modelo;
- ***xml.dom***: Armazenamento e leitura de dados em formato XML;
- ***cx_Freeze***: Geração de arquivo executável contendo o software.

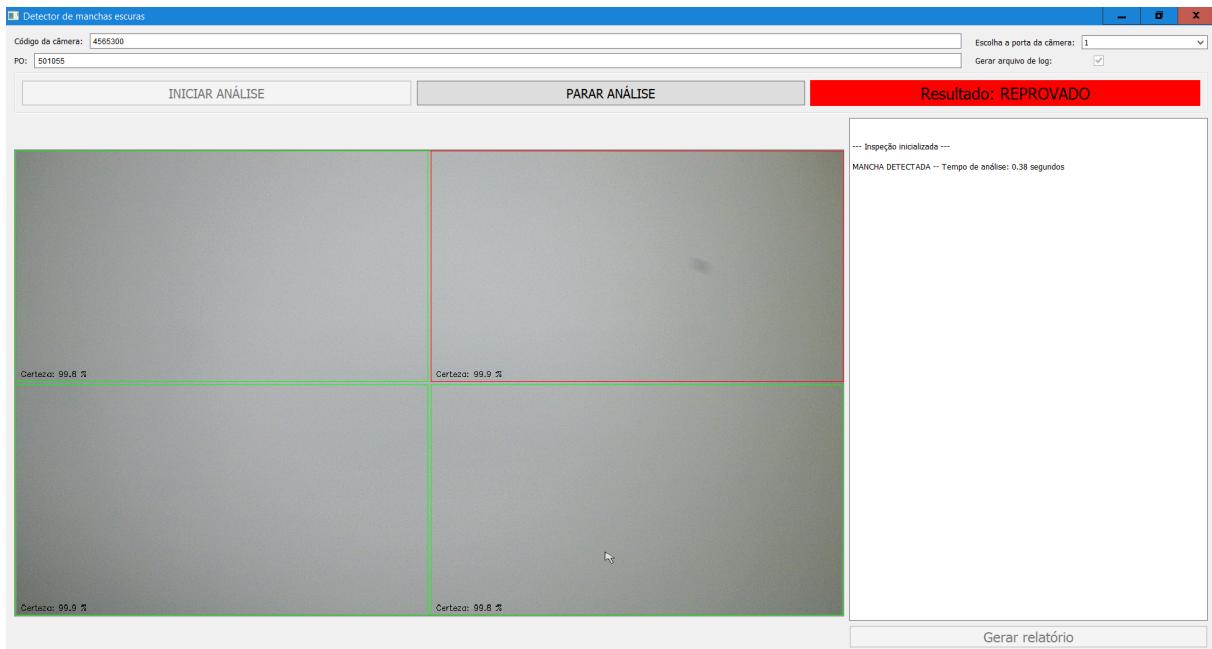
Toda a implementação de software foi realizada seguindo os requisitos de projeto e utilizando-se os diagramas de sequência e de classes, todos apresentados no Capítulo 3.

Interface com o usuário

A interface com o usuário acontece por meio de uma janela composta por campos de texto, botões e outros recursos que permitem ao usuário identificar câmera, definir os parâmetros do processo, iniciar/finalizar o procedimento de inspeção, visualizar os resultados em tempo de execução e solicitar e visualizar relatórios das

inspeções. Com base nas funcionalidades descritas, a interface desenvolvida é exibida na Figura 36.

Figura 36 – Interface do software desenvolvido.



Fonte: Autoria própria

Os campos “Câmera” e “PO” recebem dois códigos com sete e seis dígitos, respectivamente, que servem para mapear a câmera que está sendo analisada. Na Intelbras, toda câmera possui um código único e está atrelada a uma *Product Order* (PO), ou seja, a um pedido de componentes utilizados para a sua montagem. As informações contidas nesses campos serão utilizadas posteriormente para a geração de relatórios.

Quanto aos recursos de parametrização, o campo “Porta da câmera” permite ao usuário escolher o índice da câmera conectada ao computador, seja a própria *webcam* de um *notebook*, uma câmera USB ou uma placa de captura USB que simula uma câmera. Em geral, considerando o sistema operacional *Windows*, o índice será 0 quando houver apenas uma placa de captura em uma aplicação rodando em um computador, e valor 1 em uma aplicação rodando em um *notebook* (a *webcam* assumirá o valor 0). Próximo a esse campo há um *checkbox* “Gerar arquivo de log” que, quando marcado, fará com que o sistema gere um arquivo em XML ao término do processo contendo as informações da inspeção.

Por sua vez os botões “INICIAR ANÁLISE” e “PARAR ANÁLISE” são responsáveis por iniciar e finalizar o processo de inspeção. O botão “INICIAR ANÁLISE” ficará habilitado para ser clicado sempre que uma inspeção não estiver em andamento.

Quando o usuário inicia o processo, o botão de início é desabilitado e o de parada, habilitado.

O resultado de cada classificação da inspeção será exibido através de três formas. A primeira é visual e consiste na aplicação de uma máscara em forma de retângulos sobre a imagem nas cores verde e vermelho, representando, respectivamente, os resultados sem mancha e com mancha. A segunda forma é através de um campo de texto, indicando também o tempo da análise e a certeza da predição. Por fim, a exibição do resultado também se dará através de uma *label* que assumirá os valores “aprovado”, “reprovado”, “inconclusivo” e “troca”, associados às cores verde, vermelho, azul e cinza, respectivamente.

Por fim, no canto inferior direito, há um botão “Gerar relatório”, o qual permite o usuário selecionar um arquivo XML com as informações da inspeção para obter estatísticas sobre a inspeção.

Análise de imagens

O processo de aquisição, pré-processamento e a verificação da imagem é realizado dentro de uma classe própria, chamada de *ImageProcessor*. Visando evitar que a interface trave enquanto o sistema analisa as imagens, foi necessário utilizar uma abordagem de implementação em *threads* (RAMOS, 2021).

Antes do início do processo de classificação, uma conexão com o dispositivo de captura é estabelecida para a aquisição das imagens. Em seguida, o sistema entra em um ciclo, cuja primeira instrução é verificar a disponibilidade da câmera. Caso a câmera tenha sido desconectada, um erro de conexão é emitido; se estiver disponível, o sistema faz a aquisição de um *frame*. Posteriormente, a imagem obtida passa pelas etapas de diminuição da sua resolução, segmentação e conversão para o formato *Tensor*. Finalmente, na última etapa, as imagens segmentadas são submetidas ao algoritmo de predição, que, ao término do processo, emite um sinal contendo os resultados, o tempo da análise e a imagem original capturada.

Visto que em uma implementação sequencial a classe *MainWindow* aguardaria o fim do processamento da classe *ImageProcessor* (o que só aconteceria através de uma parada manual pelo usuário) para que pudesse exibir os resultados, foi necessário realizar o processo de aquisição, processamento e predição de forma paralela, através de uma *thread*, para evitar travamento. Dessa forma, a *thread* comunica-se com o controlador para emitir sinais de erro e resultados de predições, os quais são exibidos em tempo real na interface.

A sintetização da implementação dos processos descritos é apresentada no Algoritmo 1.

```

1 while  $\neg$ parar do
2   camera_disponivel  $\leftarrow$  verifica disponibilidade da câmera;
3   if  $\neg$ camera_disponivel then
4     emite erro de conexão;
5     parar  $\leftarrow$  true;
6   else
7     captura imagem da câmera;
8     pré-processa a imagem capturada;
9     classifica a imagem;
10    emite o resultado;
11  end
12 end

```

Algorithm 1: Processamento de imagens na *thread ImageProcessor*

Controlador

A maior parte do processamento realizado pelo software ocorre dentro da classe *Controller*. Essa classe contém métodos para iniciar/finalizar uma classificação, processar resultados obtidas do algoritmo de predição e processar indicadores de desempenho, cuja implementação segue os diagramas de sequência do Capítulo 3.3.

O início do processo de classificação se dá com a instância de uma *thread* de *ImageProcessor*, a qual realizará a predição sobre as imagens adquiridas de forma continuamente, até que o usuário decida parar o processo.

De forma que o sistema armazene apenas um resultado de interesse por câmera (com mancha ou sem mancha), mesmo que a mesma câmera seja submetida ao teste mais de uma vez, foi necessária a utilização das classes **inconclusivo** e **troca** da seguinte forma:

- O sistema identifica que uma nova câmera foi submetida ao teste ao detectar uma imagem da classe **troca**;
- O sistema armazena o resultado das classes **mancha** ou **normal** apenas uma vez por câmera, após a realização do teste;
- O sistema só voltará a realizar um novo armazenamento de resultado quando uma **troca** for detectada e o primeiro resultado do tipo **mancha** ou **normal** for identificado.

A sintetização da implementação da funcionalidade do armazenamento de apenas um resultado por câmera é apresentada no Algoritmo 2.

Relatórios

Visando o armazenamento das informações das inspeções para posterior consulta e obtenção de estatísticas sobre o processo, o software conta com a funcional-

```

1 if predicao = 'normal' ∨ predicao = 'mancha' then
2   if trocou_camera then
3     trocou_camera ← false;
4     armazena resultado;
5 else if resultado = 'troca' then
6   trocou_camera ← true;
7 end

```

Algorithm 2: Armazenamento de um único resultado por câmera.

lidade de gerar um arquivo de *log* em XML contendo as principais informações da estrutura de dados da classe *Inspection*, apresentada no Capítulo 3.3. A Figura 37 mostra a estrutura de dados do arquivo gerado a partir de uma inspeção e cada uma das *tags* é explicada a seguir:

Figura 37 – Estrutura de dados do arquivo XML.

```

<?xml version="1.0" encoding="utf-8"?>

<inspection>

  <model>4565299</model>

  <PO>500042</PO>

  <start_time>2021-12-21 14:00:09.044115</start_time>

  <stop_time>2021-12-21 14:08:16.066248</stop_time>

  <classifications>

    <classification>

      <analysis_time>0.5954117774963379</analysis_time>

      <date_time>2021-12-21 14:00:18.588241</date_time>

      <result>mancha</result>

      <assurance>0.9338309466838837</assurance>

    </classification>

  </classifications>

</inspection>

```

Fonte: Autoria própria

- **model:** Código do modelo da câmera;

- **PO:** Código do pedido;
- **start_time:** Data e hora do início da inspeção;
- **stop_time:** Data e hora do término da inspeção;
- **classifications:** Lista com os resultados de todas as classificações realizadas durante a inspeção, sendo uma classificação por câmera;
- **analysis_time:** Tempo de processamento de uma classificação, desde a aquisição da imagem, até a emissão do resultado;
- **date_time:** Data e hora da classificação;
- **result:** Resultado da classificação de uma câmera;
- **assurance:** Grau de pertencimento do resultado à classe, cujo valor máximo é 1. Pode ser entendido como o nível de certeza do algoritmo na realização da predição.

Sempre que a inspeção for paralisada ou que o programa seja fechado, o sistema criará uma pasta local chamada *logs* (caso ela não exista) e fará o armazenamento do arquivo em XML. O nome do arquivo será composto pela data e hora de início da inspeção e hora de término da inspeção.

Para visualizar as estatísticas de determinada inspeção, o usuário poderá selecionar um arquivo XML de interesse. O sistema disponibilizará as informações detalhadamente na forma de texto e apresentará o índice de manchas de forma gráfica, conforme a Figura 38.

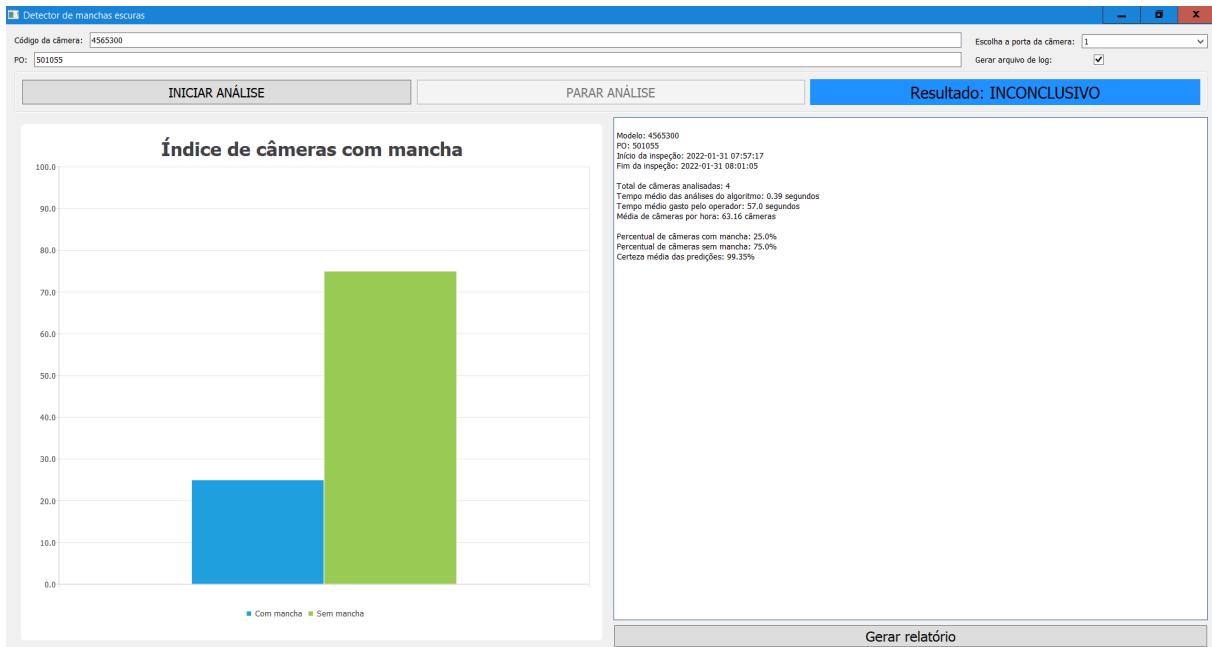
Tratativas de erros

O tratamento de erros e exceções se faz importante para que a aplicação não apresente erros de execução e seja fechada repentinamente. Dessa forma, os eventos que poderiam gerar erros foram identificados e as tratativas para esses erros foram implementadas.

Uma das possíveis fontes de erro na aplicação seria causada ao usuário tentar iniciar uma inspeção sem que a câmera esteja conectada fisicamente ao sistema. Dessa forma, antes do início do processo, é feita uma tentativa de conexão com a câmera. No caso de mal-sucedida, o usuário é requisitado para verificar a conexão com a câmera.

Uma variante desse erro seria ocasionada caso a câmera fosse desconectada durante a inspeção. Nesse caso, cada *frame* obtido é verificado, sendo que a obtenção de um valor *None* indica um problema de conexão com a câmera. Neste caso, a

Figura 38 – Funcionalidade da geração de relatórios.



Fonte: Autoria própria

inspeção é paralisada e o usuário é também solicitado a checar a conexão com a câmera.

Por fim, a tentativa de leitura de um arquivo fora dos padrões especificados de um arquivo de *log* em XML geraria outro erro na aplicação. Para evitar esse tipo de problema, caso algo fora do esperado seja detectado durante a leitura do arquivo, o processo é interrompido e o usuário recebe a mensagem de que o arquivo escolhido é inválido.

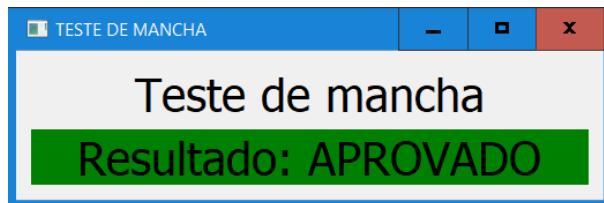
4.3 IMPLEMENTAÇÃO NA FÁBRICA

Após a realização de testes com diferentes modelos de câmera, o sistema tornou-se pronto para ser implementado na fábrica em Manaus. Entretanto, de forma a se adequar a um sistema utilizado no posto de testes, o software precisou ser modificado. Optou-se pela execução do software desenvolvido de forma totalmente paralela ao sistema corrente, de forma que as duas soluções operassem de forma independente.

Visto que o sistema corrente já oferece a visualização na tela do vídeo da câmera, a apresentação das imagens pelo sistema desenvolvido foi removida da versão final. Além disso, os botões de início e parada, o campo de texto e a opção de gerar relatórios também foram retiradas, de forma que restasse apenas a *label* para indicar o resultado ao usuário, conforme a Figura 39. Dessa forma, o programa foi reduzido

a uma janela compactada que poderá ser visualizada e utilizada junto com o sistema corrente.

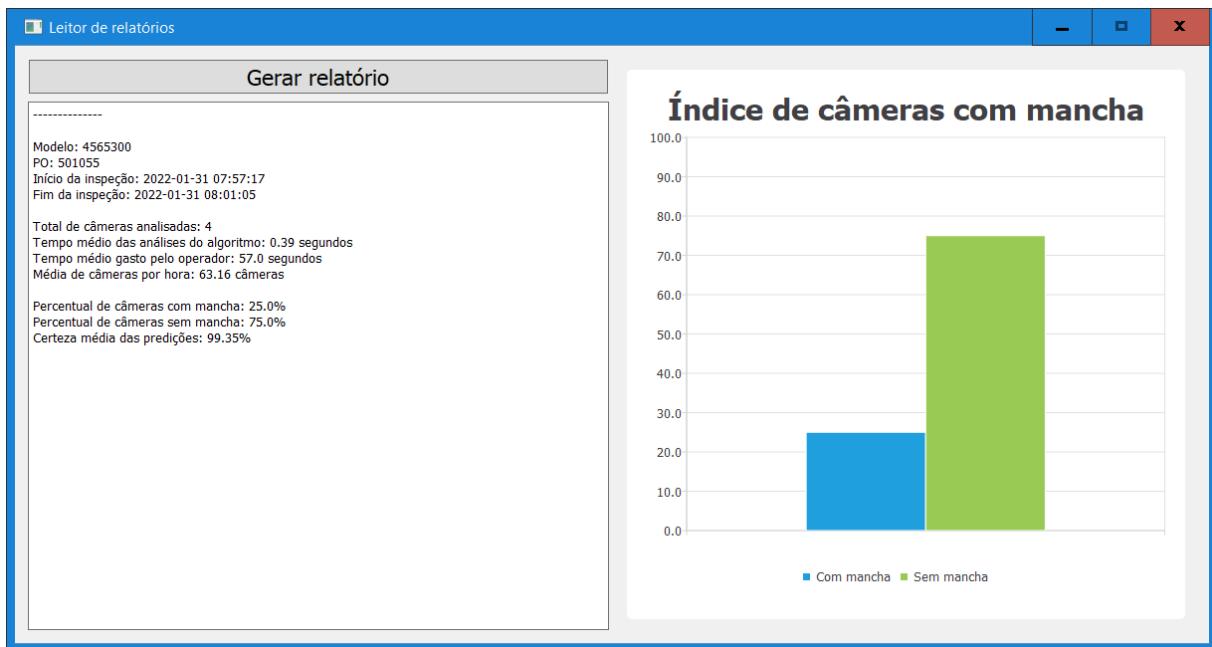
Figura 39 – Interface do software implementado na fábrica.



Fonte: Autoria própria

Em virtude da alta aceitação das funcionalidades de relatório, a criação do arquivo em XML após a finalização do software foi mantida e um novo *script* foi desenvolvido com a finalidade exclusiva de interpretar os arquivos XML e disponibilizar as informações ao usuário de maneira muito similar à implementação inicial. A Figura 40 mostra a interface do programa de geração de relatórios.

Figura 40 – Programa da geração de relatórios implementado na fábrica.



Fonte: Autoria própria

De forma a não exigir a instalação da linguagem *Python* e de todas as bibliotecas utilizadas no desenvolvimento do software, optou-se pela geração de arquivos executáveis, sendo um para o software de análise de imagem e outro para a visuali-

zação de relatórios. Isso é possível pois na criação dos executáveis foram embutidas tanto a linguagem de programação quanto as bibliotecas.

Após a realização das modificações descritas, a solução desenvolvida foi finalmente implementada na linha de produção das câmeras de segurança, ficando sob análise e acompanhamento presencial por duas semanas. Durante esse período, os operadores foram instruídos a comparar o resultado do software com a sua própria avaliação, de forma a verificar a acurácia do sistema durante a sua fase de validação.

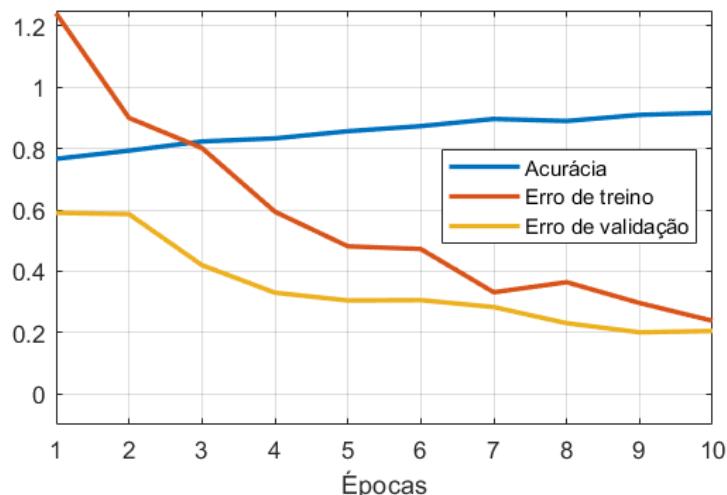
5 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos através da análise das imagens do conjunto de validação, bem como, os resultados alcançados com a implementação do sistema na fábrica.

5.1 RESULTADOS DA PREDIÇÃO DE MANCHAS

A Figura 41 mostra o gráfico do erro de treino dos conjuntos de treino e validação e a acurácia ao longo das épocas, durante o treinamento da versão definitiva do modelo. Os resultados são apresentados de forma numérica na Tabela 3.

Figura 41 – Resultados do treinamento do modelo em ResNet18 em forma gráfica.



Fonte: Autoria própria

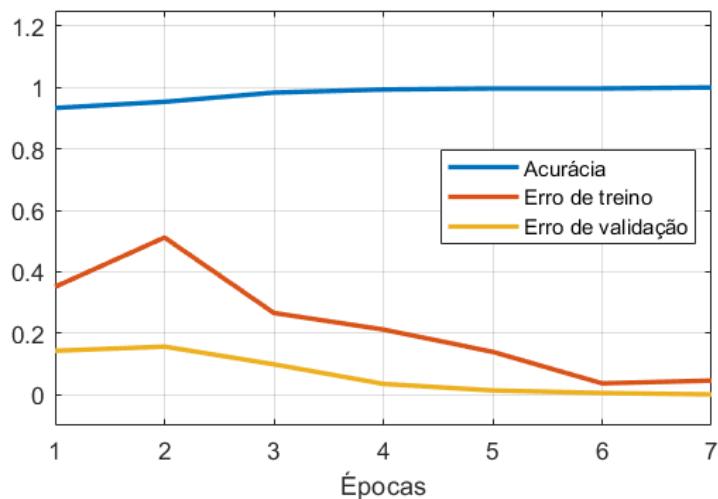
Tabela 3 – Resultados do treinamento do modelo em ResNet18 em forma numérica.

Época	Erro de treino	Erro de validação	Acurácia
1	1.240	0.590	0.766
2	0.900	0.586	0.793
3	0.801	0.402	0.823
4	0.594	0.330	0.833
5	0.481	0.304	0.856
6	0.472	0.306	0.873
7	0.331	0.283	0.896
8	0.364	0.230	0.890
9	0.296	0.201	0.910
10	0.238	0.205	0.916

A análise dos dados permite concluir que a acurácia do modelo sobre o conjunto de validação cresce conforme o andamento do treinamento, mas apresenta sinais de estabilização próximo a 92%. Visto que o erro do conjunto de treino aproxima-se do erro do conjunto de validação, o aumento do número de épocas poderia levar o modelo a um *overfitting*. Dessa forma, encontra-se aqui uma limitação no treinamento do modelo com a configuração atual, motivo pelo qual o modelo é submetido ao ajuste fino.

A Figura 42 mostra o erro de treino dos conjuntos de treino e validação e a acurácia ao longo das épocas durante o ajuste fino do modelo. Os resultados finais são sumarizados na Tabela 4.

Figura 42 – Resultados do ajuste fino do modelo em ResNet18 em forma gráfica.



Fonte: Autoria própria

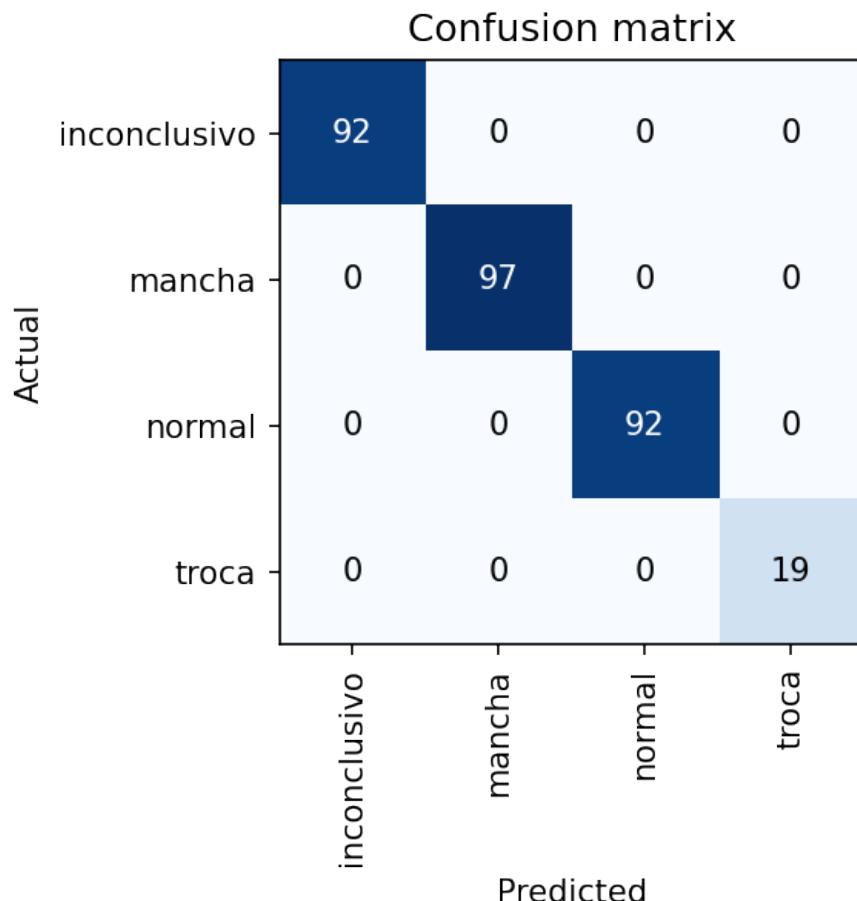
Tabela 4 – Resultados do ajuste fino do modelo em ResNet18 em forma numérica.

Época	Erro de treino	Erro de validação	Acurácia
1	0.351	0.143	0.933
2	0.511	0.156	0.953
3	0.266	0.099	0.983
4	0.212	0.035	0.993
5	0.139	0.014	0.996
6	0.037	0.006	0.996
7	0.046	0.001	1.00

A Figura 43 mostra a matriz de confusão do sistema após a realização do ajuste fino. Com base nos dados apresentados, observa-se que o algoritmo de predição de manchas desenvolvido foi capaz de classificar todas as imagens do conjunto de

validação (300 imagens) corretamente, mesmo utilizando a arquitetura com o menor número de camadas (18) da família das *ResNets*.

Figura 43 – Matriz de confusão do conjunto de validação.



Fonte: Autoria própria

5.2 RESULTADOS DA IMPLEMENTAÇÃO NA FÁBRICA

A implementação na fábrica em Manaus foi realizada em 2 dos 14 postos de verificação de manchas. Durante as duas semanas de acompanhamento, observou-se que o sistema operou da maneira esperada e não apresentou erros de execução. Além disso, durante esse período de acompanhamento, não foram constatadas previsões realizadas de forma errônea pelo programa.

De forma a apresentar os resultados obtidos em números, uma amostragem de 410 câmeras foi realizada, mapeando o processo desde a chegada no posto de verificação de manchas até a segunda verificação de manchas após a montagem final da câmera. O resultado obtido foi de 5 câmeras com mancha no teste da montagem final (etapa 3), o equivalente a 1,22% de índice de mancha. Com base na estimativa do

índice de mancha por transporte calculada em (1), de 1,24%, pode-se considerar que, na amostragem utilizada, todas as manchas foram originadas do próprio processo de transporte . Dessa forma, considera-se que o sistema foi capaz de realizar as predições na linha de produção com erro nulo, com um tempo médio de predição do algoritmo de 0,4 segundos por câmera.

Por fim, outra contribuição valiosa do projeto é referente à obtenção automática de estatísticas do processo. Dessa forma, as informações podem ser coletadas com maior confiabilidade e riqueza de detalhes, servindo, por exemplo, para verificar o tempo médio de ociosidade do operador, para, assim, aumentar a meta do número de câmeras testadas por hora.

Frente à observação do bom desempenho da solução desenvolvida, o sistema será implementado nos 12 postos restantes após a readequação das bancadas. Projetando-se o desempenho observado nas 2 bancadas onde a solução foi implementada e utilizando a base de cálculo realizada na Seção 1.1, espera-se que, após a implementação completa na fábrica, o sistema desenvolvido gere, para a empresa, uma economia anual ainda maior, de R\$ 400 mil.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um software capaz de identificar a presença de manchas escuras na imagem de câmeras de segurança, a ser implementado em linha de produção durante a etapa de testes, visando à redução dos custos de produção.

O projeto foi iniciado com uma viagem à fabrica, em Manaus, para aprendizado detalhado do processo produtivo. Em seguida, buscou-se entendimento do problema da mancha, obtendo-se estatísticas do processo de forma a calcular os ganhos de uma possível implementação de solução. Após a comprovação da viabilidade da ideia e dos ganhos em se desenvolver um sistema para lidar com o problema, o projeto, de fato, iniciou-se.

Buscou-se, desde o começo, documentar cada atividade cumprida e realizar apresentações periódicas para a equipe acerca do andamento do projeto, nas quais foram obtidos *feedbacks* e sugestões que contribuíram para as decisões tomadas durante a fase de desenvolvimento. Além disso, toda implementação em software foi realizada seguindo a modelagem desenvolvida e devidamente documentada, o que, além de ser uma boa prática de programação, torna futuras modificações ou inclusões no software muito mais fáceis.

Por fim, o projeto foi implementado, na prática, ainda que em escala reduzida, na linha de montagem de câmeras de segurança da Intelbras. Após o período de acompanhamento e análise dos resultados obtidos da implementação, constatou-se que todos os objetivos traçados foram cumpridos com êxito.

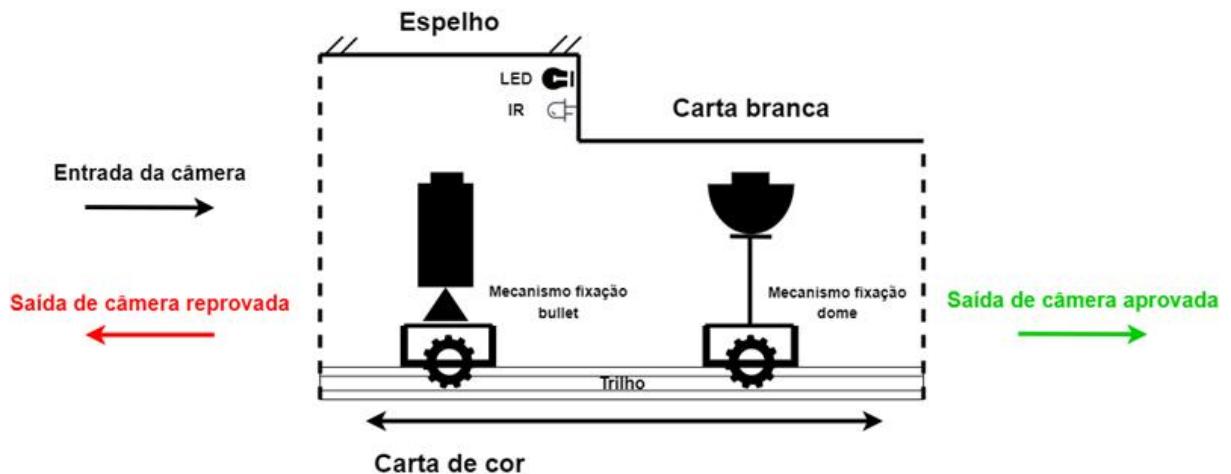
6.1 IMPLICAÇÕES DO PROJETO

A aplicação bem-sucedida deste trabalho abriu caminho para o desenvolvimento de novos sistemas que utilizam processamento de imagem com Redes Neurais com a finalidade de outras otimizações do processo produtivo.

O novo sistema trata-se de uma unidade automatizada de testes, contemplando todos os testes realizados durante a montagem final da câmera. O sistema é composto por um trilho, o qual permite o deslocamento linear da câmera entre os cenários de teste; circuitos elétricos para acionamento do motor, dos LEDs do cenário e leitura de sensores; e um Raspberry Pi, onde é executado um software composto por um algoritmo de processamento de imagens que utiliza diferentes modelos treinados de Redes Neurais para realização dos testes.

A Figura 44 mostra um esquemático da estrutura e do funcionamento do sistema apresentado.

Figura 44 – Projeto de mecanismo para automação dos testes na montagem final.



Fonte: Autoria própria

6.2 PERSPECTIVAS FUTURAS

Como possibilidade de trabalhos futuros, sugere-se o desenvolvimento de um sistema eletromecânico, composto por um algoritmo de Inteligência Artificial, capaz de medir o nível de desfoque e realizar o ajuste de foco de forma automática em câmeras de segurança.

Integrado ao sistema de identificação automática de manchas desenvolvido nesse trabalho, a solução permitiria reduzir o número de câmeras desfocadas que, possivelmente, chegariam ao cliente final, além de eliminar a necessidade de um operador humano no posto e diminuir o tempo de ajuste de foco. Como consequência, teria-se um aumento de qualidade, redução do tempo de produção e redução dos custos de operação.

6.3 CONCLUSÃO

Todo o trabalho realizado durante o Projeto de Fim de Curso foi de suma importância para aplicação dos conhecimentos adquiridos durante a graduação no curso de Engenharia de Controle e Automação, com destaque para os tópicos de Inteligência Artificial e desenvolvimento de software, além da aplicação dos conhecimentos de avaliação de desempenho e modelagem de software.

No âmbito profissional, o trabalho realizado possibilitou a experiência completa de desenvolvimento de projeto, contemplando desde as etapas iniciais de concepção e planejamento, até as etapas finais de implementação e acompanhamento. Além disso, o projeto proporcionou a realização de diversas apresentações para times de

engenharia e visitas ao ambiente fabril, configurando-se excelentes oportunidades de crescimento e aprendizado.

Diante do exposto, considera-se que, além do cumprimento dos requisitos de projeto e realização dos objetivos técnicos traçados, o Projeto de Fim de Curso apresentado cumpriu o objetivo principal de aplicar os conhecimento adquiridos durante a graduação de forma prática em um problema real de engenharia de suma importância para o mercado de trabalho.

REFERÊNCIAS

- BAHETI, Pragati. The Essential Guide to Neural Network Architectures. **V7 – Convolutional Neural Networks (CNN)**, 2021. Disponível em: <https://www.v7labs.com/blog/neural-network-architectures-guide#cnn>. Acesso em: 17/01/2022.
- BRITANNICA, Encyclopedia. Lens | Meaning, Principles, Manufacture, Facts, 2021. Disponível em: <https://www.britannica.com/technology/lens-optics>. Acesso em: 17/01/2022.
- BROWNLEE, Jason. Overfitting and Underfitting With Machine Learning Algorithms, 2019. Disponível em: <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/#:~:text=Overfitting%3A%20Good%20performance%20on%20the,poor%20generalization%20to%20other%20data>. Acesso em: 26/01/2022.
- GANDHI, Arun. Data Augmentation | How to use Deep Learning when you have Limited Data—Part 2. **Nanonets**, 2021. Disponível em: <https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/>. Acesso em: 17/02/2021.
- HEBB, Donald Olding. **The Organization of Behavior**. New York: JOHN WILEY & SONS, Inc, 1949. Disponível em: https://pure.mpg.de/rest/items/item_2346268_3/component/file_2346267/content. Acesso em: 03/03/2021.
- KAIMING HE *et al.* Deep Residual Learning for Image Recognition, 2015. Disponível em: <https://arxiv.org/pdf/1512.03385.pdf>. Acesso em: 06/03/2021.
- KAPATKER, Jayant. UNDERSTANDING THE VIDEO SIGNAL. **STAM Multimedia Inc**, 2020. Disponível em: <https://clearview-communications.com/wp-content/uploads/2020/04/Understanding-the-Video-Signal.pdf>. Acesso em: 17/01/2022.
- KRIZHEVSKY, Alex. ImageNet Classification with Deep Convolutional Neural Networks, 2012. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Acesso em: 03/03/2021.

LECUN, Yann. Gradient-Based Learning Applied to Document Recognition, 1998. Disponível em: <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>. Acesso em: 03/03/2021.

LEFKOWITZ, Melanie. Professor's perceptron paved the way for AI – 60 years too soon, 2019. Disponível em:
<https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon>. Acesso em: 03/03/2021.

MARKETS, Research. Global Surveillance Camera Market Insights Forecast Report 2021-2025: Growth in Traffic Management, Upsurge in Crime Index, Demand for Big Data, Sales of Home Security Systems, 2020. Disponível em:
<https://www.globenewswire.com/news-release/2021/06/11/2245683/28124/en/Global-Surveillance-Camera-Market-Insights-Forecast-Report-2021-2025-Growth-in-Traffic-Management-Upsurge-in-Crime-Index-Demand-for-Big-Data-Sales-of-Home-Security-Systems.html>. Acesso em: 24/11/2021.

NAZNEEN, A. Camera Tuning : Understanding the Image Signal Processor and ISP Tuning. **PathPartner**, 2020. Disponível em:
<https://www.pathpartnertech.com/camera-tuning-understanding-the-image-signal-processor-and-isp-tuning/>. Acesso em: 17/01/2022.

PAESSLER. What is CCTV? Definition and Details, 2021. Disponível em:
<https://www.paessler.com/it-explained/cctv>. Acesso em: 06/01/2022.

PYKES, Kurtis. The Difference Between Classification and Regression in Machine Learning, 2021. Disponível em: <https://towardsdatascience.com/the-difference-between-classification-and-regression-in-machine-learning-4ccdb5b18fd3>. Acesso em: 16/02/2022.

RADHAKRISHNAN, Pranoy. What are Hyperparameters? and How to tune the Hyperparameters in a Deep Neural Network? **Towards Data Science**, 2017. Disponível em: <https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a>. Acesso em: 17/01/2022.

RAMOS, Leodanis Pozo. Use PyQt's QThread to Prevent Freezing GUIs, 2021. Disponível em: <https://realpython.com/python-pyqt-qthread/>. Acesso em: 09/02/2022.

SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, 2018. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Acesso em: 03/03/2021.

SATHYA, R. Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. **International Journal of Advanced Research in Artificial Intelligence Vol. 2, No. 2**, 2013. Disponível em: https://www.researchgate.net/publication/273246843_Comparison_of_Supervised_and_Unsupervised_Learning_Algorithms_for_Pattern_Classification. Acesso em: 10/03/2022.

SECURITY, Adeva. How do CCTV cameras work at night?, 2018. Disponível em: [https://adevasecurity.com.au/how-do-cctv-cameras-work-at-night/#:~:text=In%20order%20to%20see%20at,use%20infrared%20\(IR\)%20technology.&text=These%20emit%20infrared%20light%20at,the%20CCTV%20camera%20is%20activated..](https://adevasecurity.com.au/how-do-cctv-cameras-work-at-night/#:~:text=In%20order%20to%20see%20at,use%20infrared%20(IR)%20technology.&text=These%20emit%20infrared%20light%20at,the%20CCTV%20camera%20is%20activated..) Acesso em: 17/01/2022.

SHEN, Kevin. Effect of batch size on training dynamics. **Medium**, 2018. Disponível em: <https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e>. Acesso em: 20/01/2021.

SMITH, Leslie N. Cyclical Learning Rates for Training Neural Networks, 2015. Disponível em: <https://arxiv.org/pdf/1506.01186.pdf>. Acesso em: 17/02/2021.

STRACHNYI, Kate. Brief History of Neural Networks, 2019. Disponível em: <https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec>. Acesso em: 20/01/2022.

TCH, Andrew. The mostly complete chart of Neural Networks, explained, 2017. Disponível em: <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>. Acesso em: 20/01/2022.

VIEIRA, Sérgio. Sorria, a Intelbras está te filmando, 2021. Disponível em:
<https://www.istoeedinheiro.com.br/sorria-a-intel-bras-esta-te-filmando/>.
Acesso em: 06/01/2022.

WIRE, Business. Global Surveillance Camera Market Report 2021, 2020. Disponível em: <https://www.businesswire.com/news/home/20210310005487/en/Global-Surveillance-Camera-Market-Report-2021-Market-was-Valued-at-26.41-Billion-in-2020---Forecast-to-2026---ResearchAndMarkets.com>. Acesso em: 24/11/2021.

WORLDWIDE, Edmund Optics. Imaging Electronics 101: Understanding Camera Sensors for Machine Vision Applications, 2021. Disponível em:
<https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/>. Acesso em: 17/01/2022.