INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

LETICIA MELLO VIEIRA

ESTUDO SOBRE A UTILIZAÇÃO DE IMAGENS SINTÉTICAS NA CRIAÇÃO DE MODELOS DE CLASSIFICAÇÃO EMPREGANDO REDES NEURAIS CONVOLUCIONAIS

FLORIANÓPOLIS, 2022.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

LETICIA MELLO VIEIRA

ESTUDO SOBRE A UTILIZAÇÃO DE IMAGENS SINTÉTICAS NA CRIAÇÃO DE MODELOS DE CLASSIFICAÇÃO EMPREGANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheira Mecatrônica.

Orientador: Prof. Raimundo Ricardo Matos da Cunha, Dr. Eng.
Coorientador: Prof. Maurício Edgar Stivanello, Dr. Eng.

FLORIANÓPOLIS, 2022.

Ficha de identificação da obra elaborada pelo autor.

Vieira, Leticia Mello
Estudo sobre a utilização de imagens sintéticas na **criação de modelos de classificação empregando redes** neurais **convolucionais / Leticia Mello Vieira**; orientação **de Raimundo Matos da Cunha**; **coorientação** de Maurício **Edgar Stivanello. - Florianópolis, SC**, 2022.
57 p.
Trabalho de Conclusão de Curso (TCC) - **Instituto** Federal **de Santa Catarina, Câmpus Florianópolis**. Bacharelado **em Engenharia Mecatrônica**. Departamento **Acadêmico de Metal** Mecânica.
Inclui Referências.

1. Redes neurais convolucionais. 2. Dados sintéticos. **3. Modelo de classificação. 4. Deep** learning. 5. **Visão computacional. I. Cunha, Raimundo Matos da. II.** Stivanello, **Maurício Edgar. III.** Instituto Federal **de Santa Catarina. IV. Estudo sobre a utilização de** imagens **sintéticas na criação de modelos de** classificação

ESTUDO SOBRE A UTILIZAÇÃO DE IMAGENS SINTÉTICAS NA CRIAÇÃO DE MODELOS DE CLASSIFICAÇÃO EMPREGANDO REDES NEURAIS CONVOLUCIONAIS

LETICIA MELLO VIEIRA

Este trabalho foi julgado adequado para obtenção do Título de Engenheira Mecatrônica e aprovado na sua forma final pela banca examinadora do curso de Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 12 de dezembro de 2022.

Banca examinadora:

gov.br
Documento assinado digitalmente
RAIMUNDO RICARDO MATOS DA CUNHA
Data: 10/02/2023 17:16:05-0300
Verifique em https://verificador.iti.br

Prof. Raimundo Ricardo Matos da Cunha, Dr. Eng.

IFSC
Documento assinado digitalmente
MAURICIO EDGAR STIVANELLO
Data: 12/02/2023 15:50:27-0300
CPF: 901.390.629-04
Verifique as assinaturas em https://v.ifsc.edu.br

Prof. Maurício Edgar Stivanello, Dr. Eng.

gov.br
Documento assinado digitalmente
ANDRE ROBERTO DE SOUSA
Data: 16/02/2023 11:08:50-0300
Verifique em https://verificador.iti.br

Prof. André Roberto de Sousa, Dr. Eng.

GOV.BR
Documento assinado digitalmente
FRANCISCO RAFAEL MOREIRA DA MOTA
Data: 13/02/2023 14:22:41-0300
CPF: ***.003.153-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Francisco Rafael Moreira da Mota, Dr. Eng.

AGRADECIMENTOS

Inicio agradecendo ao Instituto Federal de Santa Catarina - campus Florianópolis, seu corpo docente, direção e administração que juntos oferecem um curso de extrema qualidade. Em especial, agradeço ao professor Maurício Edgar Stivanello por ter me supervisionado durante a bolsa de pesquisa que gerou o tema desse trabalho, pela assistência e por seu apoio durante a execução. Ao professor Raimundo Ricardo Matos da Cunha que acompanhou o desenvolvimento do trabalho desde o início da pesquisa, sempre me auxiliando.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) que me abriu portas para aprender sobre o tema e me ajudou a crescer pessoal e profissionalmente.

À Fundação CERTI, em especial ao CMI e ao SMIT, onde tive a oportunidade de trabalhar com profissionais excelentes.

À minha família, por sempre apoiar e incentivar meus estudos, a correr atrás dos meus sonhos e dar meu melhor sempre.

RESUMO

Esse trabalho teve como objetivo estudar a efetividade da utilização de dados sintéticos na criação de modelos de classificação utilizando redes neurais convolucionais. Foram abordados conceitos, técnicas e metodologias para entendimento sobre a aplicabilidade de redes neurais convolucionais modelada para classificação de objeto. Inicialmente foi realizada uma seleção do caso de uso através da análise bibliográfica de trabalhos correlatos a fim de comparar diferentes desenvolvimentos e soluções já obtidas. Em seguida, foi criada a base de dados sintéticos para desenvolvimento do projeto com o auxílio dos *software* de modelagem 3D, Blender e SolidWorks. Com a base de dados pronta, iniciou-se a implementação dos algoritmos e criação do modelo de classificação utilizando a biblioteca de programação Fastai e a plataforma *online* Google Colaboratory. Nessa etapa foram realizados testes e ajustes de parâmetros que forneceram à rede um treinamento eficaz para alcançar resultados adequados. Uma vez que o algoritmo esteve bem ajustado, foram analisadas e apresentadas as decorrências com imagens sintéticas e imagens reais, levando em consideração os fatores mais relevantes.

Palavras-chaves: Redes neurais convolucionais. Dados sintéticos. Modelo de classificação. *Deep learning*. Visão computacional.

ABSTRACT

This work aimed to study the effectiveness of using synthetic data in the creation of classification models using convolutional neural networks. Concepts, techniques and methodologies for understanding the applicability of modeled convolutional neural networks for object classification were addressed. Initially, a selection of the use case was carried out through the bibliographical analysis of related works in order to compare different developments and solutions already obtained. Then, the synthetic database was created for project development with the help of 3D modeling software, Blender and SolidWorks. With the database ready, the implementation of the algorithms and the creation of the classification model began using the Fastai programming library and the Google Collaboratory online platform. At this stage, tests and parameter adjustments were performed that provided the network with effective training to achieve adequate results. Once the algorithm was well adjusted, the consequences with synthetic images and real images were analyzed and presented, taking into account the most relevant factors.

Palavras-chaves: Convolutional neural networks. Synthetic data. Classification model. Deep learning. Computer vision.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A classificação de objetos através do aprendizado de máquina (*machine learning*) cresce a cada ano, possibilitando melhora em diferentes processos, como por exemplo, os processos industriais e comerciais. Um sistema de visão computacional pode ensinar o computador a enxergar um processo e tomar decisões como se tivesse um cérebro humano, sendo necessário realizar uma coleta de dados que servirão para o aprendizado da máquina e realizar um pré-processamento para tratá-los e assim conseguir extrair informações úteis para o sistema. Em muitos casos a geração da base de dados possui custo elevado e por isso a utilização de materiais criados a partir de imagens sintéticas vêm crescendo e se tornando cada vez mais comum.

As bases de dados sintéticas podem em muitas situações serem mais facilmente criadas a partir de modelos geométricos que representam com maior ou menor nível de detalhes os objetos a serem analisados. Estas bases podem incluir descomplicadamente diferentes condições de aquisição ou mesmo variações geométricas, de posição e orientação dos objetos de interesse quando comparado à geração de bases na forma tradicional. Assim, são facilmente incorporadas pelos modelos de classificação, possuindo rapidez para atingir um alto nível de dados e uma vez totalmente treinados de forma eficiente, modelos de visão computacional podem ser utilizados em tarefas repetitivas e até mesmo complexas, aumentando o nível de produtividade e diminuindo custos nos processos.

Sistemas que possuem inteligência artificial (IA) e *machine learning* necessitam de uma grande quantidade de dados para que possam ser treinados de forma eficaz. Muitas vezes a coleta desses dados se torna difícil, seja pela disponibilidade, seja pelo elevado custo na geração desses dados. Os dados sintéticos podem ser utilizados para acelerar o desenvolvimento de modelos de classificação, desde que sejam estatisticamente significativos e espelhem dados reais com eficácia em seu uso.

A utilização de dados sintéticos para treinamento de redes neurais vem se mostrando cada vez mais eficiente. De acordo com Svetlana Sicular, analista do Gartner, até 2024, 60% dos dados utilizados para o desenvolvimento de soluções de IA e *analytics* serão gerados sinteticamente, contra 1% em 2021 (IT FORUM, 2022).

1.1 Objetivos

Após uma breve introdução do contexto em que o trabalho está inserido, tem-se a seguir a apresentação dos objetivos geral e específicos para a realização do trabalho.

1.1.1 Objetivo Geral

Realizar estudo sobre a efetividade da utilização de imagens sintéticas na criação de modelos de classificação de objetos empregando redes neurais convolucionais.

1.1.2 Objetivos Específicos

a) definição da base de dados;
b) implementação dos algoritmos e criação de modelos de classificação;
c) validação da eficácia de modelos treinados com imagens sintéticas;
d) verificação da acurácia dos modelos quando testados com imagens reais;
e) avaliação dos parâmetros importantes a serem considerados na geração das bases de dados sintéticas.

2 FUNDAMENTAÇÃO TEÓRICA

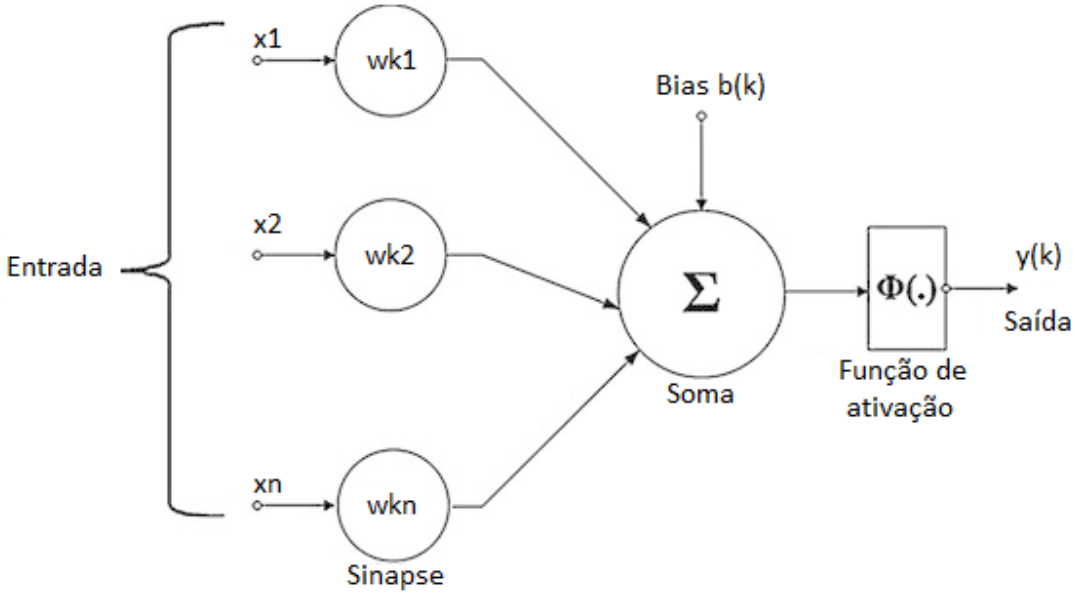
Neste capítulo são apresentados conceitos que são essenciais para o entendimento do trabalho realizado. Primeiramente são apresentadas informações sobre redes neurais convolucionais, suas camadas, parâmetros e seus diferentes tipos de arquitetura. Em seguida, são abordados o processo de treinamento, o uso de base de dados sintéticos e as métricas de avaliação dos resultados. Por fim, são explorados trabalhos correlatos.

2.1 Redes Neurais Convolucionais

No ano de 1943 em seu artigo seminal sobre como os neurônios devem funcionar, Warren McCulloch e Walter Pitts criaram uma rede neural simples com circuitos elétricos. O objetivo original da abordagem de rede neural era criar um sistema computacional capaz de resolver problemas como um cérebro humano. No entanto, com o passar do tempo, os pesquisadores mudaram o foco e passaram a usar redes neurais para resolver tarefas específicas, desviando-se de uma abordagem estritamente biológica. Desde então, as redes neurais têm oferecido suporte às mais diversas tarefas, incluindo visão computacional, reconhecimento de fala, tradução de máquina, filtragem de redes sociais, jogos de tabuleiro ou videogame e diagnósticos médicos (IT FORUM, 2022).

Neste modelo de neurônio matemático, os impulsos elétricos provenientes de outros neurônios são representados pelos chamados sinais de entrada, ilustrados com a letra "x" na Figura 1, que nada mais são do que os dados que alimentam o modelo de rede neural artificial. Dentre os vários estímulos recebidos, alguns excitarão mais e outros menos o neurônio receptor e essa medida de quão excitatório é o estímulo é caracterizado no modelo de Warren McCulloch e Walter Pitts através dos pesos sinápticos, exprimido por “wkn” onde “k” representa o índice do neurônio em questão e “n” se refere ao terminal de entrada da sinapse a qual o peso sináptico se refere. Quanto maior o valor do peso, mais excitatório é o estímulo (DATA SCIENCE ACADEMY, 2022).

Figura 1 - Representação simplificada do neurônio matemático



Fonte: Adaptado de DATA SCIENCE ACADEMY (2022).

A soma ou corpo da célula é representada por uma composição de dois módulos, o primeiro é uma junção aditiva, somatório dos estímulos (sinais de entrada) multiplicado pelo seu fator excitatório (pesos sinápticos), e posteriormente uma função de ativação, que definirá com base nas entradas e pesos sinápticos, qual será a saída do neurônio. O axônio é aqui representado pela saída “yk” obtida pela aplicação da função de ativação. Assim como no modelo biológico, o estímulo pode ser excitatório ou inibitório, representado pelo peso sináptico positivo ou negativo respectivamente (DATA SCIENCE ACADEMY, 2022).

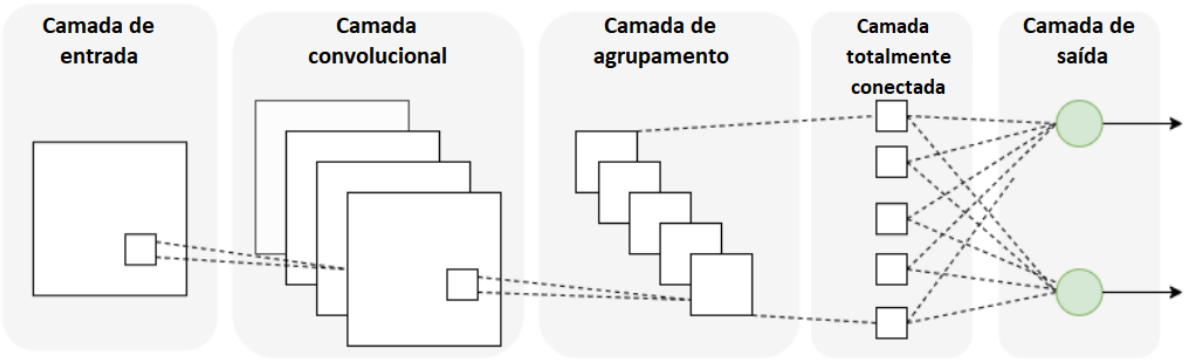
O modelo do neurônio matemático também pode incluir uma polarização ou bias de entrada. Esta variável é incluída no somatório da função de ativação, com o intuito de aumentar o grau de liberdade desta função e, consequentemente, a capacidade de aproximação da rede. O valor do bias é ajustado da mesma forma que os pesos sinápticos. O bias possibilita que um neurônio apresente saída não nula ainda que todas as suas entradas sejam nulas (DATA SCIENCE ACADEMY, 2022).

A partir dessa estrutura inicial, surgiram diversos novos modelos de redes neurais, sendo a rede neural convolucional (CNN) um deles.

As CNNs foram propostas para classificar imagens, agrupando-as por similaridade e realizando o reconhecimento de objetos em diferentes cenas. De acordo com BARBOSA *et. al.* (2021), o aprendizado profundo de uma CNN é caracterizado pela repetição de camadas convolucionais e de agrupamento. A cada

par de camadas de convolução e agrupamento são extraídas características de mais alto nível. A camada totalmente conectada realiza a classificação através das características extraídas. A consolidação do resultado ocorre na camada de saída. Na Figura 2 é ilustrada a arquitetura básica de uma CNN.

Figura 2 - Arquitetura básica de uma CNN



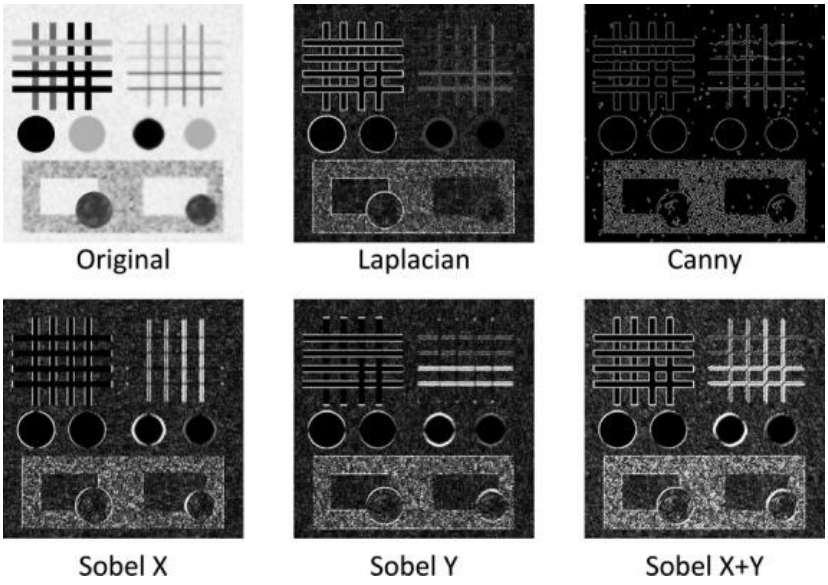
Fonte: Adaptado de BARBOSA *et. al.* (2021).

2.1.1 Camada de Convolução

Na primeira camada de uma rede neural convolucional acontece o processo de convolução, que consiste na aplicação de um ou mais filtros na imagem de entrada, normalmente chamados de *kernels*, com o objetivo de reconhecer padrões.

A convolução é uma operação linear que usa *kernels* de tamanhos distintos para gerar diferentes mapas de características (*features maps*) da imagem de entrada. Pode-se aplicar filtros para diferentes finalidades, como por exemplo: suavizar a imagem, aumentar ou diminuir o contraste, destacar as bordas, aplicar diferentes texturas, dentre outras. Um exemplo com aplicações de diferentes filtros em uma imagem é apresentado na Figura 3.

Figura 3 - Diferentes tipos de filtros

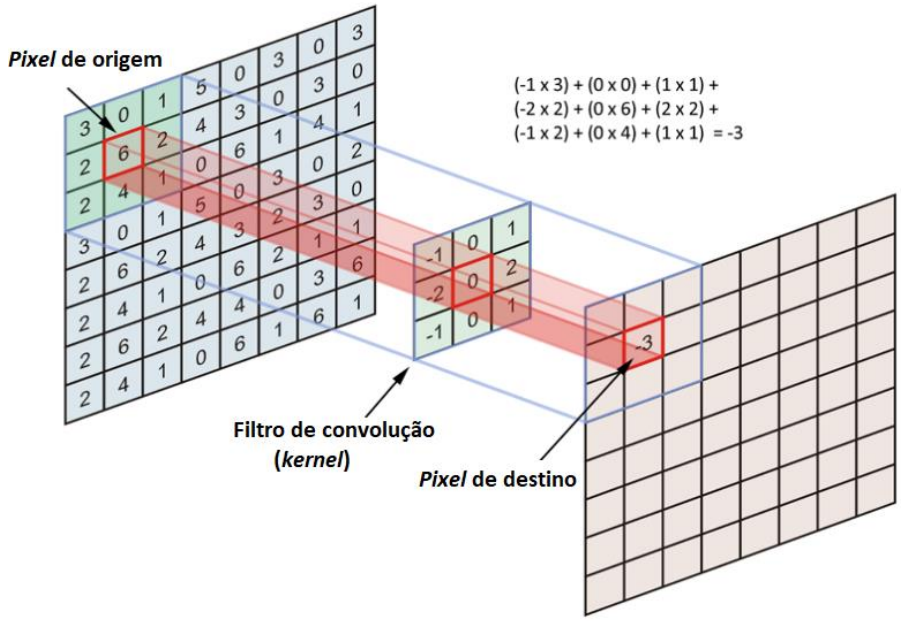


Fonte: GITHUB (2022).

Basicamente um *kernel* é uma matriz MxN, ou seja, não é necessário que tenha tamanhos de altura e largura iguais, que varre toda a imagem de entrada e realiza a operação de somatório do produto, pixel a pixel. Geralmente sua dimensão é definida em números ímpares para obter uma imagem de saída simétrica, com um pixel central.

Tamanhos grandes de *kernels* são utilizados para a extração de recursos maiores, obtendo menos informações e levando a uma redução mais rápida das dimensões da camada. Por outro lado, *kernels* menores são capazes de extrair uma quantidade muito maior de informações e possuem um desempenho melhor para classificação de imagens porque são capazes de empilhar várias camadas para aprender cada vez mais características complexas. A escolha do tamanho do kernel depende da tarefa a ser executada e do conjunto de dados. A Figura 4 exemplifica a aplicação de um *kernel* 3x3.

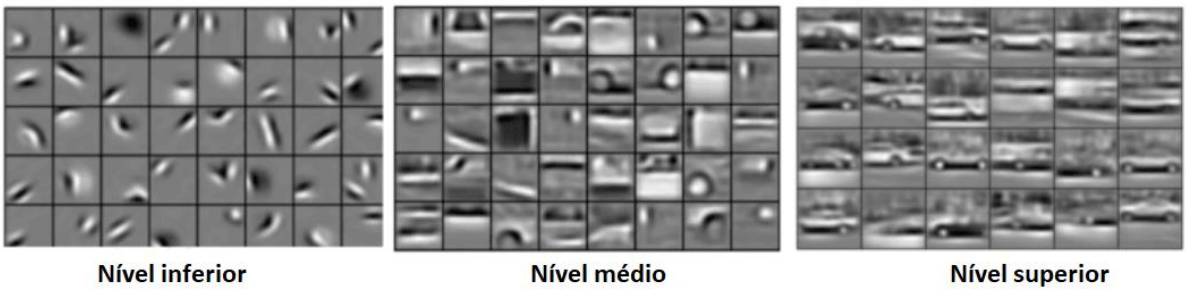
Figura 4 - Exemplo de um *kernel* 3x3



Fonte: Adaptado de DATA SCIENCE STACK EXCHANGE (2022).

Conforme as camadas de convolução se aprofundam, o modelo aprende características mais complexas usando informações de segmentos maiores de imagens e informações de outros neurônios. Esse comportamento pode ser visto na Figura 5, onde na primeira camada são aprendidas características simples, como arestas e cantos. Na camada intermediária, são aprendidas como formas e partes de objetos e nas últimas camadas são aprendidas partes mais abstratas dos objetos, neste caso veículos (ERAZO, 2021).

Figura 5 - Extração de características em várias camadas da rede

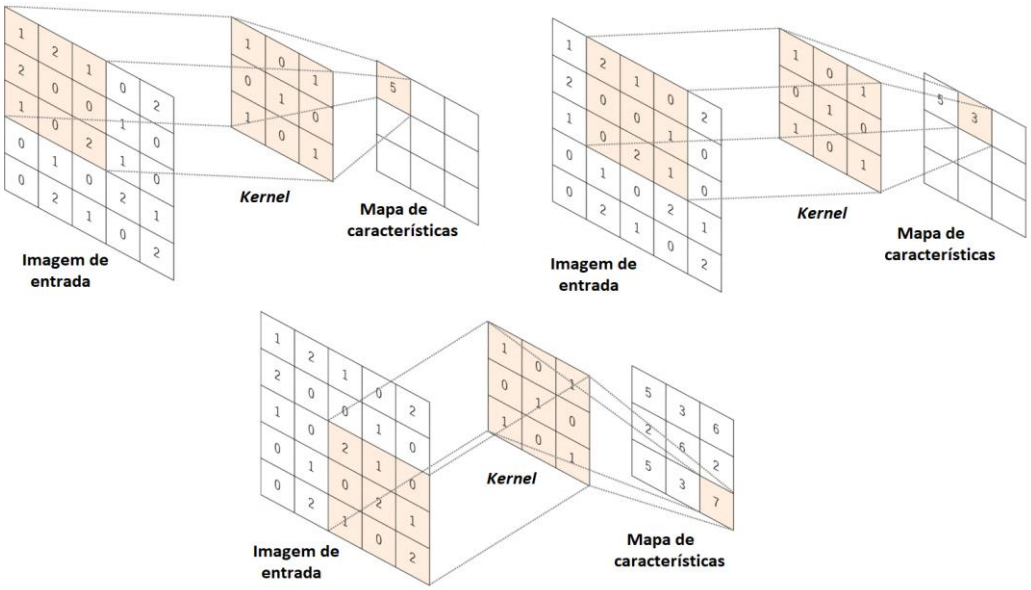


Fonte: Adaptado de ERAZO (2021).

2.1.1.1 Passo (Stride)

É possível alterar o movimento do *kernel*, que normalmente desloca-se da esquerda para a direita e de cima para baixo, através da mudança do seu passo, também chamado de *stride*. Por exemplo, um *stride* de (1, 3) significa que o *kernel* irá se deslocar 1x1 na vertical e 3x3 na horizontal, reduzindo a dimensão da imagem de entrada em 3 vezes. Quanto menor o valor do *stride*, maior o tempo computacional. A Figura 6 exemplifica a utilização de um *stride* 1x1.

Figura 6 - Exemplo com stride 1x1

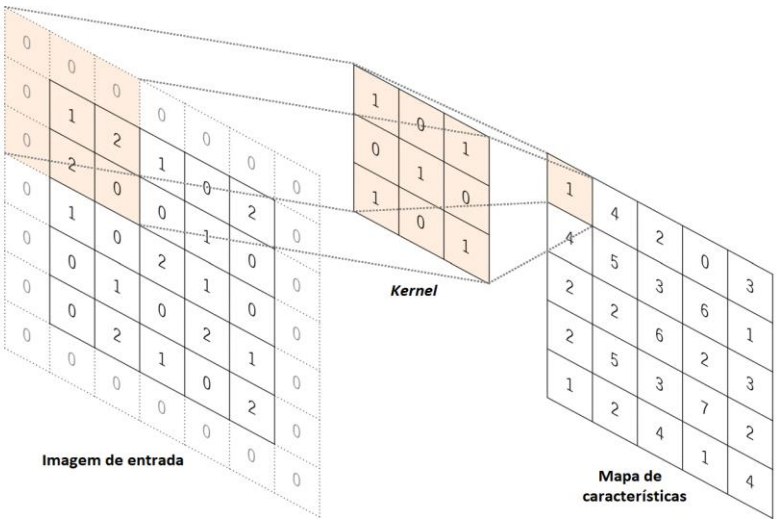


Fonte: Adaptado de YAMASHITA (2018).

2.1.1.2 Preenchimento (Padding)

O *kernel* diminui o tamanho da imagem de entrada, mas em determinados casos é necessário que o tamanho seja preservado na imagem de saída. Existem diferentes técnicas de preenchimento (*padding*) mas a abordagem mais utilizada é a adição de zeros devido ao seu desempenho, simplicidade e eficiência computacional. O processo consiste em acrescentar zeros simetricamente ao redor das bordas da imagem de entrada. A Figura 7 ilustra um exemplo utilizando *padding*.

Figura 7 - Exemplo com padding



Fonte: Adaptado de YAMASHITA (2018).

2.1.1.3 Função de Ativação

A convolução é uma operação linear que deve ter suas imagens de saídas passadas por uma função de ativação não linear e então enviadas para a próxima camada de neurônios como imagens de entradas. A não linearidade é necessária para produzir limites de decisão não lineares, de modo que a saída não possa ser escrita como uma combinação linear das entradas.

Existem diferentes tipos de funções de ativação com diferentes graus de não-linearidade. Alguns exemplos são:

a) função *sigmoide*: amplamente utilizada em redes neurais clássicas e produz valores no intervalo [0, 1]. Na camada de saída, é usada para produzir probabilidades em problemas de classificação binária já que seus resultados podem ser interpretados como a probabilidade de determinada instância pertencer ou não determinada classe. É expressa na forma da Equação 1;

$$f(x) = \frac{1}{1+e^{-x}} \quad (1)$$

b) função *softmax*: é a generalização da função *sigmoide* para casos não binários e comumente utilizada em problemas de classificação

multi-classe, produzindo resultados no intervalo [0, 1] onde sua soma é igual a 1. É escrita na forma da Equação 2;

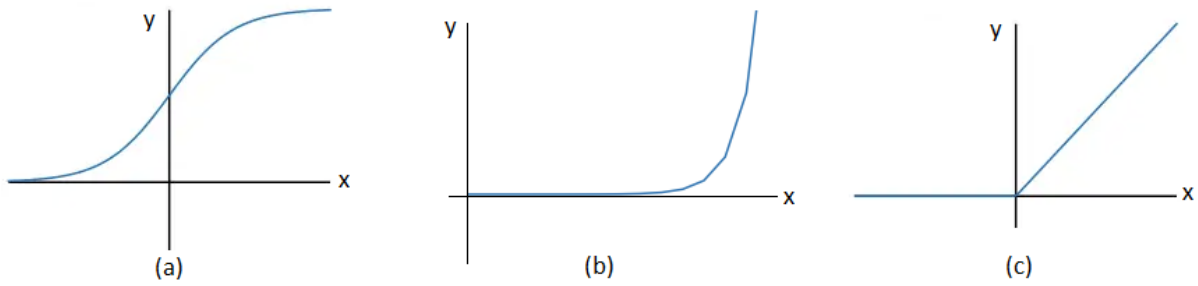
$$f(x)i = \frac{e^{xi}}{\sum_{k=1}^{k} e^{xk}} \quad (2)$$

c) função *rectified linear unit* (ReLu): retorna 0 para todos os valores negativos e o próprio valor para valores positivos, não satura na região positiva e converge mais rápido. Produz resultados no intervalo [0, ∞[e é representada na forma da Equação 3.

$$f(x) = max(0, x) \quad (3)$$

A Figura 8 ilustra os gráficos das funções descritas acima.

Figura 8 - (a) Função *sigmoide*, (b) função *softmax* e (c) função ReLu



Fonte: Da autora (2022).

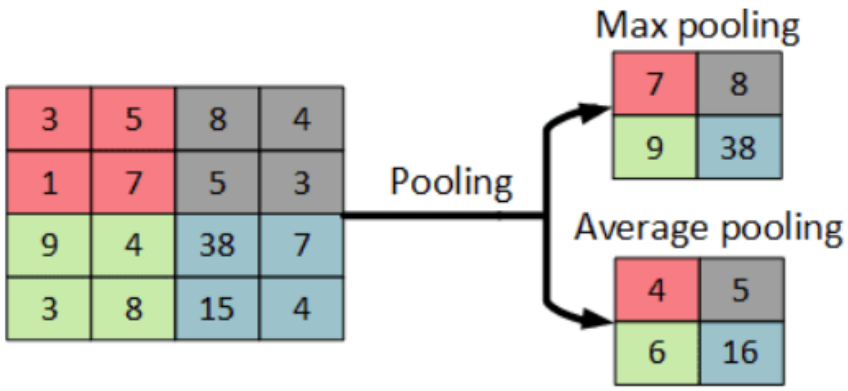
2.1.2 Camada de Agrupamento

A camada de agrupamento é adicionada após uma camada de convolução para reduzir a quantidade de variáveis da camada anterior e assim, simplificar a informação dentro de uma rede neural convolucional e esse processo pode ser repetido uma ou mais vezes.

Esta camada atua separadamente em cada mapa de características gerado na camada anterior e cria um novo conjunto com o mesmo número. Geralmente, o *kernel* aplicado possui tamanho 2x2 e possui um *stride* igual a 2, com o objetivo de reduzir pela metade o tamanho dos mapas de características. Duas funções podem ser utilizadas nessa etapa e são exemplificadas na Figura 9.

a) *average-pooling*: calcula o valor médio de pixel;
b) *max-pooling*: seleciona o maior valor de pixel.

Figura 9 - Exemplo de *max-pooling* e *average-pooling* com *stride* de 2 e filtro de 2x2 sobre uma matriz 4x4, obtendo um mapa de características de 2x2 pixels



Fonte: ERAZO (2021).

2.1.3 Camada Totalmente Conectada

A camada totalmente conectada é a última camada de uma rede neural convolucional e possui quantidade igual a de classes a serem identificadas, por exemplo, uma rede neural convolucional que possui duas classes para classificação vai possuir duas camadas totalmente conectadas.

Cada neurônio dessa camada é conectado com neurônios de camadas anteriores a fim de analisar as características extraídas por meio da convolução e classificar as imagens de acordo com suas classes.

2.2 Diferentes Tipos de Arquiteturas de Redes Neurais Convolucionais

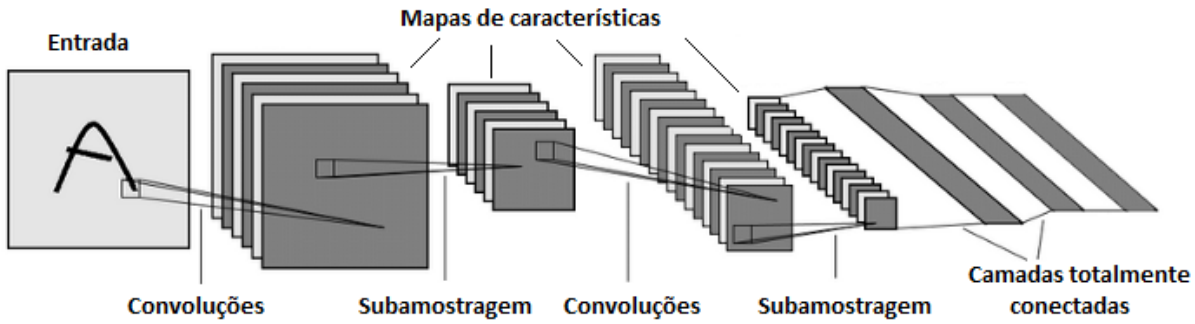
Existem diferentes tipos de arquiteturas de redes neurais convolucionais propostas para resolver o problema desejado e alguns exemplos são apresentados abaixo.

2.2.1 Arquitetura LeNet

A arquitetura LeNet foi proposta por LECUN *et. al.* (1998) com o objetivo de reconhecer dígitos manuscritos. Os dados de entrada consistiam em imagens, cada uma contendo um número, e os resultados do teste nos dados digitais do código

postal fornecidos pelo Serviço Postal dos Estados Unidos mostraram que o modelo tinha uma taxa de erro de apenas 1% e uma taxa de rejeição de cerca de 9% (LECUN *et. al.*, 1998). A Figura 10 ilustra a arquitetura LeNet.

Figura 10 - Arquitetura LeNet

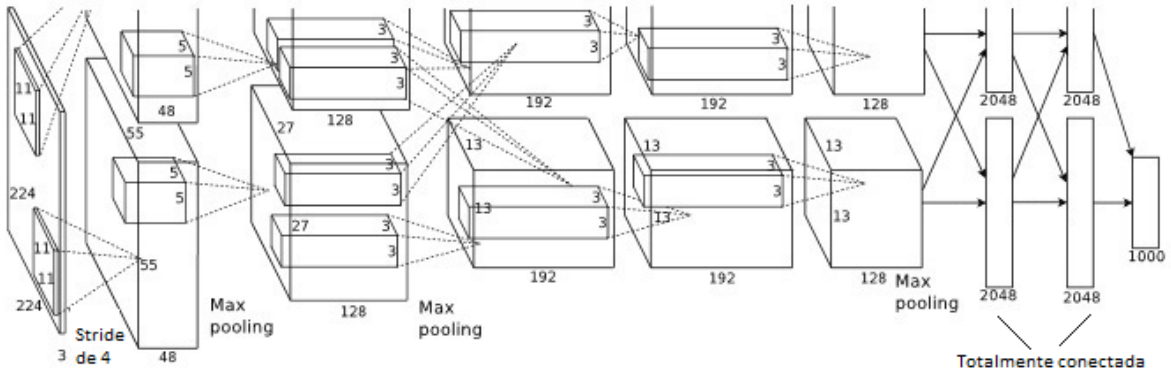


Fonte: Adaptado de LECUN *et. al.* (1998).

2.2.2 Arquitetura AlexNet

AlexNet foi apresentada por KRIZHEVSKY *et. al.* (2012) e contém uma arquitetura semelhante à LeNet, contudo é mais profunda, possui mais filtros por camada e sobrepõe camadas convolucionais. A Figura 11 ilustra a arquitetura AlexNet.

Figura 11 - Arquitetura AlexNet



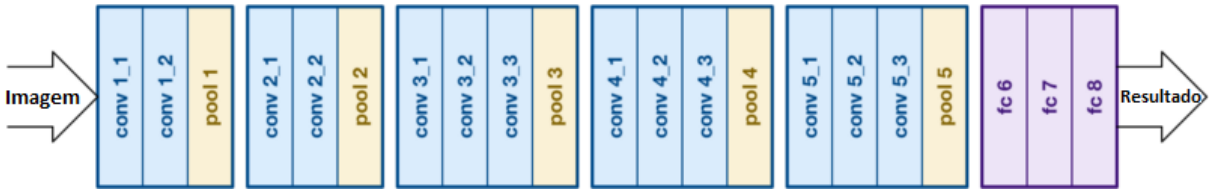
Fonte: Adaptado de KRIZHEVSKY *et. al.* (2012).

2.2.3 Arquitetura VGGNet

A arquitetura VGGNet foi desenvolvida por SIMONYAN; ZISSERMAN (2014), utilizando apenas convoluções 3x3 e agrupamento 2x2. Entretanto, a VGGNet possui muitos parâmetros, tornando-a uma rede ruim de ser treinada a partir do zero,

além de precisar de muita memória RAM. A Figura 12 ilustra a arquitetura VGGNet simplificada, onde “conv” indica a camada de convolução, “pool” a camada de agrupamento e “fc” a camada totalmente conectada.

Figura 12 - Arquitetura VGGNet

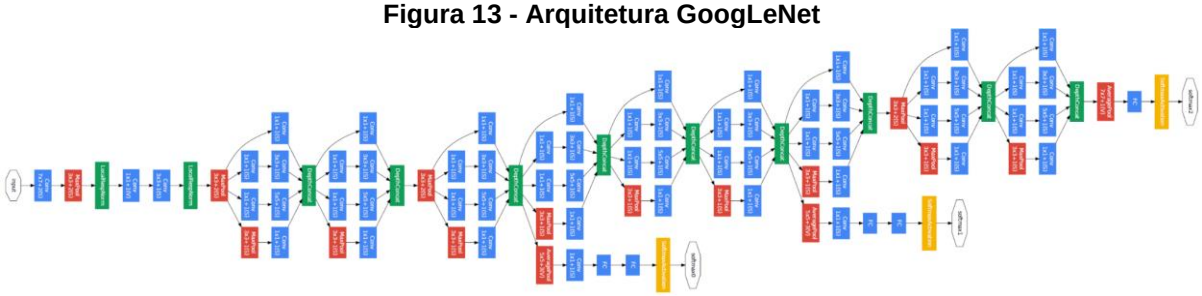


Fonte: Adaptado de VON WANGENHEIM (2018).

2.2.4 Arquitetura GoogLeNet

A GoogLeNet foi proposta por SZEGEDY *et. al.* (2015) e apresentou uma diminuição significativa na taxa de erro em tarefa de classificação de imagens, alcançando 6,67% no top 5 de erros. Ela utiliza a técnica de convolução 1x1, que diminui a quantidade de parâmetros e aumenta a profundidade da arquitetura, e *average pooling* no final da rede. A Figura 13 ilustra a arquitetura.

Figura 13 - Arquitetura GoogLeNet



Fonte: SZEGEDY *et. al.* (2015).

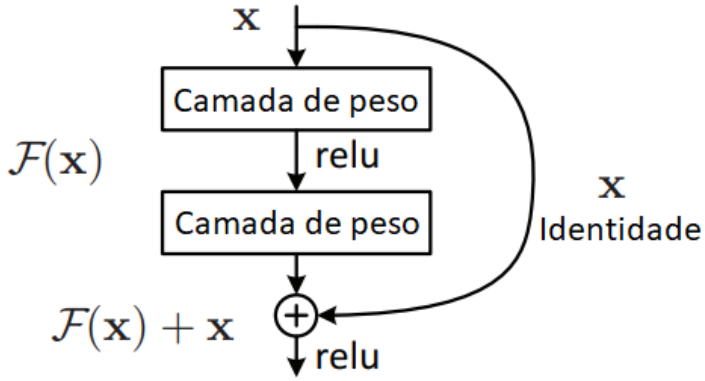
2.2.5 Arquitetura ResNet

Proposta por HE *et. al.* (2016), a arquitetura ResNet introduziu o conceito de blocos residuais, onde é utilizada a técnica chamada de conexões de salto. A técnica conecta as ativações de uma camada a outras camadas, pulando algumas camadas entre elas, formando um bloco residual.

A ResNet usa, em um determinado ponto, um sinal que é a soma do sinal produzido pelas duas camadas convolucionais anteriores mais o sinal transmitido diretamente do ponto anterior para essas camadas, unindo um sinal processado a

uma etapa antes do processamento (VON WANGENHEIM, 2018). A Figura 14 ilustra esse processo.

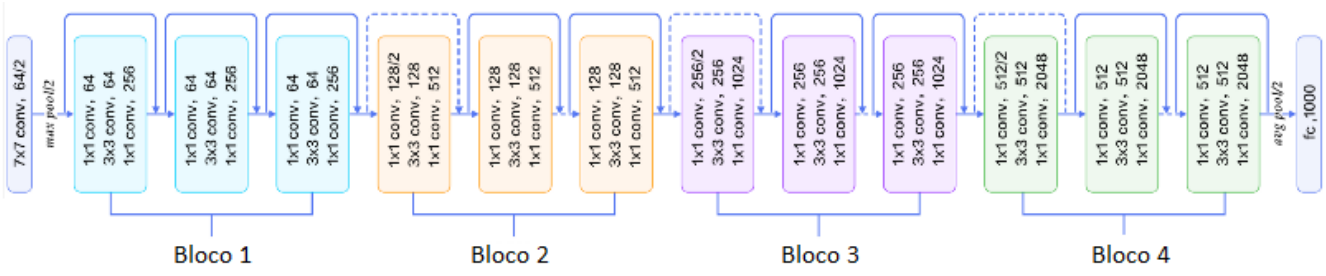
Figura 14 - Aprendizagem residual: um bloco de construção



Fonte: Adaptado de HE *el. al.* (2016).

A vantagem de adicionar esse tipo de conexão é que, se alguma camada prejudicar o desempenho da rede, ela será ignorada pela regularização. A ideia geral da arquitetura de ResNet é uma estrutura de quatro tipos de blocos de camadas, enumerados de 0 a 3, que podem ser usados para construir redes de diferentes profundidades. ResNet-50, por exemplo, usa uma configuração (3, 4, 6, 3). A Figura 15 mostra uma visão abstrata com uma configuração (3, 3, 3, 3) (VON WANGENHEIM, 2018).

Figura 15 - ResNet com configuração (3, 3, 3, 3)



Fonte: Adaptado de VON WANGENHEIM (2018).

Na Figura 16 é apresentado os diferentes modelos da arquitetura ResNet e suas configurações.

Figura 16 - Diferentes modelos de ResNet

Camada	Saída	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ResNet-152
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fonte: Adaptado de HE *el. al.* (2016).

2.3 Treinamento de uma Rede Neural Convolucional

O treinamento de uma rede é um processo de busca de filtros em camadas convolucionais e pesos em camadas totalmente conectadas que minimizam as diferenças entre previsões de saída e as classes corretas em um conjunto de dados de treinamento. O algoritmo de retropropagação é o método comumente usado para treinar redes neurais em que a função de perda e o algoritmo de otimização de descida do gradiente desempenham papéis essenciais (YAMASHITA *et. al.*, 2018).

2.3.1 Função de Perda (*Loss Function*)

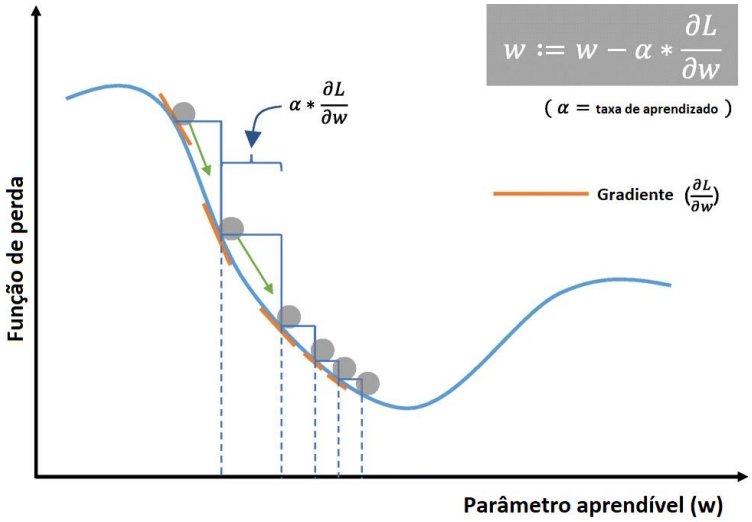
Uma função de perda, também chamada de função de custo (*cost function*), mede a compatibilidade entre as previsões de saída da rede por meio de propagação direta e verdade fundamental. A função de perda comumente usada para classificação multiclasse é a entropia cruzada (*cross entropy*), enquanto o erro quadrático médio é normalmente aplicado à regressão para valores contínuos (YAMASHITA *et. al.*, 2018).

2.3.2 Descida do Gradiente (*Gradient Descent*)

A descida do gradiente é usada como um algoritmo de otimização que atualiza iterativamente os parâmetros que podem ser aprendidos: filtros e pesos, para minimizar a perda. O gradiente da função de perda fornece a direção na qual a função tem a maior taxa de aumento e cada parâmetro é atualizado na direção negativa do gradiente com um tamanho de passo arbitrário determinado por um hiperparâmetro chamado taxa de aprendizado (*learning rate*), conforme ilustrado na Figura 17 (YAMASHITA *et. al.*, 2018).

O gradiente é, matematicamente, uma derivada parcial da perda em relação a cada parâmetro que pode ser aprendido. Onde “w” representa cada parâmetro que pode ser aprendido, “α” é uma taxa de aprendizado e “L” uma função de perda (YAMASHITA *et. al.*, 2018).

Figura 17 - Descida do gradiente



Fonte: Adaptado de YAMASHITA *et. al.* (2018).

2.3.3 Dados e Verdade Fundamental (*Data and Ground Truth Labels*)

Os dados são os itens mais importantes em pesquisas que aplicam o aprendizado profundo (*deep learning*) ou outros métodos de *machine learning*. Eles são divididos em três categorias: treinamento, validação e testes. Em treinamento, são usados para treinar a rede. Na validação, são usados para autenticar o modelo durante o processo de treinamento, ajustar os hiperparâmetros e fazer a seleção do

modelo. Por fim, na etapa de testes são usados no final do processo para avaliar a performance do modelo final.

A verdade fundamental é a resposta esperada, ou seja, a classe correta prevista na classificação.

2.3.4 Sobreajuste (*Overfit*)

Sobreajuste refere-se a uma situação em que um modelo aprende estatísticas e regularidades específicas do conjunto de treinamento, ou seja, acaba memorizando o ruído irrelevante ao invés de aprender o sinal, e, portanto, tem um desempenho inferior em um novo conjunto de dados subsequentes (YAMASHITA *et. al.*, 2018).

É possível amenizar o sobreajuste com a obtenção de mais dados de treinamento. Existem diferentes soluções que incluem regularização com *drop out* ou *weight decay*, normalização por lotes e aumento de dados.

2.3.4.1 Regularização com Drop Out ou Weigh Decay

A regularização com *drop out* foi introduzida por SRIVASTAVA *et. al.* (2014), onde os autores melhoraram a precisão da classificação no conjunto de dados MNIST em 0,4%. Nesse processo acontece o descarte de neurônios (junto com suas conexões) aleatoriamente da rede neural durante o treinamento em cada iteração, sendo equivalente a treinar diferentes redes neurais. Dessa forma, as diferentes redes irão sofrer sobreajustes de maneiras distintas e o efeito do *drop out* irá reduzi-los.

Já o *weight decay* funciona adicionando um termo de penalidade à função de perda de uma rede neural, diminuindo os pesos durante a retropropagação.

2.3.4.2 Normalização por Lotes

Durante o treinamento do modelo, a normalização em lote ajusta continuamente a saída intermediária da rede neural, subtraindo sua média e dividindo por seu desvio padrão, de modo que os valores em cada camada sejam mais estáveis.

2.3.4.3 Aumento de Dados (Data Augmentation)

O aumento de dados refere-se à aplicação aleatória de vários tipos de transformações às imagens da base de dados, ajudando a introduzir mais variedade ao conjunto. Em vez de alimentar o modelo com o mesmo conteúdo todas as vezes, são realizadas pequenas modificações aleatórias (como rotação, zoom, translação, ajuste de brilho, entre outros) que não alteram o que está dentro da figura, mas sim seus valores de pixel.

Essa é uma ferramenta útil em situações onde se possui um número relativamente pequeno de amostras de treinamento.

2.3.5 Aprendizado por Transferência (*Transfer Learning*)

Para treinar completamente uma CNN, é necessária uma quantidade elevada de dados e alta variabilidade de características. Dependendo da aplicação, esse tipo de dado pode ser caro ou inviável de adquirir (CAO *et. al.*, 2018, *apud* OLIVEIRA *et. al.*, 2019). Além disso, as redes neurais são sensíveis às diferenças de iluminação, contraste ou rotação. Se as imagens utilizadas para treinamento não apresentarem uma variedade suficiente dessas condições, o classificador poderá não se adaptar a esses detalhes, reduzindo o desempenho da classificação. Dados indesejados, como partes do plano de fundo ou ruído, podem fazer com que o sistema se especialize apenas nos dados de treinamento, levando a uma generalização baixa (TAY e CAO, 2001, *apud* OLIVEIRA *et. al.*, 2019).

Uma vez que as camadas intermediárias de uma CNN são geralmente extratoras de características que podem ser generalizadas para conjuntos de dados diferentes, uma solução comum para o treinamento é usar um conjunto de dados maior para pré-treinar a rede e, em seguida, retreinar a rede para reconhecer as classes da base de dados alvo (Oquab *et. al.*, 2014, *apud* Oliveira *et. al.*, 2019). Esta técnica é amplamente utilizada para alcançar uma alta acurácia ao treinar redes utilizando um conjunto de dados insuficiente. Outra utilidade do retreino é adicionar novas imagens em uma rede treinada. Nesse caso, é possível retreinar apenas as camadas de reconhecimento com a nova base de dados (incluindo o novo objeto) (OLIVEIRA *et. al.*, 2019).

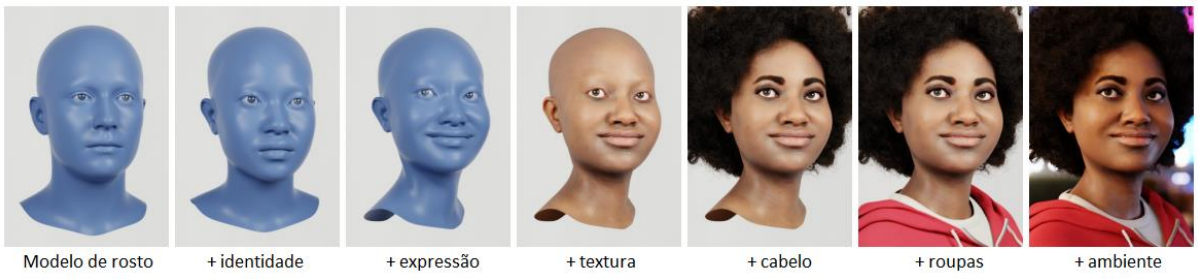
2.4 Utilização de Dados Sintéticos para Treinamento de Redes Neurais

Como mencionado no capítulo 1, sistemas que possuem IA e *machine learning* necessitam de uma grande quantidade de dados para que possam ser treinados de forma eficaz.

Dados sintéticos, como o nome sugere, são dados criados artificialmente, em vez de serem gerados por eventos reais. Comumente são originados com a ajuda de algoritmos e são usados para uma ampla gama de atividades, inclusive como conjuntos de teste para novos produtos e ferramentas, para validação de modelos e no treinamento de modelos de IA.

A Figura 18 ilustra o resultado do trabalho desenvolvido por WOOD *et. al.* (2021), um grupo de pesquisadores da Microsoft, que descreveram como combinar um modelo de rosto 3D paramétrico gerado processualmente com uma biblioteca abrangente de recursos criados à mão para renderizar imagens de treinamento com realismo e diversidade sem precedentes. O processo de síntese inclui o uso de texturas, cabelos e roupas para criar uma variedade infinita de pessoas diferentes a partir de uma imagem base.

Figura 18 - Construção de forma processual das faces sintéticas realistas



Fonte: Adaptado de WOOD *et. al.* (2021).

A utilização desse tipo de material é importante porque pode ser elaborado para atender as necessidades ou condições específicas que não estão disponíveis ou são raras em dados reais existentes, sendo possível influenciar todos os parâmetros e cenários que impactam nas imagens, minimizando as restrições associadas ao uso de dados regulados ou confidenciais.

Segundo Alexandre Linder, analista do Gartner, embora os dados reais sejam quase sempre melhores, geralmente são caros, desequilibrados, indisponíveis ou inutilizáveis devido às regulamentações de privacidade. Os dados sintéticos podem

ser um complemento eficaz ou uma alternativa aos dados reais, criando modelos de IA mais precisos. Quando combinados com dados reais, os dados sintéticos criam um conjunto de dados aprimorado que geralmente pode mitigar os pontos fracos dos dados reais.

2.4.1 Técnicas para Geração de Dados Sintéticos Utilizando *Deep Learning*

As redes neurais são especialmente hábeis em aprender uma distribuição de dados subjacente e generalizá-la. Isso permite que uma arquitetura de rede neural crie pontos de dados semelhantes, mas não idênticos, a amostras da distribuição original. Abaixo estão algumas técnicas usadas para gerar dados sintéticos.

2.4.1.1 Autoencoders Variacionais (Variational Autoencoders - VAE)

Os VAEs são modelos generativos não supervisionados que podem aprender a distribuição subjacente de dados e gerar um modelo complexo. Eles operam pegando uma distribuição original, transformando-a em uma distribuição latente e de volta ao espaço original (isso é conhecido como codificado-decodificado). Esse processo resulta em um “erro de reconstrução”, que o modelo visa minimizar.

Os VAEs são muito úteis para dados contínuos, mas menos eficazes para dados categóricos. Eles também são limitados em sua capacidade de gerar imagens ou outros tipos de dados não estruturados.

2.4.1.2 Redes Adversariais Geradoras (Generative Adversarial Networks - GAN)

GAN é um modelo generativo supervisionado que pode ser usado para gerar representações realistas e altamente detalhadas. Ele funciona treinando duas redes neurais, uma gerando pontos de dados falsos (um gerador) e a outra visando distinguir pontos de dados falsos e reais (um discriminador). Ao longo de milhares de ciclos de treinamento, o gerador se torna cada vez mais bem-sucedido na geração de pontos de dados falsos altamente realistas que podem “enganar” o gerador.

As GANs são especialmente bem-sucedidas na geração sintética de imagens, vídeos e outros dados não estruturados. Seu ponto fraco é que eles exigem

conhecimento especializado para construir e treinar, e que o modelo pode “colapsar” e começar a produzir um conjunto limitado de pontos de dados falsos muito semelhantes.

2.4.1.3 Campo de Radiância Neural (Neural Radiance Field - NeRF)

NeRF é um método de geração de novas visualizações de uma cena 3D parcialmente conhecida. O algoritmo recebe um conjunto de imagens, as interpola e adiciona novos pontos de vista do mesmo objeto.

O NeRF é uma maneira muito útil de gerar imagens adicionais e realistas a partir de um conjunto de imagens existentes. Seus pontos fracos são que é lento para treinar, lento para renderizar e pode gerar imagens de baixa qualidade.

2.4.1.4 Programas de Modelagem e Criação 3D e Python

É possível utilizar uma combinação de programas de modelagem e criação 3D (CAD) em conjunto com algoritmos em Python, linguagem de programação de alto nível criada por Guido van Rossum em 1991, para gerar uma base de dados sintética. Existem *software* que permitem modelar o objeto, alterar o tipo de material, alterar características do ambiente, entre outras configurações. Os programas SolidWorks e Blender são dois exemplos de ferramentas muito utilizadas para esse processo. O SolidWorks é um *software* CAD 3D para ilustração e montagem de peças. Com ele é possível criar, projetar e simular os projetos desenvolvidos. Já o Blender possui código aberto e permite a criação de objetos, animação, texturização, renderização, edição de vídeo, além de outros muitos atributos.

O método de criação de um conjunto sintético a partir desses recursos pode ser um pouco lento, mas é fácil de operar e entrega um resultado satisfatório.

2.5 Métricas de Avaliação de Resultados

Para analisar os resultados obtidos com CNNs são comumente utilizadas as métricas propostas por SOKOLOVA; LAPALME (2009), sendo elas: acurácia, precisão, *recall* e *score* F1, definidas abaixo.

a) acurácia: refere-se à quantidade de vezes que o modelo acertou as previsões possíveis;
b) precisão: é uma medida de quantas das previsões positivas feitas estão corretas (verdadeiros positivos);
c) *recall*: mede quantos casos positivos o classificador previu corretamente, sobre todos os casos positivos nos dados;
d) *score* F1: analisa o quão bom o modelo é para prever positivos, sendo positivo entendido como a classe que se quer prever.

As métricas relacionam as quantidades de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN), explicados abaixo.

a) verdadeiros positivos (VP): ocorre quando no conjunto real, a classe que é desejada prever foi prevista corretamente;
b) verdadeiros negativos (VN): ocorre quando no conjunto real, a classe que não é desejada prever foi prevista corretamente;
c) falsos positivos (FP): ocorre quando no conjunto real, a classe que é desejada prever foi prevista incorretamente;
d) falsos negativos (FN): ocorre quando no conjunto real, a classe que não é desejada prever foi prevista incorretamente.

Também será apresentada a matriz de confusão do modelo, uma análise de forma gráfica que relaciona as métricas citadas acima. As Equações 4, 5, 6 e 7 ilustram os cálculos de cada uma das métricas.

$$Acurácia = \frac{VP+VN}{VP+FP+VN+FN} \quad (4)$$

$$Precisão = \frac{VP}{VP+FP} \quad (5)$$

$$Recall = \frac{VP}{VP+FN} \quad (6)$$

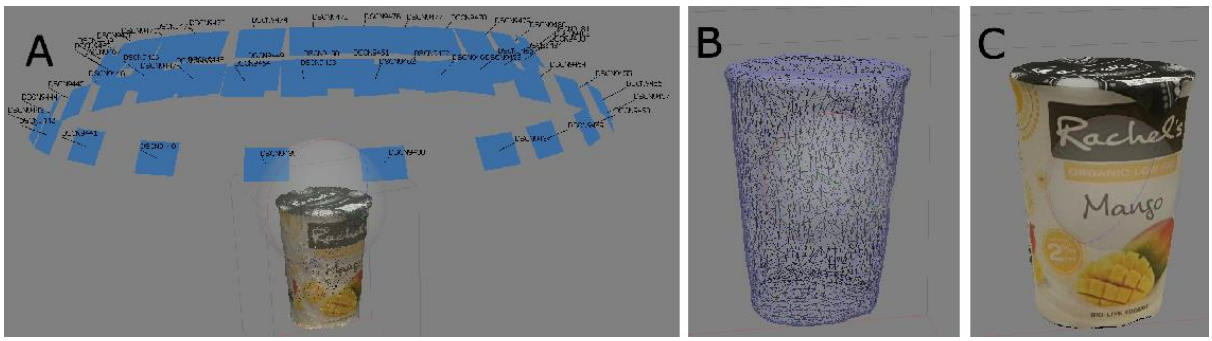
$$Score\ F1 = 2 \times \frac{Precisão \times Recall}{Precisão + Recall} \quad (7)$$

2.6 Trabalhos Correlatos

Redes neurais necessitam de um grande conjunto de dados para serem treinadas de forma eficaz. Com a indisponibilidade de bases reais que possuem boa quantidade de dados e considerando o elevado custo de criá-las a partir do zero, a utilização de dados sintéticos para treinamento e de dados reais para validação tornou-se comum para validação de modelos de classificação.

WONG *et. al.* (2019) propõe a varredura 3D de objetos físicos utilizando fotogrametria para gerar uma base de dados sintética e treinar uma CNN. Os materiais de estudo foram mantimentos de supermercado, contendo um total de 10 produtos escolhidos para a tarefa de classificação. O resultado foi promissor e conseguiu atingir uma precisão de classificação máxima de 95,8%. A Figura 19 ilustra o processo de geração de um dos objetos sintéticos.

Figura 19 - Visualização das etapas da fotogrametria: (A) calibração da câmera e geração da nuvem de pontos; (B) após a geração da malha; (C) após a geração da textura



Fonte: Adaptado de WONG *et. al.* (2019).

LOUZADA; DE PAULA (2021) apresentam uma base de dados sintética de implantes dentários formada por três diferentes modelos. Foram utilizadas imagens reais de radiografias para compor um conjunto de teste e ao treinar uma CNN com o conjunto sintético e testar com o conjunto real, o modelo preditivo obteve 71% de acurácia.

WISNIEWSKI; RANA; PETRUNIN (2022) utilizam uma CNN treinada com três diferentes modelos sintéticos de drones em cenários fictícios distintos para classificar drones verdadeiros em vídeos reais. Obteve-se uma acurácia geral de 92,4%.

KOHTALA; STEINERT (2021) avaliam a aplicabilidade de dados sintéticos baseados em modelos CAD para detectar automaticamente objetos no mundo real, utilizando uma única câmera de baixo custo. A abordagem é demonstrada e avaliada através de um estudo de caso envolvendo uma maquete física de uma planta industrial ligada a um ambiente virtual, destinado a facilitar a colaboração multidisciplinar e a visualização imersiva. Os resultados mostram a capacidade dos métodos de generalizar para dados reais, com precisão de até 87%.

O grupo de *design* e prototipagem do Centro de Pesquisa de Manufatura Avançada (Advanced Manufacturing Research Centre - AMRC) da Universidade de Sheffield realizou um estudo de caso para criar uma ferramenta que permite a geração automatizada de dados sintéticos para o treinamento de algoritmos de detecção de objetos de *machine learning*. A ferramenta é capaz de gerar os dados necessários para treinar um modelo usando apenas dados CAD e algumas imagens de fundo para colocar o objeto no contexto de onde ele será detectado na aplicação final do sistema de visão de máquina. Os resultados foram promissores, chegando a 95% de acurácia na classificação de objetos.

GASTELUM; SHEAD; HIGGINS (2020) utilizando um modelo de classificação ResNet-50 pré-treinado tem como objetivo explorar como imagens artificiais de braços manipuladores remotos podem ser usadas para reduzir - ou até eliminar - o requisito de imagens do mundo real durante o treinamento. Para o teste, foram empregados dois conjuntos de modelos: um treinado e testado exclusivamente em dados do mundo real e o outro treinado e testado exclusivamente em dados sintéticos. Os resultados obtidos foram de 86% e 90% de acurácia, respectivamente. Ao utilizar um modelo treinado apenas com imagens sintéticas e testar somente imagens reais, atingiu-se apenas 55% de acurácia.

MINHAS *et. al.* (2022) empregou um simulador virtual personalizado, especializado em sistemas climáticos variados que utiliza o Unity3d para encenar o clima com efeitos de iluminação precisos. O ambiente do simulador foi baseado na localização do mundo real do centro de Colchester, Reino Unido. Possui uma boa mistura de estradas abertas e estradas do centro da cidade. Um carro autônomo foi conduzido em diferentes horas do dia em condições climáticas variando de ensolarado, nublado, chuvoso e nebuloso com diferentes ângulos de câmera. O conjunto de dados final compreende 108.333 imagens, com aproximadamente 35.000 imagens por classe. A Figura 20 ilustra exemplos de imagens sintéticas adquiridas.

Os resultados mostram que as arquiteturas CNN de última geração treinadas no conjunto de dados sintéticos foram capazes de atingir uma precisão de até 74% quando testadas em um conjunto de dados do mundo real.

Figura 20 - Conjunto de dados meteorológicos sintéticos

Fonte: Adaptado de MINHAS *et. al.* (2022).

3 METODOLOGIA E DESENVOLVIMENTO

Neste capítulo são apresentados a metodologia e o desenvolvimento do trabalho. No início, é apresentado o objeto de estudo e em seguida os processos de criação da base de dados sintética e real. Por último, são relatados a seleção da arquitetura e do treinamento de um modelo de classificação, abordando as ferramentas utilizadas no processo.

3.1 Objetos de Estudo

Os objetos de estudo selecionados para subsidiar uma avaliação dos resultados da utilização de imagens sintéticas no treinamento de classificadores foram parafusos. Um parafuso pode ser definido pelo seu tipo de cabeça, geometria da rosca, fenda, formato do pescoço e ponta, no entanto, a presença dessas partes varia de acordo com cada um dos tipos. A Figura 21 ilustra as diferentes características que um parafuso pode conter.

Figura 21 - Diferentes características de um parafuso

Fonte: CISER (2020).

Os parafusos utilizados tiveram seus modelos classificados de acordo com a norma que os representa. Foram utilizadas duas normas: a da Organização

Internacional de Normalização (*International Organization for Standardization* - ISO) e do Instituto Alemão de Normalização (*Deutsches Institut für Normung* - DIN). Abaixo são listadas as normas utilizadas:

a) DIN 968: parafuso auto atarraxante com cabeça flangeada e fenda phillips;
b) DIN 6921: parafuso máquina com cabeça sextavada flangeada e fenda phillips;
c) ISO 1481: parafuso auto atarraxante com cabeça panela e fenda simples;
d) ISO 1580: parafuso máquina com cabeça panela e fenda simples;
e) ISO 2009: parafuso máquina com cabeça chata e fenda simples;
f) ISO 4017: parafuso máquina com cabeça sextavada;
g) ISO 4762: parafuso allen;
h) ISO 7045: parafuso máquina com cabeça panela e fenda phillips;
i) ISO 7046: parafuso máquina com cabeça chata e fenda phillips;
j) ISO 7049: parafuso auto atarraxante com cabeça lentilha e fenda phillips;
k) ISO 7050: parafuso auto atarraxante com cabeça chata e fenda phillips.

A norma DIN está presente na fabricação de parafusos, porcas, arruelas, barras roscadas e outros tipos de fixadores. Em alguns casos, ela é a única norma de referência para a fabricação. Já a norma ISO realiza a normatização de condutas e processos em organizações nos mais diferentes segmentos de mercado. O Quadro 1 relaciona as imagens dos parafusos reais utilizados no trabalho com suas respectivas classes.

Quadro 1 – Imagens reais de parafusos utilizados e classes correspondentes

DIN 968	DIN 6921	ISO 1481	ISO 1580
			
ISO 2009	**ISO 4017**	**ISO 4762**	**ISO 7045**
			
ISO 7046	**ISO 7049**	**ISO 7050**	
			

Fonte: Da autora (2022).

3.2 Criação de uma Base de Dados Sintéticos de Parafusos

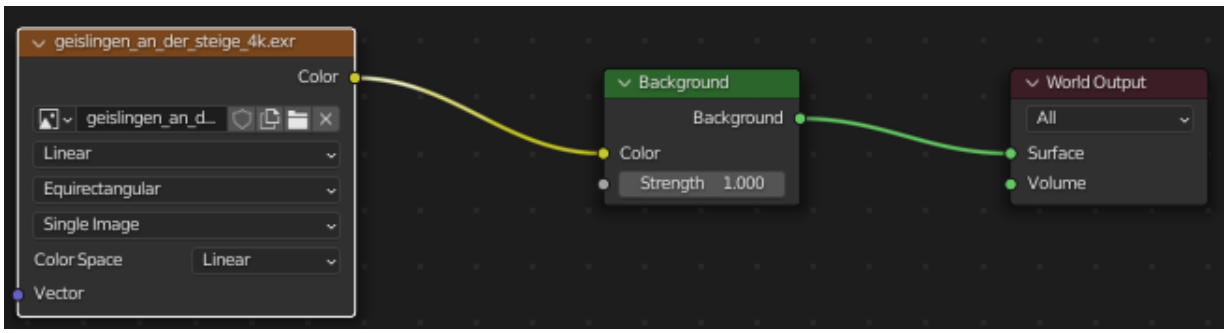
Para iniciar a formação da base de dados sintética, foram aproveitados modelos CAD 3D dos parafusos encontrados disponíveis gratuitamente na internet. Então, utilizou-se o sistema CAD 3D para adicionar roscas a eles a fim de torná-los mais parecidos com as amostras reais. Em seguida, o *software* Blender foi empregado para configurar a cena do ambiente virtual, adicionar o mapa de textura ao objeto sintético e realizar a renderização.

3.2.1 Configuração da Cena

Para melhorar a qualidade da renderização foram utilizadas configurações da cena e da textura do material do objeto. A configuração de cena 2D serve para aperfeiçoar a iluminação do ambiente da imagem. Existem sites que disponibilizam de forma gratuita arquivos *High Dynamic Range Imaging* (HDRI) e basta fazer o *download*, importar para o Blender e realizar os ajustes na aba "*layout*". No presente trabalho, esse ajuste foi através de blocos conectados entre si, onde cada bloco possui uma função com diferentes configurações.

Conforme ilustra a Figura 22, o primeiro bloco é onde o usuário importa o arquivo HDRI. O segundo bloco configura a influência da imagem no *background* (fundo) da cena e terceiro bloco indica que vai acessar as propriedades de mundo do projeto.

Figura 22 - Configuração da cena no Blender



Fonte: Da autora (2022).

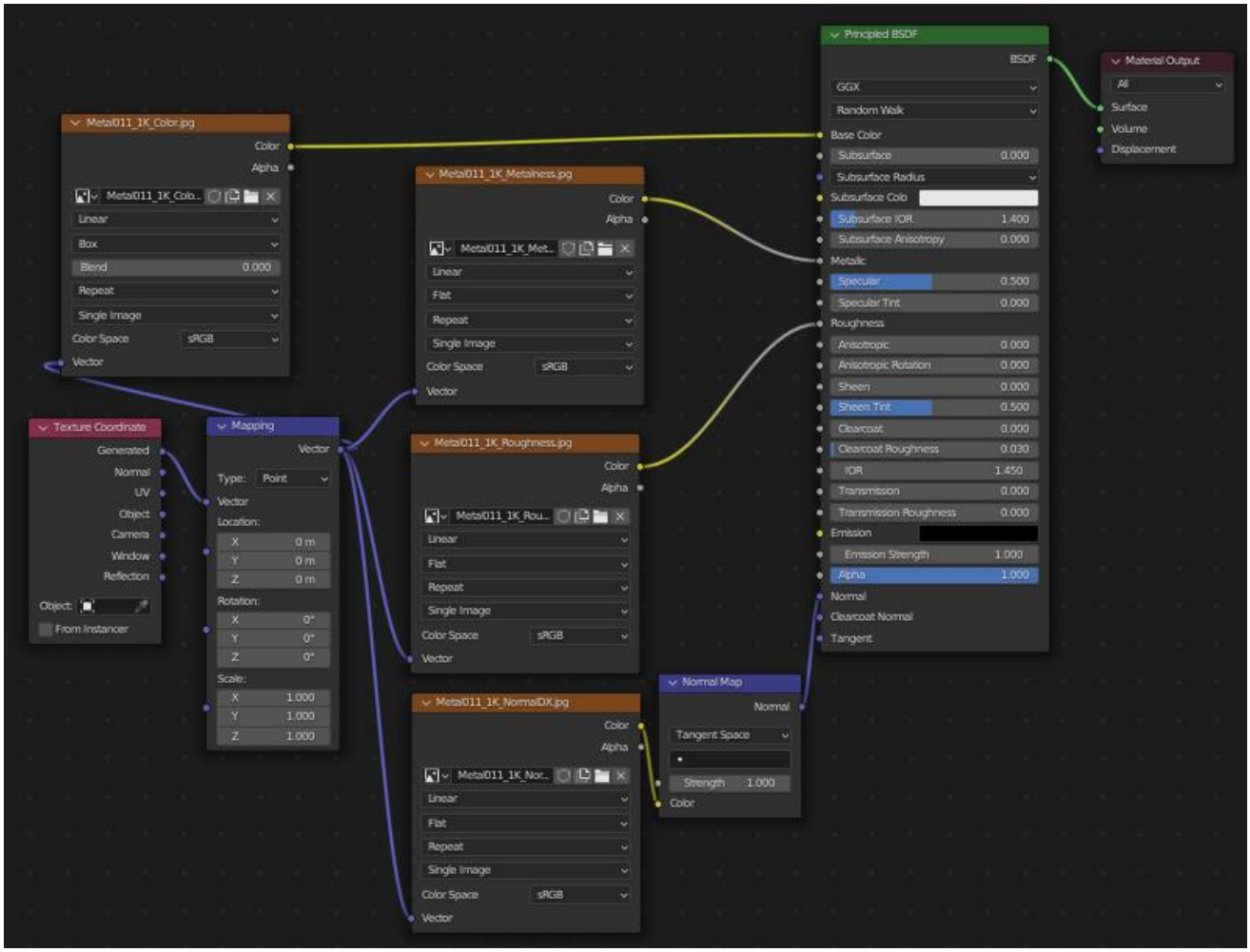
3.2.2 Configuração do Mapa de Textura

Para configurar o mapa de textura do modelo 3D com o objetivo de deixá-lo o mais próximo possível de um objeto real foi utilizada uma textura metálica *Physically Based Rendering* (PBR), que também é disponibilizada de forma gratuita na internet. Esse método de criação de materiais vem sendo amplamente utilizado em renderizações proporcionando um alto nível de realismo e consistência sob qualquer tipo de iluminação.

A configuração realizou-se através de blocos na aba "*shading*" e como ilustrado na Figura 23, foram ajustados a cor base da textura, a rugosidade (define se

a superfície será fosca ou brilhante), o mapa normal (simula o efeito de profundidade) e de metalicidade.

Figura 23 - Configuração do mapa de textura no Blender



Fonte: Da autora (2022).

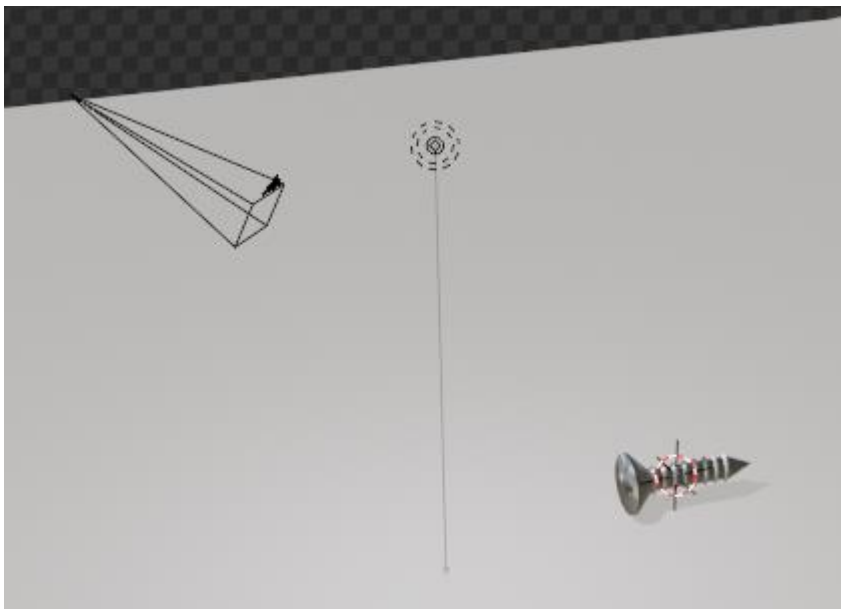
3.2.3 Renderização dos Objetos Sintéticos

O processo de renderização consiste em criar uma imagem 2D a partir de uma cena 3D. O *software* Blender oferece quatro fatores que o usuário pode controlar: iluminação, câmera, material do objeto e configurações de renderização. Diversos cálculos complexos são realizados nessa etapa com base nesses fatores e alguns renderizadores podem ser melhores que outros, dependendo da tarefa a ser executada, da matemática que usam e da maneira que o código foi escrito.

Para renderizar uma imagem de uma cena, deve-se primeiro determinar qual luz da cena está chegando a cada ponto no plano de visualização. Em seguida,

uma câmera define quais porções de uma cena serão visíveis dentro da imagem renderizada. Por padrão, uma cena contém uma câmera. O material define as qualidades artísticas de que um objeto é feito, produzindo efeito suave e uniforme, não sendo particularmente fiel à realidade. Assim, o uso de mapas de texturas que podem modificar as propriedades de superfície de um material é implementado em conjunto com a aplicação do material. Por fim, as configurações de renderização geram diferentes saídas que podem ser combinadas em uma única imagem. Podem ser adicionadas diferentes composições, como sombreamento, desfoque, nitidez, manipulação de cores, entre outros. A Figura 24 ilustra o ambiente virtual configurado para a realização das renderizações, possuindo uma câmera, uma luz tipo lâmpada, o objeto 3D e um plano de fundo.

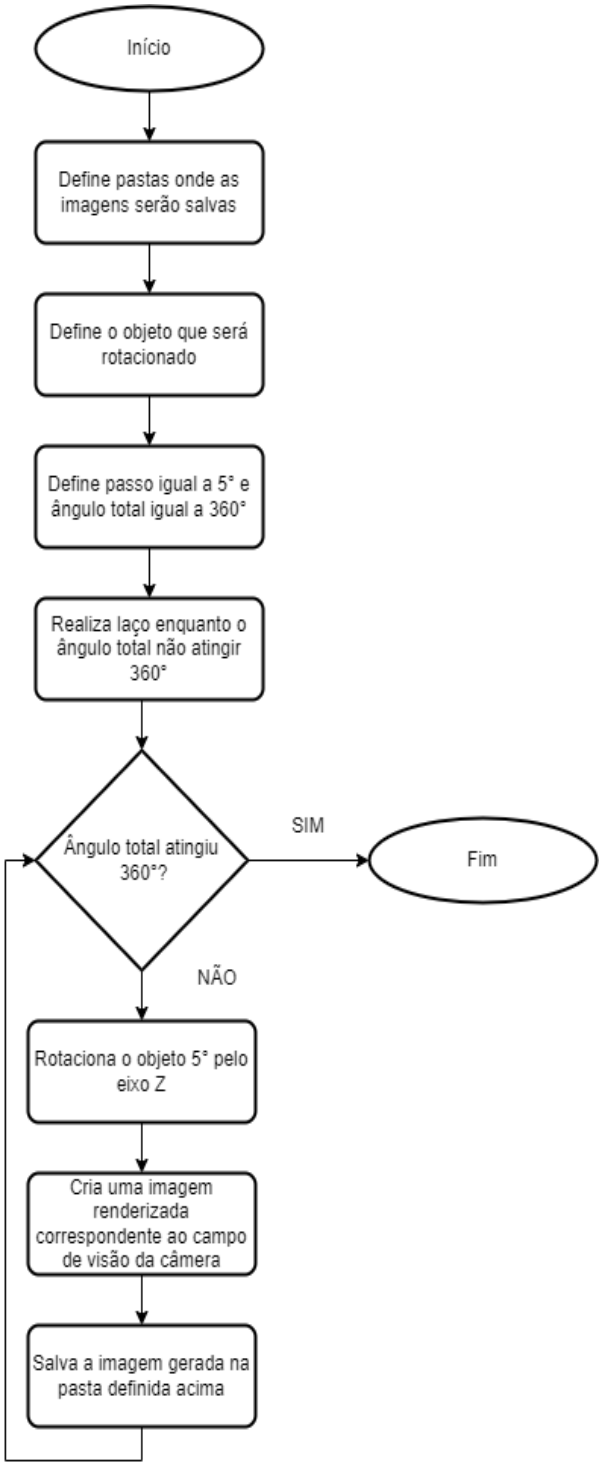
Figura 24 - Configuração do ambiente virtual do Blender



Fonte: Da autora (2022).

Após realizar as configurações do ambiente virtual e utilizando um algoritmo em Python, foi possível programar ações para executar as funções do fluxograma apresentado na Figura 25. Inicialmente define-se uma pasta alvo onde as imagens serão salvas, o objeto que será rotacionado e as configurações de renderização, que nesse caso foram duas: passo e ângulo total, ambos em graus. Em seguida, é criado um laço que definirá a condição que o algoritmo será executado, indicando o eixo em que o objeto será rotacionado, neste caso foi o eixo Z, e o nome e local do arquivo que será gerada a imagem sintética.

Figura 25 - Fluxograma do processo de renderização



Fonte: Da autora (2022).

A Figura 26 ilustra diferentes resultados do processamento realizado pelo algoritmo descrito acima para um único parafuso.

Figura 26 - Alguns resultados gerados do modelo ISO 7049

Fonte: Da autora (2022).

Outros exemplos das imagens geradas para diferentes modelos de parafusos são apresentados na Figura 27.

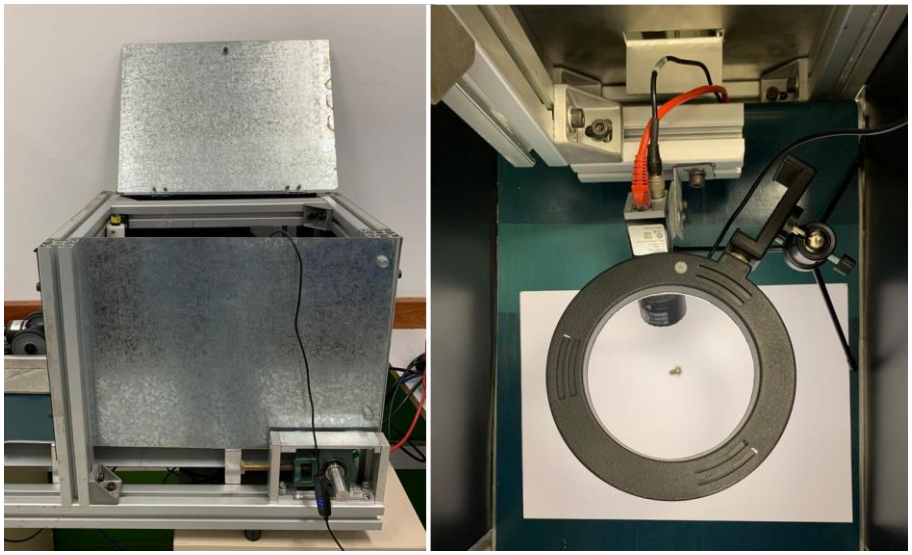
Figura 27 - Exemplos de imagens sintéticas geradas no Blender: A - ISO 7050, B - ISO 2009 e C - ISO 4017

Fonte: Da autora (2022).

3.3 Criação de uma Base de Dados Reais de Parafusos

Para a aquisição das imagens dos modelos reais de parafusos foi utilizada uma câmera industrial Basler em conjunto com uma lanterna em formato de anel (*ring light*) para ajudar a melhorar a iluminação do ambiente. A câmera foi colocada dentro de uma estrutura em forma de caixa que visava diminuir a interferência externa na captura das imagens e os parafusos, um a um, foram rotacionados manualmente. A Figura 28 ilustra o ambiente de aquisição das imagens reais.

Figura 28 - Ambiente de aquisição de imagens dos parafusos reais



Fonte: Da autora (2022).

O *software* utilizado para realizar a comunicação com a câmera foi o Pylon Viewer, que fornece um conjunto de funções para configurar a câmera, como correção de vinhetas, indicador de nitidez, gerenciador de largura de banda e muito mais. Um cabo de rede conectou a câmera ao computador, assim o *software* acessou as imagens que estavam sendo transmitidas e então bastou capturá-las e salvá-las. Alguns exemplos de imagens adquiridas são apresentados na Figura 29, as quais podem ser comparadas com suas correspondentes ilustradas, anteriormente, na Figura 27.

Figura 29 - Exemplos de imagens reais adquiridas: A - ISO 7050, B - ISO 2009 e C - ISO 4017

Fonte: Da autora (2022).

3.4 Seleção da Arquitetura e Treinamento do Modelo

Após ter gerado uma base de dados inicial com a renderização dos modelos 3D e imagens reais, utilizou-se a biblioteca Fastai para criar uma rede neural convolucional com arquitetura ResNet-34 e treiná-la para classificação de modelos de parafusos a partir das bases de dados sintética e imagens reais. A Fastai é uma biblioteca de *deep learning* proposta por HOWARD; GUGGER (2020) que fornece componentes de alto nível e tem como objetivo tornar o treinamento de redes neurais profundas o mais fácil possível e, ao mesmo tempo, rápido e preciso utilizando práticas modernas.

Como explicado na subseção 2.3.3, os dados são separados em três grupos: treinamento, validação e testes. Neste caso, 80% do banco de imagens foi utilizado para treinamento, 20% para validação e os testes foram realizados das seguintes maneiras:

a) modelo treinado com imagens sintéticas e avaliando imagens sintéticas;
b) modelo treinado com imagens sintéticas e avaliando imagens reais;
c) modelo treinado com imagens reais e avaliando imagens reais.

Durante o processo de treinamento, o modelo irá percorrer a base de dados e aprenderá características e padrões. É possível configurar a quantidade de vezes (épocas) que o conjunto de dados será apresentado para a rede neural, no caso de treinamento com imagens sintéticas foram 200 épocas e para as imagens reais, 20 épocas. Também pode-se definir uma fração de dados do conjunto total (*batch size*) que será apresentada para a CNN em cada época, ambos utilizaram um *batch size* igual a 32.

O ambiente de execução utilizado foi o Google Colaboratory, um serviço de nuvem gratuito hospedado pelo próprio Google que permite acesso a uma Unidade de Processamento Gráfico (*Graphics Processing Unit* - GPU) sem nenhum custo financeiro para o usuário. O treinamento de uma rede neural em um serviço de nuvem possibilita utilizar um ambiente virtual sem precisar gastar tempo e processamento de computador com instalação de *software* e bibliotecas.

As GPUs foram originalmente projetadas para acelerar a renderização de gráficos 3D. Com o tempo, eles se tornaram mais flexíveis e programáveis, aprimorando suas capacidades. Isso permitiu que os programadores gráficos criassem efeitos visuais mais interessantes e cenas realistas com técnicas avançadas de iluminação e sombreamento. Outros desenvolvedores também começaram a aproveitar o poder das GPUs para acelerar drasticamente cargas de trabalho adicionais em computação de alto desempenho (*High Performance Computing* - HPC), aprendizagem profunda e muito mais (INTEL, 2023).

O *deep learning* carece de um grande poder computacional e as principais bibliotecas utilizam GPUs para aumentar a velocidade de processamento. Treinar um algoritmo de *deep learning* utilizando uma Unidade de Processamento Central (*Central Processing Unit* - CPU) pode levar dias, mas com o processamento feito em sistemas baseados em GPUs, a mesma tarefa pode levar horas.

4 ANÁLISE DOS RESULTADOS

Neste capítulo é realizada a análise dos resultados obtidos com os modelos treinados com imagens sintéticas e reais. São apresentadas as matrizes de confusão, tabelas com os valores das métricas avaliadas, já definidas anteriormente na seção 2.5 desse trabalho, e comparação dos resultados.

4.1 Resultados do Treinamento do Modelo com Dados Sintéticos

O conjunto de dados utilizado no modelo de classificação foi organizado de forma a conter, para cada tipo de parafuso considerado, 72 imagens sintéticas e 5 imagens reais. Mesclar dados sintéticos com dados reais possibilita a CNN a melhorar a performance na extração de características e aprendizagem de padrões. A quantidade total de imagens no banco de dados sintéticos corresponde a 847 imagens, divididas em 678 para treinamento e 169 para validação.

A Quadro 2 apresenta os resultados obtidos para cada uma das métricas avaliadas após o treinamento do modelo.

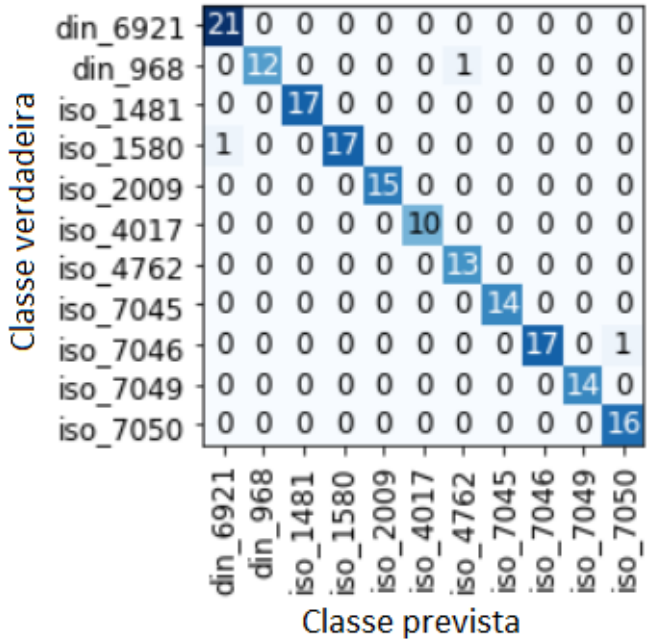
Quadro 2 - Resultados do treinamento do modelo com dados sintéticos

Acurácia (%)	**Precisão (%)**	***Recall* (%)**	***Score* F1 (%)**
98,2	98,4	98,2	98,3

Fonte: Da autora (2022).

Também é possível analisar essas métricas de forma gráfica através de uma matriz de confusão, que relaciona a classe verdadeira com a classe prevista, e apresenta quantas predições foram realizadas correta e incorretamente pelo modelo, conforme ilustrado na Figura 30.

Figura 30 - Matriz de confusão do modelo treinado com imagens sintéticas



Fonte: Da autora (2022).

4.1.1 Resultado dos Testes com as Imagens Sintéticas e Reais

Para conseguir analisar um banco de imagens, primeiro foi importado o modelo de CNN previamente salvo, que possui todos os pesos e parâmetros da etapa de treinamento. Em seguida, foi necessário informar o diretório das imagens de teste e com isso foram calculadas as predições com base no modelo treinado. Por fim, foi realizada uma comparação entre a predição e a verdadeira classe para cada uma das imagens.

Ao avaliar as imagens sintéticas, foi obtida uma acurácia de 99,6%, ou seja, 844 das 847 imagens totais foram classificadas corretamente. Já na análise com as imagens reais, foi possível chegar a uma acurácia de 93%, ou seja, 252 das 271 imagens foram preditas com acerto. Alguns exemplos são ilustrados na Figura 31, onde V é a classe verdadeira e P é a classe prevista.

Figura 31 - Exemplos de classificação utilizando o modelo treinado com imagens sintéticas

Fonte: Da autora (2022).

4.2 Resultados do Treinamento do Modelo com Dados Reais

O conjunto de dados reais possui, em média, 24 imagens para cada uma das 11 classes, sendo a quantidade total igual a 271, divididas em 217 para treinamento e 54 para validação.

O Quadro 3 apresenta os resultados obtidos para cada uma das métricas avaliadas após o treinamento do modelo.

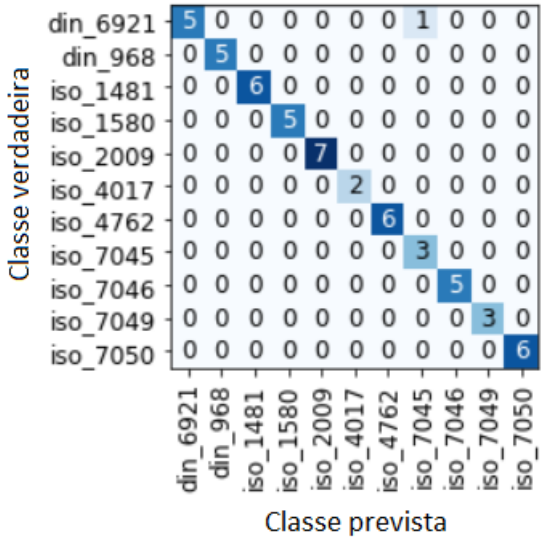
Quadro 3 - Resultados do treinamento do modelo com dados reais

Acurácia (%)	**Precisão (%)**	***Recall* (%)**	***Score* F1 (%)**
98,1	97,7	98,5	97,9

Fonte: Da autora (2022).

A Figura 32 apresenta a matriz de confusão para análise gráfica dos resultados listados acima.

Figura 32 - Matriz de confusão do modelo treinado com imagens reais

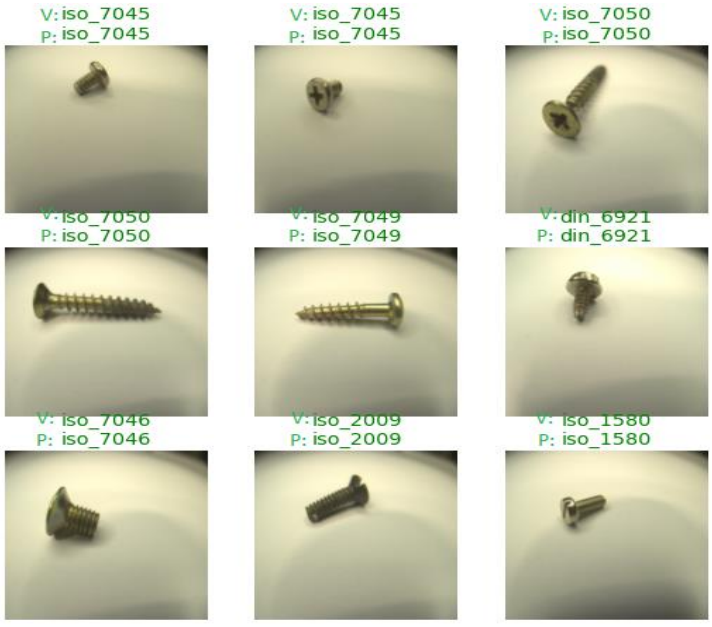


Fonte: Da autora (2022).

4.2.1 Resultado dos Testes com as Imagens Reais

Ao avaliar as imagens reais, foi obtida uma acurácia de 99,6%, ou seja, 270 das 271 imagens totais foram classificadas corretamente. Alguns exemplos de classificação são ilustrados na Figura 33.

Figura 33 - Exemplos de classificação utilizando o modelo treinado com imagens reais



Fonte: Da autora (2022).

4.3 Avaliação dos Resultados Obtidos

Com os resultados obtidos, é possível verificar que o teste realizado com o modelo treinado com imagens sintéticas atingiu um bom desempenho e a utilização de dados artificiais mesclados com dados reais é efetiva para modelos de classificação, possibilitando um aprendizado mais profundo para desenvolver a atividade proposta. Ao utilizar o modelo para classificar somente imagens reais, houve 3,6% de diferença na acurácia quando comparada com o valor obtido na classificação de apenas imagens sintéticas. Entretanto, não deixa de ser um bom resultado, pois deve-se considerar o fato de muitas vezes as imagens adquiridas em laboratórios não conseguirem apresentar uma quantidade rica em detalhes quanto as imagens geradas sinteticamente, podendo gerar uma confusão para o modelo.

O teste realizado com o modelo de classificação treinado com imagens reais também teve um resultado satisfatório, errando apenas uma classificação.

O Quadro 4 reúne os resultados obtidos durante os testes.

Quadro 4 - Resultados obtidos durante os testes

Teste	Acurácia
Modelo treinado com imagens sintéticas e avaliando imagens sintéticas	99,6% (844/847)
Modelo treinado com imagens sintéticas e avaliando imagens reais	93% (252/271)
Modelo treinado com imagens reais e avaliando imagens reais	99,6% (270/271)

Fonte: Da autora (2022).

5 CONSIDERAÇÕES FINAIS

Nesse trabalho foi realizado um estudo sobre a utilização de imagens sintéticas na criação de modelos de classificação de objetos empregando redes neurais convolucionais. Para atingir os objetivos específicos, foram aplicadas técnicas de visão computacional e *deep learning*.

O fluxo de geração de imagens sintéticas utilizando *software* de modelagem 3D e Python se mostrou competente. Tanto o SolidWorks quanto o Blender possuem inúmeras ferramentas que possibilitam gerar imagens artificiais realísticas a partir da configuração de um ambiente virtual. Já o Python, é uma linguagem fácil de entender e programar, que propicia o desenvolvimento de diversas aplicações.

A utilização de CNNs treinadas com base de dados sintéticos e com adição de pouquíssimas imagens reais se mostrou muito eficiente tanto para a classificação de imagens sintéticas quanto reais. Na maioria das vezes, as imagens artificiais conseguem transmitir um nível maior de detalhes do que as imagens reais, onde o parafuso real pode ter a sua geometria um pouco afetada devido a sua utilização no dia a dia. Então, a junção dos dois tipos de imagens é uma maneira de deixar o banco de dados mais completo e preparado para diferentes cenários.

Os resultados alcançados, ilustrados no Quadro 4, foram satisfatórios. Tais valores são condizentes e em alguns casos, até superiores, aos obtidos por outros trabalhos da literatura, apresentados na seção 2.6, que também utilizaram o método de classificação de objetos. Sugere-se para trabalhos futuros realizar o cálculo das incertezas dos valores obtidos, uma vez que a incerteza é uma estimativa que quantifica a confiabilidade do resultado.

Nesse trabalho, a escolha de acrescentar 5 imagens reais ao banco de dados sintéticos foi de forma aleatória, portanto, para trabalhos futuros sugere-se a investigação de uma porcentagem que represente a quantidade mínima de imagens reais que se deve acrescentar no banco de dados sintéticos, a fim de verificar um número que torne o modelo de classificação bom e confiável. Também sugere-se realizar outras variações de configuração do ambiente virtual e de mapas de textura no objeto 3D.

A arquitetura ResNet-34 foi utilizada com base em estudos correlacionados, porém sugere-se como trabalho futuro a análise dos resultados utilizando outras arquiteturas, a fim de realizar uma comparação.

Por fim, pelo banco de imagens sintéticas possuir um número muito maior de elementos do que a base de dados reais, foi necessário realizar um treinamento com muito mais épocas quando comparado com o modelo treinado com imagens reais para atingir um resultado satisfatório.

6 REFERÊNCIAS

MAJIN ERAZO, Jhon Jamilton et al. Desenvolvimento de um sistema de contagem e classificação de veículos utilizando redes neurais convolucionais. 2021.

IT FORUM. **O que são dados sintéticos?** Disponível em: https://itforum.com.br/noticias/o-que-sao-dados-sinteticos-dados-gerados-para-ajudar-sua-estrategia-de-ia/. Acesso em: 1 jul. 2022.

BARBOSA, Guilherme NN et al. Segurança em Redes 5G: Oportunidades e Desafios em Detecção de Anomalias e Predição de Tráfego baseadas em Aprendizado de Máquina. **Sociedade Brasileira de Computação**, 2021.

DE OLIVEIRA, BERNADO AUGUSTO GODINHO et al. Avaliação de técnicas de transferência de aprendizado em CNNs. In: **Congresso Brasileiro de Automática-CBA**. 2019.

Gartner. **Is Synthetic Data the Future of AI?** Disponível em: https://www.gartner.com/en/newsroom/press-releases/2022-06-22-is-synthetic-data-the-future-of-ai. Acesso em: 22 set. 2022.

WISNIEWSKI, Mariusz; RANA, Zeeshan A.; PETRUNIN, Ivan. Drone model classification using convolutional neural network trained on synthetic data. **Journal of Imaging**, v. 8, n. 8, p. 218, 2022.

WONG, Matthew Z. et al. Synthetic dataset generation for object-to-model deep learning in industrial applications. **PeerJ Computer Science**, v. 5, p. e222, 2019.

LOUZADA, Henrique Almeida; DE PAULA, Maria Inês Lage. Classificando Modelos de Implantes Dentários Usando Redes Neurais Convolucionais com Dados Sintetizados. In: **Anais Estendidos do XXXIV Conference on Graphics, Patterns and Images**. SBC, 2021. p. 195-200.

SOKOLOVA, Marina; LAPALME, Guy. A systematic analysis of performance measures for classification tasks. **Information processing & management**, v. 45, n. 4, p. 427-437, 2009.

DATA SCIENCE ACADEMY. Deep Learning Book. Disponível em: https://www.deeplearningbook.com.br. Acesso em: 29 set. 2022.

YAMASHITA, Rikiya et al. Convolutional neural networks: an overview and application in radiology. **Insights into imaging**, v. 9, n. 4, p. 611-629, 2018.

LECUN, Yann et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278-2324, 1998.

LECUN, Yann et al. Handwritten digit recognition with a back-propagation network. **Advances in neural information processing systems**, v. 2, 1989.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: **ADVANCES in neural information processing systems**. [S.l.: s.n.], 2012. P. 1097–1105.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SZEGEDY, Christian et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2015. p. 1-9.

HE, Kaiming et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 770-778.

SRIVASTAVA, Nitish et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, v. 15, n. 1, p. 1929-1958, 2014.

KOHTALA, Sampsa; STEINERT, Martin. Leveraging synthetic data from CAD models for training object detection models–a VR industry application case. **Procedia CIRP**, v. 100, p. 714-719, 2021.

GASTELUM, Zoe Nellie; SHEAD, Timothy; HIGGINS, Michael. **Synthetic Training Images for Real-World Object Detection**. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

MINHAS, Saad et al. Weather Classification by Utilizing Synthetic Data. **Sensors**, v. 22, n. 9, p. 3193, 2022.

AMCR. Generating synthetic data to train accurate AI object detection. Disponível em: https://www.amrc.co.uk/case-studies/generating-synthetic-data-to-train-accurate-ai-object-detection. Acesso em: 20 out. 2022.

HOWARD, Jeremy; GUGGER, Sylvain. Fastai: a layered API for deep learning. **Information**, v. 11, n. 2, p. 108, 2020.

VON WANGENHEIM, Aldo. **Reconhecimento de Imagens**: redes para classificação de imagens e reconhecimento de objetos em cenas. 2018. Disponível em: https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningreconhecimento-de-imagens. Acesso em: 20 out. 2022.

WOOD, Erroll et al. Fake it till you make it: face analysis in the wild using synthetic data alone. In: **Proceedings of the IEEE/CVF international conference on computer vision**. 2021. p. 3681-3691.

BELENUS. **Parafusos e Porcas.** Disponível em: http://aldy.com.br/wp-content/uploads/2015/11/CATALOGO-PARAFUSOS-BELENUS.pdf. Acesso em: 4 dez. 2022.

CISER. **Anatomia do parafuso: conheça cada parte da peça.** Disponível em: https://blog.ciser.com.br/tudo-sobre-fixadores/anatomia-do-parafuso-conheca-cada-parte-dessa-peca/. Acesso em: 4 dez. 2022.

INTEL. **What Is a GPU?** Disponível em: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html. Acesso em: 13 jan. 2023.