INSTITUTO FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
CURSO DE ENSINO SUPERIOR DE ENGENHARIA MECATRÔNICA

RAMON BRIGNOLI

ESTUDO DE REDES NEURAIS CONVOLUCIONAIS E SUA APLICAÇÃO PARA CLASSIFICAÇÃO DE AMBIENTES

FLORIANÓPOLIS
2021

RAMON BRIGNOLI

ESTUDO DE REDES NEURAIS CONVOLUCIONAIS E SUA APLICAÇÃO PARA CLASSIFICAÇÃO DE AMBIENTES

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 Bacharel em Engenharia Mecatrônica.

Orientador:
Dr. Eng. Mauricio Edgar Stivanello

FLORIANÓPOLIS
2021

Ficha de identificação da obra elaborada pelo autor.

Brignoli, Ramon
ESTUDO DE REDES NEURAIS CONVOLUCIONAIS E SUA APLICAÇÃO PARA CLASSIFICAÇÃO DE AMBIENTES / Ramon Brignoli; orientação de Maurício Edgar Stivanello. - Florianópolis, SC, 2021.
45 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. Classificação. 2. Redes Neurais Convolucionais. 3. Visão Computacional. I. Stivanello, Maurício Edgar. II. Instituto Federal de Santa Catarina. III. ESTUDO DE REDES NEURAIS CONVOLUCIONAIS E SUA APLICAÇÃO PARA CLASSIFICAÇÃO DE AMBIENTES.

RAMON BRIGNOLI

ESTUDO DE REDES NEURAIS CONVOLUCIONAIS E SUA APLICAÇÃO PARA CLASSIFICAÇÃO DE AMBIENTES

Este trabalho foi julgado adequado para obtenção do título de Bacharelado em Engenharia Mecatrônica, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Florianópolis, 16 de setembro de 2021.

Documento assinado digitalmente
MAURICIO EDGAR STIVANELLO
Data: 23/09/2021 14:04:24-0300
CPF: 901.390.629-04

Prof. Mauricio Edgar Stivanello, Dr
Orientador
Instituto Federal de Santa Catarina

Documento assinado digitalmente
FRANCISCO RAFAEL MOREIRA DA MOTA
Data: 23/09/2021 16:19:42-0300
CPF: 012.003.153-16

Prof. Francisco Rafael Moreira da Mota , Dr.
Instituto Federal de Santa Catarina

Documento assinado digitalmente
DELCINO PICININ JUNIOR
Data: 23/09/2021 16:04:40-0300
CPF: 801.195.300-15

Prof. Delcino Picinin Junior , Dr
Instituto Federal de Santa Catarina

À todos aqueles que se fizeram presentes
nos momentos de necessidade,
Obrigado por tudo.

AGRADECIMENTOS

Inicio meus agradecimentos exaltando o trabalho dos professores, servidores, técnicos e terceirizados do Instituto Federal de Santa Catarina. Todos certamente tiveram parte nesta caminhada, me ensinando, corrigindo ou mesmo facilitando o dia a dia na instituição. Em especial, agradeço ao professor Maurício Stivanello por me guiar durante este trabalho, não poderia ter desejado melhor orientação.

Agradeço infinitamente também à heroína Samira Cifuentes, professora e minha mãe. Durante todos meus anos de vida me mostrou e demonstrou o valor da educação, e sempre lutou para que eu tivesse a melhor delas.

Por fim, devo agradecer à minha companheira Gabriela, que esteve comigo no momento da minha aprovação no vestibular, e que depois destes anos, irá comemorar junto a mim a minha graduação. Obrigado pelo seu suporte, certamente nada disso teria acontecido se não fosse sua presença na minha vida.

RESUMO

Este trabalho apresenta a proposta de um sistema de reconhecimento de imagens baseado em aprendizado com Redes Neurais Convolucionais, com o intuito de classificar as fotos de imóveis por tipo de ambiente como sala, cozinha e quarto. Com este serviço integrado em uma aplicação web de locação e venda de imóveis, é possível categorizar todo o catálogo e a partir disso, ordenar apresentações, coletar dados, entre outras funções. Para tal, serão abordados os conceitos e técnicas necessárias para criar um classificador e assim obter um sistema automatizado para realização desta tarefa, economizando tempo e criando uma vantagem competitiva para com as outras plataformas. O sistema foi desenvolvido utilizando principalmente a biblioteca *Fastai*, escrita na linguagem de programação Python, e com o auxílio do ambiente de desenvolvimento Google Colab para treinamento e testes. Os objetivos deste trabalho foram atingidos com sucesso, ao gerar um modelo classificador com uma acurácia de 96,94% e tempo médio de inferência de 80ms utilizando a arquitetura Resnet com 50 camadas treinada por 10 épocas em um banco de dados de aproximadamente 16 mil fotos separadas em 6 classes.

Palavras-Chave: Classificação; Redes Neurais convolucionais; Visão computacional;

ABSTRACT

This paper presents the proposal of an image recognition system based on Machine Learning and Convolutional Neural Networks, to classify residential pictures based on environments such as living room, kitchen, and bedroom. With this service integrated into a real estate application, it's possible to categorize the whole catalog and order it, collect data, and more. To achieve that, the concepts and techniques necessary to create a classifier and develop an automated system that realizes the described work will be detailed. This system was developed using the Fastai library, written in Python, using the Google Colab as a developing environment for training, test, and validation. The goals of this project were achieved, by creating a classification model with an accuracy of 96,94%, and an average processing time of 80 ms, using the Resnet architecture with 50 layers, trained with 10 epochs, in a database of almost 16 thousand pictures split into 6 labels.

Keywords: Computer Vision, Convolutional Neural Networks, Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O problema em questão abordado neste projeto, é de imobiliárias convencionais que têm interesse em digitalizar seus anúncios. Segundo a revista Exame (EXAME, 2020), a busca por imobiliárias digitais aumentou mais de 33% entre agosto de 2019 e agosto de 2020, já a busca pela frase 'Casas para alugar' teve um aumento de 668% na sua procura online. Para atender este mercado e continuarem competitivas, tais imobiliárias buscam desenvolver ou ingressar em plataformas na rede e divulgar seu material neste meio. Porém, este é um processo que pode levar muito tempo dependendo da quantidade de informação gerada por cada empresa (NURIELE, 2021).

Com o intuito de acelerar este procedimento, há necessidade de reduzir o trabalho manual e repetitivo de descrever, categorizar e organizar dezenas de milhares de imóveis anunciados, cada um com até dezenas de fotos ilustrando ambientes em diversos ângulos diferentes. A proposta apresentada neste trabalho visa automatizar este sistema de classificação de imagens em uma aplicação Web de locação e vendas de imóveis, e dessa forma categorizar imagens dos ambientes internos e externos do catálogo de imobiliárias em diferentes grupos como sala, cozinha e banheiro entre outros.

A fim de classificar ambientes, primeiro é necessário interpretar a imagem. Isso é algo corriqueiro na vida de seres humanos, e também está se tornando cada dia mais presente em programas de computadores, desde um algoritmo que reconhece os caracteres de uma placa de automóvel na foto de uma multa, até um que reconheça padrões de células cancerígenas em algum exame de imagem, antes mesmo deste passar pela avaliação de um médico. Para realizar tarefas como as citadas acima, são necessárias técnicas de visão computacional. Esta é uma área que procura desenvolver formas de uma máquina enxergar o ambiente a sua volta, coletar dados de câmeras, sensores e scanners, extrair informações úteis para reconhecer, manipular e até tomar decisões sobre objetos baseando-se em inteligência artificial (BALLARD, et.al. 1982).

Para que um computador consiga efetivamente tomar as próprias decisões, é necessário antes adquirir, tratar e modelar os dados, que neste caso são imagens. Nesta etapa de modelagem, deve-se ter o conhecimento

sobre os dados, sobre as regras de negócio envolvidas, bem como em técnicas e algoritmos disponíveis.

A Aprendizagem profunda, do inglês 'deep learning', é uma técnica de aprendizado de máquina que ensina os computadores a fazer o que humanos estão acostumados: aprender pelo exemplo. Estes tipos de algoritmos processam dados em camadas de unidades básicas, ou neurônios artificiais, de forma semelhante ao que fazem os nossos cérebros (HAYKIN et.al. 2009).

A proposta do presente trabalho foi implementar as técnicas de aprendizado profundo no contexto de classificação de imagens. Para isto foi necessário estudar os diferentes tipos de redes neurais artificiais comumente utilizadas para processamento e classificação de imagens, e identificar aquela que mais se adequou ao projeto. Posteriormente, treinar este classificador a identificar os padrões de formatos e objetos de cada ambiente, que de forma autônoma aplique esta classificação a novas entradas.

1.1 Objetivos

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos para este trabalho.

Objetivo geral:

- Desenvolver um sistema de visão computacional para categorizar fotos de imóveis do banco de dados de uma imobiliária.

Objetivos específicos:

- Estudar redes neurais convolucionais e métodos de implementação;
- Organizar a base de imagens a ser utilizada no treinamento dos classificadores;
- Projetar e implementar diferentes algoritmos necessários à criação dos classificadores;
- Realizar o treinamento de diferentes classificadores;
- Validar e analisar os resultados frente aos requisitos do sistema;

- Implementar um serviço que permita utilizar o classificador através de requisições;
- Documentar o trabalho.

2 DESENVOLVIMENTO

Neste capítulo serão abordados os conceitos e definições básicas de aprendizado de máquina, redes neurais, arquiteturas e métricas.

2.1 O setor imobiliário e as mudanças motivadas pelas novas tecnologias

O setor imobiliário é composto por empresas que se dedicam ao gerenciamento de todos os processos comerciais e legais necessários para transações e aluguéis de imóveis. Neste contexto existem diversas empresas que acompanharam o ritmo de mudanças tecnológicas das últimas décadas, mas também há as que executam todo o trabalho da forma tradicional.

Neste segundo tipo de empresas, o processo habitual de operação consiste no contato pessoal entre o cliente e o corretor, a apresentação de forma presencial dos imóveis, e a documentação toda feita manualmente. Este tipo de operação dificulta, encarece e torna todo o processo mais demorado (NURIELE,2021).

Já as empresas que acompanharam o ritmo tecnológico, ou que estão em transição para tal, tendem a disponibilizar seu material de forma digital, utilizando softwares para apresentação inicial dos imóveis, gerenciamento de visitas, negociação e documentação. Estas têm como desafio toda a organização dos dados referentes aos imóveis, uma vez que a apresentação destes tem de ser atrativa ao consumidor em meio a quantidade de opções disponíveis na internet.

Realizar tal organização exige, entre diversos outros fatores, classificar as fotos por tipo de ambiente, o que proporciona o gerenciamento da apresentação dos imóveis. Fazer este processo de forma manual pode ser custoso devido a quantidade de dados disponível, portanto automatizar este processo, significaria economia de tempo e trabalho repetitivo.

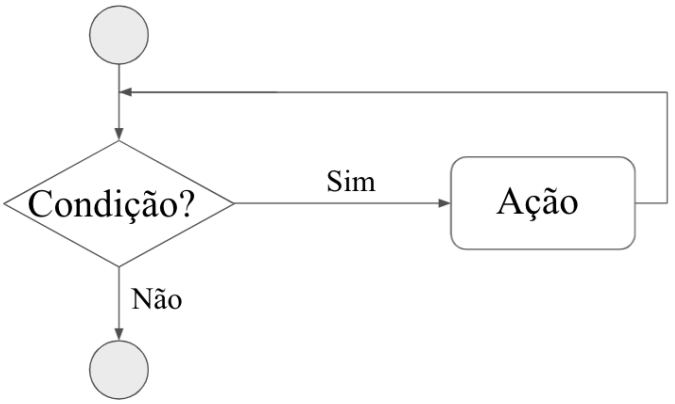
2.2 Aprendizado de máquinas e Redes Neurais Artificiais

Para falar de redes neurais artificiais (RNA) em geral, é necessário inicialmente explicar o conceito de algoritmos. "Os algoritmos fazem parte do dia-a-dia das pessoas. As instruções para o uso de medicamentos, as indicações de como montar um aparelho qualquer, uma receita de culinária são alguns exemplos

de algoritmos. Um algoritmo pode ser visto como uma sequência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema." (ZIVIANI, 2004).

A utilização de algoritmos é muito comum no ambiente computacional pois estes resolvem problemas práticos como ordenação, busca, recomendação e etc. Na figura 1, pode-se observar um tipo de algoritmo básico, que verifica uma condição, e realiza uma ação dependendo do estado.

Figura 1 - Exemplo de algoritmo básico



Elaborado pelo autor (2021)

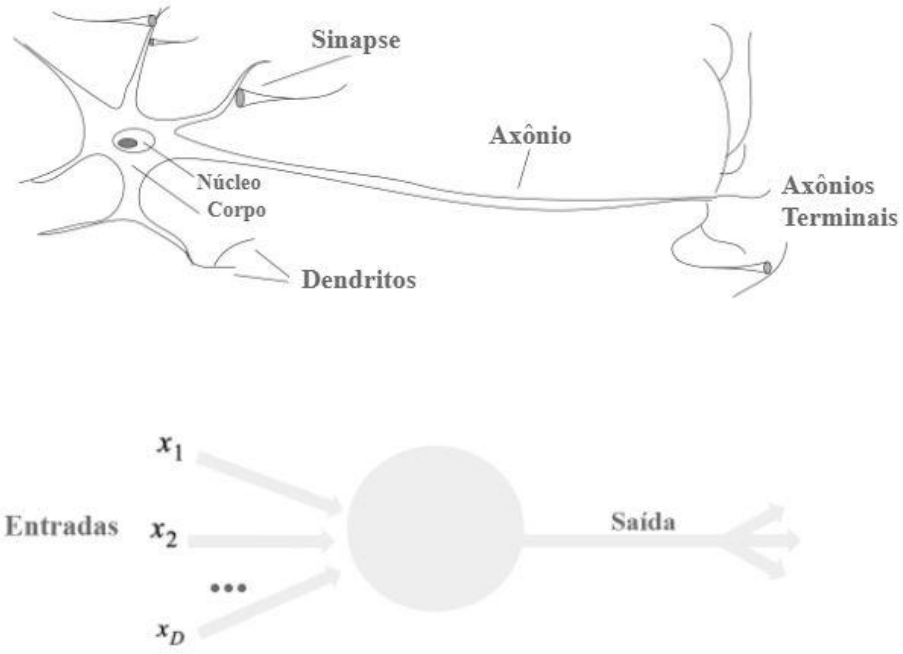
Já técnicas de aprendizado de máquina, são um subconjunto da inteligência artificial. Elas têm como principal objetivo, utilizar de algoritmos para construir modelos matemáticos mais robustos que ajudam a interpretar dados. Neste quesito, a palavra aprendizado quer indicar que estes modelos matemáticos serão adaptativos, ou seja, seus parâmetros serão modificáveis de acordo com o tipo de dado inserido, podendo de certa forma serem treinados pelo tipo de entrada (LEMOS, 2021).

Dentro destes modelos matemáticos, outro subconjunto destacável é o de redes neurais artificiais (RNA). Fazendo uma simplificação do processo de uma rede neural biológica, um neurônio se conecta aos outros utilizando uma estrutura chamada de sinapses, recebe impulsos através de uma das suas extremidades conhecida como dendritos, processa estas informações em seu núcleo celular a

partir de reações químicas e elétricas, e transmite novos impulsos resultantes pelo axônio para outros neurônios.

Já uma rede neural artificial simples é composta por neurônios artificiais, também chamados de perceptrons. Um perceptron é um algoritmo simples destinado a realizar a classificação binária, que recebe dados, realiza um processo matemático, guarda informações em memórias locais, e envia um sinal de saída (ROSENBLATT, 1958). Na figura 2 observamos a semelhança do processo biológico com o computacional.

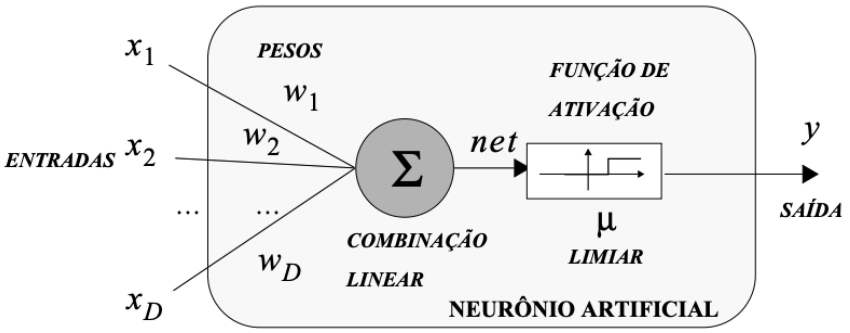
Figura 2: Semelhança entre Neurônio e Perceptron



Elaborado pelo autor (2021)

No modelo básico de um perceptron concebido por McCulloch e Pitts em 1943 e demonstrado pela figura 3, as entradas (x1.. xD) são recebidas de outros perceptrons. Cada entrada está associada a uma matriz de pesos (w1.. wD) e o processamento destes dados consiste em multiplicar a entrada pelo peso, e então realizar uma combinação linear desses valores.

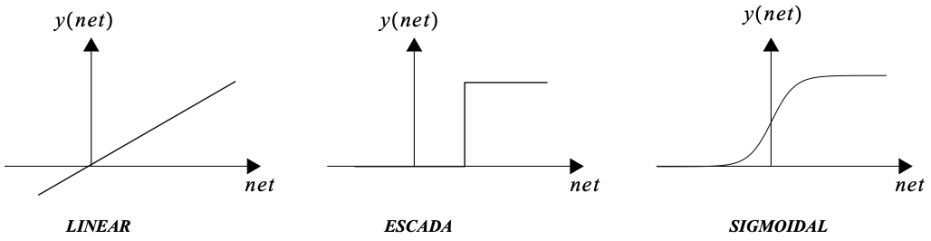
Figura 3 - Detalhamento do corpo de um Perceptron



Fonte: RAUBER (1997)

Esta combinação chamada *net* é aplicada a uma função de ativação, sendo que esta é responsável pelo processamento matemático de cada perceptron, similar ao processo químico-elétrico de um neurônio. Uma das funções mais usadas é a sigmoidal, mas existem diversas com diferentes características, algumas demonstradas na figura 4. A escolha da função de ativação é baseada nos dados de entrada da rede neural e uma determinada entrada funciona melhor com determinada função (FERNEDA, 2006). A saída chamada *ynet* desta função prevê se a entrada pertence a uma determinada categoria de interesse ou não.

Figura 4: Funções de ativação mais comuns



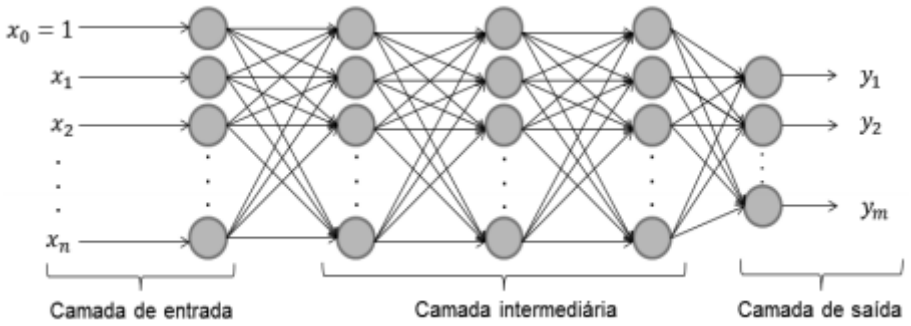
Fonte: RAUBER (1997)

Nestas RNA tradicionais, os perceptrons se organizam em forma de camadas, representada na figura 5 e são chamadas de Redes Perceptron Multicamadas (Multilayer Perceptron ou MLP). A camada mais à esquerda é chamada de entrada e recebe um vetor onde são inseridos os dados. Tais dados são transmitidos a N camadas ocultas que realizam as operações como as descritas anteriormente. Por

fim, os resultados são dispostos na camada de saída. Neste contexto, camadas ocultas nada mais são que aquelas que não são de entrada ou saída, há diferentes tipos delas com funções específicas que serão explicadas posteriormente.

Estas camadas são encadeadas e conectadas entre si pelos perceptrons, ou neurônios artificiais, formando uma rede totalmente conectada. Em cada camada, os perceptrons são responsáveis por extrair informações dos dados que estão sendo propagados pela rede, armazenando estas informações em parâmetros, ou pesos, que vão sendo ajustados a cada dado observado. Como pode-se observar na figura 5, cada neurônio está interligado a todos os outros em camadas adjacentes (RAUBER, 1997).

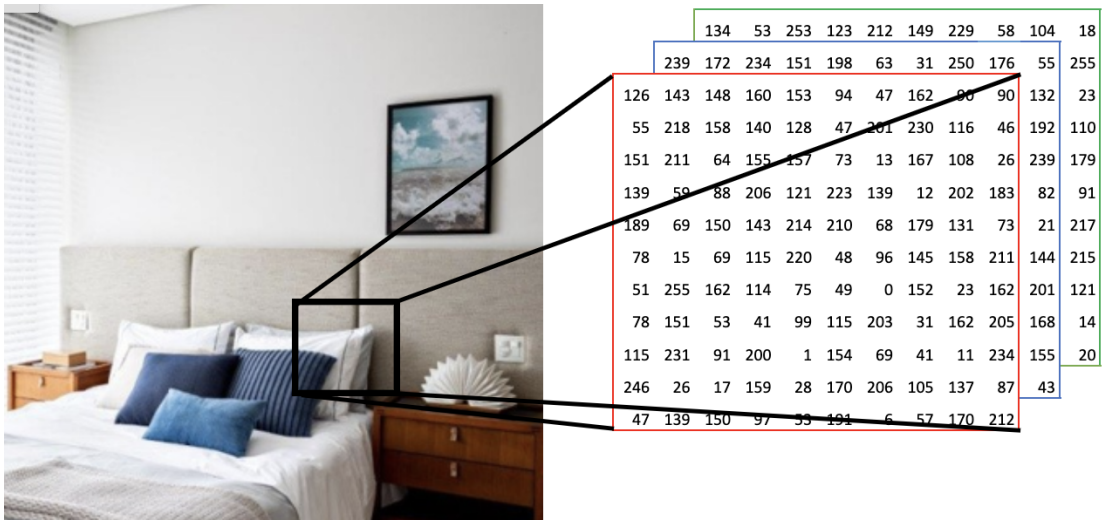
Figura 5: Organização das camadas de uma RNA Multicamada



Fonte: CUNHA (2017)

Para exemplificar o uso de uma RNA tradicional, primeiro é necessário definir como o computador representa uma imagem. Na forma digital, a menor parte de uma imagem é chamada de pixel, e a associação destes em uma matriz de tamanho *n* por *m* constroem uma imagem completa. Estes pixels têm intensidades variáveis entre 0 e 255 e podem ter uma dimensão, quando a imagem é em uma escala de cores cinzas, até três dimensões quando utilizado o sistema RGB (vermelho, verde e azul respectivamente). Isso significa que em uma imagem colorida, a associação destas três dimensões provê um pixel colorido, e posto em uma matriz, define-se a foto.

Figura 6: Representação de uma imagem digital



Elaborado pelo autor (2021)

Considerando que cada pixel de uma imagem conforme descrito anteriormente é um dado de entrada de uma RNA e que cada neurônio de uma dada camada recebe dados de todos os outros neurônios da camada anterior, uma imagem quadrada de 32 pixels de largura e 32 pixels de altura correspondem a ao menos 1024 (32*32) neurônios na primeira camada oculta, considerando as três dimensões de cores, 1024*3 dão um total de 3072 pesos.

Pode-se observar que imagens com este tamanho (32 por 32) estão muito aquém dos padrões utilizados hoje em dia, e se escalarmos para imagens mais complexas, aumenta-se excessivamente o número de pesos necessários. Porém observa-se que, nem todos os pixels de uma imagem precisam estar relacionados, tornando sua conectividade um desperdício.

A tarefa específica abordada neste trabalho, é a de classificação de imagens. Para tal, a melhor opção disponível são as redes neurais convolucionais (RNC), que são as mais indicadas para os problemas de visão computacional ou reconhecimento de linguagem natural. Um dos diferenciais das RNC em relação às outras RNA, é que elas preservam a relação de proximidade de pixels numa imagem, organizando os neurônios em 3 dimensões: altura, largura e profundidade, dessa forma reduzindo a quantidade de conexões e pesos (HOWARD, 2018).

Uma descrição mais detalhada das redes neurais convolucionais será apresentada na seção seguinte.

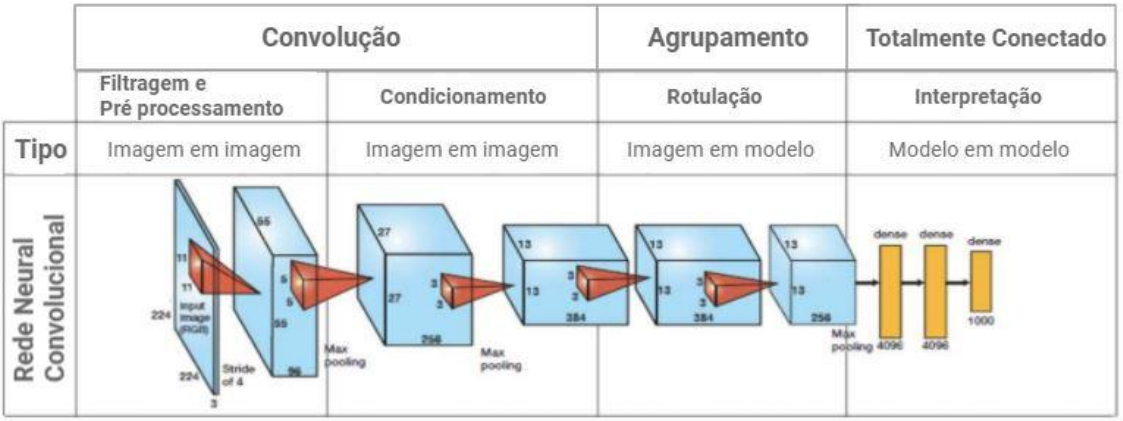
2.2.1 Descrição Geral de uma Rede Neural Convolucional

Uma RNC é um algoritmo de Aprendizado Profundo que foi projetado para captar os pixels de uma imagem, atribuir pesos a várias características da imagem e ser capaz de diferenciar uma característica da outra. O pré-processamento de uma imagem exigido em uma RNC é muito menor em comparação com outros algoritmos de classificação (HOWARD, 2018).

Em outras RNA mais primitivas as características devem ser identificadas por um humano, já uma RNC treinada de forma adequada e por tempo suficiente, tem capacidade de aprender esses filtros e características de forma autônoma.

Uma RNC se estrutura em modelos de camadas, e seu projeto foi feito de forma que diversas camadas com funções específicas sejam empilhadas e organizadas de forma a maximizar o resultado da análise de imagens. Na figura 7, há uma descrição simplificada de como se estruturam tais camadas.

Figura 7: Processamento de imagens com RNC

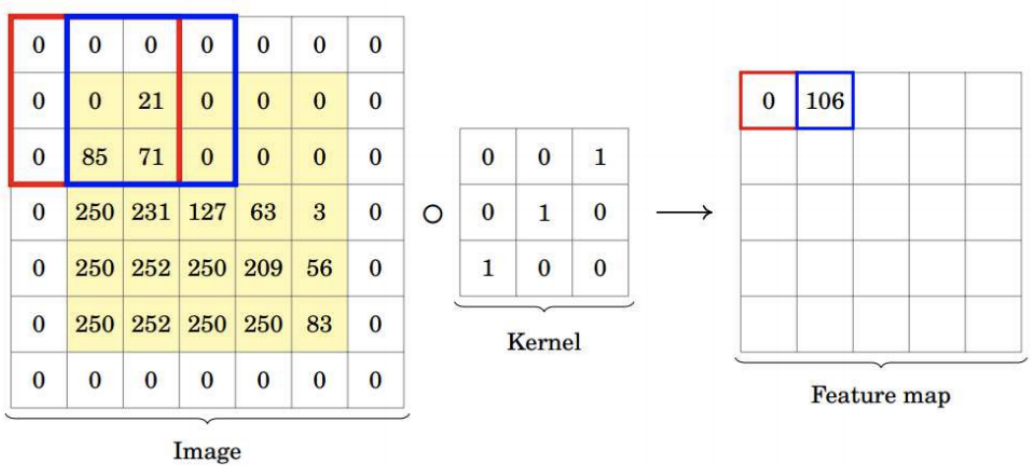


Fonte: Adaptado de VON WANGENHEIM (2020)

Neste tipo de rede neural, há geralmente três tipos de camadas ocultas: convolução, agrupamento e camadas totalmente conectadas. As duas primeiras são responsáveis por extrair características, enquanto a terceira mapeia as características e extraídas e as destina para a camada de saída. (YAMASHITA et al., 2018).

As camadas de convolução usam pequenos segmentos da imagem como, por exemplo, blocos de 3x3 (altura e largura em pixels) demonstrado em vermelho na figura 8.

Figura 8: Camada convolucional parcial, utilizando filtro 3x3



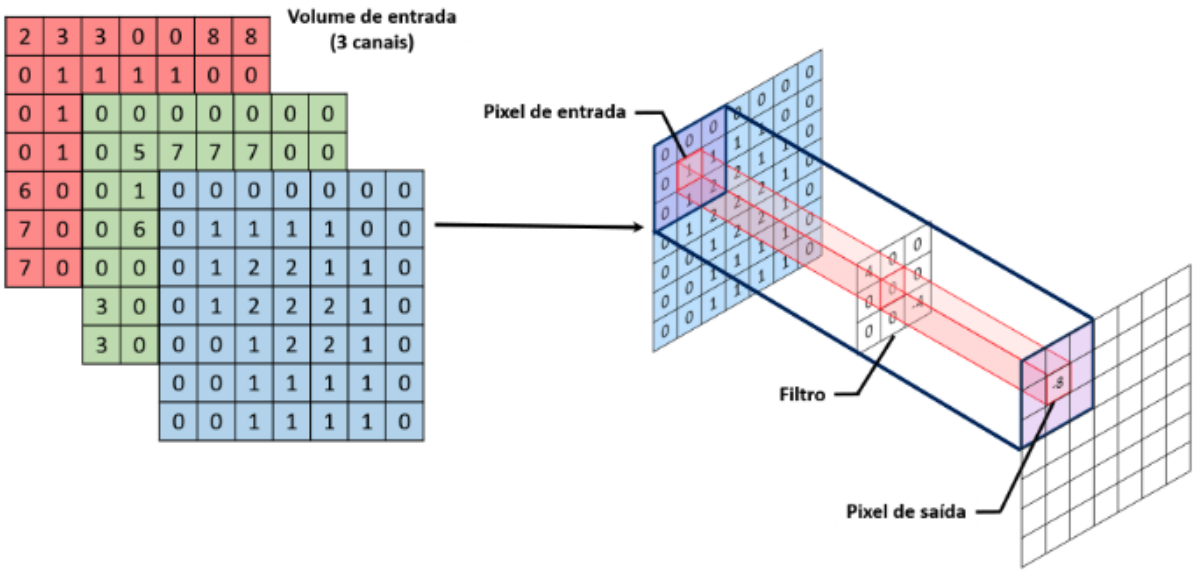
Fonte: Adaptado de Pavlovsky (2017)

Estes segmentos são chamados de filtros ou kernels, e são deslizados sobre o volume de entrada para obter uma série de mapas de características, tais como uma borda de uma determinada orientação, ou uma coloração específica. Em azul observa-se o mesmo filtro, agora movimentado 1 pixel para a direita, e destinado ao segundo neurônio na camada subsequente.

Filtros são estruturas que adaptam-se de maneira a capturar determinados coeficientes relevantes durante um treinamento, e são basicamente uma matriz de valores utilizados para a realização de operações de transformação sobre a imagem. Seu resultado é aplicado a uma função de transformação não linear e finalmente disposto na camada seguinte. Observa-se um exemplo na figura 9.

Porém há outras variáveis que devem ser configuradas, como o tamanho e quantidade de filtros, *stride* (passo) e *padding* (adicionar zeros ao redor do volume de entrada para manter a mesma dimensão após camada de convolução), estes são considerados hiperparâmetros (VALÊNCIA, 2021).

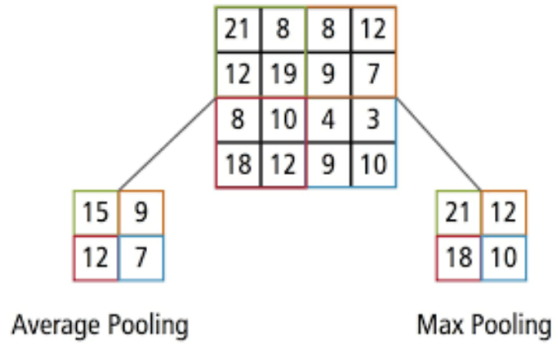
Figura 9: Ilustração da convolução entre um filtro 3x3 e o volume de entrada



Fonte: ARAÚJO (2017)

As camadas de agrupamento (Pool) são responsáveis por reduzir o número de parâmetros e o cálculo na rede, controlando o sobreajuste. Normalmente utiliza-se um filtro de tamanho 2x2 que permite diminuir pela metade tais parâmetros. Na figura 10 observa-se uma simplificação do que ocorre na camada de agrupamento dividida em dois tipos: o *Average Pooling*, onde cada pixel resultante é uma média dos anteriores; e o *Max pooling*, onde o pixel resultante é correspondente ao maior valor dos anteriores (HOWARD, 2018).

Figura 10: Ilustração de exemplo de Agrupamento



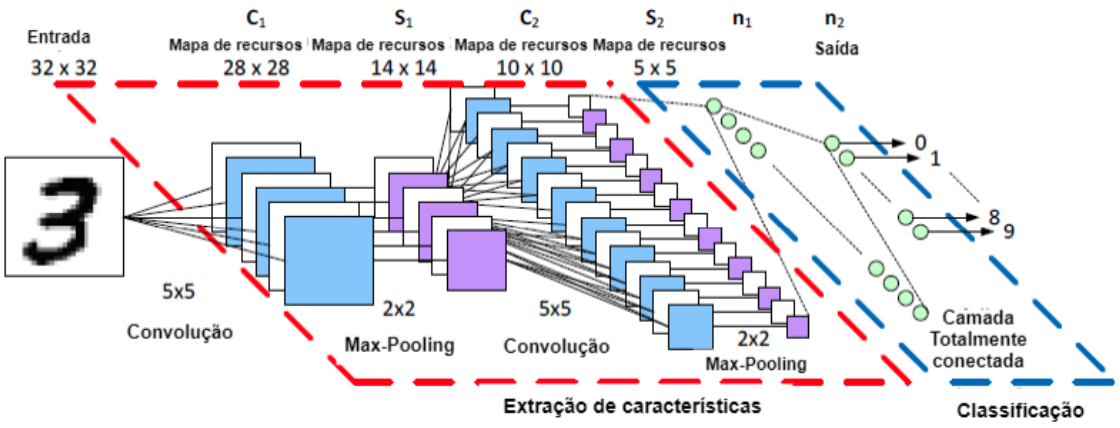
Fonte: KARPATHY (2020)

Por fim, após sequenciais camadas de convolução e agrupamento, os mapas de características estão achatados em forma de uma matriz unidimensional, e então

a camada totalmente conectada é responsável por reconectar estes dados e realizar a tarefa de classificação das imagens. (SOUZA et. al. 2020)

Um exemplo de rede neural convolucional aplicada a um exemplo específico é apresentado na figura 11. Neste caso, deseja-se realizar o reconhecimento de dígitos de 0 a 9, totalizando 10 classes possíveis. Há uma imagem de 32x32 pixels, que como pode-se observar trata-se de um número 3. A imagem é aplicada na primeira camada de convolução, utilizando filtros 5x5, e resultando no primeiro mapa de características C1 de tamanho 28x28. Em seguida é executada a camada de agrupamento com filtro 2x2, reduzindo os parâmetros pela metade para um mapa de 14x14 chamado S1.

Figura 11: Demonstração de aplicação de RNC



Fonte: Adaptado de PEEMAN et al. 2011

Tal processo é refeito, obtendo sequencialmente C2 e S2. O passo final é a camada totalmente conectada que recebe os neurônios de saída representados pelos círculos verdes e os classifica de acordo com as classes estabelecidas no treinamento.

Alguns problemas que podem ocorrer de acordo com a quantidade de camadas, são o underfitting (pouco treinamento), overfitting (treinamento em excesso), ou também o de degradação. O primeiro pode acontecer quando há poucas camadas e por consequência poucos neurônios para detectar adequadamente sinais em um conjunto de dados complexo. O segundo, por sua

vez, pode ocorrer de diversas formas, mas principalmente quando são utilizados muitos neurônios, treinados em um conjunto limitado de informação nos dados.

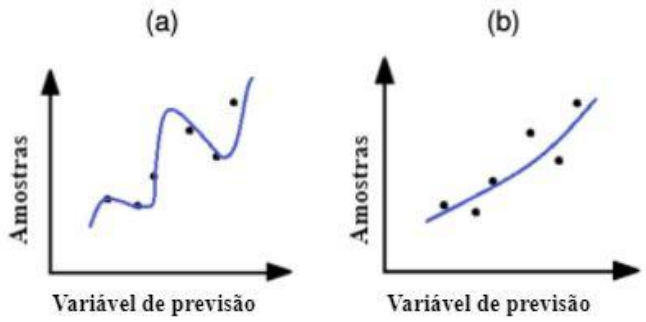
A degradação é um problema muito comum em redes neurais muito profundas, onde ao adicionar mais camadas em arquiteturas comuns, ocorre o chamado gradiente de desaparecimento, onde gradativamente informações são perdidas pela completa alteração do espaço de saída. Isto também pode levar a um tempo de treinamento muito grande, demandando muita capacidade computacional.

Para resolver tais problemas, diversas soluções foram desenvolvidas. Na seção a seguir, será detalhado o uso de uma destas, a ResNet. Esta RNC é construída por blocos residuais que realizam operações em uma entrada x. O resultado da operação f(x) é adicionado à entrada original x, e esta solução indica que um modelo mais profundo não deve produzir um erro de treinamento maior do que sua versão da rede mais rasa (HE et al., 2016).

2.2.2 Redes Neurais Residuais (ResNet)

Como citado na seção anterior, um dos problemas conhecidos de redes neurais que tentam obter resultados cada vez mais precisos, é o de degradação. Este problema pode ocorrer devido ao tamanho da RNC, onde a alocação de muitas camadas que faz com que uma RNC mais profunda memorize ruídos irrelevantes, em vez de focar no sinal, e drasticamente tenha uma queda de performance, em vez do contrário. Uma amostra é apresentada na figura 12 (HOWARD, 2018).

Figura 12: (a) Modelo com memória e (b) Modelo correto.



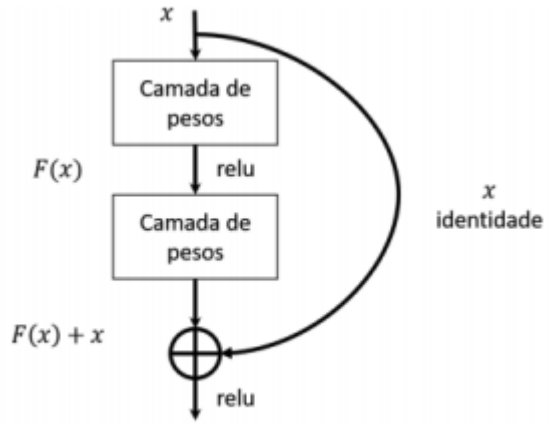
Fonte: Adaptado de LEMOS (2020)

Nesta figura é possível observar uma situação em 'a' onde o modelo, por possuir muitas camadas, se ajusta excessivamente ao conjunto de treinamento. Por sua vez, em 'b', por possuir menos camadas, o modelo se ajusta adequadamente criando uma boa fronteira de decisão sem que um ajuste excessivo tenha ocorrido. Além da degradação, outros problemas foram encontrados no aprofundamento de redes neurais.

Em resposta a isso, a ResNet foi desenvolvida e venceu o concurso anual ILSVRC 2015 (*ImageNet Large Scale Visual Recognition Challenge*) que avaliava a performance de redes neurais convolucionais na detecção de objeto e classificação de imagens.

Seu funcionamento se baseia no uso de blocos residuais (identidade) e é demonstrado na figura 13. Supondo uma entrada *x*, esta entrada passa por uma série de operações que resultam em uma saída *f(x)*. Em outros tipos de redes convolucionais, esta saída seguiria para a próxima camada completamente alterada. Já utilizando a Resnet, há uma realimentação da entrada *x* (identidade) a cada duas camadas, onde a saída agora é *h(x) = f(x) + x* (HE et al., 2016).

Figura 13: Demonstração de realimentação da identidade.



Fonte: Adaptado de HE et. al. (2016)

Com isso, a saída deixa de ser uma completa alteração da entrada, e passa a ser uma regularização. Desta forma, arquiteturas ResNet com diferentes profundidades são possíveis, desde 18 camadas até 152.

O intuito deste trabalho foi de utilizar modelos validados e conceituados, de forma que não seja necessário criar um modelo, mas sim ajustá-los e empregá-los em um novo contexto de aplicação e demonstrar sua viabilidade. Existem diversas bibliotecas de livre acesso que facilitam o uso destes modelos, algumas serão apresentadas a seguir.

2.2.3 Ferramentas disponíveis para a utilização de RNCs

Várias ferramentas foram utilizadas durante o desenvolvimento do presente trabalho. A primeira a ser citada é a biblioteca de livre acesso *Pytorch*. Esta biblioteca tem como objetivo realizar cálculos em tensores, estruturas similares a arrays, com característica adicional de poderem guardar o histórico de operações feitas. O *Pytorch* utiliza principalmente do poder de processamento de GPUs (Unidades de processamento gráfico).

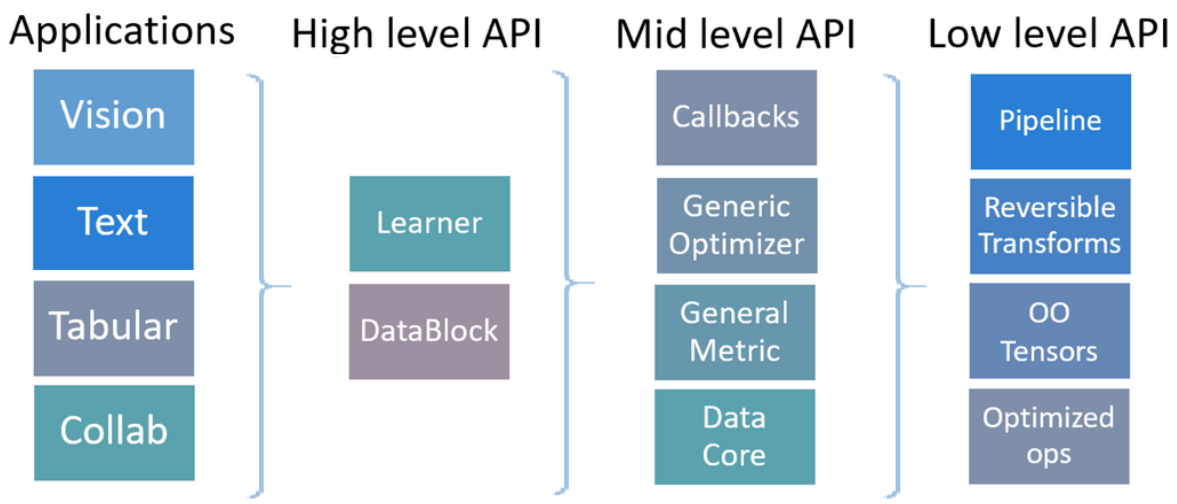
Para realizar a análise e correção dos dados, optou-se pela biblioteca open source *Pandas*. Essa biblioteca é escrita em Python, e trabalha sobre uma estrutura tabular conhecida como DataFrame. o *Pandas* é muito útil para ler, manipular, agregar e plotar os dados em poucos passos.

O terceiro recurso utilizado é a *Fastai*. Esta é uma biblioteca de aprendizagem profunda escrita na linguagem Python, desenvolvida no topo de uma hierarquia de outras bibliotecas de mais baixo nível como o *Pytorch*.

A Fastai tem como objetivo fornecer aos usuários com menos experiência uma API consistente com componentes em alto nível para que seja possível de forma rápida e fácil desenvolver modelos em poucas linhas de código, que atendam especificações básicas e tenham bom desempenho.

Ao mesmo tempo, esta biblioteca também permite a pesquisadores acesso a componentes e informações de mais baixo nível que podem ser modificados e combinados para realizar novas abordagens. Na figura 14 pode-se observar a organização da API fornecida pelo *Fastai*. No topo são apresentados os possíveis tipos de aplicação e nas etapas seguintes são demonstrados os níveis da API e suas principais funções.

Figura 14: Níveis de organização da aplicação *Fastai.*



Fonte: adaptado de HOWARD (2018)

Esta biblioteca apresenta uma grande gama de funções, que resolvem dificuldades clássicas no desenvolvimento de modelos de aprendizado profundo. Na classe chamada de DataBlock, resolvem-se problemas relacionados à entrada de dados, como coleta, identificação do problema, separação de variáveis entre dependentes e independentes, separação do banco de dados em treinamento e validação entre outras funcionalidades. É uma maneira de definir sistematicamente todas as etapas necessárias para preparar dados para um modelo de aprendizado profundo.

Para a criação de um Datablock, precisa-se definir de onde virão as imagens para o treinamento, e como estão distribuídas em relação às classes. Também é necessário definir um parâmetro que divide a totalidade do banco de dados entre treinamento e validação, juntamente com o percentual de cada um destes.

Por fim, é possível aplicar transformações de tamanho e posição nas imagens. A primeira e mais necessária transformação é a que regulariza o tamanho de cada foto, pois estas têm largura e altura diferentes, e o trabalho desta função é definir estes valores de forma que todas as imagens fiquem com a mesma proporção, algo essencial para o treinamento. Ainda é possível aplicar outras transformações utilizadas para aumentar o tamanho do banco de dados, realizando transformações randômicas de rotação, espelhamento, zoom, translação e etc.

Já o treinamento da RNC é realizado pela classe Learner. A função de treinamento possui como parâmetros principais os próprios dados pré-processados pelo DataBlock, a arquitetura que será utilizada no treinamento e as métricas de desempenho dos modelos. Com estas informações, esta função de alto nível organiza e realiza rodadas de treinamento do modelo devolvendo ao final de cada rodada, uma prévia do desempenho.

Realizado o treinamento, o bloco Learner fornece uma série de funções que podem ser utilizadas para exportar o modelo em um arquivo que pode ser integrado a qualquer tipo de sistema. Este modelo exportado pode inclusive ser novamente treinado, aplicando-se novas classes ou buscando aumentar o desempenho das existentes.

Além das bibliotecas citadas acima, outras ferramentas comumente utilizadas no desenvolvimento de RNC's são ambientes em nuvem. Dado que o treinamento de um modelo é uma tarefa custosa, e que as bibliotecas citadas anteriormente fazem grande uso de GPUs, utilizou-se o Google Colab. Este ambiente de desenvolvimento é muito prático e útil, pois não requer prévia configuração e é executado totalmente em nuvem. Desta forma, é possível escrever os códigos fontes e executá-los utilizando GPUs remotas cedidas pela própria Google. O intuito desta ferramenta é incentivar a pesquisa no âmbito de aprendizado de máquina e inteligência artificial.

2.2.4 Métricas

Para avaliar o desempenho do modelo desenvolvido, e validar o resultado do treinamento, utilizou-se o tempo de inferência médio de cada arquitetura, mas também há a necessidade de utilizar métricas que mostram ao desenvolvedor informações sobre o comportamento deles nos mais diversos aspectos. Métricas em classificações binárias são normalmente estabelecidas em quatro tipos de possíveis resultados, no quadro 1 a seguir é possível observá-los (FERRARI e SILVA, 2017).

Quadro 1 - Resultados de classificação possíveis.

Verdadeiro Positivo	O sistema classificou como Verdadeiro, e efetivamente o resultado é verdadeiro.
Verdadeiro Negativo	O sistema classificou como Falso, e efetivamente o resultado é Falso.
Falso Positivo	O sistema classificou como Verdadeiro, porém o correto seria Falso.
Falso Negativo	O sistema classificou como Falso, porém o correto seria Verdadeiro.

Elaborado pelo autor (2021)

A primeira das métricas utilizadas é a acurácia, esta se trata da medida mais intuitiva, e indica uma performance geral do modelo. Representada em (1) dentre todas as classificações, quantas o modelo classificou corretamente.

$$Acurácia = \frac{Verdadeiro\ Positivo}{Total\ de\ itens} \quad (1)$$

Outra métrica utilizada é a precisão. Esta pode ser aplicada a cada classe individualmente e mede, dos resultados positivos, quantos efetivamente eram previstos como positivos. É representada em (2)

$$Precisão = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Positivo} \quad (2)$$

Uma terceira métrica é o Recall. Ela mede a capacidade de classificar todas as amostras positivas, ou seja, é utilizada para indicar a relação entre as previsões positivas realizadas corretamente e todas as previsões que realmente são positivas. Tal métrica é dada por (3):

$$Recall = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Negativo} \quad (3)$$

A quarta métrica é o F1-Score. Ela é definida como a média harmônica entre Precisão e Recall. Tal métrica é dada por (4):

$$F1 = 2x\frac{Precisão * Recall}{Precisão + Recall} \quad (4)$$

Uma métrica muito útil e bem visual, é a matriz de confusão. Esta matriz é composta pelas labels reais nas linhas, em comparação com os valores classificados nas colunas. Na figura 15 observa-se o funcionamento de uma matriz de precisão.

Figura 15 : Matriz de confusão

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Elaborado pelo autor (2021)

O resultado de um bom modelo mostraria um tipo de linha contendo os valores previstos da diagonal da esquerda superior, em direção à direita inferior. Cada um dos campos desta linha contém os valores classificados corretamente. Todos os outros campos representam erros, e desta forma é possível observar quais classes tiveram maiores interações negativas.

Por fim, é importante falar da perda (loss). Este tipo de variável é comumente utilizado pelo próprio algoritmo para, a cada rodada de treinamento, calcular o desempenho do modelo classificador de uma forma mais precisa e realizar ajustes a partir do resultado. Existem diversas formas de calcular a perda, chamadas funções de perda. A mais comum e utilizada no presente trabalho é a Perda de entropia cruzada (Cross-entropy loss).

2.2.5 Trabalhos Correlatos

Pode-se encontrar trabalhos que utilizam visão computacional e aprendizado profundo para classificação de objetos nas mais diversas áreas. Um exemplo que pode-se citar é o trabalho desenvolvido por Yessica Valencia (2021), que realizou uma comparação entre três soluções baseadas em processamento de imagens para

realizar a inspeção automática de ovos comerciais, e classificá-los entre quatro categorias: normal, sujo, geometricamente anormal e ovos fissurados.

Seu trabalho teve êxito ao apresentar acurácia superior a 81% nas três soluções, e tempo médio de processamento inferior a 1,35 segundos. Em duas das soluções apresentadas, foram utilizadas redes neurais convolucionais e uma destas arquiteturas propostas, a ResNet, foi utilizada no sistema de classificação de ambientes realizado neste trabalho.

Um segundo trabalho relevante foi escrito por João Vitor (2020) e tem como problema classificar peças mecânicas a partir de visão computacional e aprendizado de máquina utilizando imagens sintéticas. Sua proposta também passa pelo uso de arquiteturas de redes neurais convolucionais, e com o diferencial de utilizar imagens produzidas por simulação 3D para realizar o treinamento do classificador, modificando textura, iluminação, posição e orientação da mesma. Após realizar um treinamento com 10 classes diferentes, obteve resultados de aproximadamente 90% de precisão, o que mostra grande aplicabilidade para a indústria.

Não foram encontrados trabalhos acadêmicos na área de classificação de ambientes residenciais, porém há relatos de casos de uso na internet, que se tornaram úteis para a realização deste trabalho. Destes, destaca-se o artigo produzido por Shijing Yao (2018), que demonstra a utilização de RNC's para a classificação de ambientes para uma imobiliária digital e que alcançou grande sucesso, obtendo em média precisão acima de 95%.

3 METODOLOGIA

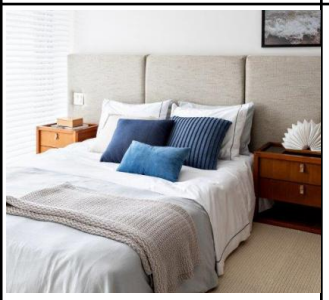
Neste capítulo será detalhada a metodologia empregada para o desenvolvimento de um sistema classificador automático de fotos de ambientes residenciais.

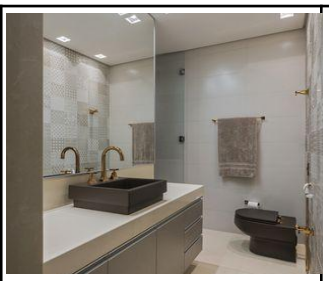
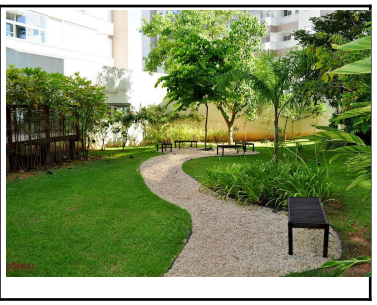
3.1 Discussão sobre os requisitos do sistema

A primeira etapa empregada na metodologia utilizada foi definir uma lista de requisitos funcionais que o sistema deveria atender. A partir de tais requisitos foi possível traçar planos de ação para atender cada ponto. Tais requisitos foram levantados em conjunto com a equipe de engenharia de dados da empresa parceira, responsável pelo desenvolvimento de produtos digitais nas mais diversas áreas.

- Requisitos funcionais:
    1. O sistema desenvolvido deve ser capaz de classificar fotos de ambientes residenciais em ao menos seis classes, sendo estas apresentadas no quadro 2:

Quadro 2 - Categorias a serem classificadas pelo sistema.

Categoria	Imagem	Categoria	Imagem
Quarto		Cozinha	
Sala		Fachada	

Banheiro		Área Externa	

Elaborado pelo autor (2021)

2. O sistema deve ser parametrizável de modo que o valor de certeza seja ajustável.

- Requisitos não funcionais:
    1. O sistema de classificação deve ser construído na forma de um serviço, de forma que outros módulos ou sistema possam realizar requisições de classificação tanto local quanto remotamente.

3.2 Descrição da solução

Com o objetivo de classificar as imagens nas categorias definidas pelos requisitos de projeto, foi selecionada a técnica de aprendizado profundo baseada em redes neurais convencionais. Este método foi implementado utilizando a biblioteca Fastai. Também foi utilizado o Google Colab e o Google Drive para salvar, carregar e formatar a base de dados, bem como para treinamento dos modelos.

Obtendo os requisitos do projeto e também estudos relacionados, foi possível traçar um plano de ação baseado em três etapas, sendo elas:

- Obter e preparar a base de imagens;
- Treinar e validar um modelo classificador;
- Disponibilizar o modelo para requisições remotas.

Tais etapas serão discutidas nas próximas subseções.

3.2.1 Obtenção e preparação do banco de dados

Em um primeiro momento, a empresa parceira JungleDevs forneceu um banco de dados contendo dezenas de milhares de imagens. Estes dados foram

fornecidos em arquivos de formato XML, contendo informações completas sobre milhares de imóveis e suas características. Dentre estas características estavam as URLs das imagens e algumas informações pertinentes, como suas classes que aqui eram chamadas de *caption*, e o tipo de arquivo chamada de *medium*. Na figura 16 pode-se observar a disposição destes dados.

Figura 16: Arquivo XML com dados dos imóveis cadastrados

Elaborado pelo autor (2021)

Para extrair estas URL's e suas informações, foi desenvolvido um código em Python, que abre cada arquivo XML disponibilizado, entra em sua raiz e procura pela marcação 'Item'. Para cada item, o código salvará em uma linha de uma tabela a URL, a classe definida e o tipo de arquivo. Ao final, cada XML gerará um arquivo CSV contendo apenas as informações necessárias para execução deste trabalho. Na tabela 1 pode-se observar como ficaram organizados estes arquivos.

Tabela 1: Criação do banco de imagens

id	url	medium	caption
0	http://fotos.brognoli.com.br/html_web/www/images/fotos/25807/big-25807-038.jpg	image	fachada
1	http://fotos.brognoli.com.br/html_web/www/images/fotos/25807/big-25807-039.jpg	image	entrada
2	http://fotos.brognoli.com.br/html_web/www/images/fotos/25807/big-25807-041.jpg	image	área externa

Elaborado pelo autor (2021)

A partir desta formatação, é possível iniciar a análise e preparação dos dados que serão utilizados no treinamento dos modelos. Para executar o treinamento do modelo posteriormente, há a necessidade de pares de treinamento, com amostras de imagens rotuladas com suas classes, para que assim o classificador possa aprender as características dessa imagem, e atribuir tais características a uma classe.

Desta forma, concatenou-se as tabelas para obter um único arquivo contendo toda a extensão de dados. Neste momento, eliminou-se do banco de dados itens que não eram imagens ou que continham algum outro erro, restando assim 33356 fotos.

Após realizada esta primeira eliminação de dados, agrupou-se os itens por classe e obteve-se 313 classes diferentes, um número muito maior do que os requisitos do projeto. Na tabela 2 pode-se observar as classes com mais entradas, e a quantidade de fotos constante em cada classe.

Tabela 2: Classificação do banco de imagens com as classes mais frequentes

Classe	Frequência	Classe	Frequência
nan	7368	**banheiro**	2387
quarto	4941	**área externa**	1583
sala	3966	**fachada**	1577
cozinha	3930	**área de serviço**	1196

Elaborado pelo autor (2021)

Observa-se que aqui a classe mais representada 'nan' não é de fato uma classe, mas sim a ausência de classificação. Em seguida, as seis classes que são definidas nos requisitos do projeto estão representadas, apresentando entre no máximo 4941 fotos de quartos e no mínimo 1577 fotos de fachada.

Após realizar uma análise nas primeiras 50 classes mais representadas, observou-se uma grande repetição de classes com diferenciais numéricos, erros de ortografia ou mesmo simples diferenças de classificação. Para dar alguns exemplos, similares a classe 'quarto' existem ao menos outras 14 classes como 'quarto_1',

'quarto_2' ou 'dormitório'. Similares à classe 'banheiro' ao menos outras 8, como 'bwc', 'wc_suite' ou 'wc_quarto'.

Para contornar esta situação, aplicou-se uma função de transformação para cada classe similar às 6 principais, de forma a reduzir a quantidade de classes representadas e aumentar o número de fotos nas principais. Ainda assim, observou-se mais de 250 classes pouco representadas e muito específicas. Na tabela 3 demonstra-se um pequeno exemplo destas classes.

Tabela 3: Classificação do banco de imagens com as classes menos frequentes.

Classe	Frequência	Classe	Frequência
hall	14	**edifício**	12
acesso praia	12	**sala de jogos**	10
lavanderia coletiva	12	**vista quarto**	10
dependência de empregada	12	**sala de jantar**	8

Elaborado pelo autor (2021)

Como o projeto não tem interesse em abranger tantas classes no primeiro momento e as classes mais representadas já têm dados o suficiente, optou-se por eliminar estes itens do banco de dados. Ao final deste processo, eliminadas também as fotos que não continham classificação, restaram 19581 fotos separadas entre as 6 classes principais.

Por último, para realizar o treinamento do modelo, é necessário ter acesso aos arquivos das fotos, e não somente as suas URL's. Dessa forma, foi necessário desenvolver uma função que acessou cada item do *DataFrame*, baixou a foto contida na URL e a salvou em uma pasta com o mesmo nome da classe no Google Drive.

Do total de imagens utilizadas, é necessário designar um volume para treinamento e outro para validação dos modelos, conforme comentado na seção 2.2.3, para a criação do Datablock. Os valores definidos foram 70% para treinamento e 30% para validação.

3.2.2 Treinamento do modelo

Após a análise e formatação da base de dados, a segunda etapa consistiu no treinamento dos modelos pela função *cnn_learner*. Como citado no capítulo 2, o treinamento dos modelos foi realizado utilizando a plataforma *Google Collaboratory*, onde foi alocado para o treinamento a GPU Tesla K80, com 24GB de memória GDDR5.

Para executar o treinamento dos modelos são necessários três principais parâmetros, o primeiro deles é o DataBlock, e faz referência ao banco de dados. O Datablock por sua vez também tem suas dependências, conforme citado anteriormente no capítulo 2.

Neste trabalho, as imagens estão armazenadas em pastas com o nome da respectiva classe, conforme demonstrado na seção 3.2.1.1. Na figura 17 pode-se observar tal disposição.

Figura 17: Disposição do banco de imagens local

Elaborado pelo autor (2021)

Ainda definiu-se que um splitter randômico seria aplicado, e que 30% das fotos seriam utilizadas para validação, logo, 70% para treinamento. Referente às transformações, um redimensionamento (*resize*) de 128x128 foi aplicado, e um aumento de dados (*augmentation*) randômico também foi definido.

O segundo parâmetro necessário para o treinamento são as arquiteturas. No presente trabalho definiu-se que as arquiteturas utilizadas serão ResNet18, Resnet34 e ResNet50. Por fim, como terceiro parâmetro é necessário definir as métricas que serão apresentadas a cada rodada (*epoch*) de treinamento. Uma rodada é finalizada quando todas imagens são apresentadas ao modelo ao menos uma vez. Neste trabalho, as métricas utilizadas em um primeiro momento serão acurácia, precisão e recall. Posteriormente a matriz de confusão será analisada.

Definidos todos estes parâmetros, utilizou-se o método de treinamento conhecido como transferência de aprendizado (*transfer learning*), para a extração de características e ajuste do resultado. Este método consiste em transferir o aprendizado obtido a partir de um cenário para outro semelhante, porém diferente do que foi originalmente treinado (GOODFELLOW et al., 2016). Neste caso, essa transferência acontece pois as arquiteturas utilizadas já são pré treinadas em um dataset, ou seja, já tem pesos inicializados de uma maneira não aleatória.

Dentro do método de transfer learning, existe uma técnica específica chamada de ajuste fino (*fine tuning*). Esta técnica consiste em atualizar os pesos dos modelos pré treinados em rodadas de treinamento adicionais, agora voltadas para as classes desejadas. A quantidade de rodadas adicionais, e a taxa de aprendizado com que os pesos serão atualizados em relação ao erro foram definidas de forma experimental, e serão discutidas na seção 3.3.

Os resultados obtidos com arquiteturas já pré treinadas em milhares de categorias, aplicadas a um treinamento local em mais algumas rodadas, retorna um desempenho muito maior do que treinar uma arquitetura treinada do zero (HOWARD, 2020).

Aplicada esta técnica sobre o conjunto Datablock/Arquitetura/Métricas, obteve-se enfim os modelos classificadores. Estes modelos podem ser exportados e utilizados de inúmeras formas, como será descrito na seção a 3.2.2 a seguir. Os resultados destes modelos serão discutidos no capítulo 4.

3.2.3 Descrição da Arquitetura do Serviço que disponibiliza a classificação

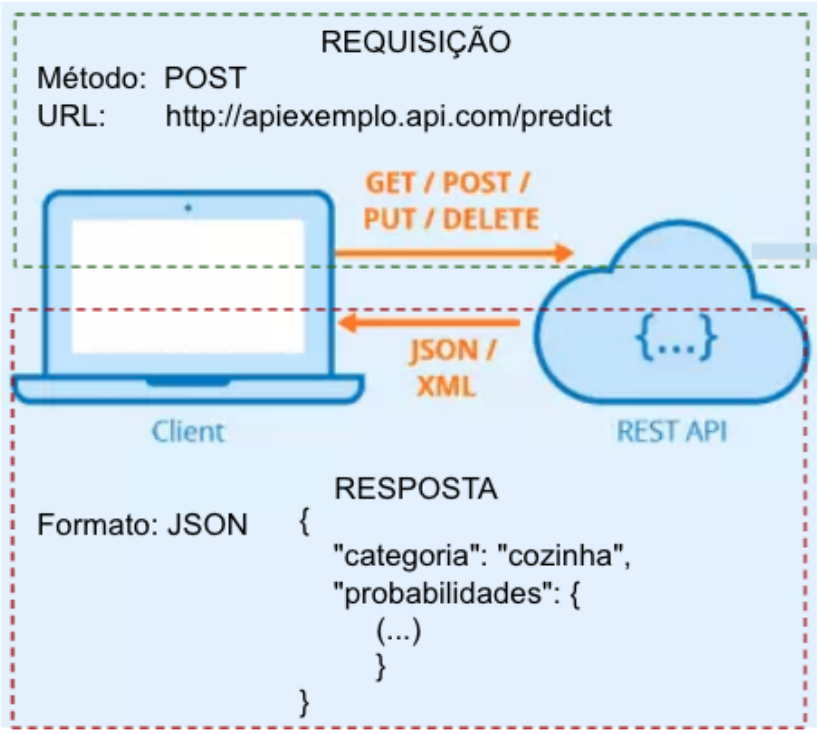
Para atender a todos os requisitos solicitados, ainda existia a necessidade de disponibilizar este modelo classificador na forma de um serviço, onde pudessem ser

recebidas requisições e retornados os respectivos resultados de classificação para o sistema cliente.

Pensando na integração deste serviço com outras aplicações *WEB*, utilizou-se um método de integração conhecido como API (Application Programming Interface) ou Interface de programação de aplicação. Uma API define um conjunto de normas que facilitam a integração de diferentes sistemas, escritos em diferentes linguagens (TEHREEM, 2020).

No presente trabalho, a *API Rest* definida utiliza requisições *HTTP*, um protocolo de requisição onde o cliente que está enviando define o tipo de ação a ser tomada pelo serviço com os métodos padrões como *GET, POST, DEL*. Nesta requisição o cliente deve enviar a imagem que será classificada. Em retorno a esta requisição, será devolvida ao cliente uma resposta no formato *JSON* contendo a classe mais provável da foto, e a probabilidade de cada classe testada. A figura 18 representa estes comportamentos.

Figura 18: Exemplo do funcionamento de API



Fonte: Adaptado de TEHREEM (2020)

Para desenvolver esta aplicação foi utilizado o *Flask*, uma 'micro framework' que permite a criação de serviços Web. Este micro framework possui um núcleo simples e expansível que permite que um projeto possua apenas os recursos necessários para sua execução. Na sua forma mais simples, o flask executa a criação de rotas de acesso, as *URLs*, para onde as requisições são enviadas. Apesar de ser um micro-framework, o *Flask* permite a criação de aplicações robustas, já que é totalmente personalizável, permitindo, caso necessário, a criação de uma arquitetura mais definida.

Utilizando o recurso definido acima, foi definida uma rota de acesso chamada de *predict*. Esta rota permite uma requisição do tipo *POST*, e espera no conteúdo desta requisição uma imagem. Dentro da aplicação, quando uma requisição é realizada, a imagem recebida é aplicada ao modelo classificador. Este por sua vez, retorna as informações como classe e probabilidades. Estas informações são finalmente formatadas para *JSON*, e retornadas ao cliente, encerrando o ciclo.

3.3 Definição de parâmetros experimentais

Como foi comentado no capítulo anterior, dois parâmetros restavam ser definidos. A quantidade de rodadas de treinamento, e a taxa de aprendizado. Ambos foram obtidos de maneira experimental, onde a taxa de aprendizado é resultante de uma função que realiza uma amostra de treinamento, utilizando uma pequena quantidade de fotos chamada de *batch*.

Figura 19: Evolução da perda *versus* taxa de aprendizado

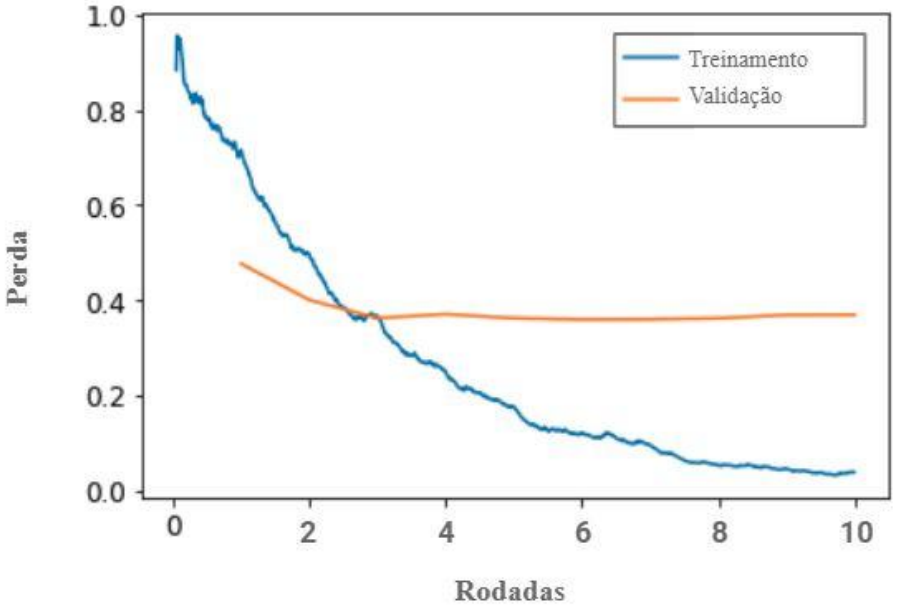
Elaborado pelo autor (2021)

Esta função aplica diferentes taxas de aprendizado ao modelo, e observa a perda em relação a cada taxa. A partir dos gráficos retornados na figura 19, é possível observar que aplicando taxas na ordem de 10^{-7}até 10^{-3} há pouca diferença na perda, mas aumentando a taxa de aprendizado acima de 10^{-3}, obtém-se uma perda menor. Até um determinado momento onde se atinge um limite próximo a 10^{-1} , então a perda aumenta exponencialmente. Este padrão se repete para os três modelos.

Para definir uma taxa de aprendizado ideal, procura-se um valor que esteja na ordem de um décimo do limite, ou seja, valores na ordem de grandeza de 10^{-2} (Fastbook, 2020).

A quantidade de rodadas que foram utilizadas no treinamento do modelo também foi obtida de forma experimental. Inicialmente já percebeu-se que o tempo é um fator importante, visto que cada rodada de treinamento, dependendo da profundidade do modelo e dos equipamentos utilizados, pode levar até dezenas de minutos para ocorrer. Outro fator observado foi a diferença entre métricas no dataset de treinamento e no de validação. Na figura 20 pode-se observar estes resultados:

Figura 20: Evolução da perda *versus* rodadas de treinamento



Elaborado pelo autor (2021)

A partir de um determinado número de rodadas, a perda na validação se mantém estática, enquanto a perda no treinamento cai vertiginosamente. Isso pode indicar um overfitting do modelo, pois o mesmo está decorando as imagens de treinamento, mas esta melhora não se reflete no ambiente de validação. Neste ponto, há pouca melhora efetiva no desempenho do modelo, e em algumas ocasiões até pioras momentâneas, portanto mais rodadas não são necessárias (Fastbook, 2020).

Também existe uma limitação de tempo na plataforma de desenvolvimento, pelo fato de o treinamento utilizar de um grande processamento computacional, e este processamento hardwares remotos, o Google Colab limita execuções muito longas. Assim, define-se que 10 rodadas serão utilizadas para o treinamento dos modelos. No capítulo 4, serão detalhados os resultados obtidos durante e após o treinamento dos modelos classificadores.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nas seções a seguir serão apresentados os resultados obtidos após o treinamento em relação a comparação das arquiteturas, a escolha do modelo mais adequado para resolução do problema, bem como os resultados da implementação deste modelo.

4.1 Avaliação dos resultados obtidos com as arquiteturas

Após a finalização do treinamento dos três modelos, é possível avaliar os resultados a partir das métricas e destas, obter o modelo com melhor desempenho para empregá-lo em uma situação de uso real. Primeiramente na tabela 4 é possível observar o tempo que cada modelo levou para ser treinado.

Como o ambiente de treinamento, a quantidade de rodadas aplicada e a GPU foi a mesma, esta diferença se deve a profundidade de cada modelo. É possível observar que o modelo mais profundo, levou mais tempo para ser treinado, o que era esperado.

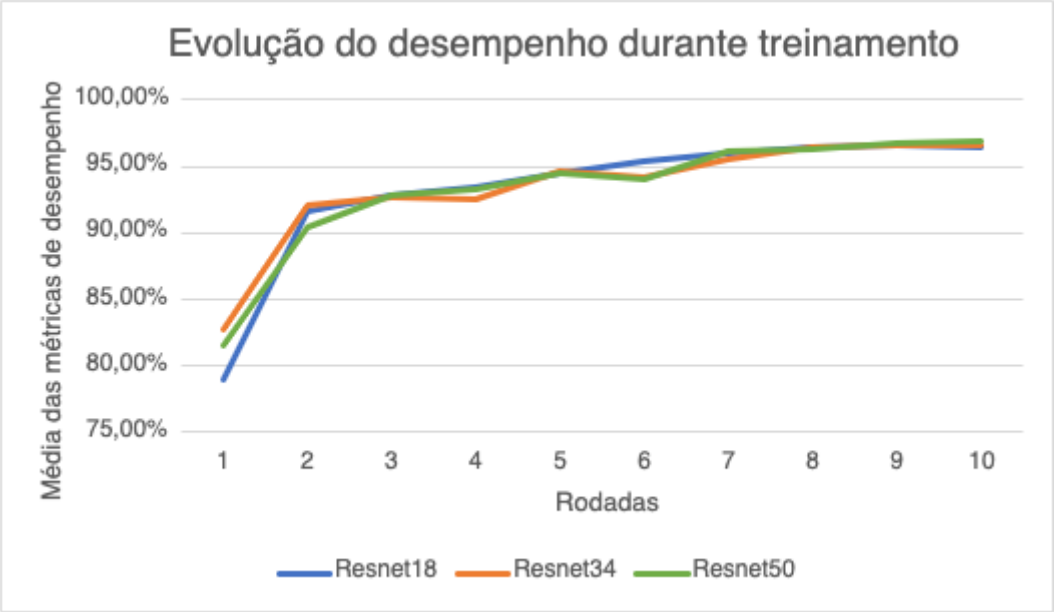
Tabela 4: Tempos de treinamento dos modelos

Arquitetura	Rodadas	Taxa de aprendizado	Tempo de treinamento (min)
Resnet18	10	0.005	113
Resnet34	10	0.005	123
Resnet50	10	0.005	141

Elaborado pelo autor (2021)

Comparando os dados de treinamento, pode-se observar na figura 21 a média das métricas de como cada modelo se comportou durante o treinamento. Percebe-se um salto na primeira para a segunda rodada, onde o modelo aprende as características mais básicas de cada classe. Posteriormente a cada rodada o treinamento busca detalhes mais específicos, onde o aumento de desempenho percentual é menor, mas também significativo.

Figura 21: Evolução do desempenho médio dos modelos durante o treinamento



Elaborado pelo autor (2021)

Considerando os resultados apenas da décima rodada, que representa o estado final do modelo, obteve-se os dados apresentados na tabela 5. Estes dados demonstram que a Resnet50 apresentou um resultado levemente maior em todas as métricas analisadas, mesmo que por diferença de poucos décimos percentuais.

Tabela 5: Resultados obtidos na comparação entre arquiteturas

Arquitetura	Acurácia	Precisão	Recall	F1
Resnet18	0,965781	0,964823	0,963484	0,964115
Resnet34	0,966633	0,966228	0,964668	0,965419
Resnet50	**0,969356**	**0,967921**	**0,970046**	**0,968861**

Elaborado pelo autor (2021)

Outro dado interessante e possível de analisar é o tempo médio que cada um dos modelos leva para realizar uma única classificação. Para obter esta informação, foram realizadas 100 classificações com imagens aleatórias do banco de dados, a partir destes valores, realiza-se uma média simples para obter o tempo de inferência de uma única imagem. Na tabela 6 pode-se observar este resultado.

Tabela 6: Resultados obtidos com relação ao tempo de classificação

Arquitetura	Quantidade de imagens	Tempo de execução (s)	Tempo de inferência para uma imagem (ms)
Resnet18	100	7,2239	72
Resnet34	100	**7,0596**	**71**
Resnet50	100	7,9971	80

Elaborado pelo autor (2021)

Nesta tabela, percebe-se que as duas primeiras arquiteturas tiveram valores bem próximos, enquanto a terceira demorou aproximadamente 10% a mais. Isso é esperado pois a Resnet50 se trata de uma arquitetura mais profunda, portanto mais cálculos são realizados.

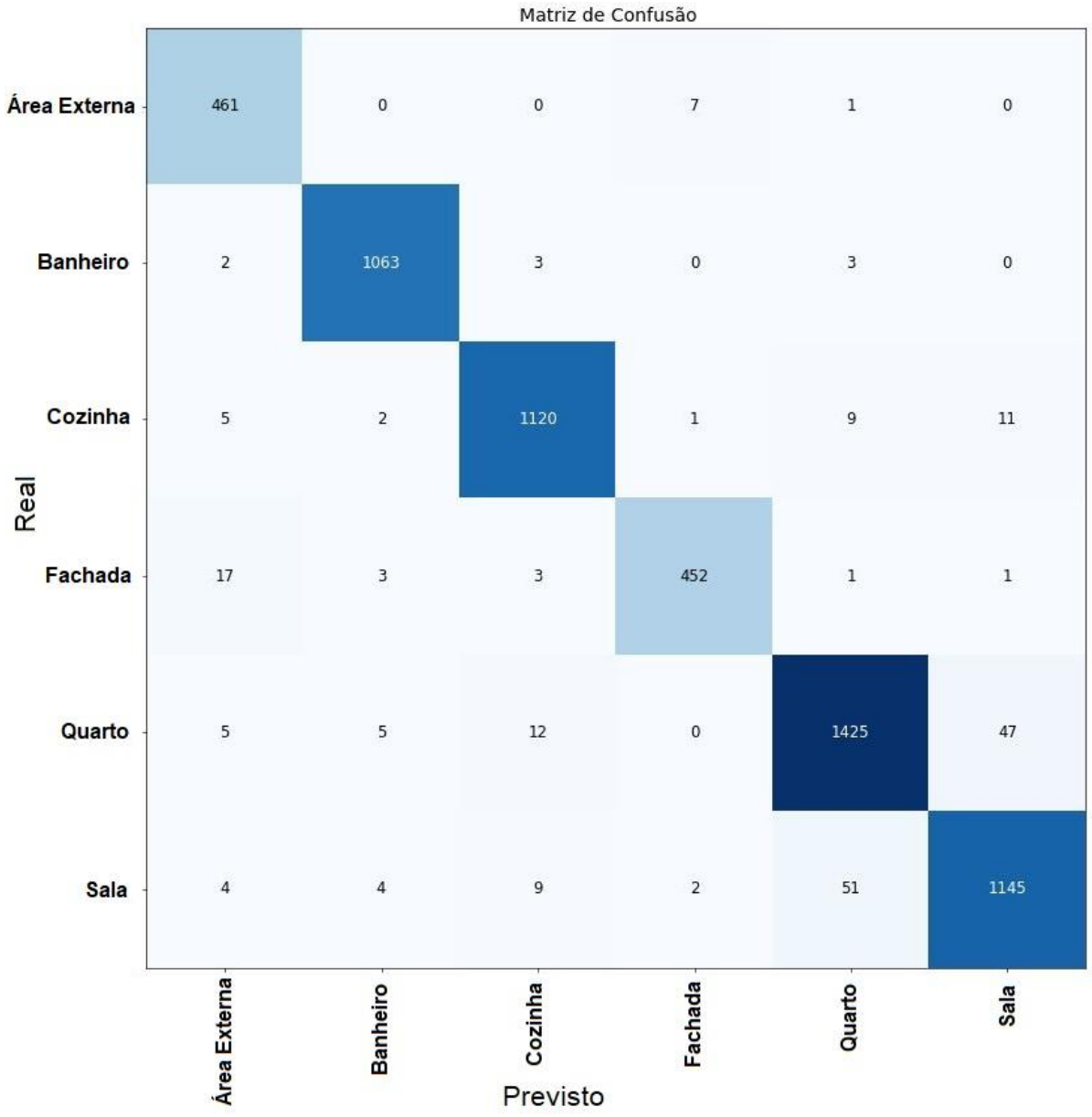
Após analisar os dados disponíveis, optou-se pelo uso do modelo treinado a partir da arquitetura Resnet50, pois este obteve melhores resultados em todas as métricas analisadas, e também a diferença no tempo de inferência é aceitável.

4.2 Avaliação detalhada da arquitetura selecionada

O modelo escolhido foi treinado a partir da arquitetura Resnet50 e teve uma acurácia média de 96%, porém se observada a matriz de confusão na figura 22, percebe-se que há uma certa disparidade no desempenho das classes. Os principais erros aconteceram entre as classes quarto e sala.

No caso das fotos de quarto, 1425 foram corretamente previstas e das restantes, quase que na sua totalidade, foram classificadas como sala. O mesmo se observa no sentido contrário, onde 51 das fotos de sala foram classificadas como quarto.

Figura 22: Resultados obtidos na matriz de confusão do modelo escolhido



Elaborado pelo autor (2021)

Esta dificuldade do modelo em classificar algumas fotos de sala e quarto, pode estar ligada ao fato de que há diversos imóveis não mobiliados nos catálogos que geraram o banco de dados, e que estes dois ambientes compartilham diversas características nesta condição. O quadro 3 apresenta as imagens que geraram maior perda ao serem classificadas, ou seja, o modelo teve maior dificuldade, ao analisar as imagens.

Quadro 3: Comparação entre classificações de sala e quarto

Classe real	Previsão do modelo	Imagem
Quarto	Sala	
Sala	Quarto	

Elaborado pelo autor (2021)

Nestas imagens observa-se justamente os fatos detalhados acima, a ausência de mobília ou de qualquer tipo de característica evidente e distinguível impede com que o classificador tenha bom desempenho neste tipo de imagem em especifico. Outro ponto onde o modelo teve dificuldade, foi ao classificar 17 fotos de fachada como sendo área externa.

Figura 23: Análise do banco de dados frente a dificuldade da classe área externa

Elaborado pelo autor (2021)

Este problema em específico se deve a grande generalização que a classe área externa representa, sendo que dentro desta classe há imagens que vêm de vários ambientes diferentes, incluindo imagens que apresentam fachadas ao fundo, ou que tenham relação com esta. Na figura 23 pode-se observar este fato, ao analisar alguns exemplos do banco de dados.

4.3 Avaliação geral do sistema frente aos requisitos

Com o modelo classificador definido, exportado e integrado a API que gerencia as requisições, pode-se realizar algumas medições para observar o comportamento do sistema em relação aos requisitos definidos no capítulo 3.

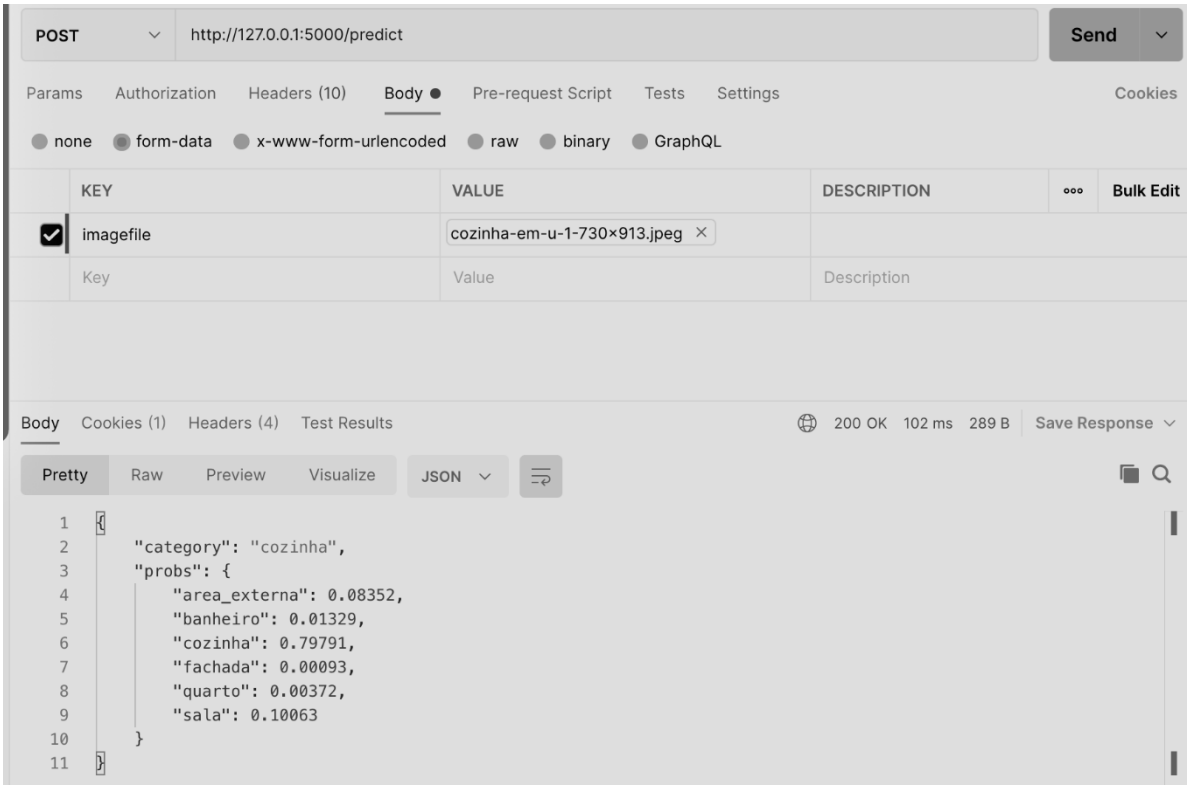
O primeiro e mais importante requisito define a função principal do sistema, classificar fotos de ao menos 6 tipos de ambientes. Este requisito foi atendido com êxito, como demonstrado nas seções anteriores.

No segundo requisito, solicitou-se que o sistema seja parametrizável, e para atender este requisito, uma variável de certeza foi inserida onde o cliente pode opcionalmente enviá-la junto com a imagem. Desta forma, caso o sistema receba este valor de certeza, ele retorna a classe mais provável caso esta esteja acima do valor de certeza. Caso não seja enviado esta variável, o sistema funciona como padrão retornando a classe com maior probabilidade.

Por último, define-se que o sistema deve receber requisições remotas, e para atender a este requisito, foi desenvolvida uma aplicação que recebe uma requisição HTTP, contendo a imagem e o parâmetro, e retorna as probabilidades das classes. Este servidor foi executado para testes localmente, e pode ser facilmente integrado a um serviço de nuvem, o que permitiria às requisições remotas.

Utilizando a ferramenta para realizar requisições remotas Postman, foi possível testar a aplicação utilizando imagens retiradas da internet, na figura 24 demonstra-se o uso desta ferramenta e a resposta da requisição. Neste caso, foi enviado uma imagem de cozinha para a API, e como retorno obteve-se a classe correta e as probabilidades de cada classe.

Figura 24: Resultados obtidos testando a API classificadora



Elaborado pelo autor (2021)

Neste servidor, foram feitos testes para obter tempo médio de uma requisição. Foram realizadas 100 requisições com imagens aleatórias do banco de dados, e obteve-se um tempo médio de resposta de 80 ms aproximadamente.

5 CONCLUSÃO

O presente trabalho descreve o desenvolvimento de um sistema de classificação de imagens de ambientes residenciais. Para atingir os requisitos, foram aplicadas técnicas de aprendizado de máquina, com foco em aprendizado profundo. Posteriormente este sistema foi integrado a um serviço que pode ser disponibilizado na internet.

O sistema desenvolvido neste trabalho demonstra que é possível aplicar técnicas de visão computacional para solucionar problemas repetitivos, como a classificação de fotos no domínio de negócio de imóveis. O modelo final utilizando a arquitetura Resnet50, treinado por 10 rodadas, atingiu uma acurácia de 96,94%, com um tempo de inferência médio de 80 ms. Após integrado em uma API, cada requisição de classificação retornou a classe mais provável em até 100 ms.

Neste trabalho, três arquiteturas foram selecionadas com base em estudos correlacionados. A fim de validar o tipo de técnica aplicada neste trabalho, sugere-se como trabalho futuro comparar os resultados aqui demonstrados com outras arquiteturas.

Também sugere-se realizar uma ampliação e otimização no banco de dados, observando categorias mais específicas que não foram abordadas neste projeto, ou mesmo detalhando categorias como área externa, composta por diversos tipos de imagens que podem ser separados em suas próprias classes como garagem, piscina ou parque.

Por fim, a API desenvolvida é um protótipo básico, que não gerencia usuários ou nenhum outro tipo de informação. Para utilizar tal serviço em produção no futuro será necessário que outros aspectos sejam incorporados e considerados.

REFERÊNCIAS

ARAÚJO, Flávio H. D., Redes Neurais Convolucionais com Tensorflow: Teoria e Prática, Escola Regional de informática do Piauí, v. 1, n. 1, p. 382-406, jun, 2017. Disponível em: https://docplayer.com.br/57119969-Redes-neurais-convolucionais-com-tensorflow-teoria-e-pratica.html. Acesso em: 19 ago. 2021.

CUNHA, Kelvin. **Reconhecimento e detecção de logotipos a partir de redes neurais convolucionais profundas**. 2017. Dissertação (Bacharelado em Engenharia da computação) – Universidade Federal de Pernambuco, Recife, 2017. Disponível em: https://www.cin.ufpe.br/~tg/2016-2/kbc.pdf. Acesso em: 20 ago. 2021.

Deep Learning for Coders with Fastai and PyTorch. Massachusetts:O'Reilly Media, Inc., 2018.

Dias, Nuriele da Silva. ADAPTAÇÃO ORGANIZACIONAL: A INFLUÊNCIA DAS NOVAS TECNOLOGIAS NA ESTRATÉGIA DE EMPRESAS DO SETOR IMOBILIÁRIO DA GRANDE FLORIANÓPOLIS . 2019. Dissertação (Bacharelado em Administração) - Universidade Federal de Santa Catarina, UFSC, 2019. Disponível em: https://repositorio.ufsc.br/handle/123456789/202218. Acesso em: 20 ago. 2021.

FERNEDA, Edberto. Redes neurais e sua aplicação em sistemas de recuperação de informação, Brasília, v. 35, n. 1, p. 25-30, jan./abr. 2006. Disponível em: https://www.scielo.br/j/ci/a/SQ9myjZWLxnyXfstXMgCdcH/?format=pdf&lang=pt. Acesso em: 10 Jul 2021.

FERRARI, D. G.; DE CASTRO SILVA, L. N. **Introdução a mineração de dados**. [s.l.] Saraiva Educação S.A., 2017.

HAYKIN, S. S.; HAYKIN, S. S.; HAYKIN, S. S.; HAYKIN, S. S. Neural networks and learning machines. [S.l.]: Pearson Upper Saddle River, NJ, USA:, 2009. v. 3.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. p. 770–778.

HOWARD, Jeremy. **Deep Learning for Coders with Fastai and PyTorch**. Notas do curso: Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD - the book and the course. Disponível em: https://github.com/fastai/fastbook Acesso em: 10 maio 2021.

KARPATHY, Andrej. Convolutional Neural Networks (CNNs / ConvNets). Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/convolutional-networks/. Acesso em: 20 jan. 2020.

LEMOS, Yessica Maria Valencia. Inspeção automática de defeitos em ovos comerciais usando visão computacional . 2021. Dissertação (Mestrado em

Engenharia de Automação e Sistemas) – UFSC, Florianópolis, 2021. Disponível em: https://repositorio.ufsc.br/handle/123456789/227084. Acesso em: 20 jun. 2021.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, Springer, v. 5, n. 4, p. 115–133, 1943.

PEEMEN, M.; MESMAN, B.; CORPORAAL, H. Speed sign detection and recognition by convolutional neural networks. In: Proceedings of the 8th International Automotive Congress. [S.l.: s.n.], 2011. p. 162–170.

RAUBER, Thomas Walter. Redes Neurais Artificiais. *In*: ERI '97 – ENCONTRO REGIONAL DE INFORMÁTICA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 1997, Nova Friburgo, RJ e Vitória, ES - 1997.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SILVA, Rodrigo Emerson Valentin da. UM ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA A CLASSIFICAÇÃO DE IMAGENS . 2018. Dissertação (Bacharelado em Sistemas de Informação) – UFC, QUIXADÁ, 2018. Disponível em: http://www.repositorio.ufc.br/handle/riufc/39475. Acesso em: 20 jun. 2021.

SOUZA, Gustavo. Google registra um aumento de 668% na busca por casas para alugar. **Exame**, São Paulo, 21 set. 2020. Notícias. Disponível em: https://exame.com/mercado-imobiliario/google-busca-por-casas-para-alugar/. Acesso em: 15 jul. 2021.

SOUZA, Victor. Análise Comparativa de Redes Neurais Convolucionais no Reconhecimento de Cenas. **Ciência da Informação**, Marabá, v. 11, n. 1, p. 419-426, set. 2020. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/view/16801. Acesso em: 19 jul. 2021.

TEHREEM, Naeem. Definição da API REST: O que é uma API REST (API RESTful)?, Astera, São Paulo, 28 Janeiro. 2020. Notícias. Disponível em: https://www.astera.com/pt/tipo/blog/defini%C3%A7%C3%A3o-de-api/. Acesso em: 20 ago. 2021.

VON WANGENHEIM, Aldo. Deep Learning::Introdução ao Novo Coneccionismo. Disponível em: http://www.lapix.ufsc.br/ensino/visao/visaocomputacionaldeep-learning/deep-learning introducao-ao-novo-coneccionismo/. Acesso em: 20 jun. 2021.

YAO, Shijing. Categorizing Listing Photos at Airbnb. Medium, 2 Maio. 2018. Notícias. Disponível em: https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3. Acesso em: 18 jun. 2021.

YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional neural networks: an overview and application in radiology. Insights into imaging, Springer, v. 9, n. 4, p. 611–629, 2018.

ZIVIANI, NIVIO. **Projeto de Algoritmos Com Implementações em Pascal e C**. São Paulo: Pioneira, 1999.

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS

DECLARAÇÃO DE FINALIZAÇÃO DE TRABALHO DE CURSO

Declaro que o estudante Ramon Brignoli**,** matrícula nº 152006309-1, do Curso de Engenharia Mecatrônica, defendeu o trabalho intitulado *Estudo de Redes Neurais Convolucionais e sua Aplicação para Classificação de Ambientes,* o qual está apto a fazer parte do banco de dados da Biblioteca Hercílio Luz do Instituto Federal de Santa Catarina, Campus Florianópolis.

Florianópolis, 16 de setembro de 2021.

Documento assinado digitalmente
MAURICIO EDGAR STIVANELLO
Data: 24/09/2021 11:18:36-0300
CPF: 901.390.629-04

_____

Prof. Orientador do TCC: Maurício Edgar Stivanello, Dr. Eng.