Classificação de Estilo Arquitetônico através de Rede Neural Convolucional Potencializada por Unidades de Processamento Neural para Dispositivos Móveis

Gustavo Henrique Maia Dias

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa - PB
2024

Gustavo Henrique Maia Dias

Classificação de Estilo Arquitetônico através de Rede Neural Convolucional Potencializada por Unidades de Processamento Neural para Dispositivos Móveis

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Thaís Gaudencio do Rêgo

Março de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

D541c Dias, Gustavo Henrique Maia.
Classificação de estilo arquitetônico através de rede neural convolucional potencializada por unidades de processamento neural para dispositivos móveis / Gustavo Henrique Maia Dias. - João Pessoa, 2024.
58 f. : il.

Orientação: Thaís Gaudencio do Rêgo.
TCC (Graduação) - UFPB/CI.

1. Redes neurais convolucional. 2. Estilo arquitetônico. 3. IA no dispositivo. I. Rêgo, Thaís Gaudencio do. II. Título.

UFPB/CI CDU 004.8

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computaçãointitulado ***Classificação de Estilo Arquitetônico através de Rede Neural Convolucional Potencializada por Unidades de Processamento Neural para Dispositivos Móveis*** de autoria de Gustavo Henrique Maia Dias, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Thaís Gaudencio do Rêgo
Universidade Federal da Paraíba

Prof. Dr. Leonardo Vidal Batista
Universidade Federal da Paraíba

Prof. Dr. Yuri de Almeida Malheiros Barbosa
Universidade Federal da Paraíba

João Pessoa, 13 de março de 2024

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

RESUMO

A identificação e classificação de estilos arquitetônicos são problemas significativos nas ciências, especialmente na arqueologia, história da arte e conservação. Essa tarefa é essencial para a preservação do patrimônio cultural, mas é frequentemente desafiadora devido à complexidade e à variação nos estilos arquitetônicos ao longo do tempo e regiões. Este estudo aborda esse problema aplicando o modelo EfficientNetV2B0, para identificar os 5 estilos arquitetônicos mais presentes no Brasil. Sendo esses o Barroco, Neoclássico, Neogótico, Eclético e Modernista; para isso, foi criado um dataset com 927 imagens para treinamento, 231 para validação e 292 para teste. A acurácia alcançada foi de 83%; com a categoria Modernista alcançando uma precisão de 95%. No entanto, o estilo Eclético apresentou um desempenho inferior, com uma precisão de apenas 64%. O desempenho do modelo foi testado em diferentes dispositivos móveis, incluindo os SoCs A10 Fusion, A14 Bionic e M2; revelando variações no tempo de execução entre os dispositivos. Esses resultados destacam o potencial do aprendizado de máquina na classificação de estilos arquitetônicos e sua viabilidade em dispositivos de diferentes capacidades de hardware, contribuindo de maneira significativa para a preservação e estudo do patrimônio arquitetônico brasileiro.

Palavras-chave: Redes Neurais Convolucionais, Estilo Arquitetônico, IA no dispositivo

ABSTRACT

The identification and classification of architectural styles are significant problems in sciences, especially in archaeology, art history, and conservation. This task is essential for the preservation of cultural heritage but is often challenging due to the complexity and variation in architectural styles over time and regions. This study addresses this issue by applying the EfficientNetV2B0 model to identify the 5 most present architectural styles in Brazil. These are Baroque, Neoclassical, Neo-Gothic, Eclectic, and Modernist; for this, a dataset was created with 927 images for training, 231 for validation, and 292 for testing. The accuracy achieved was 83%; with the Modernist category reaching a precision of 95%. However, the Eclectic style showed inferior performance, with a precision of only 64%. The model's performance was tested on different mobile devices, including the A10 Fusion, A14 Bionic, and M2 SoCs; revealing variations in execution time between devices. These results highlight the potential of machine learning in the classification of architectural styles and its feasibility on devices with different hardware capabilities, contributing significantly to the preservation and study of Brazilian architectural heritage.

Key-words: Convolutional Neural Networks, Architecture style, On-device AI

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CNN - Convolutional Neural Network (Rede Neural Convolucional)

DCNN - Deep Convolutional Neural Network (Rede Neural Convolucional Profunda)

DPM - Deformable Parts Model (Modelo de Partes Deformáveis)

IEP - Improved Set Projection (Projeção de Conjunto Melhorada)

LIME - Local Interpretable Model-agnostic Explanations (Explicações Locais Interpretáveis Independentes de Modelo)

MLLR - Multinomial Latent Logistic Regression (Regressão Logística Latente Multinomial)

NBNN - Naive Bayes Nearest Neighbor (Vizinho Mais Próximo Naive Bayes)

ResNet - Residual Networks (Redes Residuais)

SoC - System on Chip (Sistema em Chip)

Sumário

1 INTRODUÇÃO

A arquitetura, como um espelho tangível da história e cultura de uma sociedade, reflete as transformações políticas, religiosas, tecnológicas e sociais através dos tempos, desempenhando um papel crucial na preservação do patrimônio cultural e fornecendo *insights* valiosos sobre a evolução das civilizações [50]. Neste cenário, a arquitetura passou por uma evolução significativa, integrando a criatividade humana e a funcionalidade de maneira harmoniosa. Tal progresso culminou na criação de uma tapeçaria diversificada e rica, composta por uma variedade de estilos arquitetônicos distintos. Esses estilos são marcados por combinações específicas de características arquitetônicas, tais como design, material, método de construção, e elementos estruturais como janelas, portas e colunas [51], tornando a tarefa de classificá-los em categorias precisas um desafio significativo.

No setor turístico, influenciada pela arquitetura, a compreensão detalhada e acessível dos estilos arquitetônicos confere aos edifícios e monumentos históricos um papel central como atrativos da área [53]. Esta característica contribui para o estímulo da economia local e para a preservação do patrimônio cultural. Diante disso, o presente trabalho tem como objetivo investigar a aplicação da inteligência artificial (IA), em particular na visão computacional, como meios para aprimorar o processo de identificação e análise dos estilos arquitetônicos.

Desenvolvemos um modelo robusto de aprendizado de máquina, acompanhado de um aplicativo prático, visando oferecer uma solução integrada e acessível, que democratiza o acesso ao conhecimento arquitetônico e fomenta a preservação cultural. No entanto, a criação de um sistema eficiente, preciso e capaz de operar diretamente em dispositivos móveis se apresenta como um desafio notável. Este trabalho, portanto, busca superar essas barreiras, proporcionando uma metodologia ágil e eficaz para a análise em tempo real dos estilos arquitetônicos.

A adoção da IA na arquitetura e no planejamento urbano vai além da simples otimização de processos existentes, abrindo portas para novas possibilidades de análise e interpretação. A capacidade de processar rapidamente grandes volumes de dados visuais permite uma compreensão mais aprofundada das tendências arquitetônicas, beneficiando áreas como a história da arte, arquitetura e urbanismo.

Dessa forma, o impacto desta inovação é vasto e multifacetado, estendendo-se desde a preservação cultural, facilitando a identificação de estilos e a proteção de estruturas históricas, até o turismo, onde proporciona uma ferramenta educativa e interativa para visitantes e entusiastas da arquitetura. No campo educacional, estudantes e profissionais ganham acesso facilitado ao conhecimento arquitetônico, enquanto que no planejamento urbano, a habilidade de classificar e analisar rapidamente estilos arquitetônicos contribui para um entendimento mais completo do tecido urbano, auxiliando na tomada de decisões

informadas.

Ao integrar a IA no estudo e análise dos estilos arquitetônicos, este trabalho não só destaca as capacidades técnicas da tecnologia, mas também sublinha seu papel vital como facilitador para a preservação cultural, educação, turismo e planejamento urbano, evidenciando a transformação que ela pode incitar no apreço e entendimento da arquitetura em nossa sociedade.

1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um modelo de classificação de estilos arquitetônicos que seja otimizado para execução em dispositivos móveis.

1.2 Objetivos Específicos

- Desenvolver uma base com os principais estilos arquitetônicos brasileiros, coletando e catalogando imagens de diversas fontes, incluindo arquivos digitais e bibliotecas especializadas em arquitetura brasileira. Este banco de dados será fundamental para treinar e validar o modelo de classificação.
- Conduzir uma análise da arquitetura da CNN escolhida no contexto do aprendizado de máquina, com o objetivo de avaliar sua eficácia na classificação de estilos arquitetônicos. Esta análise focará na precisão do modelo, tempo de processamento e eficiência no uso de recursos computacionais.
- Otimizar o modelo para funcionamento em dispositivos móveis, utilizando a tecnologia CoreML, visando tornar o aplicativo mais acessível e de fácil utilização para o usuário final.
- Garantir a interpretabilidade das previsões do modelo, utilizando métodos como o LIME para explicar as decisões tomadas pelo modelo, proporcionando maior transparência e confiança nos resultados gerados.

1.3 Estrutura da Monografia

A estrutura desta monografia é organizada nos seguintes capítulos, detalhando cada etapa do desenvolvimento, implementação e avaliação do modelo proposto para classificação de estilos arquitetônicos:

1. **CONCEITOS GERAIS E REVISÃO DA LITERATURA:** Este capítulo estabelece a base teórica e conceitual do estudo, iniciando com a complexidade da

classificação de estilos arquitetônicos e suas características distintas. Prossegue com os fundamentos de redes neurais, destacando o modelo Perceptron e a estrutura e funcionamento das CNNs. Aborda o aprendizado por transferência em CNNs, suas vantagens e estratégias de aplicação. Compara as arquiteturas EfficientNet com outras CNNs, enfatizando sua eficiência e aplicabilidade. Discute métricas de avaliação em classificação de imagens como precisão, sensibilidade, medida-F1 e suporte. Finaliza com a importância da interpretabilidade dos modelos de aprendizado de máquina, concentrando-se na técnica LIME, e revisa trabalhos relacionados na área de classificação de estilos arquitetônicos.

2. **METODOLOGIA:** Este capítulo abrange o processo metodológico completo do projeto, iniciando com a seleção e coleta de imagens de distintos estilos arquitetônicos. Em seguida, é apresentada a descrição detalhada dos equipamentos técnicos utilizados para desenvolvimento, treinamento e avaliação do modelo de aprendizado de máquina. A criação do modelo é detalhada, enfatizando o emprego de técnicas de aprendizado transferido e o uso da arquitetura EfficientNetV2B0, incluindo estratégias de treinamento e aprimoramento do modelo. A adaptação do modelo para ser compatível com plataformas móveis, visando uma integração eficaz em diferentes dispositivos, também é discutida. A finalização da metodologia envolve a elaboração de um aplicativo, integrando elementos de interface de usuário, aprendizado de máquina e gestão de dados na nuvem. Esta abordagem holística esclarece cada etapa do processo, desde a preparação inicial dos dados até a implementação prática do modelo em um ambiente aplicativo.

3. **RESULTADOS E DISCUSSÃO:** Esta seção aborda os resultados obtidos pelo estudo, com foco no desempenho do modelo de aprendizado de máquina e a funcionalidade do aplicativo desenvolvido. A avaliação começa com uma análise do modelo, utilizando métricas de desempenho para avaliar sua precisão na classificação de estilos arquitetônicos. Em seguida, a técnica LIME é aplicada para entender melhor como o modelo faz suas previsões. A eficácia do aplicativo é analisada comparando o desempenho de diferentes SoCs, como A10 Fusion, A14 Bionic e M2, em várias versões de sistemas operacionais. Cada aspecto dos resultados fornece uma compreensão sobre a robustez e a aplicabilidade prática do projeto, desde a precisão do modelo na identificação de estilos arquitetônicos até a eficiência do aplicativo em diferentes dispositivos.

4. **CONSIDERAÇÕES FINAIS:** O capítulo final sintetiza os principais resultados da pesquisa, discutindo a eficácia da abordagem adotada para a classificação de estilos arquitetônicos no Brasil usando aprendizado de máquina e visão computacional. Destaca a importância do conjunto de dados na fase inicial e o sucesso do modelo, especialmente em identificar estilos como modernista e neogótico. A análise

do desempenho do modelo revela *insights*, especialmente no uso da técnica LIME para explicar as decisões do modelo. A pesquisa também aborda a avaliação do desempenho do modelo em diferentes dispositivos, enfatizando a acessibilidade e a viabilidade do aplicativo em uma ampla gama de hardware. As implicações práticas do estudo são discutidas, enfatizando sua contribuição para a preservação cultural, educação em arquitetura e turismo. Por fim, o capítulo sugere direções futuras para a pesquisa, como a expansão do conjunto de dados e a exploração de novas arquiteturas de redes neurais para aprimorar a precisão e eficiência do modelo.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

Este capítulo detalha os conceitos teóricos necessários para melhor compreensão de um trabalho que investiga a classificação de estilos arquitetônicos utilizando redes neurais convolucionais (*Convolutional Neural Networks*, CNN). Inicialmente, abordamos a complexidade inerente à classificação de estilos arquitetônicos, considerando os elementos distintos de design e materiais de construção associados a cada estilo. A identificação precisa destes estilos requer um modelo que possa discernir nuances arquitetônicas sutis e complexas, motivando a escolha de redes neurais profundas para esta tarefa.

As redes neurais, com sua capacidade de processar e aprender, a partir de grandes conjuntos de dados, são exploradas em profundidade. Além disso, enfocamos nas estratégias de otimização e regularização, que são essenciais para o treinamento eficaz desses modelos complexos.

Aprofundamos na aplicabilidade das CNNs para a identificação de estilos arquitetônicos, destacando a importância de sua estrutura hierárquica na extração de características visuais. Neste contexto, focamos exclusivamente na série EfficientNet das CNNs, examinando em detalhe suas peculiaridades e aplicações no reconhecimento de padrões arquitetônicos. Esta escolha se deve à versatilidade e eficiência da EfficientNet, que a torna particularmente adequada para analisar e classificar complexidades visuais encontradas em estilos arquitetônicos diversos.

Além disso, este capítulo introduz as métricas de avaliação utilizadas para medir o desempenho dos modelos de classificação de imagens. A precisão, a sensibilidade, a medida-f e o suporte são discutidos em detalhes, elucidando como essas métricas fornecem *insights* valiosos sobre a eficácia e as limitações dos modelos.

Por fim, abordamos a interpretabilidade dos modelos de aprendizado de máquina, com ênfase no uso da técnica de Explicação de Modelo Local Interpretabilidade Agnóstica (*Local Interpretable Model-Agnostic Explanations*, LIME). Essa técnica é crucial para entender como as decisões dos modelos são tomadas, garantindo transparência e confiabilidade nos sistemas de classificação automatizados.

2.1 Estilos Arquitetônicos e Componentes Arquitetônicos

A tarefa de classificar estilos arquitetônicos por meio de modelos de aprendizado de máquina representa um desafio significativo, exigindo uma compreensão detalhada dos vários componentes que formam as estruturas arquitetônicas. Estilos como barroco, eclético, neogótico, neoclássico e modernista possuem características únicas, as quais necessitam ser precisamente identificadas e entendidas para garantir o êxito do modelo. Esta seção explora a integração desses estilos no processo de modelagem e examina como os

componentes arquitetônicos contribuem para definir cada estilo específico.

2.2 Design Arquitetônico

Cada estilo arquitetônico possui um conjunto distinto de princípios e padrões de design, essenciais para a sua identificação correta:

- **Barroco:** Este estilo é caracterizado por suas formas dramáticas e detalhes ornamentais complexos, além do uso expressivo de cores e luz [33]. A Igreja de São Francisco, mostrada na Figura 1, é um exemplo, com sua fachada decorada com entalhes detalhados, volutas e figuras religiosas que criam um senso de movimento e dinamismo. As torres sinuosas e a abundância de decoração em alto relevo são típicas do Barroco [33]. A grandiosidade da estrutura é acentuada pelas balaustradas ornamentadas e pela complexidade das aberturas das janelas [3].

Figura 1: Igreja de São Francisco

Fonte: joaopessoa.pb.gov.br

- **Eclético:** Este estilo é marcado pela combinação harmoniosa de elementos de diferentes períodos e estilos arquitetônicos [37]. O Theatro Municipal do Rio de Janeiro, ilustrado na Figura 2, destaca-se por sua fusão de estilos que incluem a opulência do Barroco, a simetria do Neoclássico e os ornamentos do Art Nouveau. Notam-se colunas coríntias, frisos decorados, a utilização de abóbadas e cúpulas grandiosas, além de uma fachada que apresenta uma variedade de texturas e materiais. A complexidade arquitetônica é evidenciada pelas esculturas que adornam a entrada principal e pelas balaustradas que coroam o edifício, demonstrando a habilidade do modelo em discernir uma rica tapeçaria de influências históricas.

Figura 2: Theatro Municipal do Rio de Janeiro

Fonte: globo.com

- **Neogótico:** O estilo neogótico pode ser identificado por características marcantes como arcos pontiagudos, torres altas e um uso predominante de pedra, criando estruturas que são ao mesmo tempo majestosas e intrincadamente detalhadas [36]. A Catedral Metropolitana de São Paulo, ilustrada na Figura 3, é um exemplo paradigmático deste estilo. A fachada da catedral é dominada por duas torres gêmeas, que se elevam imponentes contra o céu, exemplificando a aspiração neogótica de direcionar o olhar para o alto [35]. Os arcos pontiagudos, elemento-chave do neogótico, são visíveis nos portais principais, enquanto o uso extenso de pedra na construção reforça a sensação de permanência e solidez [35]. A habilidade de distinguir esses elementos é crucial para o reconhecimento do estilo neogótico, diferenciando-o de outros estilos que também podem utilizar arcos e torres, mas que não apresentam a mesma ênfase vertical ou detalhamento ornamentado.

Figura 3: Catedral Metropolitana de São Paulo

Fonte: uol.com.br

- **Neoclássico:** O estilo neoclássico é uma expressão artística e arquitetônica que

busca reavivar os princípios estéticos da Antiguidade clássica [34]. Caracteriza-se por colunas imponentes, simetria rigorosa e proporções harmônicas, elementos que coletivamente evocam um senso de ordem e clareza [34]. O Museu de Arte Contemporânea da Bahia, apresentado na Figura 4, reflete esses ideais neoclássicos através de sua fachada equilibrada e o uso de elementos clássicos como colunas e entablamentos. A estrutura simétrica do edifício, junto com a fachada clara e a presença de decorações sutis, ressoa com a estética neoclássica. A associação desses elementos com o neoclassicismo é fundamental para a identificação correta do estilo, destacando-se de outras influências arquitetônicas, que podem compartilhar certas características, mas não o conjunto coeso que define o neoclássico.

Figura 4: Museu de Arte Contemporânea da Bahia

Fonte: melhoresdestinos.com.br

- **Modernista:** O modernismo na arquitetura é marcado pela celebração da funcionalidade, formas puras e a eliminação de ornamentos supérfluos [5]. As linhas são frequentemente retas e limpas, promovendo uma estética de simplicidade e elegância [5]. O Museu de Arte Contemporânea de Niterói, ilustrado na Figura 5, é um exemplo icônico da arquitetura modernista. Projetado por Oscar Niemeyer, o edifício é notável por sua forma escultural e curvilínea, que desafia as convenções de linhas retas, mas mantém a clareza e a simplicidade do modernismo. A estrutura elevada sobre uma plataforma como um disco voador, e a ausência de decoração elaborada, ressaltam a funcionalidade e o foco no futuro que são fundamentais para o modernismo. A capacidade de reconhecer esses traços minimalistas e associá-los ao modernismo é essencial para a compreensão desse movimento arquitetônico.

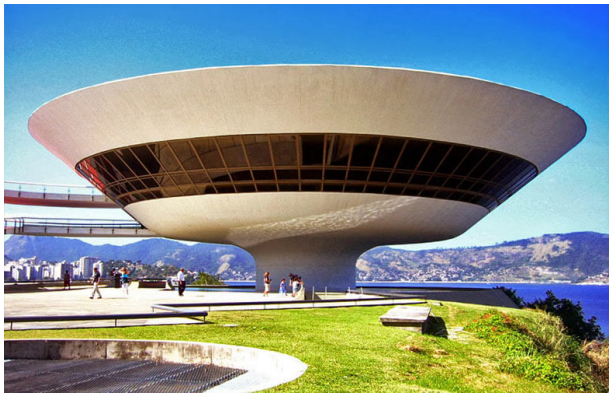


Figura 5: Museu de Arte Contemporânea de Niterói

Fonte: westwing.com.br

Assim como na arquitetura, onde cada estilo possui suas características e funcionalidades únicas, no campo das redes neurais, diferentes estruturas e modelos são desenvolvidos para atender a requisitos específicos de tarefas de processamento de dados e reconhecimento de padrões [28]. Na próxima seção, exploraremos os fundamentos de redes neurais, uma tecnologia chave na inteligência artificial, que, assim como os estilos arquitetônicos, tem seu próprio conjunto de princípios e padrões definidores.

2.3 Fundamentos de Redes Neurais

As redes neurais, um campo do aprendizado de máquina, buscam emular o funcionamento do cérebro humano para resolver problemas complexos em diversas áreas [42]. O elemento fundamental dessas redes é o neurônio artificial, que opera como uma abstração matemática do neurônio biológico.

O Perceptron, um dos modelos básicos de rede neural, consiste em um único neurônio com um limiar de ativação. Este modelo é ilustrado na Figura 6, onde se pode observar os seguintes componentes:

- **Dados de Entrada** (X_m): Representam os sinais de entrada ou estímulos externos processados pelo neurônio. Na computação neural, esses dados podem ser valores de pixels em uma imagem, elementos de dados em um conjunto de dados estruturado, ou outras formas de dados quantificáveis.

- **Pesos Sinápticos** (W_{jm}): Parâmetros fundamentais em um modelo de rede neural, que determinam a influência de cada sinal de entrada na saída do neurônio. Estes pesos ajustam a força com que cada entrada afeta a saída, análogos às conexões sinápticas no cérebro humano.

- **Limiar de Ativação ou *Bias*** (b_j): Um parâmetro adicional que permite ajustar a saída ao longo da função de ativação, independente das entradas. O bias modifica o ponto em que a função de ativação é ativada, facilitando ou dificultando a ativação do neurônio.
- **Junção Aditiva ou Somatório** ($\sum$): Neste ponto, todas as entradas, multiplicadas pelos respectivos pesos e somadas ao *bias*, são combinadas para formar um valor único. Esse processo é similar à soma de sinais de entrada por um neurônio biológico.
- **Função de Ativação** ($\phi(.)$): Aplicada ao somatório das entradas ponderadas e do *bias*, esta função determina a ativação do neurônio.
- **Saída** (y_j): A saída do neurônio é o resultado do processamento, transmitido para outros neurônios na rede ou como a saída final da rede. Esta saída é o resultado da aplicação da função de ativação ao somatório das entradas ponderadas e do bias.

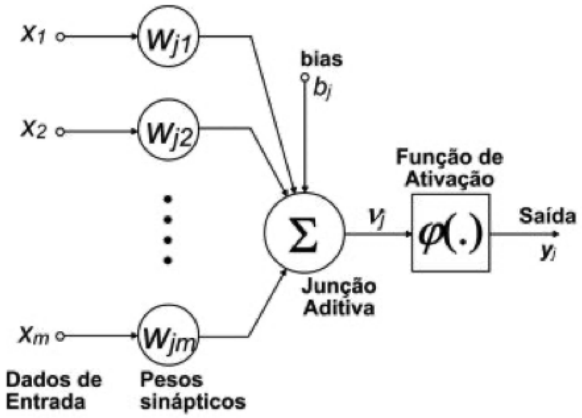


Figura 6: Modelo de um neurônio artificial

Fonte: researchgate.net

As redes neurais, evoluindo a partir de modelos básicos como o Perceptron, desempenham um papel vital em uma variedade de aplicações modernas. Elas são usadas em reconhecimento de voz, processamento de linguagem natural, diagnóstico médico, sistemas de recomendação, condução autônoma, entre muitos outros campos. A capacidade das redes neurais de modelar relações complexas e extrair padrões de grandes conjuntos de dados as torna uma ferramenta poderosa na era da grande quantidade de dados e da computação avançada.

A flexibilidade e a eficácia desses sistemas fazem com que as redes neurais sejam um componente essencial na vanguarda do desenvolvimento tecnológico, proporcionando avanços significativos em diversas áreas e apresentando um potencial ilimitado para futuras aplicações e inovações.

2.4 CNNs

Após explorar os princípios básicos das redes neurais, focamos em uma aplicação avançada destas teorias: as Redes Neurais Convolucionais (CNNs). As CNNs são altamente eficientes na identificação e aprendizado de padrões locais em dados. Abaixo, detalhamos os componentes fundamentais das CNNs:

1. **Convolução:** No cerne de uma CNN está o processo de convolução, ilustrado na Figura 7. Este processo envolve a aplicação de um filtro ou kernel sobre a entrada para extrair características relevantes. O filtro percorre a entrada, executando uma operação de multiplicação, elemento a elemento, e somando os resultados, produzindo um mapa de características. Este processo permite que a CNN aprenda padrões locais nos dados [43].

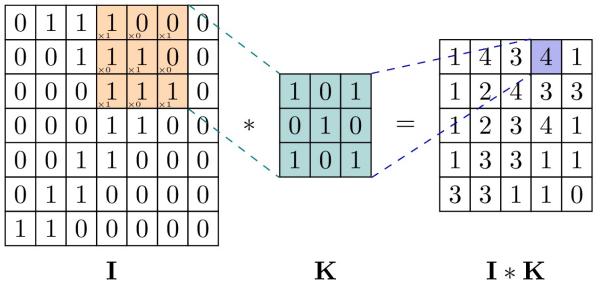


Figura 7: Operação de convolução

Fonte: serpro.gov.br

2. **Camadas de *Pooling*:** As camadas de *pooling* são aplicadas após a convolução para reduzir as dimensões espaciais do mapa de características, como demonstrado na Figura 8. O *max pooling*, o tipo mais comum de *pooling*, seleciona o valor máximo de uma região específica do mapa. Esta redução de dimensão contribui para a eficiência computacional e robustez do modelo [44].

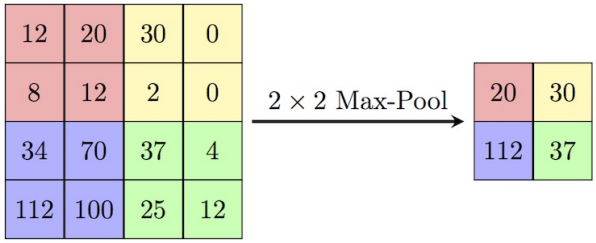


Figura 8: Operação de *Max Pooling*

Fonte: medium.com

3. **ReLU e Funções de Ativação:** Após as camadas convolucionais e de *pooling*, são frequentemente utilizadas funções de ativação como a ReLU, representada na

Figura 9. A ReLU introduz não-linearidades no modelo, permitindo que a rede aprenda padrões mais complexos [44].

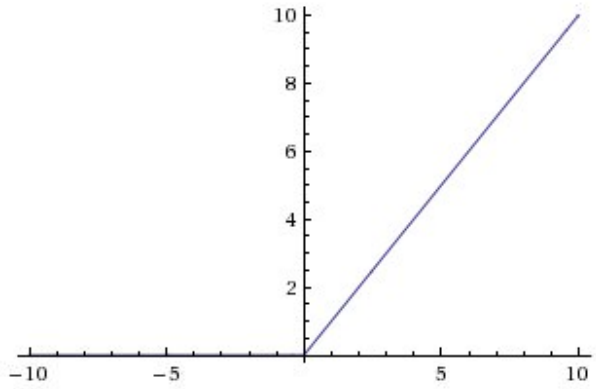


Figura 9: Função de Ativação ReLU

Fonte: medium.com

4. **Camadas Densas:** Após as camadas convolucionais e de *pooling*, os dados são transformados em um vetor unidimensional e processados por camadas densas, que combinam as características aprendidas para realizar tarefas específicas, como a classificação.

5. ***Backpropagation* e Aprendizado de Filtros:** Durante o treinamento, as CNNs ajustam os pesos dos filtros por meio do algoritmo de *backpropagation*, otimizando-os para tarefas específicas. Esse processo é orientado por uma função de perda que avalia o desempenho da rede [54].

As CNNs são especialmente eficazes para tarefas que requerem a compreensão da estrutura espacial ou temporal dos dados, como reconhecimento de imagens e análise de vídeo, devido à sua capacidade de aprender hierarquias de características diretamente dos dados [42].

2.5 Aprendizado por Transferência em Redes Neurais Convolucionais

O Aprendizado por Transferência (*Transfer Learning*) é uma abordagem significativa no contexto das CNNs, especialmente valiosa quando se lida com tarefas de classificação de imagens complexas. Esta técnica envolve o reaproveitamento de um modelo pré-treinado em uma tarefa ampla e sua subsequente adaptação para uma tarefa específica [46].

- **Vantagens do Aprendizado por Transferência:** A principal vantagem do Aprendizado por Transferência reside na economia de recursos e tempo, já que

reduz a necessidade de um grande conjunto de dados anotados para o treinamento [46]. Modelos pré-treinados em bases de dados abrangentes, como o ImageNet, já possuem um conhecimento preliminar de uma ampla gama de características visuais. Essa base de conhecimento, ao ser aplicada a tarefas específicas, permite ao modelo se concentrar em ajustar e refinar características importantes para a nova tarefa, melhorando a eficiência e a eficácia do treinamento.

- **Desafios do Aprendizado por Transferência:** Um dos principais desafios do Aprendizado por Transferência é a diferença entre os dados do pré-treino e os do domínio da tarefa alvo. Se os dados forem muito diferentes, a transferência de conhecimento pode ser menos eficaz. Outro desafio é o risco de "esquecimento catastrófico", onde o aprendizado de novas informações pode interferir e degradar o conhecimento adquirido anteriormente.

O Aprendizado por Transferência se apresenta, assim, como uma abordagem robusta e eficiente para aprimorar o desempenho de modelos de CNN em tarefas específicas. Com a estratégia correta, os modelos pré-treinados podem ser adaptados para aprender e destacar características cruciais necessárias para a nova tarefa, podendo otimizar tanto a acurácia, quanto a eficiência do modelo.

2.6 Arquiteturas EfficientNet e comparação com outras CNNs

Aprofundando a discussão sobre Redes Neurais Convolucionais (CNNs), é essencial destacar o papel das arquiteturas EfficientNet da Google, que representam um marco significativo na evolução dessas redes [47]. Essas arquiteturas emergem como uma evolução natural na sequência do Aprendizado por Transferência, ilustrando o progresso contínuo na busca por redes mais eficientes e precisas na análise de imagens.

As arquiteturas EfficientNet são particularmente relevantes no contexto do Aprendizado por Transferência, pois oferecem um modelo altamente otimizado e eficiente que pode ser facilmente adaptado para tarefas específicas [26]. Isso é crucial em aplicações que exigem análise detalhada de imagens, como no reconhecimento de imagens médicas, identificação de frutas, análise de dados de satélite e detecção de doenças em plantas [27][26].

A eficácia das EfficientNets é fundamentada em sua inovadora estratégia de escala, que incrementa de forma proporcional e sistemática a profundidade, largura e resolução da rede. Essa metodologia não apenas amplia a performance em uma variedade de tarefas de visão computacional, mas também preserva a eficiência de recursos, viabilizando sua aplicação em uma extensa gama de contextos, desde dispositivos móveis até infraestruturas de computação em nuvem [47]. A seguir, serão detalhadas as diversas arquiteturas

existentes da EfficientNet, ilustrando a versatilidade e capacidade destes modelos em diferentes cenários de aplicação e suas acurácias na ImageNet:

EfficientNet-B0: Este modelo inicial, com uma dimensão de entrada de 224 x 224 pixels e 5.3 milhões de parâmetros, tem uma precisão Top-1 de 77.1% [47]. É um modelo leve e compacto, ideal para aplicações com restrições de memória e capacidade de processamento, mantendo um bom desempenho em tarefas de classificação.

EfficientNet-B1: Com uma dimensão de entrada de 240 x 240 pixels e aproximadamente 7.8 milhões de parâmetros, este modelo tem uma precisão Top-1 de 79.1% [47]. Oferece um equilíbrio aprimorado entre acurácia e eficiência, sendo adequado para dispositivos com um pouco mais de poder de processamento do que o B0.

EfficientNet-B2: Aumentando a dimensão de entrada para 260 x 260 pixels e com cerca de 9.2 milhões de parâmetros, este modelo tem uma precisão Top-1 de 80.1% [47]. É otimizado para oferecer maior precisão sem necessitar de um grande aumento em recursos computacionais.

EfficientNet-B3: Com uma dimensão de entrada de 300 x 300 pixels e 12 milhões de parâmetros, este modelo tem uma precisão Top-1 de 81.6% [47]. Significativamente mais poderoso, é adequado para tarefas que exigem alta precisão em classificações de imagens.

EfficientNet-B4: Este modelo, com 380 x 380 pixels de dimensão de entrada e 19 milhões de parâmetros, tem uma precisão Top-1 de 82.9% [47]. Oferece um desempenho substancialmente melhorado, sendo apropriado para cenários onde a acurácia é crítica, como em diagnósticos médicos.

EfficientNet-B5: Com uma dimensão de entrada de 456 x 456 pixels e 30 milhões de parâmetros, este modelo tem uma precisão Top-1 de 83.6% [47]. Capaz de lidar com tarefas extremamente complexas, oferece uma das melhores precisões entre as variantes.

EfficientNet-B6: Esta versão, com uma entrada de 528 x 528 pixels e 43 milhões de parâmetros, tem uma precisão Top-1 de 84.0% [47]. É projetada para aplicações de ponta em inteligência artificial, onde os recursos computacionais são abundantes.

EfficientNet-B7: O modelo mais avançado da série, com uma dimensão de entrada de 600 x 600 pixels e 66 milhões de parâmetros, tem uma precisão Top-1 de 84.3% [47]. Define o estado da arte em termos de acurácia em classificação de imagens, ideal para tarefas computacionalmente intensivas em grandes conjuntos de dados.

EfficientNetV2-S: Comparada à EfficientNet-B0 original, a V2-S tem uma dimensão de entrada maior, com 300 x 300 pixels, e uma precisão Top-1 de 83.9% [48]. Isso representa um aumento significativo em relação ao B0, que tem 224 x 224 pixels. A V2-S é otimizada para eficiência em dispositivos móveis e aplicações com restrição de recursos,

oferecendo uma melhor precisão e eficiência em relação ao modelo original.

EfficientNetV2-M: Esta versão tem uma dimensão de entrada de 384 x 384 pixels e uma precisão Top-1 de 85.1% [48]. Em comparação com a EfficientNet-B3 original, a V2-M oferece uma melhoria notável em precisão e eficiência, equilibrando desempenho e consumo de recursos.

EfficientNetV2-L: Com uma entrada de 480 x 480 pixels e uma precisão Top-1 de 85.7%, a V2-L supera a EfficientNet-B4 original, que possui 380 x 380 pixels [48]. Este modelo é projetado para tarefas que exigem alta precisão, mas com uma eficiência computacional melhorada em relação à série original.

EfficientNetV2-XL: A versão mais avançada da série, a V2-XL, tem uma dimensão de entrada de 512 x 512 pixels e uma precisão Top-1 de 87.3% [48]. Comparado com a EfficientNet-B7 original, a V2-XL oferece melhorias em termos de eficiência de processamento, mantendo um alto nível de precisão em tarefas de classificação de imagens complexas.

Cada variante da EfficientNet é cuidadosamente calibrada para equilibrar o número de camadas com os requisitos de largura e resolução, otimizando o desempenho conforme o poder computacional disponível. A Figura 10 ilustra a arquitetura da EfficientNetB0, fornecendo uma referência para entender as variações nas versões subsequentes.

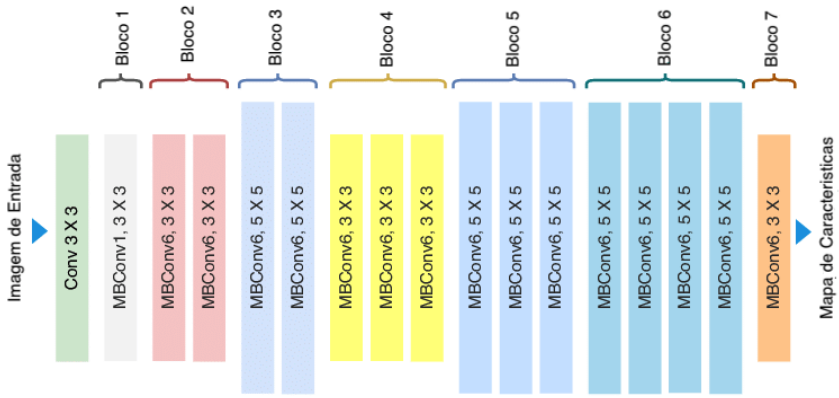


Figura 10: Arquitetura EfficientNetB0

Fonte: Adaptado de roboflow.com

Após uma análise abrangente de várias arquiteturas de rede neural, determinamos que a EfficientNetV2B0, é a mais adequada para nossas aplicações móveis. Esta arquitetura destaca-se por sua combinação única de eficiência computacional e acurácia, oferecendo um desempenho de acurácia Top-1 de 78,7% e Top-5 de 94,3% na base ImageNet, em um modelo compacto de apenas 29 MB [29]. Essa eficiência e compactação são cruciais em aplicações móveis, onde o armazenamento e a capacidade de processamento são limitados. A EfficientNetV2B0 equilibra o alto desempenho com as restrições de ta-

manho e capacidade, tornando-a uma escolha ideal para o ambiente de recursos restritos de dispositivos móveis [48]. Para ilustrar as diferenças entre a EfficientNetV2B0 e outras arquiteturas populares, a seguir apresentamos a tabela 1 que destaca o tamanho, acurácia e outros parâmetros relevantes de cada arquitetura.

Tabela 1: Comparação de Modelos de Machine Learning no Dataset ImageNet

Modelo	**Tamanho (MB)**	**Acurácia Top-1 (%)**	**Acurácia Top-5 (%)**	**Comentários**
Xception	88 MB	79.0	94.5	Boa acurácia, mas tamanho superior a outras modelos com acurácia similar
VGG16	528 MB	71.3	90.1	Tamanho inadequado para dispositivos móveis
VGG19	549 MB	71.3	90.0	Tamanho inadequado para dispositivos móveis
ResNet50	98 MB	74.9	92.1	Tamanho maior a outros modelos com acurácia superior
InceptionV3	92 MB	77.9	93.7	Tamanho maior a outros modelos com acurácia superior
InceptionResNetV2	215 MB	80.3	95.3	Tamanho inadequado para dispositivos móveis
MobileNetV2	14 MB	71.3	90.1	Baixa Acurácia
DenseNet121	33 MB	75.0	92.3	Bom tamanho, mas acurácia inferior a outros modelos com tamanho similar
NASNetLarge	343 MB	82.5	96.0	Tamanho inadequado para dispositivos móveis
EfficientNetV2B0	29 MB	78.7	94.3	Modelo escolhido com o melhor balanço entre acurácia e tamanho

Nota: Métricas de acurácia são baseadas no desempenho dos modelos no dataset ImageNet.

2.7 Métricas de Avaliação em Classificação de Imagens

Para uma avaliação do desempenho dos modelos de classificação de imagens, é crucial considerar múltiplos aspectos do seu comportamento preditivo. As métricas de Precisão, Sensibilidade, Medida-F1 e Suporte fornecem uma perspectiva abrangente e integrada, permitindo aos pesquisadores discernir não apenas a eficácia geral do modelo, mas também entender suas potenciais áreas de melhoria [40].

A matriz de confusão [25], representada na Tabela 2, é uma ferramenta poderosa nessa análise. Ela é composta por quatro valores: Verdadeiros Positivos (VP), Falsos Negativos (FN), Verdadeiros Negativos (VN) e Falsos Positivos (FP). Cada valor reflete a contagem de ocorrências para cada combinação de classe prevista e real. Os Verdadeiros Positivos e Verdadeiros Negativos representam as previsões corretas do modelo, enquanto os Falsos Positivos e Falsos Negativos representam os erros.

Tabela 2: Matriz de Confusão

	Previsto Positivo	**Previsto Negativo**
Real Positivo	VP	FN
Real Negativo	FP	VN

Os VP ocorrem quando o modelo corretamente prediz a classe positiva. Os FN acontecem quando o modelo incorretamente prediz a classe negativa para um resultado que é verdadeiramente positivo. De maneira similar, os FP surgem quando a classe negativa é incorretamente identificada como positiva, e os VN quando o modelo acertadamente identifica a classe negativa.

A Precisão quantifica a acurácia do modelo nas instâncias que ele classifica positivamente, respondendo à pergunta: "De todas as imagens classificadas como pertencentes a uma determinada categoria, quantas realmente pertencem a ela?". Uma Precisão elevada indica que o modelo tem uma baixa taxa de FPs, sendo particularmente importante em cenários onde o custo de um FP é elevado [39]. Matematicamente, é definida como:

$$Precisão = \frac{VP}{VP + FP}$$

A Sensibilidade, ou *Recall*, mede a capacidade do modelo de identificar todas as instâncias relevantes, buscando responder à pergunta: "De todas as imagens que realmente pertencem a uma determinada categoria, quantas o modelo conseguiu classificar corretamente?". Uma Sensibilidade elevada é crucial em situações onde é imperativo capturar todas as instâncias positivas, mesmo à custa de incorrer em mais Falsos Positivos

[39]. Matematicamente, é expresso por:

$$Sensibilidade = \frac{VP}{VP + FN}$$

A Medida-F combina Precisão e Sensibilidade em uma única pontuação, oferecendo uma visão balanceada do desempenho do modelo, sendo especialmente útil quando as classes estão desbalanceadas. Varia entre 0 e 1, com 1 indicando o desempenho perfeito e 0 a ausência completa de capacidade preditiva [39]. É calculado através da fórmula:

$$Medida - F = 2 \cdot \frac{Precisão \cdot Sensibilidade}{Precisão + Sensibilidade}$$

O Suporte refere-se ao número de instâncias reais da classe em questão no conjunto de dados, fornecendo contexto adicional às outras métricas e ajudando a interpretar a sua significância. Por exemplo, uma Sensibilidade elevada pode ser menos impactante se o Suporte for baixo, indicando que havia poucas instâncias da classe para serem identificadas [39].

2.8 Aplicação do LIME na Interpretabilidade de Modelos

Na sequência da nossa discussão sobre as métricas de avaliação em classificação de imagens, onde abordamos a importância de compreender a precisão, sensibilidade e outras métricas chave, é essencial reconhecer o papel da interpretabilidade dos modelos na análise dessas métricas. Neste contexto, a técnica LIME (Local Interpretable Model-agnostic Explanations) emerge como uma ferramenta fundamental. O LIME é aplicado à interpretabilidade de modelos de aprendizado de máquina, com foco particular na análise detalhada das decisões tomadas pelos modelos [23]. Esta abordagem permite que pesquisadores e profissionais compreendam melhor como as previsões são feitas, especialmente em relação aos resultados evidenciados pelas métricas de avaliação. Ao proporcionar explicações locais para as previsões de modelos de classificação, independentemente de sua complexidade, o LIME ajuda a elucidar as razões por trás do desempenho do modelo, seja ele medido em termos de precisão, sensibilidade ou qualquer outra métrica relevante.

O processo do LIME inicia-se com a perturbação dos dados de entrada, o que envolve a modificação sutil desses dados e a observação das alterações correspondentes nas previsões do modelo. Essas modificações são feitas de forma a simular variações nos dados que o modelo pode encontrar em cenários reais. Através desta técnica, é possível analisar como diferentes entradas afetam as saídas do modelo.

Após a coleta de dados sobre como as previsões do modelo variam com as perturbações, um modelo interpretável é treinado. Este modelo secundário, geralmente um

modelo linear como regressão linear ou logística, serve para aproximar as previsões do modelo complexo original. A escolha de um modelo linear deve-se à sua facilidade de interpretação; os coeficientes do modelo podem ser diretamente associados à importância de cada característica dos dados [24].

Este modelo interpretável é treinado especificamente para cada instância de previsão, focando em uma "região local"em torno da entrada em questão. Isso permite que o LIME forneça explicações específicas para cada previsão, em vez de tentar explicar o funcionamento geral do modelo.

Como ilustrado na Figura 11, o LIME inicia perturbando os dados de entrada, que é um passo fundamental para entender como diferentes características influenciam o modelo de aprendizado de máquina. A imagem à esquerda mostra um objeto de interesse segmentado de seu fundo, enquanto a imagem à direita apresenta uma interpretação de sobreposição de cores que destaca as regiões mais importantes para a previsão do modelo. A área verde destaca as características que mais contribuem positivamente para a classificação, enquanto as áreas vermelhas indicam uma contribuição negativa. Esta visualização fornece uma explicação intuitiva das previsões do modelo, alinhando-se com a funcionalidade central do LIME de gerar interpretações locais para previsões individuais.

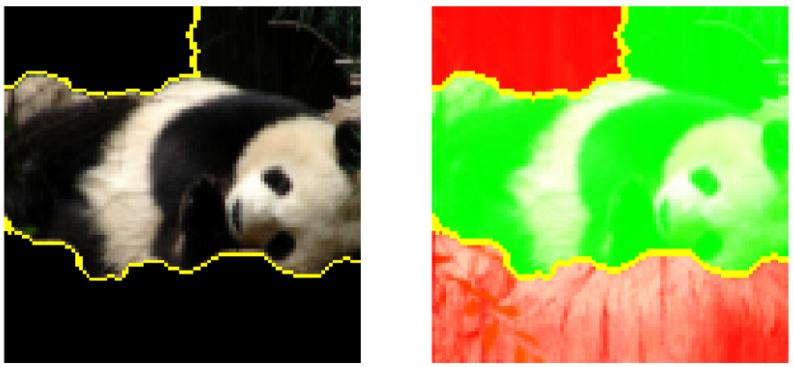


Figura 11: Exemplo de funcionamento do LIME

Fonte: medium.com

A aplicação do LIME é particularmente útil em contextos que exigem alta transparência, como em sistemas de recomendação, diagnósticos médicos ou qualquer aplicação onde compreender a base das previsões do modelo é tão crucial quanto as previsões em si. Ao desvendar os fatores que influenciam as decisões dos modelos, o LIME ajuda a garantir que esses sistemas sejam mais transparentes, confiáveis e, consequentemente, mais aceitos pelos usuários finais e pelas partes interessadas.

2.9 Trabalhos Relacionados

A classificação de estilos arquitetônicos é uma área de pesquisa que tem ganhado destaque devido à sua aplicação prática em diversos campos, incluindo história da arte, preservação patrimonial e tecnologia da informação. Este capítulo revisa estudos significativos nesse campo, destacando inovações e desafios enfrentados em diferentes abordagens.

Um estudo na área de classificação de estilos arquitetônicos emprega a técnica de Regressão Logística Latente Multinomial (*Multinomial Latent Logistic Regression*, MLLR) analisando 25 estilos arquitetônicos diferentes, alcançando uma acurácia de 46,21% [18]. Embora essa abordagem seja eficaz, ela enfrenta desafios significativos, incluindo a complexidade do modelo e o risco de sobreajuste, além de dificuldades na obtenção de alta acurácia em bases de dados amplas e complexas. Este estudo também lida com a classificação de estilos arquitetônicos misturados ou semelhantes e a escassez de bases de dados organizadas em larga escala. Uma questão adicional, particularmente relevante no contexto de dispositivos móveis, é a viabilidade de implementar esse método em telefones móveis, tendo em vista que a complexidade computacional e as demandas de memória da MLLR podem ser excessivas para dispositivos com capacidades limitadas [18].

Em uma abordagem distinta da nossa, um estudo relevante no campo de aprendizado profundo e visão computacional focou na medição de semelhanças visuais entre projetos arquitetônicos específicos, com ênfase na identificação dos trabalhos com base nos arquitetos responsáveis [10]. Este estudo, ao contrário do nosso, que se concentra na classificação de estilos arquitetônicos, aplicou um modelo de rede neural convolucional (*Convolutional Neural Network*, CNN) para diferenciar obras de 34 diferentes arquitetos. O modelo alcançou uma acurácia geral de 73%, embora tenha apresentado variações na capacidade de distinguir entre os trabalhos de vários arquitetos. Além disso, o estudo aponta limitações, como dificuldades na captura de características externas dos projetos e na consideração dos aspectos temporais na evolução do trabalho dos arquitetos [10].

Outro trabalho introduziu um método inovador na classificação de estilos arquitetônicos, utilizando um módulo de extração de características baseado em um Modelo de Partes Deformaveis (*Deformable Parts Model*, DPM) e um método de Projeção de Conjunto Melhorada (*Improved Set Projection*, IEP) [9]. Este estudo analisou 25 estilos arquitetônicos diferentes, incluindo estilos como Gótico, Barroco e Moderno e alcançou uma acurácia de 98,57%. Embora essa abordagem represente um avanço na área, ela enfrenta desafios significativos. A complexidade inerente à classificação de estilos arquitetônicos, a dependência de grandes volumes de dados rotulados, e a dificuldade em distinguir entre estilos visualmente semelhantes são alguns dos desafios identificados. Além disso, a variação no número de imagens por estilo na base de dados utilizada, a necessidade de ajuste fino dos parâmetros e a comparação com outras abordagens são aspectos

cruciais que destacam a complexidade e a necessidade de um aprofundamento contínuo na pesquisa nesta área [9].

Uma outra alternativa se concentra no uso do algoritmo Vizinho Mais Próximo Naive Bayes (*Naive Bayes Nearest Neighbor*, NBNN) para a classificação de estilo, alcançando uma acurácia de 82,25% [11]. Este algoritmo, apesar de sua simplicidade e capacidade de evitar sobreajuste, pode enfrentar limitações ao lidar com uma ampla variedade de estilos arquitetônicos. Este aspecto é particularmente relevante quando consideramos que o estudo se limita a apenas três estilos distintos: Renascentista Flamengo, Haussmanniano e Neoclássico. Embora essa escolha focada permita uma análise detalhada e um tratamento aprofundado desses estilos específicos, ela, no entanto, restringe a abrangência e a aplicabilidade do modelo a uma variedade mais ampla de arquiteturas, representando um ponto de atenção na avaliação geral do trabalho [11].

Já outro trabalho propõe uma abordagem inovadora na classificação de estilos arquitetônicos focando especificamente em janelas de fachadas de edifícios [19]. Este estudo enfrenta a complexa tarefa de classificar janelas por estilos arquitetônicos, desafiado pela alta diversidade intraclasse e pela presença de reflexos nas imagens. Enquanto o estudo avança na classificação de janelas com uma taxa de sucesso significativa (95,16%), apontando a necessidade de futuros aperfeiçoamentos para reduzir classificações incorretas, ele se limita na análise de apenas três estilos arquitetônicos principais europeus: Românico, Gótico e Renascença/Barroco. Esta limitação, embora permita um foco detalhado nestes estilos, consequentemente restringe a abrangência e a aplicabilidade do modelo a um espectro mais amplo de estilos arquitetônicos, representando uma lacuna notável em seu escopo. [19]

Estes estudos refletem a diversidade de métodos e desafios na classificação de estilos arquitetônicos. Embora cada abordagem traga contribuições valiosas, as limitações, como a restrição no número de estilos estudados e a dependência de grandes volumes de dados, destacam a necessidade de contínuo desenvolvimento e inovação na área. A compreensão desses trabalhos enriquece nosso próprio estudo, permitindo-nos abordar as limitações identificadas e explorar novas possibilidades na classificação de estilos arquitetônicos em dispositivos móveis.

Uma síntese comparativa das abordagens em estudos anteriores é apresentada na Tabela 3, onde resumimos as principais técnicas utilizadas, o número de classes, a acurácia alcançada e a compatibilidade com dispositivos móveis. Esta tabela oferece uma visão geral útil das tendências e resultados na área, enfatizando tanto os sucessos quanto os desafios enfrentados por diferentes pesquisadores.

Tabela 3: Trabalhos Relacionados

Estudo	**Metodo**	**Classes**	**Acurácia**	**Otimizado *Mobile***
Xu et al. (2014)	Regressão Logística Latente Multinomial (MLLR)	25 estilos	46,21%	Não
Zhao et al. (2018)	Modelo de Partes Deformáveis e Projeção de Conjunto Melhorada (DPM/IEP)	25 estilos	98,57%	Não
Mathias et al. (2011)	Vizinho Mais Próximo Naive Bayes (NBNN)	3 estilos	82,25%	Não
Shalunts et al. (2011)	Classificação de Janelas por Estilo	3 estilos	95,16%	Não
Yoshimura et al. (2018)	Rede Neural Convolucional (CNN)	34 arquitetos	73,84%	Não
Nosso modelo	Rede Neural Convolucional (CNN)	5 estilos	83%	Sim

3 METODOLOGIA

Este capítulo descreve a metodologia empregada no projeto de identificação e classificação de estilos arquitetônicos no Brasil, dividida em 5 seções essenciais.

Na seção "Construção do Conjunto de Dados de Estilos Arquitetônicos", detalhamos a elaboração do conjunto de dados. O processo inclui a seleção de estilos arquitetônicos significativos, a identificação de edifícios representativos e a coleta de imagens apropriadas. As imagens foram escolhidas com base na clareza, perspectiva, iluminação e representação precisa do estilo arquitetônico, respeitando os direitos autorais.

A seção "Hardware Utilizado" expõe o hardware utilizado no desenvolvimento do projeto. As especificações técnicas dos dispositivos para treinamento e teste do modelo são essenciais para a eficiência do processo de treinamento e a precisão dos testes.

"Criação do Modelo" discute a utilização do aprendizado transferido através de bibliotecas de código aberto para desenvolver um modelo baseado na arquitetura EfficientNetV2B0. Esta parte abrange desde a seleção do modelo até seu teste, com ênfase em técnicas de aumento de dados para aprimorar a capacidade de generalização do modelo.

A seção "Conversão CoreML" detalha a transformação do modelo desenvolvido para um formato compatível com plataformas de dispositivos móveis específicas, faci-

litando sua integração em diferentes dispositivos. Este processo permite a utilização eficiente dos recursos de hardware destes dispositivos.

Por último, "Integração do Aplicativo" relata o desenvolvimento do aplicativo, integrando tecnologias de interface de usuário declarativa, aprendizado de máquina e gerenciamento de dados baseado em nuvem. A combinação dessas tecnologias resultou em um aplicativo interativo com uma interface eficiente, modelo de aprendizado de máquina integrado e gestão eficiente de dados.

Cada seção oferece uma visão detalhada da metodologia aplicada, abrangendo desde a coleta de dados até a implementação do modelo em um aplicativo funcional.

3.1 Construção do Conjunto de Dados de Estilos Arquitetônicos

Esta seção detalhará as etapas envolvidas na montagem deste conjunto de dados, desde a identificação de estilos arquitetônicos chave, até a seleção de imagens livres de direitos autorais.

O primeiro passo no processo envolveu uma seleção sobre os principais estilos arquitetônicos presentes no Brasil. Realizamos um estudo, com base em literatura acadêmica, bancos de dados arquitetônicos e opiniões de especialistas, para identificar cinco estilos arquitetônicos chave que definem o Brasil: Barroco, Neoclássico, Neogótico, Eclético e Modernista [2][14][5][3][4]. Uma compilação visual destes estilos é apresentada na Figura 12, ilustrando exemplos de edifícios para cada estilo arquitetônico.

Após a identificação dos principais estilos arquitetônicos, procedemos com a elaboração de uma lista contendo os edifícios mais representativos de cada estilo. Essa compilação foi realizada com base em uma revisão bibliográfica, incluindo fontes reconhecidas no campo da arquitetura. Utilizamos obras e autores que oferecem análises aprofundadas e detalhadas sobre cada estilo arquitetônico e seus exemplos mais emblemáticos [30][31].

Com os estilos arquitetônicos e os edifícios-chave identificados, o próximo passo foi a coleta de imagens de alta qualidade e sem direitos autorais, com um requisito mínimo de resolução de 300x300 pixels. Recorremos ao Google Images, um vasto repositório de conteúdo visual, para encontrar imagens adequadas para o nosso conjunto de dados. O processo de seleção não foi automatizado; ao contrário, optamos por uma abordagem manual para garantir a melhor qualidade e clareza. Cada imagem foi revisada individualmente por sua clareza visual, perspectiva e iluminação, bem como a extensão em que o edifício e seu estilo arquitetônico eram claramente exibidos. Apenas imagens que atenderam a esses critérios e também estavam livres de restrições de direitos autorais foram selecionadas para o nosso conjunto de dados.

Após o processo de seleção de imagens, obtivemos um total de 927 imagens para

Figura 12: Estilos arquitetônicos usados

Fonte: acervodigital.unesp.br

treinamento e 231 imagens para validação e 292 imagens para testes. A seguir, a Tabela 4 ilustra a distribuição destas imagens por estilo arquitetônico.

Tabela 4: Número de Imagens de Estilos Arquitetônicos

Estilo Arquitetônico	Treinamento	Validação	Teste
Barroco	208	52	65
Neoclássico	180	45	58
Neogótico	160	40	51
Eclético	185	46	58
Modernista	192	48	60

3.2 Hardware Utilizado

No desenvolvimento deste projeto, o hardware teve um papel crítico. O treinamento e a conversão do modelo foram executados em um laptop com processador Apple M2, que possui uma velocidade de clock de 3,5 GHz, 4 núcleos de eficiência, 4 núcleos de desempenho, uma GPU integrada de 8 núcleos, um *Neural Engine* de 16 núcleos e 8 GB de RAM, operando no macOS Ventura 13.3.

Para testar o modelo, foram utilizados três dispositivos: o próprio laptop Apple M2, operando no sistema macOS 13.5; um telefone móvel com CPU Apple A14, operando no sistema iOS 16.1; e um dispositivo adicional com a CPU Apple A10, executando o sistema iOS 15.7.6.

O telefone móvel utilizado possui uma CPU Apple A14, opera a 3,1 GHz, contando com 4 núcleos de eficiência, 2 núcleos de desempenho, uma GPU integrada de 4 núcleos, um *Neural Engine* de 16 núcleos e 4 GB de RAM. O dispositivo com a CPU Apple A10, por sua vez, representa uma geração anterior de hardware, ainda assim oferecendo uma performance comparável para o teste do modelo.

Essa configuração de hardware permitiu uma avaliação detalhada do modelo em ambientes distintos, assegurando a verificação de seu desempenho tanto no dispositivo de desenvolvimento, quanto em um dispositivo móvel.

3.3 Criação do Modelo

Para agilizar o processo de criação do modelo, utilizamos o poder do aprendizado transferido usando TensorFlow 2.13.0 e Keras 2.13.1, duas bibliotecas de código aberto próprias para aplicações de aprendizado de máquina. TensorFlow, desenvolvido pelo Google Brain, é uma ferramenta abrangente para criar e treinar vários tipos de redes neurais, enquanto Keras é uma biblioteca de redes neurais escrita em Python, que roda

sobre TensorFlow, fornecendo uma interface mais simples para construções complexas de redes neurais.

No contexto do nosso trabalho, enfrentamos o desafio único da diversidade e variabilidade de estilos arquitetônicos. Esses estilos podem se apresentar em uma infinidade de formas e orientações, dependendo da perspectiva da imagem capturada, das condições de iluminação, ou até mesmo das mudanças sazonais ao redor. Esta variabilidade, embora enriqueça nosso conjunto de dados, pode potencialmente causar problemas para o modelo, fazendo-o se ajustar demais às nuances específicas presentes nos dados de treinamento e, assim, ter dificuldades com novos dados nunca vistos antes.

No processo de *transfer learning*, começamos com o modelo EfficientNetV2B0 pré-treinado, que já havia sido treinado em um grande conjunto de dados para tarefas de visão computacional. Isso significa que o modelo já havia aprendido uma variedade de características visuais úteis em imagens, tornando-o uma base sólida para nosso problema de reconhecimento de estilos arquitetônicos.

Implementamos diversas técnicas de aumento de dados utilizando a biblioteca *TensorFlow*, mais especificamente por meio do módulo *ImageDataGenerator*. Abaixo, detalhamos as técnicas aplicadas:

- **Inversão Horizontal:** As imagens foram espelhadas horizontalmente de forma aleatória, definido pelo parâmetro `horizontal_flip=True`. Isso ajuda o modelo a reconhecer estruturas simétricas e edifícios capturados de diferentes perspectivas.
- **Rotação:** As imagens foram rotacionadas aleatoriamente dentro de um intervalo de ± 5 graus, configurado por `rotation_range=5`. Isso permite que o modelo se adapte a variações nos ângulos de captura das imagens.
- **Deslocamento de Largura:** As imagens foram deslocadas horizontalmente em até 10% da largura total, como indicado por `width_shift_range=0.1`.
- **Deslocamento de Altura:** As imagens foram deslocadas verticalmente em até 10% da altura total, definido por `height_shift_range=0.1`.
- **Cisalhamento:** Aplicamos uma pequena transformação de cisalhamento, especificada por `shear_range=0.1`.
- **Zoom:** Realizamos um pequeno zoom nas imagens, dentro de um intervalo de $\pm 10\%$, configurado por `zoom_range=0.1`.
- **Ajuste de Brilho:** Variação aleatória do brilho das imagens entre 80% e 120% do original, controlada por `brightness_range=(0.8, 1.2)`.

Para lidar com pontos fora dos limites da imagem, após a aplicação dessas transformações, optamos por preenchê-los de acordo com a estratégia "nearest", especificada por `fill_mode=‘‘nearest"`.

É importante destacar que, para o gerador de validação, não aplicamos essas técnicas de aumento de dados, exceto pelo redimensionamento das imagens. Além disso, optamos por não embaralhar os dados de validação e teste, definindo `shuffle=False` para garantir a consistência durante a avaliação do modelo.

A primeira etapa foi importar o modelo EfficientNetV2B0 junto com suas camadas pré-treinadas. Em seguida, adaptamos o modelo para nossa tarefa específica, que era a classificação de estilos arquitetônicos. Para isso, substituímos a camada de saída original do modelo por uma nova camada de classificação, com o número de classes correspondentes aos diferentes estilos arquitetônicos que queríamos reconhecer.

Além disso, congelamos as camadas pré-treinadas do EfficientNetV2B0 durante o treinamento, o que significa que essas camadas não foram atualizadas durante o processo de treinamento. Isso foi importante para manter as características visuais aprendidas durante o pré-treinamento, evitando que fossem perdidas.

O próximo passo foi a calibração dos hiperparâmetros do modelo para nossa tarefa específica. Selecionamos uma taxa de aprendizado e determinamos o otimizador Adam com uma agenda de decaimento exponencial da taxa de aprendizado, e uma função de perda de entropia cruzada categórica, adequados para o treinamento de uma tarefa de classificação multi-classe. Configuramos também o treinamento para um total de 300 épocas, com um mecanismo de parada antecipada baseado no monitoramento da perda de validação (val_loss) e um limite de paciência de 20 épocas. O tamanho do lote foi definido conforme mencionado anteriormente.

Durante o treinamento, utilizamos nosso conjunto de dados diversificado, que incluía imagens de vários estilos arquitetônicos e as técnicas de aumento de dados mencionadas anteriormente. Isso permitiu que o modelo aprendesse a reconhecer os estilos arquitetônicos, mesmo diante da variabilidade nas condições de captura das imagens.

3.4 Conversão CoreML

O CoreML, *framework* de aprendizado de máquina da Apple, é uma ferramenta essencial para desenvolvedores que desejam integrar modelos de aprendizado de máquina em aplicações para iOS, macOS, watchOS e tvOS. Este *framework* suporta uma ampla gama de tipos de modelos, incluindo, mas não se limitando a redes neurais, árvores de decisão e máquinas de vetores de suporte [16]. O design do *framework* se concentra no desempenho em dispositivo, com otimizações para reduzir a memória e o consumo de energia, garantindo que o modelo possa utilizar a CPU, GPU e *Neural Engine* para

máxima eficiência e velocidade.

O *Neural Engine*, presente em dispositivos recentes como iPhones e iPads, é uma unidade de processamento neural de 16 núcleos capaz de realizar 15.8 teraflops, oferecendo um aumento significativo de 26 vezes em relação ao modelo original do iPhone X. Ele permite a execução eficiente de algoritmos de aprendizado de máquina e inteligência artificial, otimizando o desempenho e a vida útil da bateria [49]. Além disso, é possível converter modelos de aprendizado de máquina das plataformas PyTorch e TensorFlow para o formato CoreML da Apple, aproveitando assim as capacidades do *Neural Engine* nos dispositivos Apple. Nossa decisão de implantar nosso modelo no formato CoreML nos permite aproveitar totalmente essas capacidades, melhorando assim a experiência do usuário.

Após as fases de treinamento, validação e teste, o modelo TensorFlow é normalmente salvo em formato .keras. No entanto, para implantar o modelo em dispositivos Apple, ele precisa ser convertido para o formato CoreML (.mlmodel). Esse processo de conversão é facilitado pelo pacote Python *coremltools* na sua versão 7.1. Internamente, o processo de conversão envolve o mapeamento de operações do TensorFlow para suas equivalentes no CoreML. Por exemplo, uma operação de convolução no TensorFlow será convertida em uma camada de convolução no CoreML. No entanto, nem todas as operações do TensorFlow têm equivalentes diretos no CoreML e, em alguns casos, uma única operação do TensorFlow pode ser traduzida em várias operações CoreML para alcançar a mesma funcionalidade.

3.5 Integração do Aplicativo

O desenvolvimento de nosso aplicativo foi uma combinação de tecnologias modernas, incluindo SwiftUI, CoreML e Firebase.

O núcleo da interface do nosso aplicativo foi desenvolvido usando SwiftUI, uma ferramenta de interface de usuário da Apple. Ele permite que os desenvolvedores desenhem e componham interfaces de usuário de forma declarativa [17]. Isso significa que, em vez de instruir o programa sobre como criar a interface de usuário, especificamos o que a interface de usuário deve fazer. Isso simplifica fundamentalmente o código, tornando-o mais legível. SwiftUI também usa uma abordagem baseada em dados, o que significa que a interface de usuário é atualizada conforme o estado dos dados muda. Desta forma, a interface de usuário está sempre em sincronia com os dados subjacentes, sem a necessidade de intervenção manual para atualizá-la [7]. O recurso de visualização ao vivo do SwiftUI foi fundamental em nosso processo de desenvolvimento, permitindo a visualização em tempo real das mudanças na interface do usuário.

Como parte significativa de nossa estratégia de desenvolvimento, integramos nosso

modelo CoreML em nosso aplicativo. Este modelo, projetado para atender às nossas necessidades específicas, foi incorporado usando o *framework* CoreML da Apple.

Optamos pelo uso do Firebase na sua versão de iOS 10.16.0 e seu banco de dados de documentos NoSQL, Firestore, como uma solução eficiente para armazenar *logs* de tempo de execução e informações sobre o hardware de execução. A suíte de ferramentas baseadas na nuvem do Firebase facilitou a implementação dessa estratégia, proporcionando um ambiente otimizado para o armazenamento e recuperação desses dados específicos. A capacidade do Firestore de realizar armazenamento de dados em tempo real e sincronização provou ser vital para assegurar que todas as informações fossem capturadas e armazenadas de forma confiável, mesmo em situações de intermitência na conexão com a internet. Este acervo de dados permitiu uma avaliação detalhada do desempenho do sistema e do hardware ao longo do tempo.

Em conclusão, a combinação de SwiftUI para uma interface de usuário eficiente, CoreML para a integração de nosso modelo personalizado de aprendizado de máquina, e Firebase com Firestore para a gestão eficaz dos dados do usuário, culminou na criação de um aplicativo interativo.

4 RESULTADOS E DISCUSSÃO

Neste capítulo, conduzimos uma avaliação dos resultados obtidos em nosso estudo, com ênfase no desempenho de um modelo de aprendizado de máquina executado em diferentes dispositivos e sistemas operacionais. Avaliamos a eficácia do modelo em interpretar e classificar estilos arquitetônicos por meio de um conjunto diversificado de métricas de desempenho e exploramos a aplicabilidade desses resultados no contexto prático do aplicativo.

O estudo começa com a análise do modelo de aprendizado de máquina, onde métricas importantes como precisão, sensibilidade, e medida-F1 são investigadas para cada categoria arquitetônica identificada. Prosseguimos com uma avaliação detalhada da interpretação das decisões do modelo, enriquecida pelo uso da técnica LIME, que fornece *insights* sobre as predições.

A discussão avança com uma análise comparativa do desempenho de três SoCs da Apple — A10 Fusion, A14 Bionic e M2 — em diferentes versões de sistemas operacionais.

4.1 Desempenho do Modelo

A avaliação de modelos de aprendizado de máquina em classificação de estilos arquitetônicos revela *insights* importantes sobre suas capacidades e limitações. O conjunto de teste composto por 292 imagens, divididas entre estilos como Barroco, Neoclássico, Neogótico, Modernista e Eclético, oferece a base para análise.

O desempenho na categoria Neogótico, com sensibilidade de 85%, sugere uma habilidade do modelo em identificar características marcantes desse estilo. Já o estilo Modernista, com precisão impressionante de 95%, mostra que o modelo é eficaz na correta classificação de imagens desse estilo, indicando uma clara distinção das suas características visuais.

Por outro lado, a categoria Eclético, com precisão de apenas 64%, destaca a complexidade inerente a este estilo. A natureza diversa do Eclético, que amalgama diferentes influências e elementos arquitetônicos, parece induzir confusão no modelo, resultando em uma classificação menos precisa.

Os estilos Barroco e Neoclássico apresentam resultados de medida-F1 de 0,84 e 0,73, respectivamente, mas ainda com margem para melhoria. Em particular, a sensibilidade mais baixa na categoria Neoclássico sugere a necessidade de refinamento do modelo para melhor capturar as nuances desse estilo. Isso poderia levar a uma classificação mais abrangente e precisa, melhorando a eficácia geral do modelo.

Embora nosso modelo, baseado em uma Rede Neural Convolucional (CNN) mais

simples, apresente um desempenho ligeiramente inferior com uma precisão de 83%, sua vantagem reside na viabilidade de execução em dispositivos móveis. Em contraste, abordagens como o DPM e IEP, que alcançam uma acurácia elevada de 98,57% [9], geralmente exigem mais recursos computacionais, tornando-os menos práticos para aplicações em dispositivos com capacidades limitadas.

A Tabela 5 apresenta os resultados quantitativos do desempenho do modelo, oferecendo uma visão detalhada do seu comportamento em relação a cada estilo arquitetônico.

Tabela 5: Métricas de Classificação

Classe	**Precisão**	**Sensibilidade**	**Medida-F1**	**Suporte**
Barroco	0,87	0,82	0,84	65
Eclético	0,64	0,78	0,70	58
Modernista	0,95	1,00	0,98	60
Neoclássico	0,78	0,69	0,73	58
Neogótico	0,93	0,85	0,89	47
Acurácia		0,83		
Média Macro		0,84		
Média Ponderada		0,83		

Um componente crítico para avaliar o desempenho de um modelo de classificação é a sua Matriz de Confusão, ilustrada na Figura 13. Esta matriz permite uma apreciação aprofundada da precisão do modelo ao confrontar as previsões realizadas com os valores verdadeiros. Por meio da Matriz de Confusão, não só identificamos os acertos do modelo mas também discernimos os diversos tipos de equívocos que ocorrem. Este diagnóstico é essencial para reconhecer as categorias em que o modelo exibe alta eficiência e aquelas onde melhorias são necessárias. A análise da Matriz de Confusão disposta fornece uma visão clara das forças e fraquezas do modelo:

- Para o estilo Barroco, o modelo teve um alto número de acertos (53 corretamente classificados), com poucos casos confundidos com os estilos Eclético (3 casos) e Neogótico (1 caso).
- A classe Eclética teve 42 acertos, mas também mostra um número significativo de confusões com o estilo Neoclássico (11 casos), indicando uma área que pode precisar de mais discernimento por parte do modelo.
- A classe Modernista foi perfeitamente classificada, com todos os 60 casos corretos e nenhuma confusão com outros estilos, o que sugere que o modelo é muito eficaz em identificar as características desse estilo arquitetônico.
- Para o estilo Neoclássico, o modelo acertou 44 casos, mas houve confusão notável

com o estilo Eclético (8 casos) e Barroco (3 casos), o que pode ser um ponto para melhoria.

- Por fim, a classe Neogótica teve 42 acertos, com algumas confusões, especialmente com a classe Eclética (4 casos).

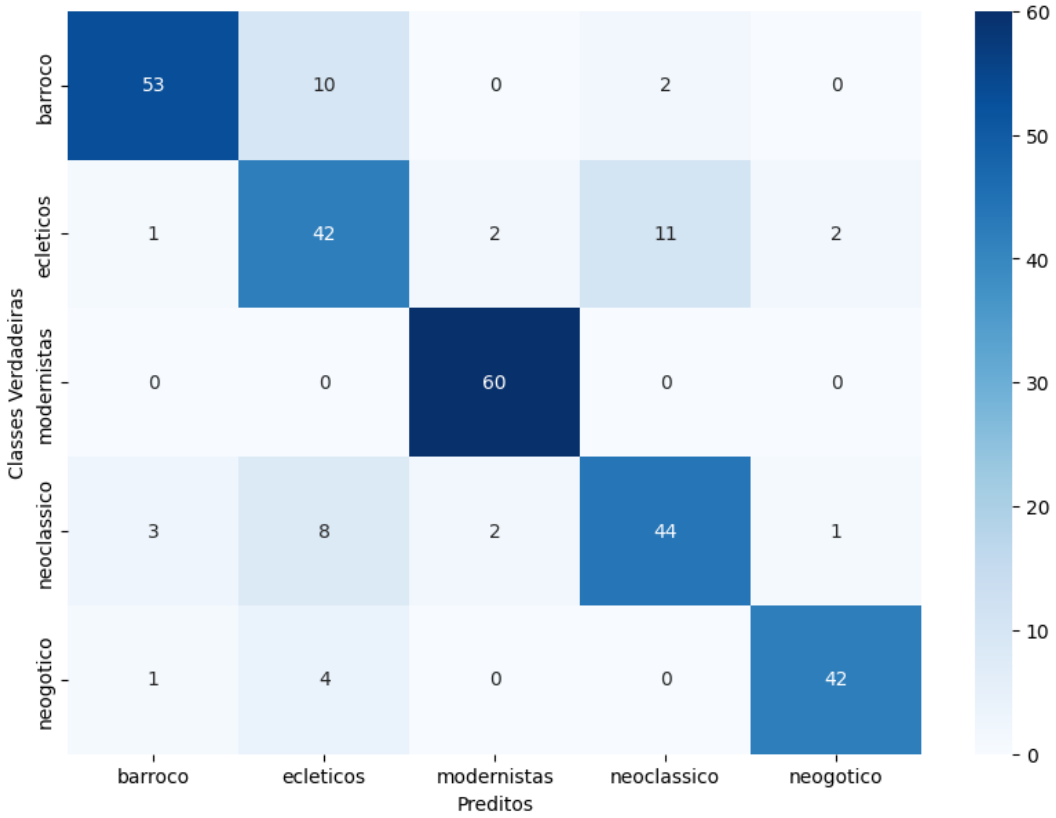


Figura 13: Matriz de Confusão.

4.2 Avaliação das Previsões do Modelo com LIME

A aplicação da técnica LIME contribuiu significativamente para uma avaliação detalhada das previsões do modelo de aprendizado de máquina, com enfoque na identificação de estilos arquitetônicos em edificações selecionadas. O principal objetivo desta aplicação era desvendar os critérios e fundamentos adotados pelo modelo ao atribuir estilos arquitetônicos específicos, assegurando assim um entendimento dos processos de decisão automatizados.

Para alcançar este propósito, o LIME foi configurado para destacar e colorir em vermelho as cinco áreas mais influentes na determinação de cada estilo arquitetônico detectado pelo modelo. Esta configuração permitiu uma análise mais intuitiva e visual das características arquitetônicas predominantes que influenciaram as decisões do modelo.

A Figura 14 focaliza na Diocese de Petrolina, onde o LIME foi aplicado para detalhar as previsões do modelo de classificação arquitetônica. A subfigura (a) exibe a imagem original do edifício, e a subfigura (b) ilustra a aplicação do LIME, evidenciando as regiões que o modelo identificou como influentes na classificação do estilo arquitetônico neo-gótico.

Na subfigura (b), o mapa de calor realça elementos como as agulhas pontiagudas das torres e os arcos ogivais, que são característicos do estilo neo-gótico [36] e foram identificados pelo modelo como influentes na classificação. A análise visual proporcionada pelo LIME confirma que o modelo reconhece adequadamente os elementos arquitetônicos relevantes do estilo neo-gótico presentes na Diocese de Petrolina.

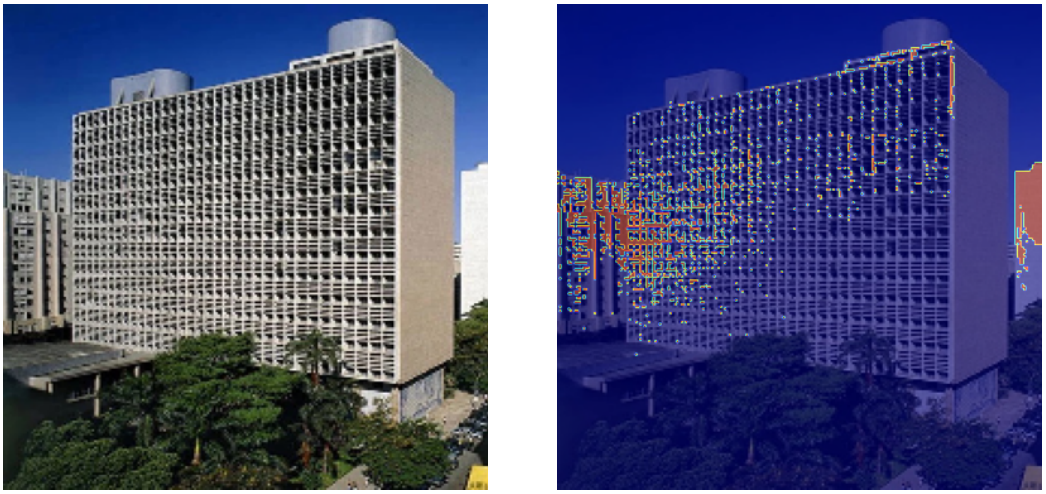
(a) Foto original da Diocese de Petrolina

(b) Diocese de Petrolina com análise do LIME

Figura 14: Diocese de Petrolina antes e depois do mapa de calor do LIME

A Figura 15 apresenta o Palácio Gustavo Capanema, uma edificação emblemática do modernismo brasileiro, submetida à análise do modelo de aprendizado de máquina com a técnica LIME. A subfigura (a) mostra a foto original do prédio, enquanto a subfigura (b) destaca, através de um mapa de calor do LIME, as regiões identificadas pelo modelo como significativas para a classificação do estilo modernista.

Na subfigura (b), o mapa de calor do LIME marca em vermelho as cinco áreas mais influentes que orientaram a detecção do estilo modernista pelo modelo. Elementos como as linhas retas da fachada, a ausência de ornamentos excessivos e a estrutura de pilotis são distintos desta corrente arquitetônica [5] e aparecem destacados, sublinhando a importância destes na decisão do modelo. A precisão com que o LIME aponta esses elementos confirma a competência do modelo em reconhecer e interpretar corretamente os traços do modernismo na arquitetura do Palácio Gustavo Capanema.



(a) Foto original do Palácio Gustavo Capanema

(b) Palácio Gustavo Capanema com análise do LIME

Figura 15: Palácio Gustavo Capanema antes e depois do mapa de calor do LIME

A Figura 16 compara duas representações do Teatro Santa Roza, um edifício que mescla elementos do barroco com características neoclássicas, o qual foi classificado pelo modelo como pertencente ao estilo neoclássico. A subfigura (a) exibe a foto original do teatro, e a subfigura (b) mostra a imagem após a aplicação do LIME, destacando as áreas que o modelo considerou decisivas para essa classificação.

O mapa de calor do LIME, na subfigura (b), identifica e realça em vermelho as características que levaram à detecção do neoclassicismo pelo modelo. Entre essas características, o modelo enfatizou aspectos como a simetria da fachada, a forma das janelas e a ornamentação mais sóbria e geométrica, que são distintivos do neoclassicismo [34], apesar da presença de elementos barrocos.

A análise do LIME sugere que, embora o Teatro Santa Roza possua atributos barrocos, o modelo deu predominância às características neoclássicas na sua previsão. Esse resultado ressalta a necessidade de um exame detalhado das bases de conhecimento do modelo, pois reflete sobre sua interpretação das características arquitetônicas e a influência disso na classificação final.

(a) Foto original do Teatro Santa Roza

(b) Teatro Santa Roza com análise do LIME

Figura 16: Teatro Santa Roza antes e depois do mapa de calor do LIME

A Figura 17 destaca o Teatro Municipal do Rio de Janeiro, refletindo a mistura de influências arquitetônicas que caracterizam o estilo eclético do edifício. A subfigura (a) mostra a fachada original do teatro, e a subfigura (b) demonstra a aplicação do LIME, que aponta para as características determinantes na classificação do modelo.

No mapa de calor do LIME, visualizado na subfigura (b), áreas específicas são ressaltadas, denotando a combinação de elementos neoclássicos, como colunas coríntias e frontões [34], com detalhes do renascimento francês evidenciados nas cúpulas e balaustradas [38]. Esses elementos, destacados pelo modelo, sublinham a diversidade estilística que define o ecletismo do teatro.

(a) Foto original do Teatro Municipal do Rio de Janeiro

(b) Teatro Municipal do Rio de Janeiro com análise do LIME

Figura 17: Teatro Municipal do Rio de Janeiro antes e depois do mapa de calor do LIME

4.3 Análise de Desempenho do Aplicativo

Esta seção apresenta uma análise comparativa do desempenho de diferentes SoCs da Apple, executando operações em variados sistemas operacionais da companhia. Utilizando a Tabela 6, observamos os tempos de execução e as porcentagens relativas de desempenho, com base no SoC M2.

Tabela 6: Comparação de Modelos de CPU e Versões de Sistema

SoC	Sistema Operacional	Tempo (segundos)	% relativa
Apple M2	macOS 13.5	0,038	100,00%
Apple A14	iOS 16.1	0,049	128,94%
Apple A10	iOS 15.7.6	0,103	271,05%

Os resultados indicam uma correlação direta entre a arquitetura dos SoCs e o desempenho computacional. O SoC Apple M2 destaca-se com o menor tempo de execução, uma conquista atribuída não só aos seus 4 núcleos de alto desempenho e 4 núcleos de eficiência energética, mas também à sua GPU de 8 núcleos e a Neural Engine de 16 núcleos, que executa 15,8 trilhões de operações por segundo. Essa combinação otimiza o desempenho em tarefas intensivas de aprendizado de máquina, traduzindo-se em uma eficiência significativamente maior em comparação aos seus predecessores.

O SoC A14 Bionic, apesar de apresentar um incremento no tempo de execução em relação ao M2, ainda assim mostra um avanço arquitetural com sua GPU de 4 núcleos e a Neural Engine de 16 núcleos, que realizam 11 trilhões de operações por segundo. Os aceleradores de aprendizado de máquina de segunda geração do A14, conhecidos como blocos AMX, complementam a *Neural Engine*, contribuindo para uma melhora no desempenho de tarefas relacionadas ao aprendizado de máquina em comparação ao A10, mas não alcançando a eficiência do M2.

Por outro lado, o SoC A10 Fusion, o mais antigo dos três, possui uma GPU de 6 núcleos e não conta com uma *Neural Engine*. As limitações da GPU, que apesar de ser 50% mais rápida que a do A9, não consegue igualar o desempenho das arquiteturas mais recentes em tarefas de aprendizado de máquina. A falta de uma *Neural Engine* dedicada no A10 leva a uma dependência maior da GPU para essas tarefas, resultando em um desempenho significativamente inferior, como demonstrado pelo tempo de execução 271,05% maior em comparação ao M2.

A análise dos tempos de execução dos diferentes SoCs da Apple revela que, mesmo com um processador como o A10 Fusion, lançado em 2016, é possível alcançar um desempenho de processamento local satisfatório para o aplicativo em questão. Embora o A10 não disponha de uma *Neural Engine* e sua GPU seja menos eficiente para operações específicas de aprendizado de máquina em comparação com modelos mais novos, o tempo

de execução observado corrobora a viabilidade de processamento local sem depender de soluções baseadas em nuvem, que implicariam em custos adicionais.

Esta conclusão é particularmente relevante para o desenvolvimento de aplicativos que devem operar eficientemente em uma ampla gama de dispositivos, incluindo aqueles com hardware menos recente. A capacidade de executar tarefas computacionalmente intensivas de maneira local, sem a necessidade de recorrer a serviços de computação em nuvem, não apenas economiza custos, mas também melhora a acessibilidade e a privacidade do usuário. Portanto, a manutenção do suporte a modelos de hardware anteriores, como o A10 Fusion, é uma consideração importante no design de aplicativos, permitindo um desempenho robusto e uma experiência de usuário consistente, mesmo em dispositivos de gerações anteriores.

5 CONSIDERAÇOES FINAIS

Este trabalho investigou a identificação e classificação de estilos arquitetônicos no Brasil, utilizando técnicas de aprendizado de máquina e visão computacional. A pesquisa abrangeu a construção de um conjunto de dados diversificado, o desenvolvimento e a avaliação de um modelo de aprendizado de máquina, e a integração do modelo em um aplicativo funcional.

5.1 Conjunto de Dados e Metodologia

A fase de construção do conjunto de dados foi essencial, envolvendo a seleção de imagens representativas de estilos arquitetônicos. A qualidade das imagens foi crucial para o treinamento eficiente do modelo. A metodologia adotada incluiu o uso de redes neurais convolucionais profundas, com foco na arquitetura EfficientNetV2B0, otimizada para dispositivos móveis através do CoreML.

5.2 Análise de Desempenho do Modelo

O modelo foi avaliado utilizando métricas como precisão, sensibilidade e medida-F1. Os resultados indicaram eficácia na identificação de certos estilos arquitetônicos como o modernista e o neogótico com, respectivamente, 100% e 88% de acurácia, enquanto apontaram desafios na classificação de estilos com maior variação como o eclético e o neoclássico, com, respectivamente, 77% e 82% de acurácia. A técnica LIME forneceu *insights* sobre os fundamentos das decisões do modelo, mostrando no caso do Teatro Santa Roza, uma das características mais importantes para classificá-lo como neoclássico foi o frontão.

5.3 Avaliação de Desempenho em Diferentes Hardware

A análise em diferentes SoCs da Apple mostrou que, apesar das variações entre gerações, o modelo manteve eficiência aceitável em dispositivos mais antigos. Este aspecto é importante para a acessibilidade e viabilidade do aplicativo em uma variedade de dispositivos.

5.4 Implicações Práticas e Futuras Direções

A integração do modelo em um aplicativo móvel demonstrou a aplicabilidade prática da pesquisa, fornecendo uma ferramenta para a identificação de estilos arquitetônicos. O estudo contribui para campos como preservação cultural, educação em arquitetura e turismo.

Para futuras pesquisas, recomenda-se expandir o conjunto de dados para abranger mais estilos arquitetônicos e explorar abordagens para melhorar a precisão do modelo, especialmente em estilos com alta variabilidade. Investigar novas arquiteturas de redes neurais e técnicas de otimização pode levar a aprimoramentos na eficiência e eficácia do modelo.

Em suma, este trabalho evidencia o potencial da inteligência artificial na análise e compreensão da arquitetura, oferecendo uma ferramenta útil para o estudo de estilos arquitetônicos e abrindo caminho para inovações futuras na área.

REFERÊNCIAS

[1] CRAGOE, Carol Davidson. **How to read buildings: a crash course in architectural styles**. New York: Rizzoli, 2008.

[2] SEGAWA, Hugo. **Architecture of Brazil: 1900-1990**. Springer Science & Business Media, 2012.

[3] LEONARD, Irving A. **Baroque and Rococo in Latin America**. Durham: Duke University Press, 1969.

[4] BERGDOLL, Barry; COMAS, Carlos Eduardo; LIERNUR, Jorge Francisco; DEL REAL, Patricio. **Latin America in construction: architecture 1955-1980**. New York: Museum of Modern Art, 2015.

[5] DECKKER, Zilah Quezado. **Brazil built: the architecture of the modern movement in Brazil**. Taylor & Francis, 2001.

[6] KUHN, Max; JOHNSON, Kjell. **Applied predictive modeling**. Vol. 26. Springer, 2013.

[7] CAHILL, Bear. **UI Design for iOS App Development: Using SwiftUI**. Springer, 2021.

[8] ZHANG, Luming; SONG, Mingli; LIU, Xiao; SUN, Li; CHEN, Chun; BU, Jiajun. Recognizing architecture styles by hierarchical sparse coding of blocklets. **Information Sciences**, v. 254, p. 141–154, 2014.

[9] ZHAO, Peipei; MIAO, Qiguang; SONG, Jianfeng; QI, Yutao; LIU, Ruyi; GE, Daohui. Architectural style classification based on feature extraction module. **IEEE Access**, v. 6, p. 52598–52606, 2018.

[10] YOSHIMURA, Yuji; CAI, Bill; WANG, Zhoutong; RATTI, Carlo. Deep learning architect: classification for architectural design through the eye of artificial intelligence. **Computational Urban Planning and Management for Smart Cities 16**, p. 249–265, 2019.

[11] MATHIAS, Markus; MARTINOVIC, Andelo; WEISSENBERG, Julien; HAEGLER, Simon; VAN GOOL, Luc. Automatic architectural style recognition. **The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, v. 38, p. 171–176, 2012.

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

[13] KOSTOF, Spiro. A history of architecture: settings and rituals. **New York: Oxford**, 1995.

[14] BRANCO, Bernardo Castello. Arquitetura indígena brasileira: da descoberta aos dias atuais. **Revista de Arqueologia**, v. 7, n. 1, p. 69–85, 1993.

[15] PEDREGOSA, Fabian; VAROQUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand; GRISEL, Olivier; BLONDEL, Mathieu; PRETTENHOFER, Peter; WEISS, Ron; DUBOURG, Vincent and others. Scikit-learn: Machine learning in Python. **the Journal of machine Learning research**, v. 12, p. 2825–2830, 2011.

[16] MARQUES, Oge; MARQUES, Oge. Machine learning with core ml. **Image Processing and Computer Vision in iOS**, p. 29–40, 2020.

[17] VARMA, Jayant; VARMA, Jayant. What Is SwiftUI. **SwiftUI for Absolute Beginners: Program Controls and Views for iPhone, iPad, and Mac Apps**, p. 1–8, 2019.

[18] XU, Zhe; TAO, Dacheng; ZHANG, Ya; WU, Junjie; TSOI, Ah Chung. Architectural style classification using multinomial latent logistic regression. In: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. **Anais eletrônicos...** Springer, 2014.

[19] SHALUNTS, Gayane; HAXHIMUSA, Yll; SABLATNIG, Robert. Architectural Style Classification of Building Facade Windows. In: ISVC (2). **Anais eletrônicos...** 2011.

[20] MASCARENHAS, Sheldon; AGARWAL, Mukul. A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. In: 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON). **Anais eletrônicos...** IEEE, 2021.

[21] DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. **Anais eletrônicos...** IEEE, 2009.

[22] JACOB, Benoit; KLIGYS, Skirmantas; CHEN, Bo; ZHU, Menglong; TANG, Matthew; HOWARD, Andrew; ADAM, Hartwig; KALENICHENKO, Dmitry. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. **Anais eletrônicos...** 2018.

[23] RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why should i trust you?"Explaining the predictions of any classifier. In: PROCEEDINGS OF THE

22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Anais eletrônicos...** p. 1135–1144, 2016.

[24] Christoph Molnar. *Interpretable machine learning*. Lulu.com, 2020.

[25] **Géron, Aurélien**. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.

[26] **Shahoveisi, Fereshteh; Taheri Gorji, Hamed; Shahabi, Seyedmojtaba; Hosseinirad, Seyedali; Markell, Samuel; Vasefi, Fartash**. *Application of image processing and transfer learning for the detection of rust disease*. Scientific Reports, Nature Publishing Group UK London, vol. 13, no. 1, pp. 5133, 2023.

[27] **Duong, Linh T; Nguyen, Phuong T; Di Sipio, Claudio; Di Ruscio, Davide**. *Automated fruit recognition using EfficientNet and MixNet*. Computers and Electronics in Agriculture, Elsevier, vol. 171, pp. 105326, 2020.

[28] **Suzuki, Kenji**. *Artificial neural networks: Architectures and applications*. BoD–Books on Demand, 2013.

[29] **Equipe Keras**. *Aplicações Keras*. Disponível em: https://keras.io/applications. Acesso em: 15 de dezembro de 2023.

[30] Augusto Carlos da Silva Telles. *Atlas dos monumentos históricos e artísticos do Brasil*. 1975.

[31] Mateus Rosada. *Igrejas paulistas da colônia e do império: Arquitetura e ornamentação*. Ph.D. thesis, Universidade de São Paulo, 2016.

[32] TAN, Mingxing; LE, Quoc. Efficientnetv2: Smaller models and faster training. In: International conference on machine learning. Proceedings of Machine Learning Research (PMLR), 2021, p. 10096–10106.

[33] BAETA, Rodrigo Espinha. **O barroco, a arquitetura e a cidade nos séculos XVII e XVIII**. EDUFBA, 2010.

[34] SOUSA, Alberto. **Arquitetura neoclássica brasileira: um reexame**. Pini, 1994.

[35] **Borngässer, Barbara and Klein, Bruno (Hrsg/eds)**. *Neugotik global–kolonial–postkolonial: Gotisierende Sakralarchitektur auf der Iberischen Halbinsel und in Lateinamerika vom 19. bis zum 21. Jahrhundert= Neogótico global–colonial–postcolonial: Arquitectura sagrada neogótica en la Peninsula Ibérica y América Latina del siglo XIX al XXI*. Iberoamericana Vervuert, 2020.

[36] Dias, Pollyanna d'Avila e D'avilla, G. "O século XIX e o neogótico na Arquitetura brasileira: um estudo de caracterização". *Revista Ohun*, 2008, número 4, páginas 100-115.

[37] Fabris, Annateresa. *Arquitetura eclética no Brasil: o cenário da modernização. Anais do Museu Paulista: História e Cultura Material*, volume 1, páginas 131-143, 1993. Editora: SciELO Brasil.

[38] David Thomson. *Renaissance Paris: Architecture and Growth, 1475-1600*. Univ of California Press, 1984.

[39] Alice Zheng. *Evaluating machine learning models: A beginner's guide to key concepts and pitfalls*. O'Reilly Media, 2015.

[40] Oliver Theobald. *Machine Learning for Absolute Beginners: A Plain English Introduction*. Scatterplot Press London, UK, 2017.

[41] **Kelleher, John D. and Mac Namee, Brian and D'Arcy, Aoife**. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT Press, 2020.

[42] Fausett, Laurene V. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. Pearson Education India, 2006.

[43] Nielsen, Michael A. *Neural Networks and Deep Learning*, volume 25. Determination press, San Francisco, CA, USA, 2015.

[44] Aggarwal, Charu C. and others. *Neural Networks and Deep Learning*. Springer, vol. 10, no. 978, pp. 3, Springer, 2018.

[45] Chollet, Francois. *Deep Learning with Python*. Simon and Schuster, 2021.

[46] Pan, Sinno Jialin and Yang, Qiang. *A Survey on Transfer Learning*. IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, 2010. DOI: 10.1109/TKDE.2009.191.

[47] Mingxing Tan e Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Em *International Conference on Machine Learning*, páginas 6105–6114, 2019. PMLR.

[48] Mingxing Tan e Quoc Le. EfficientNetV2: Smaller Models and Faster Training. Em *International Conference on Machine Learning*, páginas 10096–10106, 2021. PMLR.

[49] APPLE MACHINE LEARNING RESEARCH. *Deploying Transformers on the Apple Neural Engine*. In: Apple Machine Learning Research, 2021.

[50] EMMONS, Paul; LOMHOLT, Jane; HENDRIX, John Shannon. The Cultural Role of Architecture: Contemporary and Historical Perspectives. **Routledge**, 2012.

[51] JONES, Denna; ROGERS, Richard; GUMUCHDJIAN, Philip. Architecture: the whole story. **(No Title)**, 2014.

[52] SANTOS, Nadja Ferreira. Interface entre Arquitetura e Arqueologia na Preservação do Patrimônio Cultural Urbano. Pelotas: Universidade Federal de Pelotas, 2009.

[53] MEDINA, Lasansky; MCLAREN, Brian. Architecture and Tourism: Perception, Performance and Place. 2015.

[54] BOUÉ, Laurent. Deep learning for pedestrians: backpropagation in CNNs. arXiv preprint arXiv:1811.11987, 2018.