UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MÁRCIO VINÍCIUS DA SILVA DOS SANTOS

IDENTIFICAÇÃO DE FACHADAS RESIDENCIAIS E COMERCIAIS USANDO REDE NEURAL CONVOLUCIONAL

São Luís - MA
2025

MÁRCIO VINÍCIUS DA SILVA DOS SANTOS

IDENTIFICAÇÃO DE FACHADAS RESIDENCIAIS E COMERCIAIS USANDO REDE NEURAL CONVOLUCIONAL

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso de Engenharia de Computação, da Universidade Estadual do Maranhão, como pré-requisito para a obtenção de título de Bacharel em Engenharia de Computação.

Orientador: Prof Msc. Pedro Brandão Neto

São Luís - MA

2025

Santos, Márcio Vinícius da Silva dos
Identificação de fachadas residenciais e comerciais usando rede neural convolucional. / Márcio Vinícius da Silva dos Santos. – São Luis, MA, 2025.
50 f

TCC (Curso de Graduação em Engenharia da computação) - Universidade Estadual do Maranhão, 2025.

Orientador: Prof. Me. Pedro Brandão Neto.

1.Fachadas. 2.Classificação. 3.Rede Neural Convolucional. 4.ResNet50. 5.Transfer Learning. I.Titulo.

CDU: 004.032.26

Elaborado por Cássia Diniz - CRB 13/910

MÁRCIO VINÍCIUS DA SILVA DOS SANTOS

IDENTIFICAÇÃO DE FACHADAS RESIDENCIAIS E COMERCIAIS USANDO REDE NEURAL CONVOLUCIONAL

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso de Engenharia de Computação, da Universidade Estadual do Maranhão, como pré-requisito para a obtenção de título de Bacharel em Engenharia de Computação.

Aprovado em: 20 de Fevereiro de 2025.

BANCA EXAMINADORA

Prof. Me. Pedro Brandão Neto
Orientador

Prof. Dr. Luís Carlos Costa Fonseca
Examinador

Documento assinado digitalmente
gov.br
YONARA COSTA MAGALHAES
Data: 20/02/2025 14:31:59-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Yonara Costa Magalhães
Examinadora

Este trabalho é dedicado à minha familia.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por me conceder a vida, saúde e força para seguir em frente. Sou grato por Sua presença constante em minha jornada. À minha família, que sempre esteve ao meu lado, me apoiando, muitas conquistas não seriam possíveis sem o suporte deles. Também sou imensamente grato a todos os professores e professoras da Universidade Estadual do Maranhão, que contribuíram para a minha formação.

RESUMO

A classificação de fachadas como residenciais ou empresariais é um assunto importante em diversas áreas como segurança pública, análise imobiliária e planejamento urbano. Com o *Deep Learning* (DL), redes neurais convolucionais (CNNs) são utilizadas para reconhecimento de padrões em imagens. Neste trabalho, utiliza-se um modelo baseado na arquitetura *ResNet50*, empregando *Transfer Learning* para melhorar o desempenho na classificação de fachadas, adaptando o modelo pré treinado para permitir a generalização ao contexto específico das fachadas urbanas. Além disso, técnicas de *Data Augmentation* são usadas para expandir o conjunto de imagens disponíveis e evitar problemas de *overfitting*. A abordagem proposta utiliza camadas de *Global Average Pooling* para reduzir a dimensionalidade dos dados extraídos pela *ResNet50* e camadas densas para a classificação binária entre fachadas residenciais e empresariais. A função de perda utilizada é a *Binary Crossentropy*, sendo o modelo otimizado com o algoritmo *Adam*, que garante estabilidade durante o treinamento. Os experimentos realizados demonstraram que a combinação de modelos pré-treinados com um *dataset* regionalizado permitiu a criação de um classificador eficiente para essa tarefa, alcançando bons níveis de acurácia e área sob a curva *ROC(AUC)*. Os resultados indicam que a metodologia pode ser aplicada em sistemas automatizados para identificação de fachadas urbanas, contribuindo para diversas áreas, como a gestão municipal e o setor imobiliário.

Palavras-chave: Fachadas, Classificação, Rede Neural Convolucional, ResNet50, Transfer Learning.

ABSTRACT

The classification of facades as residential or commercial is an important issue in several areas such as public safety, real estate analysis and urban planning. With Deep Learning (DL), convolutional neural networks (CNNs) are used for pattern recognition in images. In this work, a model based on the ResNet50 architecture is used, employing Transfer Learning to improve the performance in facade classification, adapting the pre-trained model to allow generalization to the specific context of urban facades. In addition, Data Augmentation techniques are used to expand the set of available images and avoid over-fitting problems. The proposed approach uses Global Average Pooling layers to reduce the dimensionality of the data extracted by ResNet50 and dense layers for binary classification between residential and commercial facades. The loss function used is Binary Crossentropy, and the model is optimized with the Adam algorithm, which guarantees stability during training. The experiments carried out demonstrated that the combination of pre-trained models with a regionalized dataset allowed the creation of an efficient classifier for this task, achieving good levels of accuracy and area under the ROC curve (AUC). The results indicate that the methodology can be applied in automated systems for identifying urban facades, contributing to several areas, such as municipal management and the real estate sector.

Keywords: Facades, Classification, Convolutional Neural Network, ResNet50, Transfer Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FCN-8s	Redes Totalmente Convolucionais com um passo de 8 pixels na camada final
3D	Tridimensional
CNNs	Redes Neurais Convolucionais
DNN	Deep Neural Network
DCNN	Redes Neurais Convolucionais Profundas
DL	Deep Learning
ReLu	Rectified Linear Unit
DLB	Deep Learning Book
FN	Falso Negativo
FP	Falso Positivo
ReLU	Rectified Linear Unit
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
CPU	Central Processing Unit
GPU	Graphical Processing Unit
TPU	Tensor Processing Unit
ROC	Receiver Operating Characteristic
AUC	Área sob a Curva

SUMÁRIO

1 INTRODUÇÃO

A classificação de fachadas de edificações como residenciais ou empresariais é um desafio relevante em áreas como segurança pública, análise imobiliária e planejamento urbano. Com o avanço do aprendizado profundo, do inglês *Deep Learning* (DL), técnicas baseadas em redes neurais convolucionais (*CNNs*) são amplamente utilizadas para tarefas de reconhecimento de padrões em imagens (GOODFELLOW; BENGIO; COURVILLE, 2016).

A arquitetura *ResNet* revolucionou o campo do aprendizado profundo ao introduzir conexões residuais que facilitam o treinamento de redes neurais muito profundas. Apresentada por He et al. (2016) em 2016, a *ResNet* permitiu a criação de modelos com até 152 camadas, como a *ResNet*-152, sem enfrentar os problemas de degradação que anteriormente limitavam a profundidade das redes neurais. Neste trabalho, propõe-se a utilização de um modelo baseado na arquitetura *ResNet50* (HE et al., 2016), empregando Transfer Learning para melhorar o desempenho na classificação de fachadas. Como os modelos pré-treinados foram desenvolvidos para grandes bases de dados, como o ImageNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), adaptações são feitas para permitir a generalização ao contexto específico das fachadas urbanas. Além disso, técnicas de Data Augmentation são aplicadas para expandir o conjunto de imagens disponíveis e evitar problemas de overfitting (SHORTEN; KHOSHGOFTAAR, 2019).

Este trabalho utiliza uma camada de *Global Average Pooling* para reduzir a complexidade dos dados extraídos pelo modelo pré-treinado *ResNet* e prepará-los para a camada densa, responsável pela classificação binária entre fachadas residenciais e comerciais. A função de perda utilizada é a *Binary Crossentropy*, e o modelo é otimizado com o algoritmo *Adam* para garantir estabilidade durante o treinamento. O *dataset* utilizado foi criado a partir de imagens do Google (2023), permitindo uma abordagem mais regionalizada. Os resultados demonstram que a combinação de modelos pré-treinados com *Transfer Learning* e um *dataset* específico possibilita a criação de um classificador com bons níveis de acurácia e *AUC-ROC*. Esse modelo pode ser aplicado em sistemas para identificação de fachadas urbanas, contribuindo para diversas áreas, como a gestão municipal e o setor imobiliário.

1.1 Objetivos

A seguir é apresentado o objetivo geral deste trabalho e também os objetivos específicos.

1.1.1 Objetivo Geral

- Identificar se uma fachada é residencial ou empresarial a partir de uma imagem.

1.1.2 Objetivos Específicos

- Criar uma arquitetura de redes convolucionais para identificar fachadas como residenciais ou empresariais, contribuindo para áreas investigativas;
- Organizar uma base de dados para treinar e testar o modelo treinado, com imagens locais;

1.2 Trabalhos Relacionados

Em Kang et al. (2018), abordam a tarefa de classificar instâncias individuais de edifícios utilizando imagens coletadas do *Google Street View*. A pesquisa explora o uso de técnicas avançadas de aprendizado de máquina, em especial redes neurais convolucionais (*CNNs*), para identificar e classificar edifícios em diferentes categorias, com base nas características visuais das fachadas. Os autores propõem uma metodologia que integra informações de múltiplas visualizações das imagens de rua, permitindo uma análise mais robusta e precisa, mesmo em cenários urbanos complexos e com variações de ângulos e iluminação.

O Wang et al. (2023) propôs uma abordagem inovadora para a classificação de estilos arquitetônicos utilizando redes neurais convolucionais (*CNN*) combinadas com mecanismos de atenção espacial e de canal. A pesquisa foca na melhoria da precisão de classificação ao incorporar uma arquitetura de atenção que permite ao modelo focar de forma seletiva nas regiões mais relevantes das imagens, além de destacar características importantes relacionadas aos estilos arquitetônicos. A combinação dessas duas estratégias oferece ao modelo uma maior capacidade de diferenciar os padrões sutis nas fachadas dos edifícios, mesmo em contextos complexos e variados.

Em Liu et al. (2017) propõe a segmentação de fachadas de edifícios utilizando redes neurais profundas. Os autores introduzem uma função de perda simétrica que incorpora características estruturais comuns em construções humanas, como simetria, aprimorando a precisão na segmentação de elementos como janelas, portas e varandas. Além disso, a pesquisa utiliza redes neurais convolucionais totalmente convolucionais (*FCN-8s*) treinadas com essa função de perda, obtendo resultados superiores em comparação com métodos anteriores nos conjuntos de dados *ECP* e *eTRIMS*.

Já em Yoshimura et al. (2019) os autores aplicam técnicas avançadas de aprendizado profundo e visão computacional para medir semelhanças visuais entre projetos arquitetônicos de diferentes arquitetos. Utilizando um conjunto de dados composto por

imagens coletadas da web e uma coleção original de obras arquitetônicas, eles treinam uma rede neural convolucional profunda (*DCNN*) capaz de classificar com 73% de precisão obras pertencentes a 34 arquitetos distintos. Ao analisar os pesos da rede treinada, os pesquisadores conseguem medir quantitativamente as semelhanças visuais entre os arquitetos, agrupando-os de acordo com características compartilhadas. Esses agrupamentos corroboram visões convencionais na história da arquitetura, demonstrando que as características aprendidas pela rede neural estão alinhadas com a compreensão tradicional dos designs arquitetônicos.

1.3 Estrutura do Trabalho

Este trabalho de conclusão de curso apresenta um modelo de classificação de imagens, com base em duas classes: residencial e comercial. O capítulo 2 aborda os conceitos como *Deep Learning*, Redes Neurais Convolucionais e *ResNet*. No capítulo 3 são apresentadas a metodologia utilizada neste trabalho, a base de dados, a arquitetura utilizada, a arquitetura da rede pré-treinada e as métricas de avaliação de desempenho. No capítulo 4 são apresentados os resultados. Por fim, no capítulo 5 são abordadas as considerações finais deste trabalho e as propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo traz uma revisão utilizada como base para a elaboração do trabalho, abordando sobre Redes Neurais, Redes Neurais Convolucionais, *Deep Learning* e *ResNet*.

2.1 Redes Neurais

As Redes Neurais Artificiais (RNAs), ou simplesmente Redes Neurais, são modelos computacionais inspirados na estrutura e no funcionamento do cérebro humano (HAYKIN, 2009). Elas são compostas por unidades de processamento interconectadas, chamadas neurônios artificiais ou nós, que se organizam em camadas. Essas camadas são geralmente divididas em três tipos principais:

- Camada de Entrada (*Input Layer*): Recebe os dados brutos de entrada, representando as características ou atributos do problema a ser resolvido. Cada neurônio nesta camada corresponde a uma característica específica.
- Camadas Ocultas (*Hidden Layers*): Realizam o processamento intermediário dos dados. Uma rede neural pode ter uma ou várias camadas ocultas, e cada camada aplica transformações matemáticas aos dados recebidos da camada anterior. O número de camadas ocultas e o número de neurônios em cada camada são parâmetros importantes que afetam a capacidade da rede de aprender padrões complexos (GOODFELLOW, 2016).
- Camada de Saída (*Output Layer*): Produz a saída final da rede, que representa a solução do problema. O formato da saída depende da tarefa a ser realizada (classificação, regressão, etc.).

2.1.1 Funcionamento de um Neurônio Artificial

Cada neurônio artificial recebe múltiplas entradas, cada uma associada a um peso sináptico. O neurônio calcula uma soma ponderada das entradas, adiciona um bias (viés) e aplica uma função de ativação ao resultado. A função de ativação introduz não-linearidade no modelo, permitindo que a rede aprenda relações complexas entre as entradas e as saídas (NIELSEN, 2015). Matematicamente, o processo pode ser descrito da seguinte forma:

Soma Ponderada:

$$z = (w1 * x1) + (w2 * x2) + ... + (wn * xn) + b \tag{2.1}$$

Temos:

- z é a soma ponderada das entradas.
- wi são os pesos sinápticos associados a cada entrada.
- xi são os valores das entradas.
- b é o bias.

para a Função de Ativação:

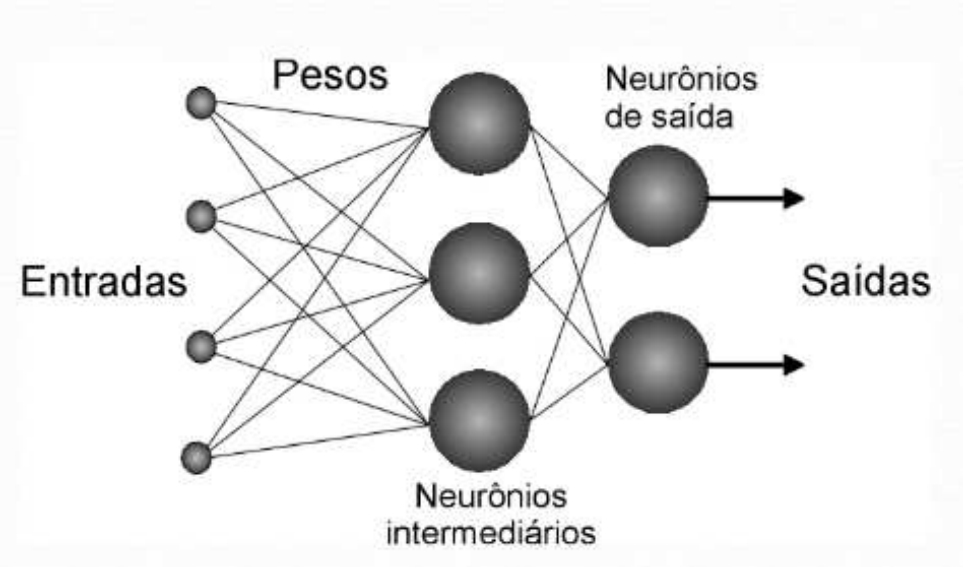
$$a = f(z) \tag{2.2}$$

Onde:

- a é a saída do neurônio.
- f é a função de ativação.

Na figura 1 demonstra como é a estrutura de uma rede neural.

Figura 1 – Estrutura de uma rede neural.



Fonte: (Cérebro Mente, Acesso em 2025)

2.1.2 Funções de ativação

É importante destacar também e detalhar as funções de ativação comuns:

2.1.2.1 *Sigmoid*

Mapeia a entrada para um valor entre 0 e 1. Útil para problemas de classificação binária, mas pode sofrer com o problema do desaparecimento do gradiente (RUMELHART; HINTON; WILLIAMS, 1986).

2.1.2.2 Tangente Hiperbólica (Tanh)

Similar à *Sigmoid*, mas mapeia a entrada para um valor entre -1 e 1. Também pode sofrer com o problema do desaparecimento do gradiente (GLOROT; BENGIO, 2010).

2.1.2.3 *ReLU (Rectified Linear Unit)*

Retorna a entrada se for positiva, caso contrário, retorna zero. Amplamente utilizada em redes profundas devido à sua eficiência computacional e capacidade de mitigar o problema do desaparecimento do gradiente (NAIR; HINTON, 2010).

A função de ativação *ReLU* é uma das mais populares e amplamente utilizadas em Redes Neurais Convolucionais e outras arquiteturas de *Deep Learning* (NAIR; HINTON, 2010). Possui uma excelente eficiência computacional, simplicidade e capacidade de reduzir o problema do desaparecimento do gradiente, que ocorre em redes profundas.

2.1.2.4 *Leaky ReLU*

Similar à *ReLU*, mas permite uma pequena inclinação para valores negativos, evitando o problema do "neurônio morto"(MAAS et al., 2013).

2.1.3 Aprendizado em Redes Neurais

O aprendizado em redes neurais envolve ajustar os pesos sinápticos e os *biases* de forma a minimizar o erro entre a saída da rede e a saída desejada. Esse processo é realizado através de algoritmos de otimização, como o Gradiente Descendente (*Gradient Descent*) e suas variantes (Adam, SGD, RMSprop, etc.) (KINGMA, 2014). O algoritmo de retropropagação (*Backpropagation*) é utilizado para calcular o gradiente do erro em relação aos pesos e *biases*, permitindo que o algoritmo de otimização ajuste os parâmetros na direção que reduz o erro (RUMELHART; HINTON; WILLIAMS, 1986).

Tipos de Redes Neurais:

Existem diversos tipos de redes neurais, cada um adequado para diferentes tipos de problemas:

- Redes *Feedforward*: O tipo mais básico de rede neural, onde os dados fluem em uma única direção, da entrada para a saída (ROSENBLATT, 1958).

- Redes Recorrentes (RNNs): Possuem conexões de *feedback*, permitindo que a rede mantenha um estado interno e processe sequências de dados (HOCHREITER, 1997).
- Redes Convolucionais (CNNs): Especializadas no processamento de imagens e vídeos, utilizando camadas convolucionais para extrair características relevantes dos dados (LECUN et al., 1998).
- *Autoencoders*: Utilizadas para aprendizado não supervisionado, buscando aprender uma representação compacta dos dados de entrada (HINTON; SALAKHUTDINOV, 2006).
- Redes Generativas Adversariais (GANs): Compostas por duas redes (gerador e discriminador) que competem entre si para gerar dados sintéticos semelhantes aos dados reais (GOODFELLOW et al., 2014).

As Redes Neurais são ferramentas poderosas para resolver uma variedade de problemas, desde classificação de imagens, reconhecimento de voz até previsão de séries temporais e processamento de linguagem natural. No entanto, é importante entender os princípios básicos do seu funcionamento e os diferentes tipos de arquiteturas para escolher a mais adequada para cada tarefa.

2.1.4 Arquitetura das Redes Neurais

A arquitetura de uma rede neural é um fator crítico que influencia seu desempenho. O design da rede envolve a escolha do número de camadas ocultas, o número de neurônios em cada camada e as funções de ativação utilizadas. Redes mais profundas, conhecidas como *Deep Neural Networks*, têm várias camadas ocultas e são capazes de aprender representações hierárquicas dos dados (BENGIO; COURVILLE; VINCENT, 2013). A profundidade permite que a rede capture características de alto nível a partir de características de baixo nível, o que se torna muito útil em tarefas como reconhecimento de imagens.

2.1.5 Regularização em Redes Neurais

Um desafio comum no treinamento de redes neurais é o *overfitting*, que nada mais é quando o modelo aprende a memorizar os dados de treinamento em vez de generalizar para novos dados. Para solucionar esse problema, várias técnicas de regularização são aplicadas, como por exemplo a *Dropout*, *L2 Regularization* e a *Early Stopping*.

O processo de *Dropout* ocorre durante o treinamento, onde uma fração dos neurônios é aleatoriamente "desligada"(ou seja, suas ativações são definidas como zero)

em cada iteração. Isso força a rede a aprender representações mais robustas e reduz a dependência entre os neurônios (SRIVASTAVA et al., 2014).

A *L2 Regularization*, também conhecida como *weight decay*, é uma técnica que penaliza grandes pesos na função de custo, incentivando a rede a manter pesos menores e mais distribuídos (NG, 2004).

E por últmo temos a *Early Stopping*, que é uma técnica responsável por monitorar e interromper o treinamento do modelo quando o desempenho começa a decair, evitando assim o *overfitting*.

2.1.6 Treinamento e otimização

O treinamento eficaz das redes neurais envolve não apenas a escolha do algoritmo de otimização, mas também o ajuste dos hiperparâmetros. Hiperparâmetros como taxa de aprendizado, tamanho do lote (*batch size*) e número de épocas têm um impacto significativo na convergência do modelo. A escolha da taxa de aprendizado é particularmente crítica; uma taxa muito alta pode levar a oscilações e divergência no treinamento, enquanto uma taxa muito baixa pode resultar em um processo extremamente lento (BERGSTRA; BENGIO, 2012).

É um processo crítico que envolve a adaptação dos pesos das conexões entre os neurônios para minimizar o erro na previsão das saídas. Este processo é geralmente realizado através de um método conhecido como retropropagação (*backpropagation*), que utiliza o algoritmo de gradiente descendente para ajustar os pesos com base no erro calculado entre a saída prevista e a saída real (RUMELHART; HINTON; WILLIAMS, 1986).

2.2 Redes Neurais Convolucionais

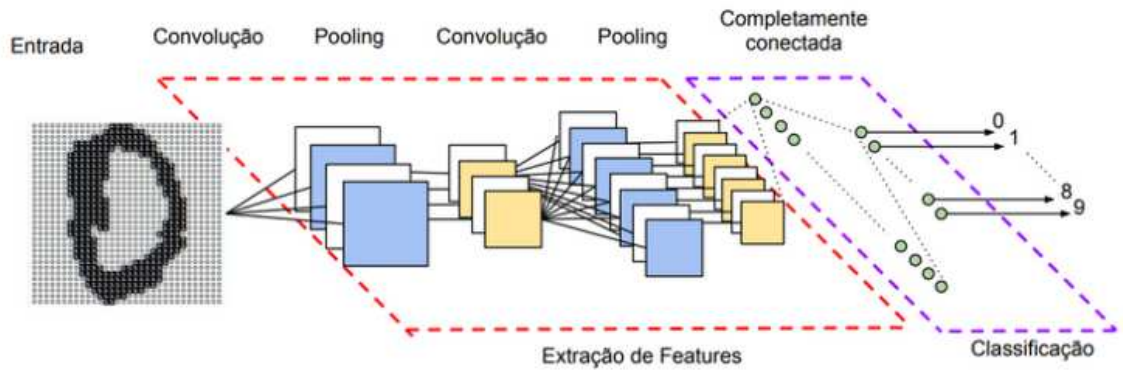
Redes Neurais Convolucionais (*CNNs*) são um tipo de rede neural específica para lidar com o processamento de imagens, para processar dados com uma estrutura de grade, vídeos e sinais de áudio (LECUN et al., 1998). Elas são capazes de receber uma imagem de entrada e extrair características adicionando pesos para aprender os padrões das imagens. É um tipo de rede que exige pouco pré-processamento.

CNNs são modelos de rede *Deep Learning* mais conhecidos e utilizados atualmente em projetos que evolvam imagens como principal insumo (SHAKIBHAMEDAN et al., 2024). O que caracteriza esse tipo de rede é ser composta de camadas convolucionais, que processa as entradas considerando campos receptivos locais. Adicionalmente inclui operações conhecidas como *pooling*, responsáveis por reduzir a dimensionalidade espacial das representações (PONTI; COSTA, 2018).

As *CNNs* têm um excelente desempenho em problemas de aprendizado de máquina. Especialmente as aplicações que lidam com dados de imagem, como o maior conjunto de dados de classificação de imagens (ImageNet), visão computacional e processamento de linguagem natural (NLP) e os resultados alcançados foram muito surpreendentes (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

De acordo com Gu et al. (2018) existem diversas variantes de arquiteturas de CNNs na literatura, porém, seus componentes básicos são muito semelhantes. A arquitetura básica normalmente consiste em três tipos de camadas: *convolutional layer* ou camada convolucional, *pooling layer* ou camadas de agrupamento e *fully-connected layer* ou camada totalmente conectada.

Figura 2 – Componentes de uma CNN.



Fonte: Retirado de (VARGAS; PAES; VASCONCELOS, 2016)

2.2.1 Camadas Convolucionais

É uma camada essencial para a CNN, que consiste em *kernels* (ou filtros) responsáveis por realizar operações de convolução, assim as imagens de entrada passam por esses filtros e cada filtro aprende a detectar um padrão ou característica específico daquela entrada. A operação de Convolução é uma operação que combina o filtro com uma pequena região da entrada, que resulta em um único valor representando a ativação do filtro, para ser repassado para a próxima camada.

Dois parâmetros importantes que controlam o comportamento das camadas convolucionais são o *stride* e o *padding* (DUMOULIN; VISIN, 2016).

- Stride: Responsável por definir o número de *pixels* que o filtro se move, um *stride* maior tem como consequência um mapa de características de saída menor e reduz a sobreposição entre as regiões processadas.

- *Padding*: Permite que o filtro processe as bordas da imagem sem precisar reduzir excessivamente o tamanho do mapa de características, isso ocorre devido ao *padding* adicionar *pixels* extras a borda da imagem de entrada. Há também outros tipos de *padding*, como por exemplo o "zero-padding"que adiciona *pixels* com valores zeros, e o "same-padding"que adiciona *pixels* com o objetivo de garantir que o mapa de características de saída tenha o mesmo tamanho da entrada.

2.2.1.1 Múltiplos filtros e canais

Camadas convolucionais geralmente utilizam múltiplos filtros para detectar diferentes padrões na entrada. Cada filtro gera um mapa de características diferente, e todos os mapas de características são combinados para formar a saída da camada. Além disso, as camadas convolucionais podem processar entradas com múltiplos canais (ex: imagens coloridas com canais vermelho, verde e azul). Nesse caso, cada filtro tem a mesma profundidade da entrada e realiza a convolução em todos os canais simultaneamente (GOODFELLOW, 2016).

2.3 *Deep Learning*

O aprendizado profundo, conhecido como DL, é um segmento da inteligência artificial que começou a ser explorado na década de 1950 (GOODFELLOW; BENGIO; COURVILLE, 2016). O DL é uma rede neural formada por camadas ocultas e um número elevado de operações. Seu objetivo é desmembrar problemas complexos em partes mais simples. Assim, cada camada desempenha o papel de extrair características específicas, e a combinação destas resulta no produto final. As camadas ocultas consistem nas operações realizadas entre a camada de entrada da rede e a camada de saída. As arquiteturas de DL podem ser projetadas de acordo com a aplicação, pois são adaptáveis. Por esse motivo, existem várias arquiteturas e suas variações, todas baseadas em modelos previamente estabelecidos e bem fundamentados no meio acadêmico.

Desde então, essa técnica tem sido objeto de intensas pesquisas e melhorias. Embora tenha sido introduzido há várias décadas, sua relevância só ocorreu nos dias atuais. Havia caído em desuso devido à limitação em capacidade de processamento e armazenamento de informações, que para a época ainda não existia. Com avanço tecnológico em velocidade de processamento e em capacidade de armazenamento, o DL começou a ser aplicado em diversas áreas para solucionar uma variedade de desafios. Outro aspecto crucial é a abundância de dados disponíveis. A facilidade em reunir conjuntos de dados grandes, essencial para as estruturas de DL, favoreceu a aplicação desse campo da IA (PIRES; SANTOS; PEREIRA, 2025)

A função de ativação não-linear conhecida como *ReLu* é amplamente utilizada na técnica de DL e possui a seguinte forma $f(z) = max(0, z)$, onde z é o valor de entrada, o valor z irá se propagar se ele for maior que 0. Os pesquisadores adotaram essa função pois obtiveram bons resultados mesmo com uma base de dados grande, e por isso ela é a mais usada, porém, até o momento, não existe uma justificativa plausível por a $ReLU$ ser melhor que outras funções de ativação na literatura quando se trata de DL (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.4 Camada de *Data Augmentation*

É uma técnica para aumentar os dados de treinamento, aplicando transformações aleatórias, para a criar novos dados a partir dos já existentes. Esse aumento de dados ajuda a diminuir o *overfitting* (LEMLEY; BAZRAFKAN; CORCORAN, 2017). O processo de *data augmentation* consiste em aplicar, por exemplo, variações de brilho, saturação, zoom, reflexões horizontais e verticais, recortes, entre outros métodos, para ampliar os dados de treinamento.

Através dessa técnica conseguimos aprimorar o desempenho da rede sem introduzir características novas para aprendizado, o que resulta em uma drástica redução do *overfitting* e torna a rede mais robusta. A figura 3 é um exemplo dessa técnica.

Figura 3 – Exemplo de *data augmentation*

Fonte: Retirado do Google (2023)

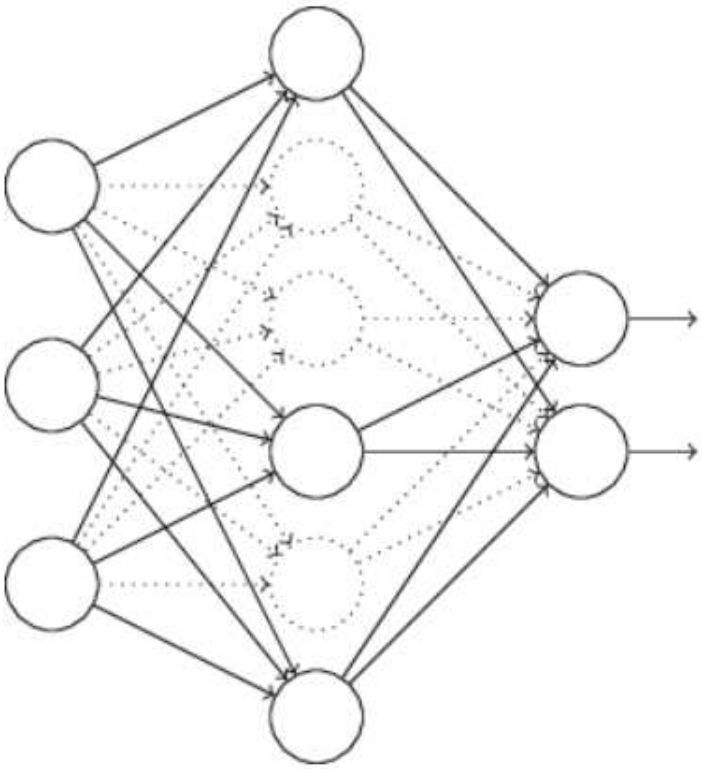
2.5 Camada de *Pooling*

É comum reduzir a dimensão espacial dos mapas ao longo das camadas da rede. Essa redução em tamanho é chamada de *poolig* sendo a operação de máximo (*maxpooling*) comumente empregada. Essa operação tem dois propósitos: em primeiro lugar, o custo computacional, pois como a profundidade dos tensores, d, costuma aumentar ao longo das camadas, é conveniente reduzir a dimensão espacial dos mesmos. Em segundo, reduzindo o tamanho das imagens obtém-se um tipo de composição de banco de filtros multirresolução que processa imagens em diferentes espaços-escala (PONTI; COSTA, 2018).

2.6 Camada de *Dropout*

A técnica de *Dropout*, sugerida por Hinton (2012), tem como objetivo diminuir o sobreajuste (*overfitting*) em redes neurais profundas. O procedimento acontece ao desligar aleatoriamente um grupo de neurônios durante a formação da rede, prevenindo que a aprendizagem se baseie excessivamente em combinações particulares de neurônios. A imagem 4 demonstra esse processo.

Figura 4 – Camada de *Dropout*



Fonte: Retirado de (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.7 *ResNet(Residual Network)*

No treinamento de redes neurais profundas existe um problema conhecido como degradação do gradiente, onde o desempenho do modelo se degrada à medida que mais camadas eram adicionadas (HE et al., 2016). Antes da *ResNet*, abordagens como *VGG* e *GoogLeNet* demonstravam que redes mais profundas podiam alcançar resultados superiores, mas apresentavam dificuldades na convergência devido ao desaparecimento do gradiente (SIMONYAN, 2014). Assim surge a *ResNet* com as conexões residuais, permitindo que a informação fluísse diretamente entre as camadas, facilitando o treinamento de redes muito profundas.

A arquitetura *ResNet* desenvolvida por He et al. (2016), revolucionou o campo das *CNNs* ao introduzir os blocos residuais, que utilizam conexões de atalho (conhecidas como *skip connections*) para suavizar o problema da degradação do gradiente em redes muito profundas. Em vez de aprender diretamente um mapeamento desejado ($H(x)$), a *ResNet* aprende um residual $F(x) = H(x) - x$, isso facilita a otimização do modelo, e melhora sua precisão. Nos experimentos conduzidos pelos autores, a *ResNet-152*, com 152 camadas, superou modelos anteriores, como *VGG* e *GoogLeNet*, no desafio *ImageNet*, demonstrando que redes mais profundas podem ser treinadas de forma eficiente sem perda

de desempenho (HE et al., 2016). Após esse trabalho, essa arquitetura tem sido adotada em diversas aplicações de visão computacional, incluindo classificação, segmentação e detecção de objetos.

Embora a *ResNet* tenha sido projetada para classificação de imagens, sua estrutura foi rapidamente adotada em outras tarefas de visão computacional. Modelos baseados na *ResNet*, como *Mask R-CNN*, são amplamente utilizados para segmentação de objetos (HE et al., 2016). Além disso, a *ResNet* também influencia redes para aprendizado por reforço, como no *AlphaGo*, e aplicações médicas, como detecção de anomalias em exames de imagem (RAJPURKAR et al., 2017).

2.7.0.1 Blocos Residuais

A *ResNet* possui dois tipos de blocos residuais:

- Blocos de identidade: Responsável por fazer a conexão residual adicionar a entrada à saída das camadas convolucionais, isso ocorre quando elas possuem a mesma dimensão.
- Bloco de convolução: Neste bloco o processo é ocorre de outra forma e são utilizados justamente quando a entrada e a saída do bloco possuem dimensões diferentes. Antes de adicionar a entrada, é feito um ajuste na dimensão em uma camada convolucional 1x1 , após esse ajuste a conexão residual adiciona a entrada à saída das camadas convolucionais.

2.7.1 Vantagens da *ResNet*

Uma das principais vantagens da *ResNet* é fazer com que as conexões residuais permitam que o gradiente flua pelas camadas, diretamente, facilitando o treinamento de redes profundas. A capacidade de treinar redes mais profundas permite que a *ResNet* aprenda padrões e características complexas dos dados. Por isso redes profundas tendem a generalizar melhor para novos dados, uma vez que são capazes de detectar padrões mais sutis e invariantes.

2.7.2 *ResNet* para classificação de imagens

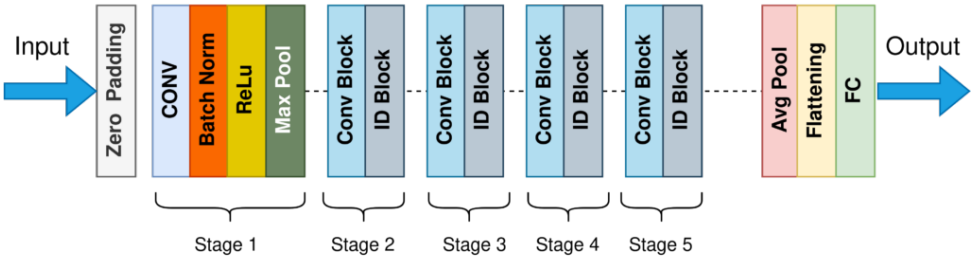
A *ResNet* tem se mostrado altamente eficaz para tarefas de classificação de imagens, alcançando resultados de última geração em diversos *benchmarks* (HE et al., 2016). A arquitetura da *ResNet* pode ser adaptada para diferentes classes e tamanho de entrada, além de permitir ser combinada com outras técnicas como *Data Augmentation* e *Transfer Learning*, que ajudam a melhorar aindamais o modelo de uma forma geral.

2.7.3 *ResNet50*

A *ResNet50* (*Residual Network* com 50 camadas) é uma arquitetura de rede neural convolucional que revolucionou o campo do *Deep Learning* ao introduzir o conceito de conexões residuais. Desenvolvida por He et al. (2016), a *ResNet50* foi projetada para resolver o problema de degradação do gradiente, que ocorre em redes neurais muito profundas e dificulta o treinamento eficiente (HE et al., 2016)

A figura 5 representa a arquitetura desse modelo.

Figura 5 – Arquitetura do modelo *ResNet50*



Fonte: Retirado do Gorlapraveen123 (2021)

2.7.3.1 Degradação do Gradiente

Em redes neurais profundas, à medida que o número de camadas aumenta, o gradiente, usado para ajustar os pesos durante o treinamento, pode se tornar muito pequeno, praticamente desaparecendo. Esse fenômeno, conhecido como desaparecimento do gradiente, impede que as camadas iniciais da rede aprendam de forma eficaz, limitando o desempenho do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.7.3.2 Conexões Residuais

A *ResNet50* resolve o problema da degradação do gradiente por meio de conexões residuais (ou "skip connections"). Essas conexões permitem que a informação "pule"uma ou mais camadas, sendo somada diretamente à saída de camadas mais profundas. Matematicamente, em vez de aprender uma função complexa $H(x)$, a ResNet aprende uma função residual $F(x) = H(x) - x$, onde x é a entrada da camada. A saída final da camada é então dada por $H(x) = F(x) + x$ (HE et al., 2016)

Essa abordagem facilita o treinamento de redes muito profundas, pois as conexões residuais permitem que o gradiente flua diretamente através das camadas, evitando o desaparecimento do gradiente. Além disso, as conexões residuais ajudam a preservar informações importantes que podem ser perdidas em redes muito profundas (HE et al., 2016)

2.7.3.3 Estrutura da *ResNet50*

A *ResNet50* é composta por 50 camadas convolucionais, estruturada em blocos residuais. Cada bloco residual contém várias camadas convolucionais, seguidas por uma conexão residual que soma a entrada original à saída do bloco. A arquitetura da *ResNet50* pode ser dividida em quatro estágios principais, cada um com um número diferente de blocos residuais:

- Estágio 1: Contém 3 blocos residuais, com 64 filtros cada.
- Estágio 2: Contém 4 blocos residuais, com 128 filtros cada.
- Estágio 3: Contém 6 blocos residuais, com 256 filtros cada.
- Estágio 4: Contém 3 blocos residuais, com 512 filtros cada.

Uma camada de *Global Average Pooling*, presente no final da rede, reduz a dimensionalidade dos mapas de características, seguida por uma camada totalmente conectada (Dense) para a classificação final (HE et al., 2016).

2.7.3.4 Vantagens da *ResNet50*

As principais vantagens a *ResNet50* em comparação com outras arquiteturas de redes neurais:

- Facilidade de Treinamento: As conexões residuais permitem o treinamento eficiente de redes muito profundas, evitando o desaparecimento do gradiente (HE et al., 2016).
- Desempenho Superior: A *ResNet50* alcançou resultados *state-of-the-art* em competições como o *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, superando arquiteturas anteriores como *VGG* e *GoogleNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2017).
- Flexibilidade: A *ResNet50* pode ser facilmente adaptada para diferentes tarefas de visão computacional, como classificação, detecção de objetos e segmentação semântica (HE et al., 2016)

2.7.3.5 Transfer Learning com a ResNet50

Uma das principais vantagens da *ResNet50* é a possibilidade de utilizar *Transfer Learning*, uma técnica que permite aproveitar os pesos pré-treinados da rede em grandes bases de dados, como o *ImageNet*, e adaptá-los para novas tarefas. No contexto deste trabalho, a *ResNet50* foi utilizada com *Transfer Learning* para classificar fachadas residenciais

e comerciais. Os pesos da rede foram congelados durante o treinamento, permitindo que o modelo aproveitasse o conhecimento adquirido no ImageNet e se adaptasse ao contexto específico das fachadas urbanas (HE et al., 2016)

2.7.3.6 Aplicações da *ResNet50*

A *ResNet50* pode ser utilizada em diversas aplicações de visão computacional:

- Classificação de Imagens: A *ResNet50* é comumente usada para classificar imagens em grandes conjuntos de dados, como o *ImageNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2017).
- Detecção de Objetos: Modelos baseados na *ResNet50*, como o *Faster R-CNN*, são usados para detectar objetos em imagens (HE et al., 2016)
- Segmentação Semântica: A *ResNet50* é utilizada em redes como o *Mask R-CNN* para segmentar objetos em imagens (HE et al., 2016).
- Aplicações Médicas: A *ResNet50* tem sido aplicada em tarefas como detecção de anomalias em exames de imagem e diagnóstico médico (RAJPURKAR et al., 2017).

2.7.3.7 *ResNet50* no Contexto deste Trabalho

Neste trabalho, a *ResNet50* foi escolhida como a arquitetura base devido ao seu desempenho comprovado em tarefas de classificação de imagens. A utilização de *Transfer Learning* permitiu adaptar a *ResNet50* para o contexto específico de classificação de fachadas residenciais e comerciais, alcançando resultados satisfatórios mesmo com um *dataset* pequeno (HE et al., 2016).

3 MATERIAIS E MÉTODOS

Nesta seção, serão abordados as etapas do processo deste trabalho. metodologia inclui a coleta e preparação do *dataset*, a arquitetura do modelo, as técnicas de pré-processamento e aumento de dados, e as métricas de avaliação utilizadas.

3.1 Base de dados

As imagens utilizadas neste trabalho foram obtidas por meio do *Google Street View*, com o intuito apenas para fins de estudo e armazenadas em uma pasta do *Google Drive*. A base de dados está divida em imagens de treino (*train*), validação (*val*) e teste (*test*), como podemos observar na figura 9, e ao todo possui 100 imagens e 2 classes, uma classe de fachadas comerciais e outra de fachadas residenciais. Os dados estão divididos da seguinte forma:

- Treino: 70% das imagens (70 imagens).
- Validação: 15% das imagens (15 imagens).
- Teste: 15% das imagens (15 imagens).

Figura 6 – Estrutura da base de dados.

Fonte: Elaborado pelo autor, 2025.

As imagens foram carregadas utilizando uma função do Tensorflow (2025), como podemos observar na figura 7.

Figura 7 – Exemplo do carregamento das imagens.

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
  '/content/drive/MyDrive/TCC/Data/train',
  shuffle=True,
  seed=123,
  image_size=IMG_SIZE,
  batch_size=batch_size)


validation_dataset = tf.keras.utils.image_dataset_from_directory(
  '/content/drive/MyDrive/TCC/Data/val',
  shuffle=True,
  seed=123,
  image_size=IMG_SIZE,
  batch_size=batch_size)

test_dataset = tf.keras.utils.image_dataset_from_directory(
  '/content/drive/MyDrive/TCC/Data/test',
  shuffle=True,
  seed=1234,
  image_size=IMG_SIZE,
  batch_size=batch_size)
```

Fonte: Elaborado pelo autor, 2025.

3.2 Pré-processamento dos dados

Antes de alimentar as imagens no modelo, realizou-se um processo de pré-processamento para garantir que todas as imagens estivessem no formato adequado ao modelo pré-treinado. As etapas de pré-processamento incluíram:

- Redimensionamento: Todas as imagens foram redimensionadas para o formato $224x224$ *pixels*, que é o tamanho de entrada esperado pela arquitetura *ResNet50*.
- Normalização: As imagens foram normalizadas para que os valores dos *pixels* estivessem na faixa de 0 a 1, dividindo cada pixel por 255.
- *Data Augmentation*: Para aumentar a diversidade do *dataset* e evitar *overfitting*, foram aplicadas técnicas de aumento de dados, como rotação e inversão horizontal. A Figura 2 mostra exemplos de imagens após as transformações de aumento de dados.

Na figura 8 é possível observar como ocorreu o pré processamento.

Figura 8 – Etapas do pré-processamento.

```
# camada para aplicar rotacao e inversao horizontal/vertical
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomTranslation(0.2, 0.2),
])

inputs = tf.keras.Input(shape=(224, 224, 3))
x = data_augmentation(inputs)  # Aplicação de Data Augmentation
x = tf.keras.applications.resnet50.preprocess_input(x)  # Pré-processamento para ResNet
```

Fonte: Elaborado pelo autor, 2025.

3.3 Arquitetura do modelo

Para a construção do modelo utilizou-se da *ResNet50* como base, uma rede neural convolucional profunda que utiliza conexões residuais para facilitar o treinamento de redes com muitas camadas. A *ResNet50* foi escolhida devido ao seu desempenho comprovado em tarefas de classificação de imagens e à disponibilidade de pesos pré-treinados no *ImageNet*.

A Tabela 1 resume a arquitetura do modelo, mostrando o tipo de cada camada, o tamanho da saída e o número de parâmetros treináveis.

Tabela 1 – Resumo técnico da arquitetura do modelo baseado na *ResNet50*

Camada	**Tipo**	**Tamanho de Saída**	**Parâmetros Treináveis**
`InputLayer`	Entrada de Imagem	(224, 224, 3)	0
`Data Augmentation`	Pré-processamento	(224, 224, 3)	0
`ResNet50`	Convolucional	(7, 7, 2048)	23.587.712 (congelados)
`GlobalAveragePooling2D`	Pooling Global	(2048,)	0
`Dense`	Totalmente Conectada	(128,)	262.272
`Dropout`	Regularização (20%)	(128,)	0
`Dense (Saída)`	Camada de Classificação	(1,)	129

Fonte: Elaborado pelo autor, 2025.

A arquitetura do modelo foi adaptada para a tarefa de classificação binária (residencial versus comercial) da seguinte forma:

3.3.1 *InputLayer*

Essa camada é responsável por receber as imagens, para então iniciar o processo de normalização e o pré-processamento antes de passar para a camada do modelo base. As imagens esperadas na camada de entrada precisam estar formato (224,224,3), ou seja, com 224 x 224 *pixels* e com 3 canais (*RGB*) de cor.

3.3.2 *Data Augmentation*

Esta camada é opcional, tornando-se mais útil para os casos onde possuímos um *dataset* pequeno. Neste trabalho, realizamos dois tipos de transformações nas imagens,

uma de rotação e outra de inversão horizontal, para expandir o *dataset* de treino e evitar o *overffiting*, aumentando assim a robustez do modelo.

Figura 9 – Exemplo de uma imagem após as transformações.

Fonte: Elaborado pelo autor, 2025.

3.3.3 Camada de pré-processamento

A camada de pré-processamento desempenha um papel fundamental na preparação das imagens para a entrada na rede *ResNet50*. Essa etapa é crucial para garantir que o modelo receba dados consistentes e dentro de uma faixa de valores adequada, o que pode acelerar o treinamento e melhorar o desempenho final (GOODFELLOW, 2016)

Essa camada tem como objetivo principal normalizar os valores dos *pixels* das imagens, garantindo que os dados de entrada estejam em uma escala adequada para o treinamento da *ResNet50* (KRIZHEVSKY; SUTSKEVER; HINTON, 2017). Para deixar as imagens no padrão esperado pela camada de entrada do modelo, utilizou-se da função

própria disponibilizada pelo *Tensorflow* que pré-processa um lote de imagens, ou seja, teremos um tensor ou uma matriz *numpy* pré-processada que irá codificar um lote de imagens.

Para isso, utilizamos a função *preprocess_input* da *ResNet50*, fornecida pelo *TensorFlow Keras* já apresentado na figura 8, que realiza uma normalização mais complexa, otimizada para a arquitetura *ResNet50* (TENSORFLOW, 2024). A função realiza as seguintes operações:

- Conversão para *Float32*: Converte os valores dos *pixels* para o tipo de dados *float32*, que é o tipo de dados padrão utilizado pelas camadas da *ResNet50*.
- Normalização: Normaliza os valores dos *pixels*, isso garante que as imagens de entrada tenham uma distribuição de valores semelhante àquelas que a rede foi pré-treinada, o que pode melhorar o desempenho e a velocidade de convergência.

A utilização da função própria do TensorFlow (2024) é uma prática recomendada ao trabalhar com a *ResNet50*, pois garante que as imagens de entrada estejam normalizadas de forma condizente com o treinamento original da rede. Isso ajuda a melhorar significativamente o desempenho, a velocidade de convergência e a eficácia do *Transfer Learning*.

3.3.4 Camada base da ResNet50

Com esta camada é possível extrair as características das imagens, mantendo os pesos da rede congelados e aproveitando a rede pré-treinada, esse processo é conhecido como *Transfer Learning*. A tabela 2 demonstra como está estruturada a arquitetura do modelo base *ResNet50*.

Tabela 2 – Arquitetura ResNet-50 para Classificação Binária

Camada	**Saída**	**Parâmetros**
InputLayer	(224, 224, 3)	0
Sequential	(224, 224, 3)	0
GetItem (R, G, B)	(224, 224)	0
Stack	(224, 224, 3)	0
Add	(224, 224, 3)	0
ResNet-50 (Backbone)	(7, 7, 2048)	23,587,712
GlobalAveragePooling2D	(2048)	0
Dropout	(2048)	0
Dense (Saída)	(1)	2,049
Total de Parâmetros		**23,589,761**
Parâmetros Treináveis		**2,049**
Parâmetros Não-Treináveis		**23,587,712**

Fonte: Elaborado pelo autor, 2025.

3.3.5 Camada *Global Average Pooling*

Esta camada de *pooling* é capaz de reduzir a dimensionalidade do tensor de características antes de repassar os dados para a camada *Dense*. Esse processo é essencial para manter a os dados compactos e com suas respectivas características.

É comum reduzir a dimensão espacial dos mapas ao longo das camadas da rede. Essa redução em tamanho é chamada de *pooling* sendo a operação de máximo *maxpooling* comumente empregada. Essa operação tem dois propósitos: em primeiro lugar, o custo computacional, pois como a profundidade dos tensores, d, costuma aumentar ao longo das camadas, é conveniente reduzir a dimensão espacial dos mesmos. Em segundo, reduzindo o tamanho das imagens obtemos um tipo de composição de banco de filtros multiresolução que processa imagens em diferentes espaços-escala (PONTI; COSTA, 2018).

3.3.6 Camada *Dense Layer*

É conhecida como camada totalmente conectada, com 128 neurônios de ativação. Nesta camada, cada neurônio recebe como entrada todos os neurônios da camada anterior, permitindo que a rede aprenda os padrões dos dados utilizados no treinamento.

É um componente fundamental das redes neurais artificiais, incluindo CNNs (GOODFELLOW, 2016). Nessa camada, cada neurônio recebe conexões de todos os neurônios da camada anterior, permitindo que a rede aprenda relações complexas entre as características extraídas pelas camadas convolucionais (BISHOP; NASRABADI, 2006).

3.3.7 Camada *Dropout*

É uma técnica para minimizar *overfitting* que, na fase de treinamento e durante o passo de propagação dos dados pela rede, aleatoriamente desativa com probabilidade p a ativação de neurônios, em particular neurônios de camadas totalmente conectadas (PONTI; COSTA, 2018).

Além de diminuir o *overfitting* é capaz também de melhorar a capacidade de generalização do modelo, uma vez que evita que a rede memorize os dados de treinamento.

3.3.8 Camada de saída (*Dense Layer* para classificar)

É a camada responsável por retornar a classificação, ou seja, após a computação interna, mapeia os dados de saída da camada anterior, submetendo-os para a computação da camada de saída para produzir uma resposta de 0 ou 1, representando as classes residencial e comercial, respectivamente.

3.4 Método proposto

3.4.1 Treinamento do Modelo

O modelo foi treinado utilizando o algoritmo de otimização Adam, com uma taxa de aprendizado de 0,001. A função de perda utilizada foi a *Binary Crossentropy*, que é adequada para problemas de classificação binária. O treinamento foi realizado por 30 épocas, com um *batch size* de 10. A figura 10 demonstra como foi realizado o processo.

Figura 10 – Treinamento do modelo.

```
# compila o modelo, definindo o otimizador, a função de perda e as métricas.
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=[tf.keras.metrics.BinaryAccuracy(name='accuracy'),
                       tf.keras.metrics.AUC(name='auc')])
```

Fonte: Elaborado pelo autor, 2025.

Para evitar *overfitting*, foram utilizadas técnicas de regularização, como o *Dropout* e o *Data Augmentation*, que está presente na figura 8. Além disso, o treinamento foi monitorado utilizando o conjunto de validação, e o modelo com o melhor desempenho foi selecionado para avaliação no conjunto de teste.

3.5 Métricas de Avaliação do Desempenho

Para avaliar o modelo treinado, foram usados as seguintes métricas de desempenho: acurácia, curva $ROC(AUC)$, precisão, recall e matriz de confusão. Iremos utilizar as seguintes definições:

- Verdadeiro Positivo (VP) são os casos que o modelo previu corretamente a classe positiva, no caso as fachadas comerciais.
- Verdadeiro Negativo (VN) são os casos que o modelo previu corretamente a classe negativa, no caso as fachadas residenciais.
- Positivo (FP) são os casos que o modelo previu a classe positiva (comercial), mas a classe real era a negativa (residencial).
- Falso Negativo (FN) são os casos que o modelo previu a classe negativa (residencial), mas a classe real era a positiva (comercial).

3.5.1 Acurácia

A acurácia nada mais é do que a proporção de previsões corretas em relação ao total de previsões. Segundo Costa (2018), mesmo a acurácia sendo a métrica mais utilizada para avaliar um modelo em aprendizado de máquina, ela sozinha não revela corretamente o desempenho geral do modelo. A acurácia possui a seguinte equação:

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{3.1}$$

3.5.2 Curva $ROC(AUC)$

É um gráfico que demonstra a capacidade do modelo em diferenciar entre classes positivas e negativas. A curva é construída plotando a taxa de verdadeiros positivos (TVP) contra a (TFP)taxa de falsos positivos para diferentes valores de *treshold*, onde o *treshold* (ou limiar) é o valor que determina quando uma previsão será classificada como positiva ou negativa. Possui a seguinte equação:

$$TVP = \frac{VP}{VP + FN} \tag{3.2}$$

$$TFP = \frac{FP}{FP + VN} \tag{3.3}$$

O desempenho do classificador medido pela curva ROC (AUC) varia entre 0 e 1. Quanto mais próximo de 1, significa que o modelo consegue diferenciar melhor as classes.

3.5.3 *Recall*

Mede a capacidade do modelo de identificar todos os exemplos positivos corretamente, sem erros. Quanto maior o *recall*, melhor será o modelo em capturar todos os casos positivos, mas pode acabar classificando erroneamente alguns negativos como positivos.

$$Recall = \frac{VP}{VP + FN} \tag{3.4}$$

3.5.4 Precisão

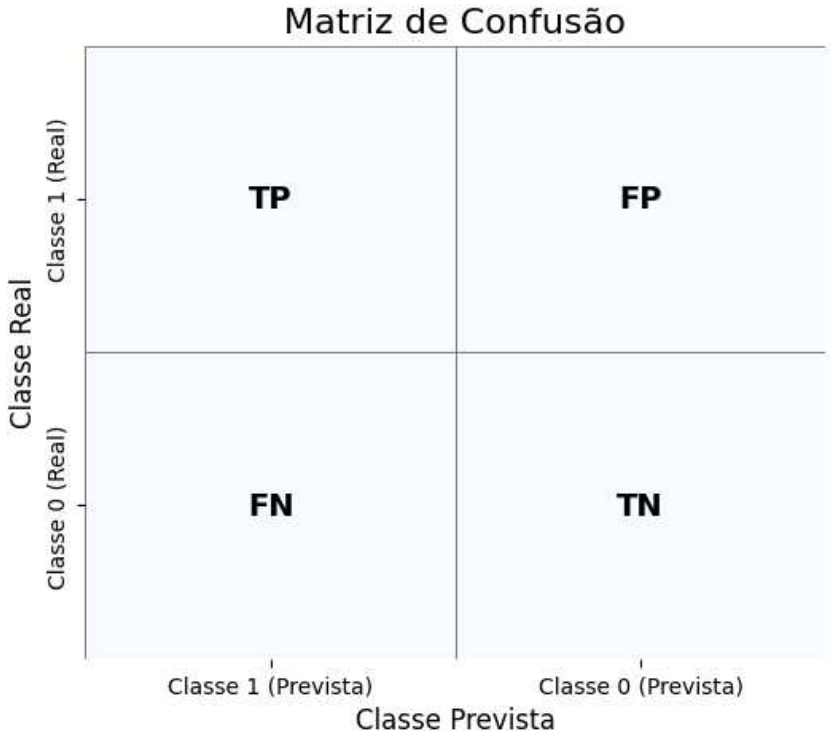
Indica a proporção de previsões positivas que foram corretas.Uma alta precisão indica que o modelo fez poucas classificações erradas como positivas. A precisão é importante quando os falsos positivos são críticos, por exemplo em situações onde uma classificação incorreta pode gerar custos ou problemas adicionais.

$$Precisão = \frac{VP}{VP + FP} \tag{3.5}$$

3.5.5 *Matriz de Confusão*

A matriz de confusão indica a quantidade de ocorrências que o algoritmo teve em cada uma das quatro categorias (VP, VN, FP e FN). Na figura 11 temos a representação dessa matriz.

Figura 11 – Matriz de Confusão.



Fonte: Elaborado pelo autor, 2025.

3.6 Ambiente de teste

Para realização do trabalho foi utilizada a linguagem de programação Python, junto com as bibliotecas como *TensorFlow* e *Keras*. Todo o código foi executado dentro

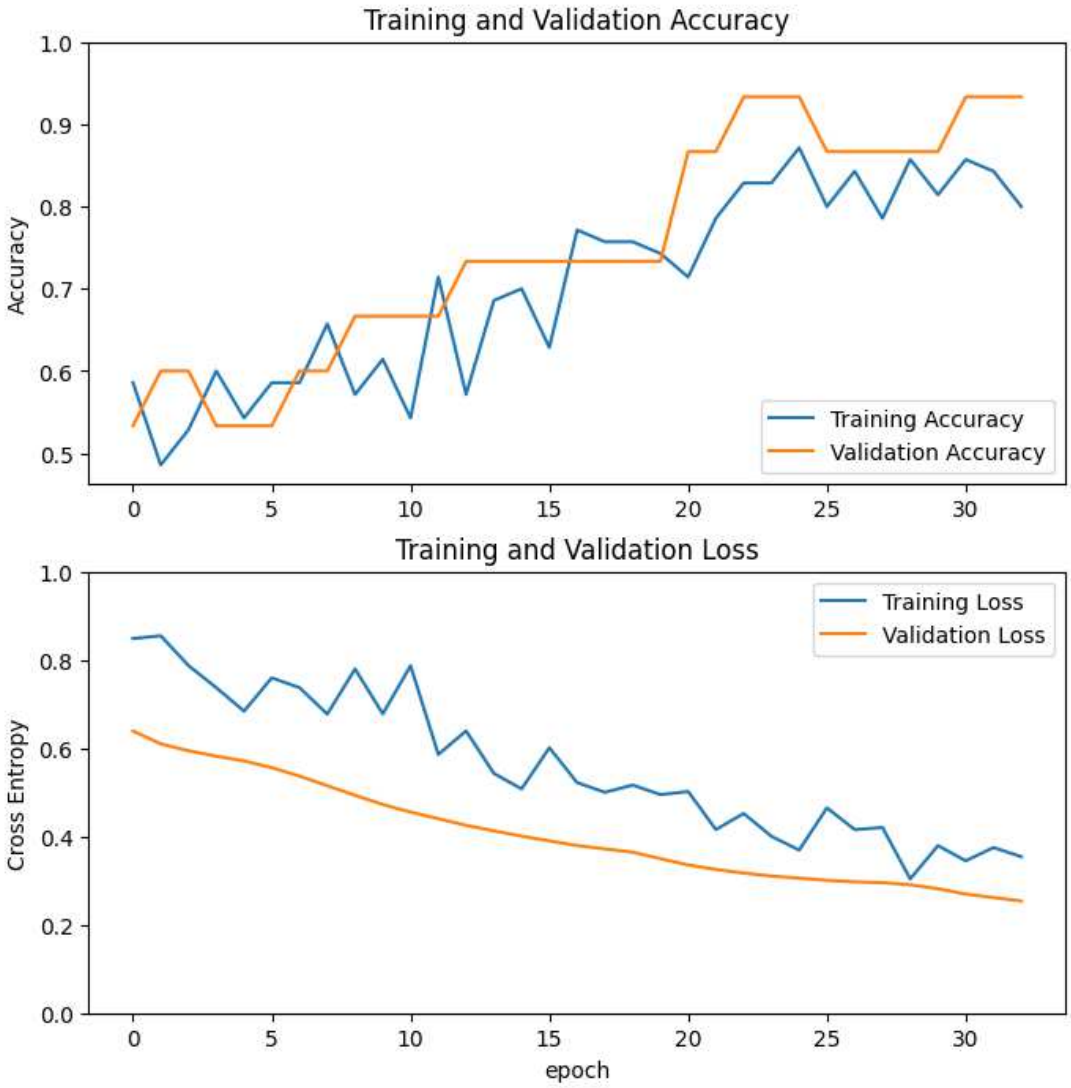
do *Google Colab*, que nada mais é que um serviço do *Jupyter Notebook* hospedado que permite executar código *Python* diretamente do navegador, sendo uma boa opção para quem trabalha com *Data Science* e *Machine Learning*, uma vez que não precisa fazer uma pré-configuração e oferece acesso gratuito a recursos computacionais, como *CPU*, *GPU* e *TPU*. O código pode ser consultado no Github no seguinte link <https://github.com/MarcioSantoos/tcc_resnet>

4 RESULTADOS E DISCUSSÕES

A figura 12 representa os valores da acurácia e da perda (*loss*) durante o treinamento, foram utilizadas ao todo 30 épocas para treinar o modelo, o tamanho do *batch size* foi de 10 imagens, e a figura mostra a evolução da acurácia e *loss* ao longo das épocas de treinamento.

As técnicas de *Data Augmentation* foram muito úteis para melhorar o desempenho do modelo, considerando o tamanho reduzido do *dataset*. Ao aplicar transformações como rotação e inversão horizontal nas imagens, foi possível aumentar a diversidade do conjunto de treinamento e evitar *overfitting*.

Figura 12 – Acurácia e *Loss*



Fonte: Elaborado pelo autor, 2025.

Os resultados mostram que o modelo teve um desempenho geral bom, com o

loss decaindo ao longo do treinamento, ou seja, o modelo realmente estava aprendendo os padrões de forma consistente. Na curva de aprendizado não foram observados sinais de *overfitting*, o que sugere que as técnicas de *Data Augmentation* e *Dropout* foram eficazes em regularizar o modelo. É possível observar na tabela 3 que a acurácia obtida foi de 90%. A precisão de 100% indica que o modelo não cometeu falsos positivos, ou seja, todas as fachadas classificadas como comerciais eram, de fato, comerciais. No entanto, o *recall* de 83% revela que o modelo deixou de identificar 1 fachada comercial (falso negativo).

Na tabela 3 podemos observar as métricas de desempenho do modelo de forma organizada, e logo após a tabela 3 os resultados são detalhados.

Tabela 3 – Métricas de desempenho do modelo

Métrica	**Valor**
Acurácia	0.90
Precisão	1.00
Recall	0.83
Curva ROC(AUC)	0.90

- Acurácia: O modelo alcançou uma acurácia de 90%, o que indica que ele classificou corretamente 90% das imagens do conjunto de teste. Esse valor é considerado satisfatório, especialmente considerando o tamanho reduzido do *dataset*.
- Precisão: A precisão de 100% indica que o modelo não cometeu falsos positivos, ou seja, todas as fachadas classificadas como comerciais eram realmente comerciais. Isso demonstra que o modelo é altamente confiável ao identificar fachadas comerciais.
- *Recall*: O *recall* de 83% revela que o modelo deixou de identificar uma fachada comercial, classificando-a como residencial (falso negativo). Isso sugere que, embora o modelo seja preciso, ele pode não capturar todos os casos positivos.
- Curva *ROC(AUC)*: A curva *ROC(AUC)* de 90% indica que o modelo tem uma boa capacidade de distinguir entre fachadas residenciais e comerciais. Quanto mais próxima de 1, melhor é a capacidade do modelo de diferenciar as classes.

A matriz de confusão confirma que o modelo não cometeu falsos positivos, mas apresentou um falso negativo, ou seja, uma classe comercial como residencial.

Tabela 4 – Matriz de Confusão do Modelo

	Falso Negativo	**Falso Positivo**
Verdadeiro Negativo	5	0
Verdadeiro Positivo	1	4

- Verdadeiros Negativos (TN): O modelo classificou corretamente 5 fachadas residenciais.
- Verdadeiros Positivos (TP): O modelo classificou corretamente 4 fachadas comerciais.
- Falsos Negativos (FN): O modelo classificou incorretamente 1 fachada comercial como residencial.
- Falsos Positivos (FP): O modelo não cometeu falsos positivos.

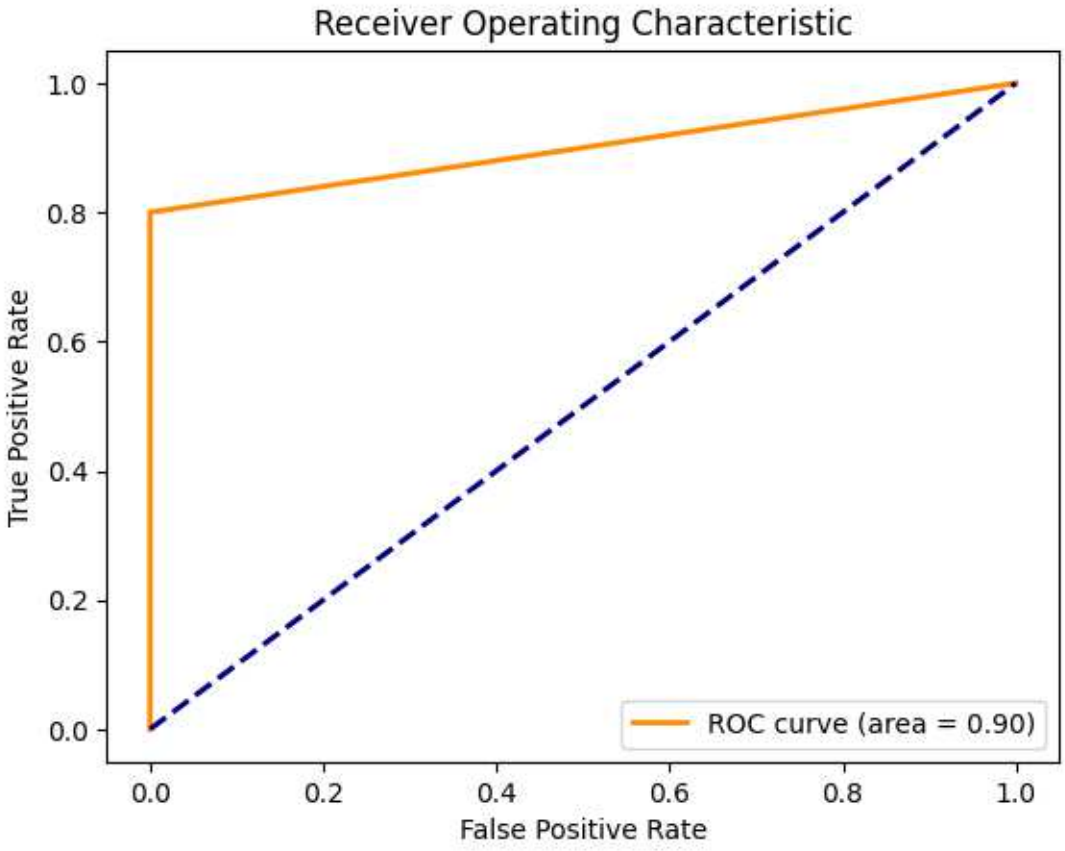
A matriz de confusão apresenta 5 verdadeiros negativos (TN), 4 verdadeiros positivos (TP), 1 falso negativo (FN) e nenhum falso positivo (FP). Isso confirma que o modelo tem uma alta confiabilidade na predição de casos positivos, mas ocasionalmente pode não detectar todos os positivos existentes. Isso sugere que o modelo é confiável ao classificar fachadas como comerciais, mas pode melhorar na identificação de todas as fachadas comerciais presentes no *dataset*, podemos observar esse comportamento na figura 13.

Figura 13 – Exemplo de classificação do modelo.

Fonte: Elaborado pelo autor, 2025.

O resultado da curva *ROC(AUC)* também demonstra que o modelo conseguiu aprender bem as características das imagens, uma vez que consegue discernir bem cada classe, atingindo um valor de 90%, ou seja, em 90% dos casos o modelo classifica corretamente a imagem. A figura 14 demonstra como ficou a curva.

Figura 14 – Curva *ROC(AUC)*



Fonte: Elaborado pelo autor, 2025.

Um dos maiores desafios enfrentados neste trabalho foi encontrar um modelo pré-treinado que alcançasse bons valores nas métricas apresentadas com um *dataset* pequeno, utilizando a técnica de *transfer learning*. O melhor modelo que atendeu este trabalho foi o *ResNet50*. A construção do *dataset* também foi um dos desafios na construção do modelo proposto, que inicialmente se pretendia fazer a obtenção das imagens com uma câmera de forma manual. Embora exista na literatura trabalhos semelhantes a este, não foi possível localizar outro trabalho que também pretendia classificar apenas fachadas de casas residenciais e comerciais.

E embora os resultados sejam promissores, é importante destacar as limitações do modelo:

- Tamanho do *Dataset*: O *dataset* utilizado é pequeno (100 imagens), o que pode limitar a generalização do modelo para outros contextos urbanos.
- Diversidade de Imagens: O *dataset* pode não representar a diversidade de fachadas encontradas em diferentes regiões ou condições de iluminação.
- Falsos Negativos: O modelo apresentou dificuldades em identificar algumas

fachadas comerciais, o que pode estar relacionado à falta de exemplos específicos no conjunto de treinamento.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentou-se um modelo pré-treinado para identificar fachadas de casas comerciais e residenciais, com a rede neural convolucional *ResNet50*, utilizando técnicas de *Transfer Learning* e *Data Augmentation*, para a classificação de fachadas residenciais e comerciais. Os resultados demonstraram que o modelo atingiu bons níveis de acurácia, alcançando uma acurácia de 90%, e de curva *ROC(AUC)*, uma curva de 90%. Por mais que a base de dados seja pequena, a rede conseguiu classificar bem as imagens, a matriz de confusão mostrou que o modelo não cometeu falsos positivos, mas apresentou um falso negativo, o que sugere que ele é confiável na classificação de fachadas comerciais, mas pode melhorar na identificação de todas as fachadas comerciais presentes no *dataset*.

É importante destacar que o trabalho foi desenvolvido com um *dataset* pequeno (100 imagens), o que pode limitar a generalização do modelo para outros contextos, ou seja, o modelo pode ter um desempenho inferior onde a arquitetura das residências for diferente. Além disso, o uso de técnicas como *Transfer Learning* e *Data Augmentation* foi essencial para melhorar o desempenho do modelo, especialmente considerando o tamanho reduzido do *dataset*. Os resultados obtidos sugerem que a combinação de modelos pré-treinados com um *dataset* regionalizado pode ser eficaz para a classificação de fachadas urbanas. O uso da *ResNet50* permitiu aproveitar o conhecimento adquirido em grandes bases de dados, como o *ImageNet*, e adaptá-lo para o contexto específico das fachadas residenciais e comerciais. A técnica de *Data Augmentation* também se mostrou crucial para evitar *overfitting* e aumentar a robustez do modelo.

No entanto, é importante reconhecer as limitações do modelo. O *dataset* utilizado, mesmo que tenha sido suficiente para os objetivos deste trabalho, é pequeno e pode não representar a diversidade de fachadas encontradas em diferentes regiões ou contextos urbanos. Além disso, o modelo apresentou dificuldades em identificar algumas fachadas comerciais, o que pode estar relacionado à falta de mais dados, ou seja, de exemplos específicos no conjunto de treinamento.

Para trabalhos futuros, propõe-se utilizar uma base de imagens maior. Assim, também será possível fazer mais testes com outros modelos pré-treinados que trabalham melhor com uma base de dados maior, e que possuam mais camadas profundas. Para aprimorar o modelo proposto e expandir suas aplicações, sugerem-se as seguintes direções para trabalhos futuros:

- Expansão do *Dataset*: Coletar um *dataset* maior e mais diversificado, incluindo fachadas de diferentes regiões, estilos arquitetônicos e condições de iluminação. Isso ajudaria a melhorar a generalização do modelo e sua capacidade de lidar com diferentes cenários urbanos.

- Estimar valores de imóveis: Para trabalhos futuros pode-se realizar a integração do modelo de classificação de fachadas com sistemas de estimativa de valor imobiliário. A aparência externa de um imóvel, ou seja, sua fachada, é um dos fatores que influenciam diretamente a percepção de valor do imóvel. Portanto, o modelo desenvolvido neste trabalho pode ser expandido para incluir a análise de características visuais das fachadas e estimar o valor de um imóvel.

- Uso de Outras Arquiteturas: Explorar outras arquiteturas de redes neurais, como *EfficientNet*, *MobileNet* ou *Vision Transformers* (ViT), que podem oferecer melhor desempenho ou eficiência computacional em comparação com a *ResNet50*.

- Integração com Técnicas de Detecção de Objetos: Integrar o modelo com técnicas de detecção de objetos para identificar elementos específicos das fachadas, como janelas, portas e placas comerciais. Isso poderia melhorar a precisão da classificação e fornecer informações adicionais sobre as características das fachadas.

- Aplicações em Tempo Real: Implementar o modelo em sistemas de tempo real, como câmeras de vigilância ou drones, para monitoramento urbano. Isso exigiria otimizações para garantir que o modelo possa processar imagens rapidamente e com baixo consumo de recursos computacionais.

- Análise de Estilos Arquitetônicos: Expandir o escopo do modelo para incluir a classificação de estilos arquitetônicos, além de fachadas residenciais e comerciais. Isso poderia ser útil para estudos de preservação histórica e análise de tendências arquitetônicas.

5.1 Impacto Prático do Trabalho

A classificação automática de fachadas tem aplicações práticas em diversas áreas, como segurança pública, análise imobiliária e planejamento urbano. Por exemplo, o modelo desenvolvido pode ser integrado a sistemas de vigilância urbana para identificar automaticamente estabelecimentos comerciais em tempo real, auxiliando na gestão municipal e no monitoramento de áreas comerciais. Além disso, no setor imobiliário, a classificação de fachadas pode ser usada para análises de mercado, identificação de tendências arquitetônicas e avaliação de imóveis.

Outra aplicação potencial, que foi uma das motivações para a realização deste trabalho, é no campo da investigação pública. Em muitos casos, investigações envolvem o cruzamento de dados de diversas fontes, e entre esses dados, há informações sobre endereços que são alvo de interesse da investigação. O modelo desenvolvido pode ser

utilizado para classificar automaticamente a fachada de um imóvel a partir de fotos, verificando se as características visuais do imóvel correspondem às descrições presentes nos dados investigativos. Isso pode auxiliar na validação de informações e na identificação de inconsistências, agilizando processos de investigação.

Além disso, no futuro, o modelo poderá ser expandido para estimar o valor de imóveis com base nas características das fachadas. Ao integrar o sistema com dados de mercado imobiliário, seria possível correlacionar aspectos visuais das fachadas, como estilo arquitetônico, estado de conservação e presença de elementos comerciais, com o valor de mercado dos imóveis. Essa funcionalidade poderia ser útil tanto para órgãos públicos, no planejamento de políticas urbanas, quanto para o setor privado, na avaliação de propriedades.

5.2 Considerações Éticas e Sociais

A aplicação de modelos de classificação de fachadas em ambientes urbanos levanta questões éticas e sociais que devem ser consideradas. Por exemplo:

- Vigilância Excessiva: A classificação automática de fachadas pode ser usada em sistemas de vigilância urbana, o que pode gerar preocupações sobre o monitoramento excessivo de espaços públicos.
- Impacto Social: A classificação de fachadas pode ter impactos sociais, como a identificação de áreas comerciais em detrimento de áreas residenciais, o que pode influenciar políticas públicas e investimentos.

5.3 Conclusão

Este trabalho demonstrou que a combinação de modelos pré-treinados com técnicas de *Transfer Learning* e *Data Augmentation* pode ser eficaz para a classificação de fachadas residenciais e comerciais. Os resultados obtidos são promissores e indicam que o modelo pode ser aplicado em sistemas automatizados para identificação de fachadas urbanas, contribuindo para áreas como segurança e investigação pública.

No entanto, é importante reconhecer as limitações do trabalho, especialmente em relação ao tamanho e diversidade do *dataset* utilizado. Para trabalhos futuros, sugere-se a expansão do *dataset*, a exploração de outras arquiteturas de redes neurais e a integração com técnicas de detecção de objetos. Além disso, é fundamental considerar as implicações éticas e sociais da aplicação dessa tecnologia, garantindo que ela seja usada de forma responsável e em benefício da sociedade.

REFERÊNCIAS

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. *2017 international conference on engineering and technology (ICET)*. [S.l.], 2017. p. 1–6.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, v. 13, n. 2, 2012.

BISHOP, C. M.; NASRABADI, N. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. v. 4.

COSTA, A. H. M. da. Aplicação de redes neurais convolucionais na identificação de tatuadores. *Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade de Brasília*, 2018.

Cérebro Mente. *Redes Neurais Artificiais: O que são e como funcionam*. Acesso em 2025. Acessado em: 20 fev. 2025. Disponível em: <https://cerebromente.org.br/n05/tecnologia/rna_i.htm>.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. [S.l.], 2010. p. 249–256.

GOODFELLOW, I. *Deep learning*. [S.l.]: MIT press, 2016.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014.

GOOGLE. *Google Street View*. 2023. Acessado em: 04 jan. 2025. Disponível em: <https://www.google.com/maps>.

GORLAPRAVEEN123. *File:ResNet50.png*. 2021. Accessed: February 18, 2025. Disponível em: <https://commons.wikimedia.org/wiki/File:ResNet50.png>.

GU, J. et al. Recent advances in convolutional neural networks. *Pattern recognition*, Elsevier, v. 77, p. 354–377, 2018.

HAYKIN, S. *Neural networks and learning machines, 3/E*. [S.l.]: Pearson Education India, 2009.

HE, K. et al. 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.

HINTON, G. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

HOCHREITER, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

KANG, J. et al. Building instance classification using street view images. *ISPRS journal of photogrammetry and remote sensing*, Elsevier, v. 145, p. 44–59, 2018.

KINGMA, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

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

LEMLEY, J.; BAZRAFKAN, S.; CORCORAN, P. Smart augmentation learning an optimal data augmentation strategy. *Ieee Access*, IEEE, v. 5, p. 5858–5869, 2017.

LIU, H. et al. Deepfacade: A deep learning approach to facade parsing. In: IJCAI. [S.l.], 2017.

MAAS, A. L. et al. Rectifier nonlinearities improve neural network acoustic models. In: ATLANTA, GA. *Proc. icml*. [S.l.], 2013. v. 30, n. 1, p. 3.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814.

NG, A. Y. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In: *Proceedings of the twenty-first international conference on Machine learning*. [S.l.: s.n.], 2004. p. 78.

NIELSEN, A. *Neural networks and deep learning*. [S.l.]: Determination Press, 2015.

PIRES, P. B.; SANTOS, J. D.; PEREIRA, I. V. Artificial neural networks: history and state of the art. *Encyclopedia of Information Science and Technology, Sixth Edition*, IGI Global, p. 1–25, 2025.

PONTI, M. A.; COSTA, G. B. P. D. Como funciona o deep learning. *arXiv preprint arXiv:1806.07908*, 2018.

RAJPURKAR, P. et al. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017.

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.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group UK London, v. 323, n. 6088, p. 533–536, 1986.

SHAKIBHAMEDAN, S. et al. Ace-cnn: Approximate carry disregard multipliers for energy-efficient cnn-based image classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, IEEE, 2024.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, Springer, v. 6, n. 1, p. 1–48, 2019.

SIMONYAN, K. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

TENSORFLOW. *tf.keras.applications.resnet50.preprocess_input*. 2024. Disponível em: <https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/preprocess_input>.

TENSORFLOW. *Transfer learning fine-tuning*. 2025. Disponível em: <https://www.tensorflow.org/tutorials/images/transfer_learning?hl=pt-br#train_the_model>. Acesso em: 30 jan. 2025.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4.

WANG, B. et al. Architectural style classification based on cnn and channel–spatial attention. *Signal, Image and Video Processing*, Springer, v. 17, n. 1, p. 99–107, 2023.

YOSHIMURA, Y. et al. Deep learning architect: classification for architectural design through the eye of artificial intelligence. *Computational Urban Planning and Management for Smart Cities 16*, Springer, p. 249–265, 2019.