UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba
Campus de São João da boa vista

Eduardo Bianucci Barcelona

Título do trabalho: Estudo do uso da Transformada de Fourier e imagens sintéticas geradas pelo método de Spot-Noise no treinamento de Redes Neurais Convolucionais para a classificação de texturas

São João da Boa Vista
2026

Eduardo Bianucci Barcelona

Título do trabalho: Estudo do uso da Transformada de Fourier e imagens sintéticas geradas pelo método de Spot-Noise no treinamento de Redes Neurais Convolucionais para a classificação de texturas

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, do Instituto de Ciência e Tecnologia de Sorocaba e do Campus de São João da Boa Vista da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Antonio Cesar Germano Martins

São João da Boa Vista
2026

B242e

Barcelona, Eduardo Bianucci

Estudo do uso da Transformada de Fourier e imagens sintéticas geradas pelo método de Spot-Noise no treinamento de Redes Neurais Convolucionais para a classificação de texturas / Eduardo Bianucci Barcelona. -- São João da Boa Vista, 2026

54 p.

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, São João da Boa Vista

Orientador: Antonio Cesar Germano Martins

1. Reconhecimento visual de textura. 2. Fourier, Transformadas de. 3. Redes neurais (Computação). 4. Aprendizagem profunda (Aprendizado do computador). 5. Ruído. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Impacto potencial desta pesquisa

O trabalho contribui para o avanço da inteligência artificial aplicada ao processamento digital de imagens. A metodologia proposta demonstra como o domínio de Fourier e o ruído estocástico refinam o aprendizado de CNNs, gerando modelos mais robustos e eficientes para a classificação de texturas complexas em ambientes acadêmicos e industriais.

Potential impact of this research

This work contributes to the advancement of artificial intelligence applied to digital image processing. The proposed methodology demonstrates how the Fourier domain and stochastic noise refine CNN learning, generating more robust and efficient models for complex texture classification in both academic and industrial environments.

EDUARDO BIANUCCI BARCELONA

ESTUDO DO USO DA TRANSFORMADA DE FOURIER E IMAGENS SINTÉTICAS GERADAS PELO MÉTODO DE SPOT-NOISE NO TREINAMENTO DE REDES NEURAIS CONVOLUCIONAIS PARA A CLASSIFICAÇÃO DE TEXTURAS

Dissertação apresentada à Universidade Estadual Paulista (UNESP), Faculdade de Engenharia — Campus de Sorocaba, em Sorocaba, para obtenção do título de Mestre em Engenharia Elétrica, junto ao Programa de Pós-Graduação em Engenharia Elétrica.

Área de Concentração: Sistemas Eletrônicos
Data da Defesa: 06/04/2026

Banca Examinadora:

gov.br Documento assinado digitalmente
ANTONIO CESAR GERMANO MARTINS
Data: 03/06/2026 12:22:14-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Antonio Cesar Germano Martins
Departamento de Engenharia Ambiental / UNESP / ICTS

gov.br Documento assinado digitalmente
WALDEMAR BONVENTI JUNIOR
Data: 04/06/2026 19:00:03-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Waldemar Bonventi Júnior
Departamento de Engenharia / Faculdade de Tecnologia de Sorocaba – FATEC

gov.br Documento assinado digitalmente
LEOPOLDO ANDRE DUTRA LUSQUINO FILHO
Data: 03/06/2026 10:55:12-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Leopoldo André Dutra Lusquino Filho
Departamento de Engenharia de Controle e Automação / UNESP / Câmpus de Sorocaba - ICTS

“If I have seen further, it is by standing on the shoulders of giants”

Isaac Newton

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela força, sabedoria e por iluminar meu caminho em todos os momentos desta jornada.

À minha família pelo apoio incondicional e por serem a fonte constante de inspiração e incentivo que me permitiu chegar até aqui.

Ao meu orientador, Dr. Antônio Cesar Germano Martins, pela paciência, pelas valiosas orientações e por compartilhar seu vasto conhecimento, sendo peça fundamental no desenvolvimento deste trabalho.

Por fim, agradeço a todos que, de forma direta ou indireta, me apoiaram e torceram por mim durante toda essa caminhada.

RESUMO

A análise de texturas em imagens digitais desempenha um papel importante em diversas aplicações, como inspeção visual automatizada, reconhecimento de padrões e classificação de superfícies. No entanto, a grande variabilidade estrutural das texturas e a complexidade das informações presentes nas imagens podem dificultar o processo de aprendizado de modelos baseados em aprendizado profundo. Nesse contexto, este trabalho investiga o impacto da representação das imagens no domínio de Fourier em comparação com o domínio espacial para o processo de aprendizado de Redes Neurais Convolucionais (CNNs) na análise de texturas. Para permitir uma avaliação controlada dessa hipótese, foi utilizado o método Spot Noise para a geração de imagens sintéticas de textura, possibilitando controlar características estruturais dos padrões gerados, como forma e dimensão dos elementos que compõem as texturas. A partir desse conjunto de dados, foram conduzidos experimentos comparando o desempenho de uma CNN treinada com imagens no domínio da frequência e outra treinada com imagens no domínio espacial. Os resultados obtidos indicam que a utilização da Transformada de Fourier favorece o processo de aprendizado da rede, proporcionando melhor desempenho e maior capacidade de generalização, inclusive para padrões intermediários não utilizados durante o treinamento. Além disso, o uso de texturas sintéticas geradas por Spot Noise mostrou-se adequado para a realização de experimentos controlados voltados ao estudo de métodos de aprendizado profundo aplicados à análise de texturas.

Palavras–chave: análise de texturas; spot noise; transformada de fourier; redes neurais convolucionais; aprendizado profundo.

ABSTRACT

Texture analysis in digital images plays an important role in several applications, such as automated visual inspection, pattern recognition, and surface classification. However, the high structural variability of textures and the complexity of the information present in images may hinder the learning process of deep learning–based models. In this context, this work investigates the impact of image representation in the Fourier domain compared to the spatial domain on the learning process of Convolutional Neural Networks (CNNs) for texture analysis. To enable a controlled evaluation of this hypothesis, the Spot Noise method was used to generate synthetic texture images, allowing the control of structural characteristics of the generated patterns, such as the shape and size of the elements composing the textures. Based on this dataset, experiments were conducted comparing the performance of a CNN trained with images in the frequency domain and another trained with images in the spatial domain. The obtained results indicate that the use of the Fourier Transform favors the network learning process, providing improved performance and greater generalization capability, including for intermediate patterns not used during training. Furthermore, the use of synthetic textures generated by Spot Noise proved to be suitable for conducting controlled experiments aimed at studying deep learning methods applied to texture analysis.

Keywords: texture analysis; spot noise; fourier transform; convolutional neural network, deep learning

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

GLCM	Matrizes de Co-ocorrência de Nível de Cinza (Gray Level Co-occurrence Matrix)
CVE	Coeficiente de Variação Espacial
LBP	Padrão Binário Local (Local Binary Pattern)
RGB	Red, Green and Blue
ALOT	Biblioteca de Texturas de Amsterdam (Amsterdam Library of Textures)
Outex	Biblioteca de Texturas Outex
VDVI	Índice de Vegetação por Diferença de Banda Visível (Visible-band Difference Vegetation Index)
DBC	Contagem Diferencial de Caixas (Differential Box-Counting)
SVM	Máquina de Vetores de Suporte (Support Vector Machine)
EWT	Transformada Wavelet Empírica (Empirical Wavelet Transform)
DCT	Transformada Discreta de Cosseno (Discrete Cosine Transform)
DWT	Transformada Wavelet Discreta (Discrete Wavelet Transform)
SFDI	Imagem no Domínio de Frequência Espacial (Spatial Frequency Domain Imaging)
RNA	Rede Neural Artificial
CT	Tomografia Computadorizada (Computed Tomography)
CNNs	Redes Neurais Convolucionais (Convolutional Neural Networks)
GANs	Redes Generativas Adversariais (Generative Adversarial Networks)
ViTs	Transformadores de Visão (Vision Transformers)
MSE	Erro Quadrático Médio (Mean Squared Error)
MAE	Erro Médio Absoluto (Mean Absolute Error)

SUMÁRIO

1 INTRODUÇÃO

A análise computacional de imagens digitais tem encontrado aplicações em diversos setores da sociedade, impactando desde sistemas de segurança e controle de acesso até diagnósticos médicos e inspeção de qualidade industrial. A automação da análise visual não apenas otimiza processos, mas também oferece agilidade, escalabilidade e confiabilidade. Dentre as múltiplas características que compõem uma imagem, a textura se destaca como um descritor importante, fornecendo informações valiosas sobre a estrutura e a organização espacial dos objetos representados.

A textura em imagens digitais pode ser utilizada em uma ampla gama de aplicações, como reconhecimento de padrões, inspeção visual automatizada, classificação de superfícies e segmentação de áreas heterogêneas. Contudo, essa diversidade de padrões, que variam desde estruturas regulares e repetitivas até arranjos aleatórios e não homogêneos, torna a análise de texturas um problema complexo e desafiador de se implementar.

Com o avanço exponencial das técnicas de aprendizado de máquina, novas e promissoras possibilidades têm surgido para extrair, interpretar e classificar informações texturais de forma cada vez mais precisa e automatizada. Um exemplo são as redes neurais convolucionais (CNNs), que possuem a capacidade de aprender automaticamente características relevantes de forma hierárquica.

No entanto, o treinamento eficaz desses modelos de aprendizado profundo, depende de fortemente da quantidade, qualidade e da variabilidade estrutural das imagens utilizadas no treinamento, especialmente em aplicações que exigem boa capacidade de generalização.

Diante desse cenário, este trabalho propõe uma abordagem inovadora para o processo de treinamento de uma CNN na análise de texturas. Para facilitar o treinamento e contornar a limitação de dados rotulados, a metodologia adota a geração de imagens sintéticas de textura utilizando o método Spot Noise, que permite criar texturas estocásticas com controle local sobre os elementos visuais (van Wijk, 1991).

A partir desse conjunto sintético, o processo é conduzido no domínio de Fourier, explorando os padrões de distribuição das frequências espaciais das imagens. Essa estratégia busca investigar se a reorganização espectral promovida pela

Transformada de Fourier pode facilitar o processo de aprendizado das CNNs, tornando mais explícitas as estruturas texturais presentes nas imagens.

A combinação da geração de imagens por Spot Noise, a análise no domínio de Fourier e as características das CNNs mostrou-se uma solução eficiente na classificação de imagens de acordo com as texturas presentes. Este trabalho está organizado da seguinte forma: a Seção 2 apresenta a revisão da literatura, a Seção 3 descreve a metodologia adotada, a Seção 4 apresenta os resultados e discussões, e a Seção 5 apresenta as conclusões do estudo.

2 REVISÃO DA LITERATURA

2.1 Métodos Clássicos de Análise de Textura

A análise de textura em imagens tem suas raízes em abordagens estatísticas e estruturais que se tornaram a base de métodos subsequentes.

O artigo de Haralick *et al.* (1973) é um marco importante nesse sentido. Nele, os autores introduziram o conceito de matrizes de co-ocorrência de nível de cinza (GLCM) que busca analisar a textura das imagens com uma abordagem estatística, apresentando descritores como contraste, homogeneidade, entropia, correlação entre outros. Esses descritores são amplamente utilizados e são eficazes para classificação de imagens, sendo empregados em tarefas como reconhecimento de padrões e segmentação.

Schwartz e Pedrini (2003) expandem essa abordagem ao aplicar as matrizes de co-ocorrência em uma análise multiescalar, o que permite capturar padrões de textura em diferentes resoluções e escalas, tornando a análise mais robusta e sensível às variações espaciais.

Conci *et al.* (2006) propõe o Coeficiente de Variação Espacial (CVE) que é calculado com base em medidas estatísticas de posição (média) e dispersão (desvio padrão) dos pixels, permitindo uma análise detalhada das variações espaciais de intensidade de cor nas imagens. A metodologia foi testada em imagens monocromáticas, coloridas e multiespectrais, com resultados que demonstram sua eficácia na segmentação de diferentes classes de texturas. O estudo também destaca a simplicidade e o baixo custo computacional do método, além de sugerir que ele pode ser adaptado para ampliar a gama de bandas espectrais analisadas, tornando-o aplicável em uma ampla variedade de áreas, como processamento de imagens médicas e de satélites.

Hosny *et al.* (2021) desenvolveram um método que une características locais e globais para reconhecimento de texturas coloridas. As características locais foram extraídas com o descritor *Local Binary Pattern* (LBP), enquanto as globais foram obtidas a partir de momentos ortogonais de Chebyshev. Propostos por Mukundan *et al.* (2000), os momentos de Chebyshev são utilizados para descrever formas geométricas, aplicados em cada canal de cor (RGB). A técnica mostrou-se robusta a transformações geométricas (rotação, escala e translação) e manteve a fidelidade das informações cromáticas. Avaliações em conjuntos de dados desafiadores, como

Outex criado por Ojala *et al.* (2002) e *ALOT* publicado por Burghouts *et al.* (2009), mostraram que essa abordagem é competitiva e superou outros métodos clássicos de extração de características.

Zhou *et al.* (2021) propuseram uma abordagem híbrida para o mapeamento de vegetação em áreas desérticas, utilizando imagens RGB capturadas por drones. A técnica combinou o uso do índice de vegetação por diferença de banda visível (VDVI) com descritores de textura baseados na matriz de co-ocorrência, aplicando a classificação com *Random Forest*. Essa integração entre informações espectrais e texturais resultou em um aumento expressivo da acurácia (de 76,69% para 84,19%), evidenciando a importância das características texturais em contextos de baixa separabilidade espectral. O estudo reforça a aplicabilidade prática da análise de textura em ambientes naturais com dados captados por sensores aéreos.

2.2 Estratégias de Processamento de Imagens Baseadas em Texturas

Amorim e Filho (2017) utilizaram análise de textura para avaliar lacunaridade em imagens urbanas captadas por satélite, com base no método de Contagem Diferencial de Caixas, proposto por Sarkar *et al.* (1994), permitindo a avaliação de espaços vazios na morfologia urbana.

Souza e Pereira (2021) exploram o uso de medidas fractais para a classificação de texturas em imagens, utilizando a dimensão fractal e a lacunaridade para descrever características locais nas imagens utilizadas e em seguida aplica algoritmos de aprendizado de máquina para classificá-las. Esta abordagem é capaz de capturar detalhes de texturas em diferentes escalas e, além disso, pode ser mais robusta a ruídos uma vez que foca em características geométricas gerais e não é tão dependente de características pixel a pixel.

Tiwari *et al.* (2023) investigaram a robustez de um método baseado em segmentação e medidas fractais para a classificação de texturas em imagens com ruído gaussiano. A abordagem segmenta a imagem em blocos e extrai medidas de dimensão fractal local, que são utilizadas em classificadores como *Random Forest* e *Support Vector Machine* (SVM). Os resultados mostraram alta acurácia mesmo em níveis elevados de ruído, evidenciando o potencial das medidas fractais segmentadas para representar texturas de forma eficaz em ambientes ruidosos.

Huang *et al.* (2023) apresentaram uma revisão detalhada sobre segmentação de texturas aplicando transformadas *wavelet* (Mallat, 1989), destacando o uso das

wavelets empíricas (EWT), que extraem modos harmônicos do espectro de Fourier de forma adaptativa. O estudo inclui uma decomposição "*cartoon* + textura" como pré-processamento (que evidencia as características de textura da imagem) e mostraram o desempenho superior às *wavelets* tradicionais.

Yesilli *et al.* (2022) propuseram um método automatizado para análise de textura de superfícies, utilizando transformadas discretas de cosseno (DCT) e *wavelet* (DWT). O foco do estudo foi eliminar a necessidade de limiares definidos manualmente, empregando critérios baseados em energia e entropia para extrair os componentes texturais. Com classificadores como SVM e *Random Forest*, o modelo alcançou acurácia acima de 95%, superando técnicas convencionais de filtragem. A abordagem se destaca pela automatização e por sua aplicabilidade em contextos industriais, onde a inspeção visual precisa ser rápida e confiável.

Esses estudos se baseiam na classificação e análise de texturas em imagens por meio de modelos matemáticos ou transformações previamente estabelecidas para extrair atributos relevantes da imagem. Em contraste, métodos de IA, especialmente *deep learning*, dispensam em grande medida essa predefinição, já que os próprios modelos constroem automaticamente os descritores mais relevantes a partir dos dados.

2.2.1 Métodos de Análise no Domínio da Frequência

A análise de imagens no domínio da frequência é uma abordagem que permite a extração de informações sobre padrões e periodicidades que não são evidentes no domínio espacial, como afirmam Gonzalez e Woods (2018). O uso da transformada de Fourier, por exemplo, decompõe a imagem em suas componentes de frequência, onde altas frequências representam detalhes finos e arestas, e baixas frequências estão associadas a áreas mais uniformes.

Matematicamente, imagens digitais podem ser representadas como funções bidimensionais no domínio espacial, nas quais cada posição $[x, y]$ corresponde à intensidade de um pixel. A Transformada Discreta de Fourier bidimensional permite converter essa representação para o domínio da frequência, evidenciando a distribuição das componentes espectrais presentes na imagem.

A Transformada Discreta de Fourier bidimensional de uma imagem $t[x, y]$ pode ser expressa pela Equação 1 onde $[x, y]$ representam coordenadas espaciais da imagem e $[u, v]$ representam coordenadas no domínio da frequência.

Equação 1 — Transformada Discreta de Fourier Bidimensional

$$T[u,v] = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} t[x,y]e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \quad (1)$$

Fonte: Adaptado de Gonzalez e Woods (2018)

No artigo de Júnior *et al.* (2021) essa análise é utilizada para determinar propriedades ópticas de tecidos, projetando padrões de luz com frequências espaciais controladas para analisar a superfície do material. Essa metodologia, que simula o processo de decomposição da transformada de Fourier no momento da aquisição, reforça a ideia de que a análise da resposta de um material a diferentes frequências espaciais é uma maneira eficaz de caracterizá-lo.

Além disso, uma propriedade importante relacionada à Transformada de Fourier é o teorema da convolução, que estabelece que a convolução de dois sinais no domínio espacial é equivalente à multiplicação de suas transformadas no domínio da frequência. Considerando dois sinais bidimensionais $w(x,y)$ e $s(x,y)$, a convolução espacial pode ser representada pela Equação 2.

Equação 2 — Convolução no Domínio Espacial

$$t(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(m,n)\ \cdot s(x-m,y-n)dm\,dn \quad (2)$$

Fonte: Adaptado de Gonzalez e Woods (2018)

No entanto, como as imagens digitais são sinais discretos, a implementação prática em sistemas computacionais utiliza a versão discreta da convolução. Para uma imagem de dimensões $M \times N$, a convolução discreta é dada pela Equação 3.

Equação 3 — Convolução Discreta para dados bidimensionais

$$t(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} w[m,n] \cdot s[x-m, y-n] \quad (3)$$

Fonte: Adaptado de Gonzalez e Woods (2018)

Ao aplicar a Transformada de Fourier $F[\,]$ à convolução espacial, obtém-se a relação apresentada na Equação 4, onde $T[u,v]$, $W[u,v]$ e $S[u,v]$ são respectivamente as Transformadas de Fourier de $t(x,y)$, $w(x,y)$ e $s(x,y)$ e $*$ representa a operação de convolução.

Equação 4 — Representação da Convolução no Domínio da Frequência

$$F[w(x,y) * s(x,y)] = W[u,v] \bullet S[u,v] \quad (4)$$

Fonte: Adaptado de Gonzalez e Woods (2018)

Essa propriedade é particularmente relevante para a geração de texturas, pois permite interpretar a formação de padrões como o resultado da multiplicação da Transformada de Fourier de duas imagens, sendo a base do processo de geração de textura pelo método de Spot Noise (Wijk,1991), a ser detalhado a seguir, e que estabelece uma base teórica para a caracterização de padrões texturais.

2.2.2 Geração Sintética de Textura

O entendimento de modelos de geração sintética de textura permite a criação de métodos para a análise de textura. Um desses modelos é o *Spot Noise*, desenvolvido por Wijk (1991) que opera a partir de um princípio simples: a técnica gera uma textura a partir da convolução de *spots*, ou figuras elementares, com uma matriz de ruído aleatório. O formato desse *spot* define a aparência final da textura.

Martins, Simões e Prado (2007) propuseram uma abordagem baseada em imagens geradas pelo método de *Spot Noise* para treinar Redes Neurais Artificiais (RNA) na tarefa de classificação de padrões de cereais. As imagens foram convertidas para o domínio da frequência, por meio da transformada de Fourier, e divididas em 8

regiões cujos primeiros momentos foram utilizados como entrada para uma RNA. Os resultados do estudo evidenciaram que a aplicação da análise espectral permitiu capturar características dos spots gerativos da textura. Essa metodologia mostra que a combinação entre Spot Noise e análise de frequência pode ser útil na modelagem de problemas práticos de classificação visual. No entanto, nesta abordagem, os descritores foram previamente definidos e os seus valores apresentados para a RNA.

2.3 Redes Neurais e Abordagens de Aprendizado Profundo

2.3.1 Estrutura e Operações em Redes Neurais Convolucionais

As redes neurais convolucionais constituem uma arquitetura especializada para o processamento de dados com estrutura espacial, como imagens. Diferentemente das redes convencionais, as CNNs exploram a proximidade entre pixels por meio de operações locais, extraindo características espaciais hierárquicas ao longo de suas camadas.

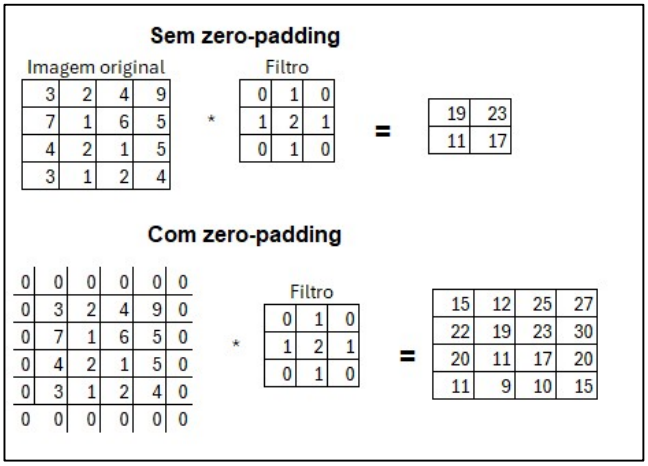
A principal operação que define esse tipo de rede se baseia na operação de convolução, na qual pequenos filtros (ou *kernels*) percorrem a imagem original aplicando multiplicações e somas locais. Cada filtro aprende a detectar padrões específicos, como bordas horizontais ou outras características presentes, e gera um mapa que representa a presença desses padrões em diferentes regiões da imagem. A profundidade da rede permite que os filtros das primeiras camadas detectem estruturas simples, enquanto os filtros das camadas posteriores aprendem características mais abstratas e complexas.

Ao passar por uma camada convolucional, a imagem sofre uma redução dimensional a depender do tamanho do filtro utilizado e do deslocamento (*stride*) definido. A redução dimensional acontece porque o filtro, ao se mover pela imagem, gera uma nova matriz de saída que é menor que a entrada.

Segundo Géron (2021, p. 347), "Para que uma camada tenha a mesma altura e largura da camada anterior, é comum acrescentar zeros ao redor das entradas, [...] Esse procedimento é conhecido como *zero-padding* (completar com zeros)."

A Figura 1 exibe uma operação de convolução com e sem esse procedimento.

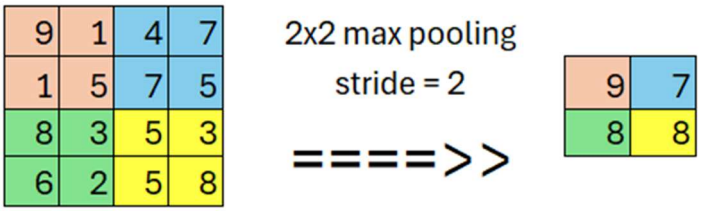
Figura 1 — Convolução e Zero-Padding



Fonte: do Autor. 2024

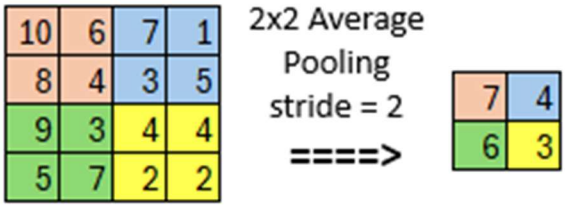
Para complementar as camadas convolucionais, são empregadas as camadas de pooling que, conforme descrito por Nielsen (2015), a principal função é reduzir a dimensionalidade espacial dos mapas de ativação, o que contribui para a diminuição da quantidade de parâmetros e do custo computacional, além de conferir maior robustez ao modelo contra pequenas variações espaciais. A forma de max pooling considera o valor máximo dentro de uma janela definida pelo tamanho do passo que será dado entre cada uma delas, preservando apenas a informação de maior valor daquela região, conforme ilustrado na Figura 2. Já o average pooling preserva o valor médio daquela matriz (Figura 3).

Figura 2 — Max Pooling



Fonte: do Autor. 2024

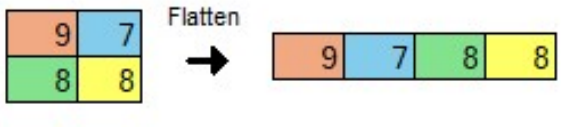
Figura 3 — Average Pooling



Fonte: do Autor. 2024

Outra operação de reestruturação dos dados é o achatamento (*flatten*), que transforma os mapas bidimensionais gerados pelas convoluções e *pooling* em um vetor unidimensional (ver Figura 4).

Figura 4 — Flatten



Fonte: do Autor. 2024

Esse vetor serve como entrada para as camadas densas (ou totalmente conectadas), que são responsáveis pela parte final do processamento, geralmente dedicada à tomada de decisão, seja ela uma classificação, uma regressão ou outra tarefa supervisionada.

A sequência de camadas (convolução, *pooling*, achatamento e camadas densas) constitui a arquitetura padrão das CNNs. Essa estrutura hierárquica capacita os modelos a extrair e aprender características visuais de forma progressiva, permitindo-lhes identificar padrões com alta precisão, mesmo em conjuntos de dados complexos e ruidosos.

Uma técnica, que serve para reduzir problemas de sobreajuste (*overfitting*) em CNNs, é a normalização em lotes (*Batch Normalization*). Proposta por Ioffe e Szegedy (2015) essa é uma técnica de regularização e otimização amplamente utilizada em redes neurais profundas, introduzida para acelerar e estabilizar o processo de treinamento, operando a normalização das entradas de cada camada durante o treinamento, reajustando seus valores para que tenham uma média próxima de zero e um desvio padrão próximo de um. Isso reduz o problema de

mudança interna de covariância, em que a distribuição das entradas para cada camada muda a cada mini lote de dados. Ao manter as distribuições das entradas estáveis, a normalização em lotes permite que a rede neural seja treinada com taxas de aprendizado mais altas e torna o modelo menos sensível à inicialização de pesos, contribuindo para uma convergência mais rápida e, em alguns casos, para a redução do overfitting.

Outra prática que ajuda a minimizar as chances da rede sofrer *overfitting* é a regularização L2. Abordado por Goodfellow *et al.* (2016), é derivada da regularização de Tikhonov (1977), a regularização L2 adiciona à função de perda um termo proporcional ao quadrado da magnitude dos pesos da rede, penalizando valores elevados e incentivando soluções mais simples.

2.3.2 Aplicações Baseadas em Redes Neurais na Análise de Texturas

Wang *et al.* (2022) usaram redes neurais convolucionais para inspecionar superfícies metálicas pós-processo de esmerilhamento, com o objetivo de classificar automaticamente a rugosidade, o tipo e o desgaste da lixa utilizada. Aplicando o aprendizado por transferência, o modelo obteve taxas de acurácia superiores a 90% em todas as tarefas. A pesquisa mostrou que é possível extrair, por meio da textura visual, informações sobre propriedades geométricas da superfície, validando o uso de CNNs em aplicações industriais que exigem precisão e eficiência em ambientes com dados limitados.

Abdallah *et al.* (2023) propuseram uma abordagem baseada na arquitetura ResNet-50 para classificar imagens de carne bovina em condições saudáveis ou rançosas, utilizando análise de textura de superfície. Como o conjunto inicial de dados era pequeno (18 imagens), utilizaram redes generativas adversariais (GANs) para aumentar artificialmente o número de amostras, totalizando 180 imagens. O modelo alcançou acurácia de 96,03% no treinamento e mostrou-se robusto mesmo com conjuntos limitados, validando o uso de CNNs e data augmentation em contextos industriais com dados escassos.

Liu e Aldrich (2022) analisaram a eficácia de diferentes métodos para a caracterização de texturas em materiais, comparando técnicas clássicas como LBP, GLCM com CNNs modernas (AlexNet, VGG19, ResNet50, GoogLeNet e MobileNetV2). Os testes envolveram imagens reais de aço carbono e simulações baseadas em diagramas de Voronoi. As CNNs, especialmente com transfer learning,

superaram as abordagens tradicionais, com destaque para MobileNetV2 e GoogLeNet, que apresentaram desempenho elevado com baixo custo computacional. Para lidar com o problema de escassez de amostras de treinamento para a CNN, os autores utilizaram técnicas de aumento de dados baseadas (*data augmentation*), aplicando operações de rotação, deslocamento, inclinação e espelhamento horizontal nas imagens originais. O estudo reforça o uso de redes neurais convolucionais em ambientes industriais com restrições de dados.

Huber *et al.* (2021) analisaram a capacidade de generalização de uma CNN treinada para redução de ruído em imagens de tomografia computadorizada (CT). A rede foi testada com amostras que apresentavam variações na resolução espacial e na densidade espectral do ruído, decorrentes de diferentes protocolos de aquisição. Os resultados mostraram que mesmo pequenas alterações nos dados de entrada podem afetar significativamente o desempenho da rede, indicando que modelos treinados com dados específicos podem ser sensíveis a variações não previstas. O estudo destaca a importância da diversidade no treinamento e do uso de técnicas de adaptação para garantir robustez, sendo relevante para projetos que pretendem aplicar redes treinadas com dados sintéticos em ambientes reais.

Scabini *et al.* (2024) realizaram uma investigação comparativa abrangente sobre o uso de *Vision Transformers* (ViTs) na análise de texturas, avaliando 21 arquiteturas distintas e comparando-as com redes neurais convolucionais e métodos tradicionais como GLCM e LBP. Os ViTs apresentaram desempenho superior em diversos cenários, especialmente quando utilizados como extratores de características, mesmo sem fine-tuning. No entanto, os autores destacam que CNNs ainda se mostram vantajosas em aplicações com recursos computacionais limitados. Além disso, embora estas exijam grandes volumes de dados para um bom desempenho, demandam um número de amostras consideravelmente menor quando comparadas aos ViTs.

Uma das principais limitações dessas abordagens é a forte dependência de bases de dados amplas e diversificadas, necessárias para garantir um bom desempenho. A falta de dados representativos pode levar a problemas de sobreajuste (overfitting), reduzindo a capacidade de generalização dos modelos em cenários reais. Nesse contexto, estratégias baseadas na geração de texturas sintéticas tornam-se particularmente relevantes, pois permitem controlar propriedades estruturais das

imagens e gerar conjuntos de dados extensos e bem caracterizados, adequados para o treinamento de modelos de aprendizado profundo.

2.3.3 Métricas de Desempenho

Além da estrutura arquitetural da rede, o processo de treinamento das Redes Neurais Convolucionais depende da definição de uma função de perda responsável por quantificar o erro entre as predições do modelo e os valores de referência presentes no conjunto de dados. Essa função orienta o processo de otimização durante o treinamento, permitindo o ajuste iterativo dos parâmetros da rede por meio de algoritmos de descida do gradiente como discutido por Goodfellow *et al.* (2016).

No contexto de problemas de regressão, diferentes funções de perda podem ser utilizadas para medir a discrepância entre o valor real y e o valor estimado pela rede $\hat{y}$. Entre elas, destaca-se a função de perda Huber (*Huber Loss*), proposta originalmente por Huber (1964), que combina propriedades do erro quadrático médio (MSE) e do erro absoluto médio (MAE). Essa função apresenta comportamento quadrático para erros pequenos e crescimento linear para erros maiores, reduzindo assim a influência de valores discrepantes no processo de treinamento.

Matematicamente, a função de perda Huber pode ser definida pela Equação 5.

Equação 5 — Função de perda Huber (Huber Loss)

$$L_\delta(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2, & se\ |y-\hat{y}| \le \delta \\ \delta\left(|y-\hat{y}| - \frac{1}{2}\delta\right), & se\ |y-\hat{y}| > \delta \end{cases} \quad (5)$$

Adaptado de Huber (1964)

Nessa equação, y representa o valor de referência, $\hat{y}$ o valor estimado pelo modelo e δ um parâmetro que define o ponto de transição entre o comportamento quadrático e linear da função.

Essa característica torna a Huber Loss particularmente adequada em cenários nos quais se deseja combinar estabilidade numérica durante o treinamento com maior robustez a possíveis valores extremos presentes nos dados.

Outra métrica amplamente empregada, quando se aplica redes neurais em tarefas de classificação é a matriz de confusão, introduzida por Congalton (1991)

como uma proposta de uma técnica de validação consistente para métodos de classificação estatísticos. Essa ferramenta permite derivar três métricas essenciais para uma análise completa do desempenho de uma classificação, sendo elas: Acurácia Global (AG) presente na Equação 6, Acurácia do Produtor (AP) ou Revocação (Equação 7) e Acurácia do Usuário (AU) ou Precisão (Equação 8).

Equação 6 — Acurácia Global

$$AG = \frac{VP + VN}{VP + VN + FP + FN} \quad (6)$$

Fonte: Adaptado de Congalton (1991)

Equação 7 — Acurácia do Produtor

$$AP = \frac{VP}{VP + FN} \quad (7)$$

Fonte: Adaptado de Congalton (1991)

Equação 8 — Acurácia do Usuário

$$AU = \frac{VP}{VP + FP} \quad (8)$$

Fonte: Adaptado de Congalton (1991)

As métricas apresentadas nas Equação 6, 7 e 8 fundamentam-se na relação entre os valores reais e as predições do modelo. Nesse contexto, os Verdadeiros Positivos (VP) e Verdadeiros Negativos (VN) representam os acertos da classificação, ocorrendo quando o algoritmo identifica corretamente a presença ou a ausência de uma classe, respectivamente. Em contrapartida, os erros são divididos em Falsos Positivos (FP), que caracterizam a atribuição incorreta de uma classe a uma amostra não pertencente a ela e Falsos Negativos (FN), que ocorrem quando o modelo omite uma amostra que deveria ter sido classificada na categoria em questão.

Pantaleão (2009) usa essas métricas para fazer uma comparação de diferentes métodos de classificação em imagens do satélite ALOS.

3 METODOLOGIA

Este capítulo descreve os procedimentos metodológicos adotados para investigar o impacto da representação espectral na aprendizagem de padrões texturais por Redes Neurais Convolucionais. Para isso, foi adotada uma abordagem experimental baseada na geração de imagens sintéticas de textura por meio do modelo de Spot Noise e na análise dessas imagens tanto no domínio espacial quanto no domínio da frequência.

Antes da definição da arquitetura final utilizada nos experimentos principais, foram realizados testes preliminares para avaliar diferentes configurações de treinamento da rede neural convolucional. Esses experimentos iniciais envolveram a investigação de uma arquitetura mais profunda, a variação no número de classes (utilizando inicialmente um conjunto de quatro classes) e a quantidade de amostras disponíveis, além do ajuste de diferentes funções de ativação e hiperparâmetros associados ao modelo.

Com base nessas observações, optou-se pela adoção de uma arquitetura convolucional mais compacta e pelo aumento do número de classes utilizadas no treinamento, buscando favorecer a capacidade de generalização do modelo e melhorar a representação contínua das propriedades estruturais das texturas.

A partir dessa reformulação, foram conduzidos os experimentos principais do trabalho, os quais buscaram avaliar o impacto da representação espectral na capacidade de aprendizado da rede neural. O experimento 1 foi realizado com imagens no domínio da frequência, enquanto o experimento 2, com imagens no domínio espacial.

As texturas utilizadas nos experimentos foram geradas a partir da convolução de ruído aleatório com diferentes *spots*, permitindo controlar diretamente as propriedades estruturais da textura resultante. Foram utilizados *spots* com formatos geométricos de círculos e de quadrados. Enquanto o primeiro gera texturas mais suaves, o segundo apresenta uma estrutura axadrezada.

Além da variação de formato, também foram consideradas variações no tamanho dos *spots*, para avaliar a capacidade de generalização do aprendizado da CNN. Assim, foram utilizadas texturas, por exemplo, geradas com *spots* de círculos de diâmetro 10, 20 e 30 pixels para o treinamento e verificada a generalização com texturas geradas com *spots* de círculos de diâmetro 12, 17 e 24 pixels.

É importante salientar que o conceito de generalização está sendo utilizado com uma interpretação mais abrangente. Não se trata apenas de verificar se a rede consegue classificar corretamente imagens de textura geradas com os mesmos *spots*, mas sim se consegue ordenar as diferentes texturas de acordo com os tamanhos dos *spots* de mesmo formato.

Para os dois experimentos foram preservadas todas as características da arquitetura da CNN e do procedimento de treinamento, permitindo avaliar diretamente o impacto da representação espectral no desempenho do modelo.

Como foram usadas imagens sintetizadas, não se tem a limitação do números de amostras para treinamento, validação e teste, uma vez que ao se utilizar imagens com ruídos aleatórios diferentes e realizar a convolução com o mesmo *spot*, as características da textura são preservadas pois o resultado da convolução de dois sinais no domínio espacial é o equivalente a multiplicação das transformadas de Fourier desses sinais no domínio de frequência e a transformada de Fourier de ruído aleatório se aproxima de uma constante para qualquer valor de frequência, significando que a transformada de Fourier da textura apresentará características do *spot*.

A geração das imagens e todas as operações matemáticas aplicadas foram realizadas com o software Octave e as implementações das Redes Neurais Convolucionais foram desenvolvidas em Python, utilizando o ambiente Google Colaboratory.

A divisão dos dados em conjuntos de treinamento e validação foi realizada na proporção de 80/20, respectivamente. Para a otimização da rede, foi utilizado o algoritmo Adam, com taxa de aprendizado inicial de 2×10^{-4}. Além disso, o tamanho de lote (*batch size*) adotado foi de 32 amostras.

O treinamento foi conduzido com a função de perda *Huber Loss* com $\delta = 1$. Como métrica de desempenho foi utilizada a Média do Erro Absoluto (MAE), que permite avaliar diretamente a proximidade entre os valores preditos pela rede e os valores de referência associados às texturas.

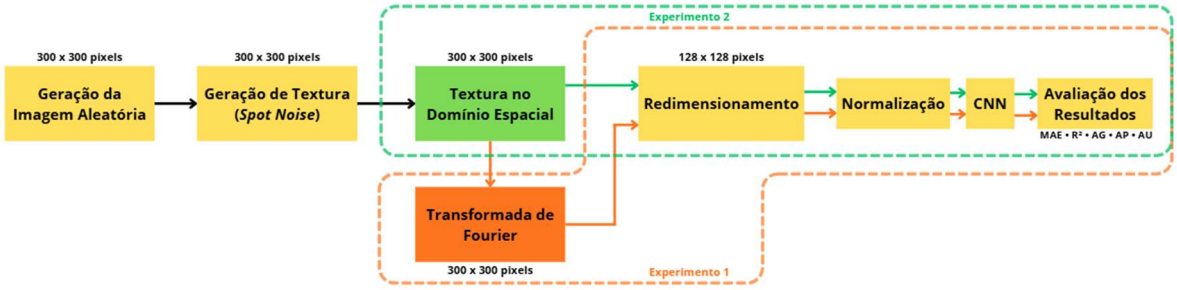
A abordagem adotada neste trabalho pode ser caracterizada como uma classificação baseada em regressão. Em vez de utilizar uma saída categórica convencional, como a função *softmax*, a rede neural foi treinada para estimar um valor contínuo associado às propriedades estruturais da textura. Nesse contexto, o sinal do valor predito representa o formato do *spot*, enquanto a magnitude do valor estimado

está associada ao seu tamanho. O objetivo dessa estratégia é permitir que o modelo seja capaz de classificar corretamente texturas não observadas durante o treinamento, incluindo aquelas geradas com tamanhos diferentes de *spots*. Dessa forma, o modelo aprende uma estrutura contínua no espaço de características texturais, caracterizando um processo de regressão.

Para a avaliação dos resultados foram utilizados gráficos de dispersão e matrizes de confusão, das quais foram extraídas as seguintes métricas: a acurácia do usuário (AU), acurácia do produtor (AP) e acurácia global (AG).

A Figura 5 apresenta uma visão geral do pipeline experimental adotado neste trabalho, incluindo as etapas de geração das texturas sintéticas pelo método Spot Noise, transformação para o domínio de Fourier, pré-processamento das imagens, treinamento da Rede Neural Convolucional e avaliação dos resultados.

Figura 5 — Pipeline Geral dos experimentos



Fonte: do Autor. 2026

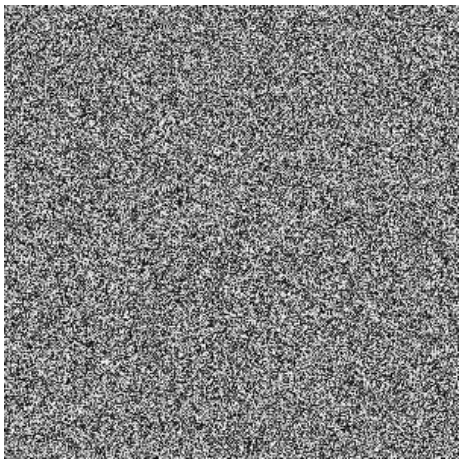
4 RESULTADOS E DISCUSSÕES

Nesta seção serão apresentados os conjuntos de dados utilizados, as arquiteturas das CNNs e os resultados dos experimentos.

4.1 Geração das imagens pelo método de Spot Noise

Inicialmente foram geradas 500 imagens aleatórias de tamanho 300x300 pixels. A Figura 6 apresenta um exemplo de uma dessas imagens.

Figura 6 — Exemplo de imagem aleatória gerada



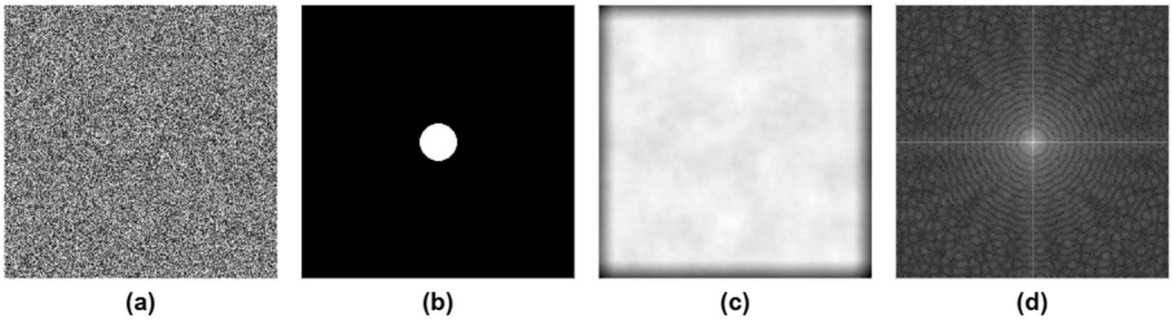
Fonte: do Autor. 2024

Essas imagens foram convoluídas com spots de círculos e quadrados, para que fossem geradas imagens de texturas através do método de Spot Noise sendo em seguida, obtidas as transformadas de Fourier.

Foram utilizados círculos de diâmetro e quadrados de lados 10, 20, 30 e 40 pixels respectivamente.

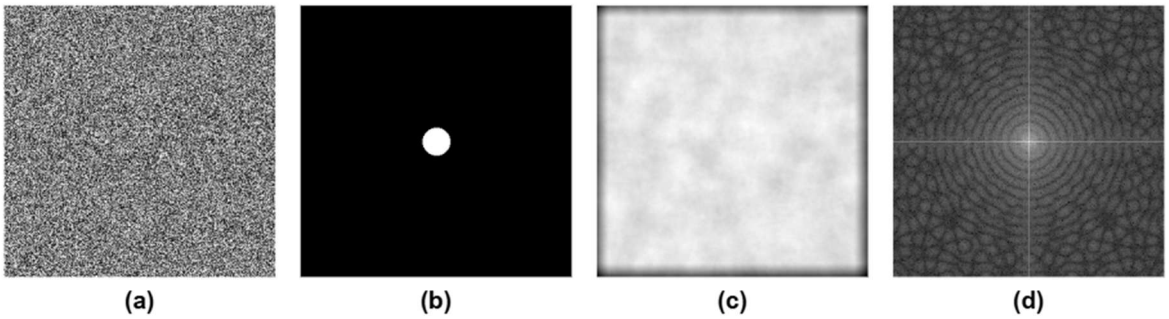
As Figura 7 à Figura 14 ilustram as imagens no processo de geração das texturas utilizando diferentes formatos e tamanhos de spots.

Figura 7 — Imagens do processo de geração de uma textura circular (a) Imagem aleatória (b) spot de Círculo de diâmetro 40 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



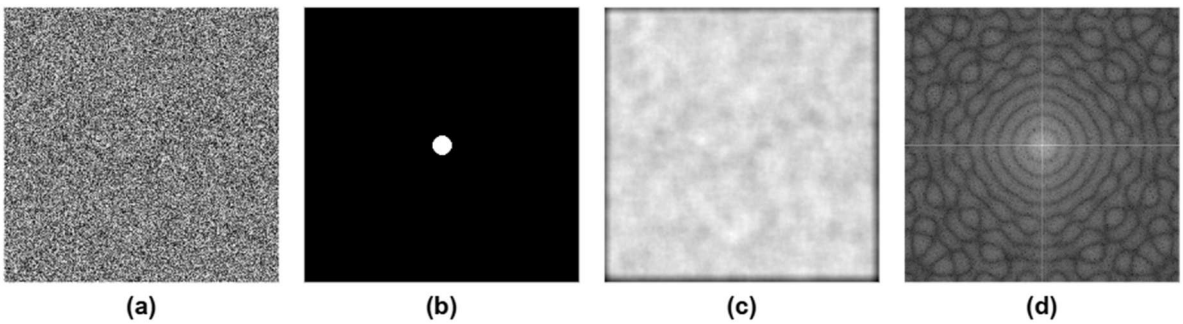
Fonte: do Autor. 2024

Figura 8 — Imagens do processo de geração de uma textura circular (a) Imagem aleatória (b) spot de Círculo de diâmetro 30 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



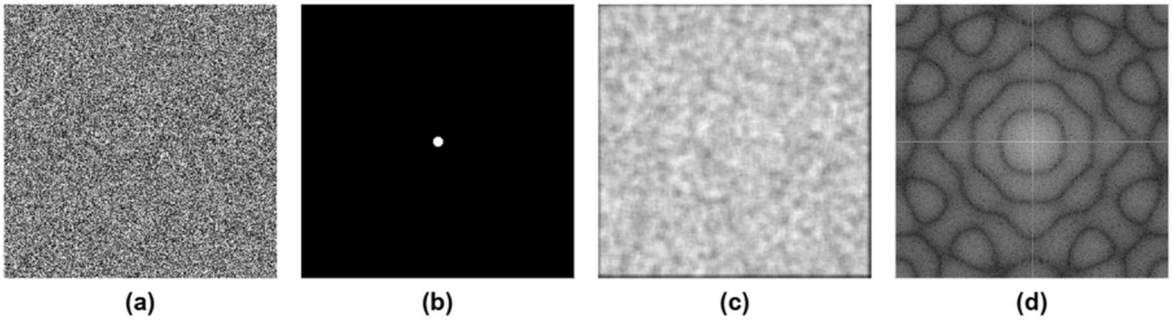
Fonte: do Autor. 2024

Figura 9 — Imagens do processo de geração de uma textura circular (a) Imagem aleatória (b) spot de Círculo de diâmetro 20 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



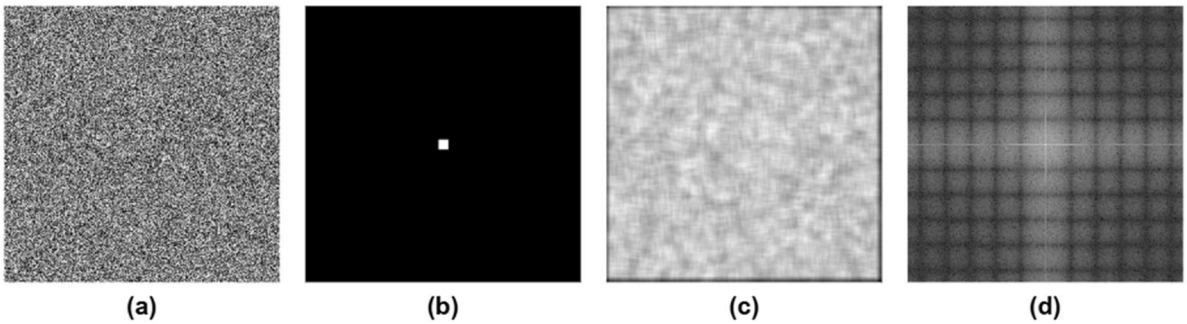
Fonte: do Autor. 2024

Figura 10 — Imagens do processo de geração de uma textura circular (a) Imagem aleatória (b) spot de Círculo de diâmetro 10 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



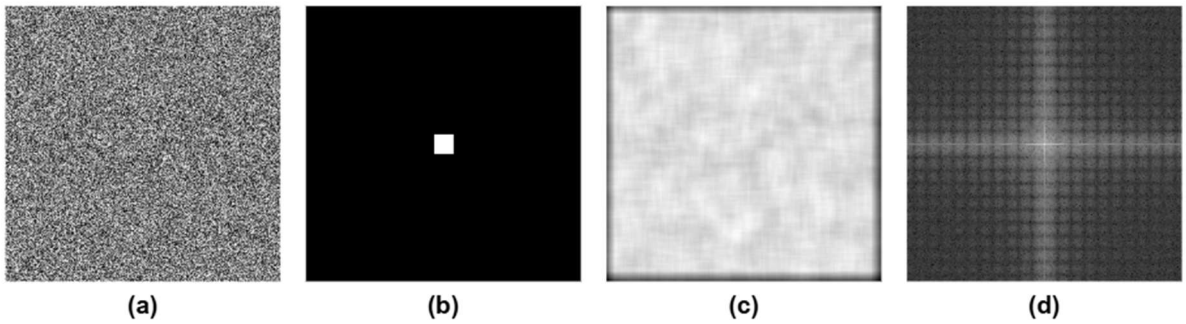
Fonte: do Autor. 2024

Figura 11 — Imagens do processo de geração de uma textura quadrada (a) Imagem aleatória (b) spot de Quadrado de lado 10 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



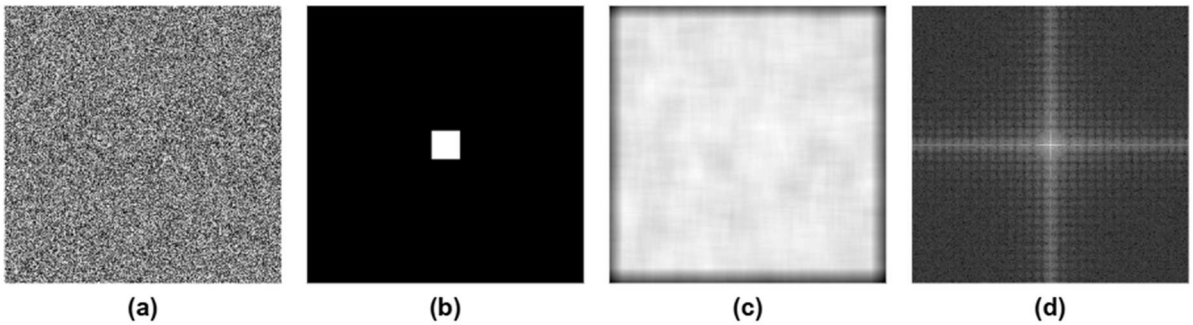
Fonte: do Autor. 2024

Figura 12 — Imagens do processo de geração de uma textura quadrada (a) Imagem aleatória (b) spot de Quadrado de lado 20 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



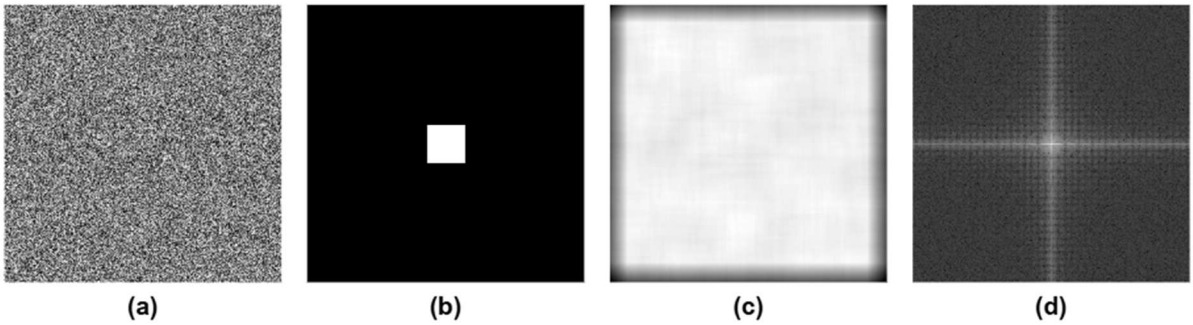
Fonte: do Autor. 2024

Figura 13 — Imagens do processo de geração de uma textura quadrada (a) Imagem aleatória (b) spot de Quadrado de lado 30 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



Fonte: do Autor. 2024

Figura 14 — Imagens do processo de geração de uma textura quadrada (a) Imagem aleatória (b) spot de Quadrado de lado 40 (c) Textura gerada (d) Imagem da textura no domínio de Fourier



Fonte: do Autor. 2024

Para preservar as dimensões originais da imagem (300 x 300 pixels) foi adotada a técnica de zero-padding.

4.2 Pré-Processamento das imagens

Antes da entrada na rede neural, as imagens passam por uma etapa de pré-processamento, na qual são redimensionadas para 128 x 128 pixels por interpolação bilinear e convertidas para escala de cinza.

Essa redução de dimensionalidade foi adotada para otimizar o custo computacional do experimento, diminuindo em mais de 80% o volume de dados de entrada. Isso reduz significativamente o número de operações convolucionais por época e agiliza o treinamento da rede, permitindo uma convergência mais rápida do modelo.

É importante ressaltar que esse redimensionamento não compromete a integridade das informações relevantes das texturas. A escolha de gerar as imagens originalmente em 300 x 300 pixels visa garantir a fidelidade geométrica dos *spots* (especialmente a curvatura dos spots circulares), evitando o serrilhamento (*aliasing*) típico de baixas resoluções. A redução posterior para 128 x 128, feita por interpolação bilinear, preserva as características estruturais e os padrões espectrais necessários para a classificação, eliminando apenas variações pontuais de alta frequência que não alteram a assinatura estrutural básica de cada classe de textura.

4.3 Testes preliminares de arquitetura

Nos testes iniciais da rede neural convolucional, embora o modelo tenha apresentado rápida convergência e baixos valores de erro durante o treinamento, observou-se uma baixa capacidade de generalização para padrões intermediários inéditos. A menor variabilidade estrutural do conjunto de treinamento inicial somada à elevada profundidade da arquitetura de CNN adotada no início, tornaram a rede altamente suscetível ao sobreajuste (*overfitting*), fazendo com que ela memorizasse características excessivamente específicas das amostras.

A análise das funções de ativação revelou comportamentos críticos. Ao testar a função tangente hiperbólica (*tanh*), o treinamento indicou uma aparente melhoria no desempenho, com redução consistente da função de perda e do erro absoluto médio. No entanto, a avaliação com os dados de teste demonstrou que a rede concentrava suas previsões próximas de zero, independentemente da textura apresentada.

Esse fenômeno decorre da simetria da função *tanh* combinada à distribuição dos rótulos adotados no problema. Como as variáveis-alvo estavam organizadas de forma aproximadamente simétrica em torno de zero, a convergência para valores centrais gerava métricas estatísticas artificialmente satisfatórias, mascarando a ausência de aprendizado efetivo das características estruturais das texturas.

Diante disso, constatou-se que arquiteturas mais profundas elevavam o custo computacional e a vulnerabilidade ao sobreajuste, principalmente quando associadas a poucas classes. Como solução para mitigar o viés de previsão central e a instabilidade numérica, a substituição pela função de ativação *LeakyReLU*, combinada a uma arquitetura mais compacta e ao aumento no número de classes, mostrou-se

eficaz, garantindo estabilidade ao treinamento e superior capacidade de generalização para texturas intermediárias.

4.4 Treinamento da Rede Neural Convolucional

Os treinamentos utilizaram 4000 imagens divididas em 8 classes balanceadas (mesmo número de amostras por classe) de acordo com a Tabela 1.

Tabela 1 — Rótulos utilizados no treinamento

Formato	Lado/Diâmetro	Rótulo
Círculo	40	-1,00
Círculo	30	-0,75
Círculo	20	-0,50
Círculo	10	-0,25
Quadrado	10	0,25
Quadrado	20	0,50
Quadrado	30	0,75
Quadrado	40	1,00

Fonte: do Autor. 2025

A CNN recebe como entrada imagens em escala de cinza com dimensões de 128 × 128 pixels.

A arquitetura é composta por três blocos principais de extração de características, seguidos por uma etapa de agregação global e por camadas totalmente conectadas responsáveis pela regressão final.

O primeiro bloco convolucional é formado por uma camada convolucional com 32 filtros de dimensão 5 x 5, utilizando *stride* igual a 2 e *zero-padding*, seguida por uma camada de Normalização de Lotes (*Batch Normalization*) e pela função de ativação *LeakyReLU*. Em seguida, é aplicada uma camada de *max pooling* de 2 x 2, responsável pela redução da dimensionalidade espacial e pela retenção das características mais relevantes.

O segundo bloco convolucional possui uma camada convolucional com 64 filtros de dimensão 3 x 3, também utilizando *stride* igual a 2 e *zero-padding*, seguida novamente por *Batch Normalization* e ativação *LeakyReLU*. Finalizando este bloco, uma nova camada de *max pooling* de 2 × 2 é aplicada para continuidade da redução espacial das representações.

O terceiro bloco convolucional é composto por uma camada convolucional com 128 filtros de dimensão 3 x 3, utilizando *zero-padding*, seguida por *Batch Normalization* e ativação *LeakyReLU*, aprofundando o processo de extração de características.

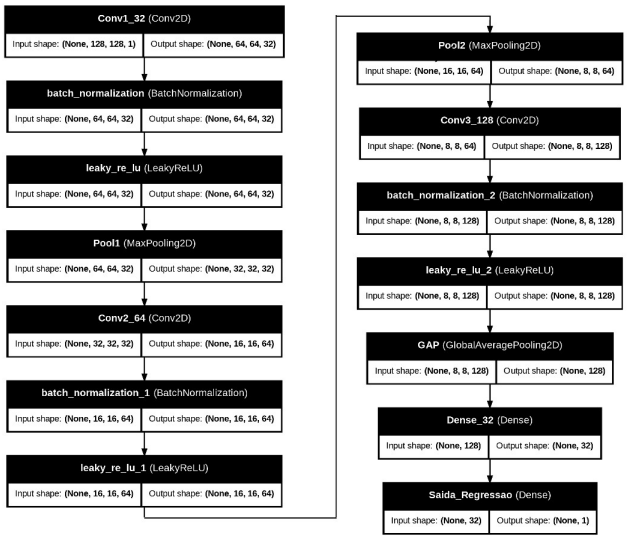
Em todas as camadas convolucionais foi aplicada regularização L2 nos pesos, com o objetivo de reduzir o risco de sobreajuste (*overfitting*) durante o treinamento.

Após a etapa de extração de características, foi utilizada uma camada de *Global Average Pooling*, que realiza uma agregação espacial das ativações geradas pelos mapas de características, reduzindo significativamente o número de parâmetros do modelo em comparação com abordagens baseadas em achatamento (*flatten*).

A etapa final da rede é composta por uma camada densa com 32 neurônios e função de ativação ReLU, seguida por uma camada de saída com um único neurônio com ativação linear, responsável por produzir a estimativa contínua associada à estrutura da textura, caracterizando o problema como uma tarefa de regressão.

A Figura 15 representa esta arquitetura de rede.

Figura 15 — Representação da Arquitetura da Rede Neural Convolucional



Fonte: do Autor. 2025

Os testes de generalização foram feitos utilizando 50 imagens de cada uma das texturas geradas usando as classes mostradas na Tabela 2.

Tabela 2 — Classes utilizadas no teste da Rede Neural Convolucional

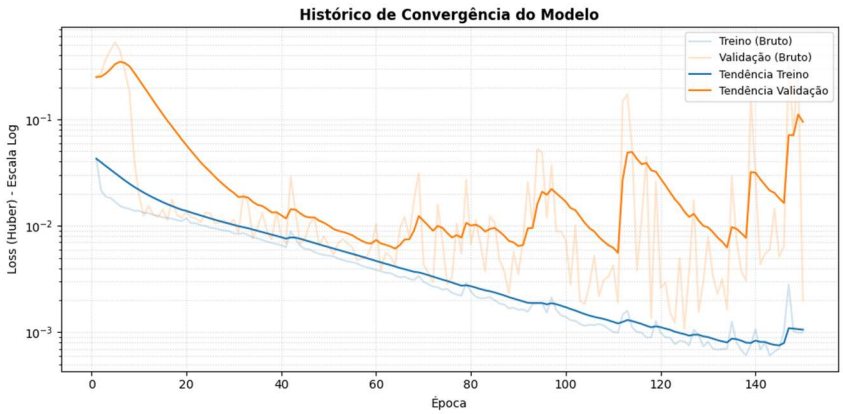
Formato	Lado/Diâmetro	Valor Esperado
Círculo	40	-1,000
Círculo	34	-0,875
Círculo	30	-0,750
Círculo	24	-0,625
Círculo	20	-0,500
Círculo	14	-0,375
Círculo	10	-0,250
Quadrado	10	0,250
Quadrado	14	0,375
Quadrado	20	0,500
Quadrado	24	0,625
Quadrado	30	0,750
Quadrado	34	0,875
Quadrado	40	1,000

Fonte: do Autor. 2025

4.5 Experimento 1: Imagens no Domínio de Fourier

A Figura 16 apresenta o histórico de treinamento da CNN no experimento 1 em escala logarítmica. As linhas sólidas representam a tendência suavizada, calculada por média móvel.

Figura 16 — Treinamento da CNN com imagens no domínio de Fourier em Escala logarítmica com linha de tendência



Fonte: do Autor. 2026

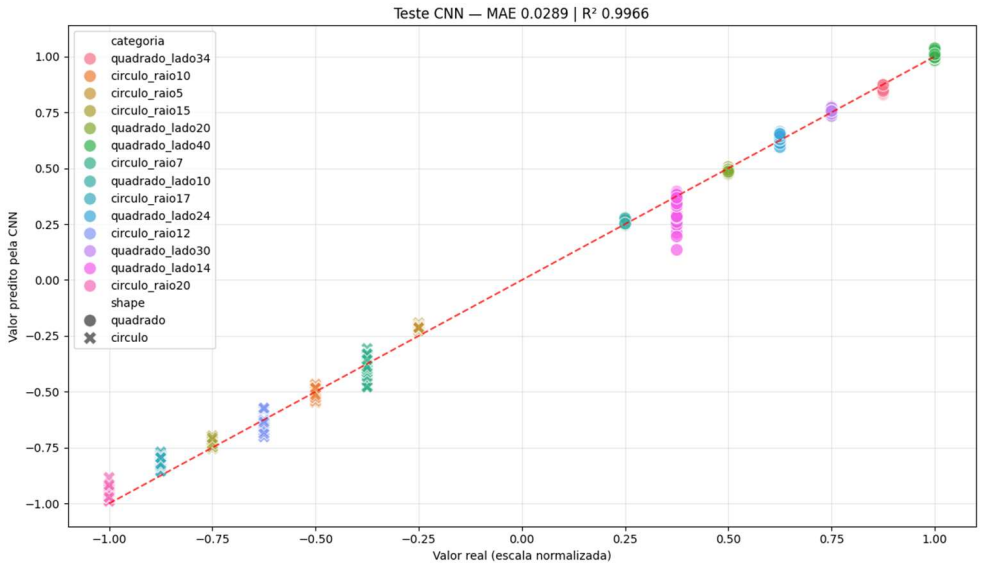
A análise da evolução da função de perda (Figura 16) revela um processo de aprendizado consistente e estável. Observa-se que, por volta da época 90, a rede consolida o aprendizado das texturas e entra em uma fase de ajuste fino, buscando

refinar o desempenho geral. As oscilações residuais verificadas na curva de validação nesta fase final são atribuídas à natureza estocástica do *Spot Noise* e à escala logarítmica da representação, que amplifica visualmente variações numéricas de baixa magnitude (10^{-3}). A proximidade entre as curvas de treino e validação, aliada à ausência de tendências ascendentes na perda de validação, confirma a robustez do modelo e a capacidade de generalização, sem evidências de *overfitting*.

Adicionalmente, a implementação de mecanismos de *Early Stopping* e a persistência exclusiva dos melhores pesos (*Save Best Weights Only*) garantiram que o modelo final selecionado fosse aquele com o desempenho ótimo no conjunto de validação. Dessa forma, eventuais instabilidades pontuais nas épocas finais do treinamento não comprometem a precisão do sistema, uma vez que o estado preservado da rede neural corresponde ao ponto de máxima eficiência estatística registrado durante todo o ciclo de épocas.

A Figura 17 apresenta o desempenho da CNN quando apresentada às transformadas de Fourier das imagens do conjunto de validação utilizando uma amostragem de 50 imagens por classe. Com um MAE de 0,0289 e R^2 de 0.9966, nota-se que o modelo manteve um alto poder de generalização, sendo capaz de identificar tanto as características de formato quanto as de tamanho do spot. Embora classes não apresentadas durante o treinamento (como o quadrado de lado 14) exibam uma dispersão ligeiramente superior, observa-se que essas variações são coerentes, uma vez que as predições se concentram em classes adjacentes de dimensões próximas e mesma assinatura estrutural.

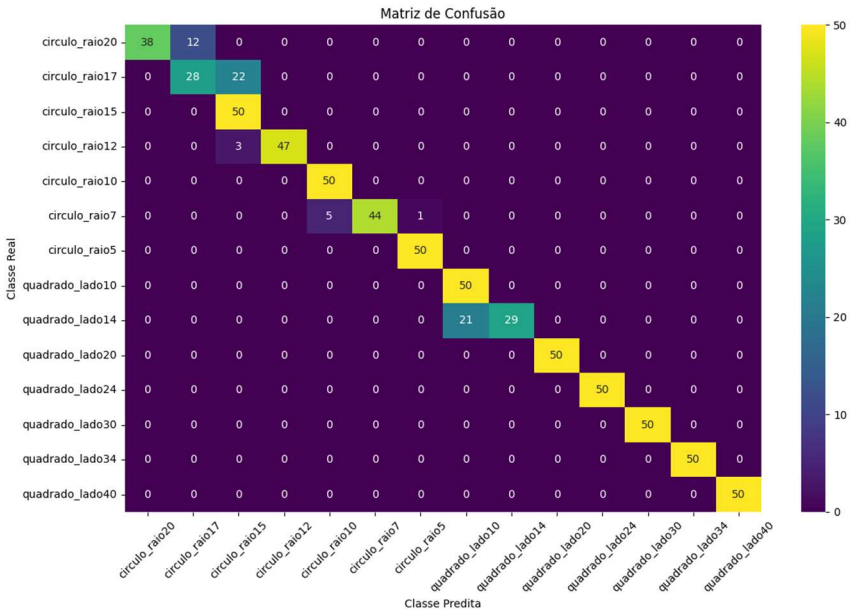
Figura 17 — Desempenho da CNN com imagens de validação no domínio de Fourier



Fonte: do Autor. 2026

A Figura 18 apresenta a Matriz confusão do desempenho da CNN com imagens no domínio de Fourier e a Tabela 3 apresenta a Acurácia do Produtor e do Usuário.

Figura 18 — Matriz Confusão da Classificação de imagens de validação no domínio de Fourier



Fonte: do Autor. 2026

O desempenho representa uma Acurácia Global (AG) de 0,91.

Tabela 3 — Acurácia do Produtor e do Usuário da CNN com imagens de validação no Domínio de Fourier

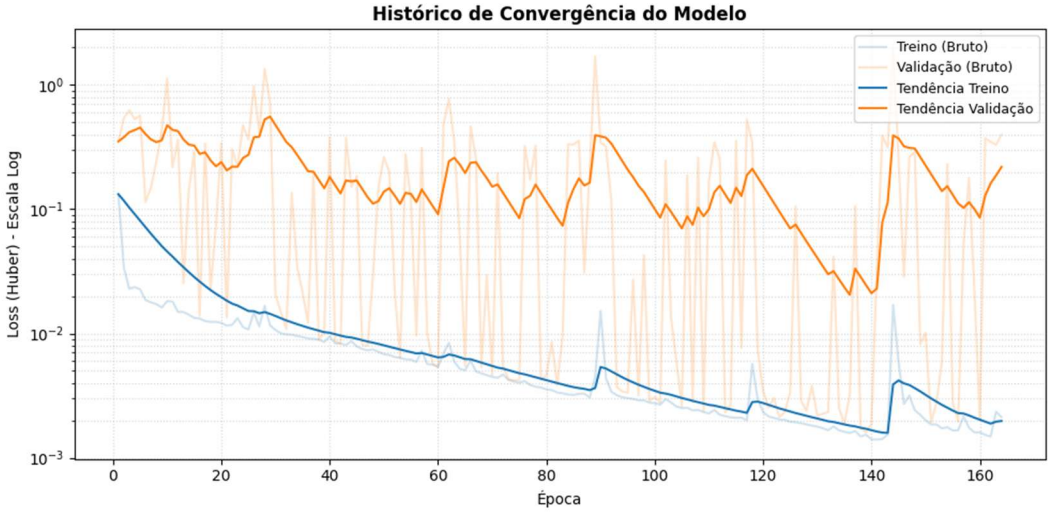
Formato	Lado/Diâmetro	Acurácia do Produtor (AP)	Acurácia do usuário (AU)
Círculo	40	0,76	1,00
Círculo	34	0,56	0,70
Círculo	30	1,00	0,67
Círculo	24	0,94	1,00
Círculo	20	1,00	0,91
Círculo	14	0,88	1,00
Círculo	10	1,00	0,98
Quadrado	10	1,00	0,70
Quadrado	14	0,58	1,00
Quadrado	20	1,00	1,00
Quadrado	24	1,00	1,00
Quadrado	30	1,00	1,00
Quadrado	34	1,00	1,00
Quadrado	40	1,00	1,00

Fonte: do Autor. 2026

4.6 Experimento 2: imagens no Domínio Espacial

A Figura 19 apresenta o histórico de treinamento da CNN no experimento 2, que utilizou imagens no domínio espacial. Diferentemente do observado no domínio de Fourier, a evolução da função de perda revela um processo de aprendizado marcado por instabilidades severas e uma convergência menos eficiente. Embora a curva de treinamento (azul) mantenha uma tendência descendente, a curva de validação (laranja) exibe oscilações de alta amplitude e picos de erro que ultrapassam a ordem de 10^0, indicando dificuldades do modelo em generalizar os padrões espaciais das texturas sintéticas.

Figura 19 — Treinamento da CNN com imagens no domínio espacial em Escala logarítmica com linha de tendência



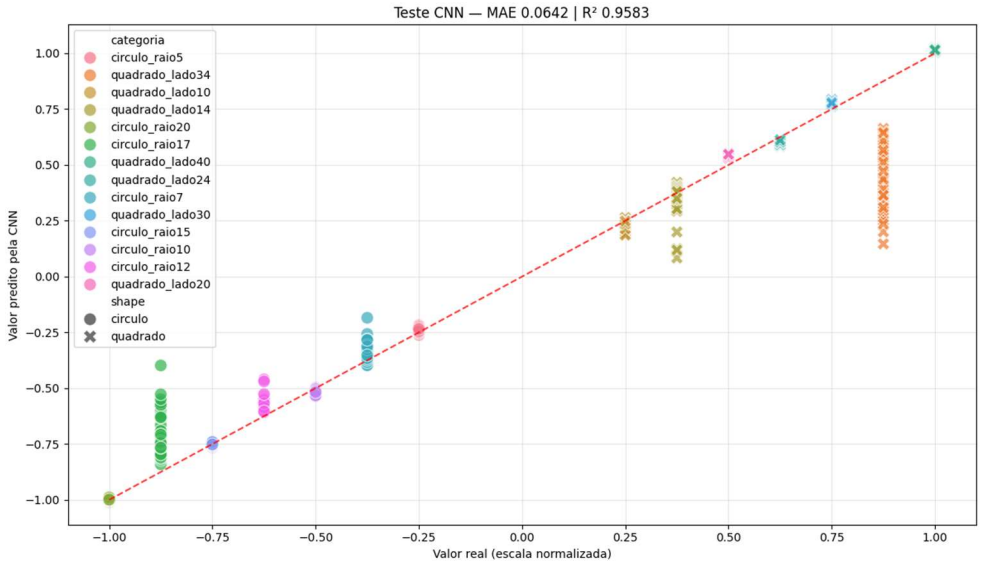
Fonte: do Autor. 2026

As linhas sólidas de tendência confirmam que apesar da tendência de queda, o erro apresenta grande oscilação e o valor médio permanece significativamente superior ao obtido no experimento anterior. Esse comportamento sugere que, no domínio espacial, as características da textura aleatória dificultam a distinção clara entre as assinaturas morfológicas das classes, resultando em um gradiente de erro com maior amplitude durante a retropropagação.

Novamente, o uso de *Early Stopping* e *Save Best Weights Only* foi fundamental. Mesmo diante de um cenário de baixa estabilidade, essas técnicas permitiram isolar o estado da rede que apresentou o menor erro relativo antes das flutuações mais críticas, garantindo que o modelo final, embora inferior ao do Experimento 1, representasse o melhor ajuste possível para o domínio espacial.

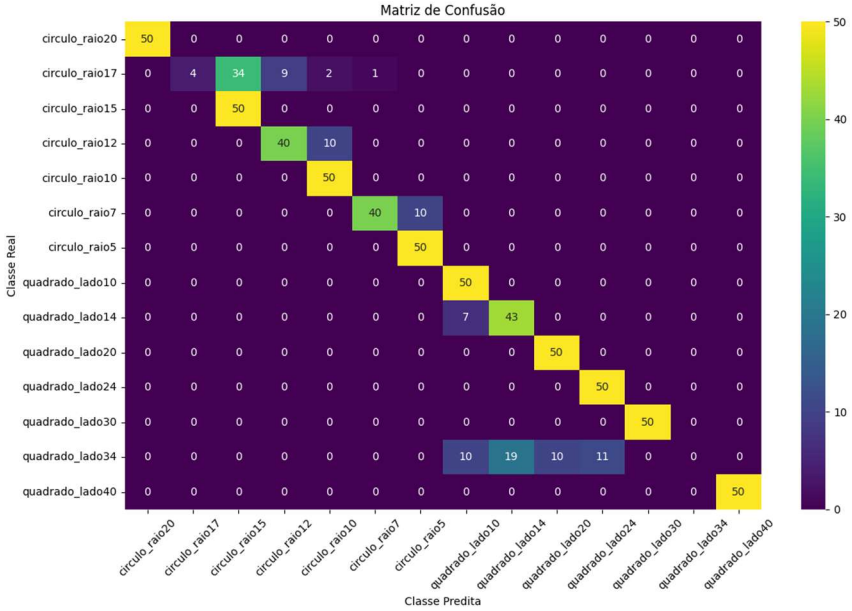
A Figura 20 apresenta o desempenho da CNN quando aplicada às imagens de validação no domínio espacial. Com um MAE de 0,0642 e R^2 de 0.9583, percebe-se que a rede teve dificuldades em generalizar, uma vez que vários valores de saída estão muito dispersos e longe da linha de tendência. Além disso, a rede foi praticamente incapaz de prever corretamente as texturas geradas com spots de quadrados de lados 14 e 34. A Figura 21 apresenta a Matriz Confusão deste experimento e a Tabela 4 a Acurácia do produtor e do usuário.

Figura 20 — Desempenho da CNN com imagens de validação no domínio espacial



Fonte: do Autor. 2026

Figura 21 — Matriz Confusão da Classificação das imagens de validação no domínio espacial



Fonte: do Autor. 2026

O desempenho do experimento 2 representa uma Acurácia Global (AG) de 0,82.

Tabela 4 — Acurácia do Produtor e do Usuário da CNN com imagens de validação no Domínio Espacial

Formato	Lado/Diâmetro	Acurácia do Produtor (AP)	Acurácia do usuário (AU)
Círculo	40	1,00	1,00
Círculo	34	0,08	1,00
Círculo	30	1,00	0,60
Círculo	24	0,80	0,82
Círculo	20	1,00	0,81
Círculo	14	0,80	0,98
Círculo	10	1,00	0,83
Quadrado	10	1,00	0,75
Quadrado	14	0,86	0,69
Quadrado	20	1,00	0,83
Quadrado	24	1,00	0,82
Quadrado	30	1,00	1,00
Quadrado	34	0,00	0,00
Quadrado	40	1,00	1,00

Fonte: do Autor. 2026

Observa-se que o experimento realizado no domínio de Fourier apresentou desempenho superior ao obtido no domínio espacial, tanto em termos de erro médio absoluto quanto em termos de capacidade de generalização para texturas não observadas durante o treinamento.

Esse comportamento pode ser explicado pela natureza espectral do processo de geração das texturas, uma vez que o espectro da textura resultante tende a preservar predominantemente as características espectrais do spot utilizado na geração da textura.

Consequentemente, no domínio de Fourier, as assinaturas de cada textura tornam-se evidentes e permitem a criação de uma ordem de acordo com os espectros das imagens. Essa organização da informação pode facilitar o processo de aprendizagem da rede neural convolucional, permitindo que os filtros convolucionais identifiquem padrões estruturais de forma mais consistente do que no domínio espacial.

5 CONCLUSÃO

Este trabalho investigou a aplicação de Redes Neurais Convolucionais (CNNs) na análise de texturas sintéticas geradas pelo método de *Spot Noise*, avaliando o impacto das representações nos domínios espacial e de Fourier no desempenho do modelo.

Os experimentos mostraram que a utilização da Transformada de Fourier como etapa de pré-processamento contribuiu significativamente para a capacidade de generalização da rede neural. No domínio espectral, o modelo apresentou erro médio absoluto (MAE) de 0,0289 e um coeficiente de determinação (R^2) de 0,9966, além de uma elevada acurácia global. Estes índices indicam que existe uma robustez no processo, inclusive na estimativa de classes intermediárias não observadas durante o treinamento.

Em contrapartida, o treinamento realizado diretamente no domínio espacial resultou em maior dispersão e redução do desempenho geral com um MAE de 0,0642 e um R^2 de 0,9583. As dificuldades observadas na previsão de dimensões intermediárias evidenciaram uma sensibilidade maior da rede às variações locais de pixel, limitando sua eficácia em cenários de generalização.

A escolha do método Spot Noise como base para a geração do conjunto de dados foi determinante para o sucesso dos experimentos, principalmente devido à sua natureza paramétrica. Esse método permite controlar variáveis estruturais das texturas, como forma e dimensão dos elementos geradores, possibilitando a construção de um conjunto de dados sintético consistente e bem definido. Esse controle foi essencial para avaliar a capacidade da rede neural em generalizar e estimar padrões intermediários não apresentados durante o treinamento.

Além disso, o uso do Spot Noise traz uma forte relação entre o processo de geração das texturas e sua representação no domínio de Fourier. Como o espectro do ruído aleatório apresenta distribuição aproximadamente uniforme, a estrutura espectral resultante das imagens passa a refletir predominantemente as características do spot utilizado na geração da textura. Dessa forma, a Transformada de Fourier reorganiza a informação estrutural das imagens em padrões espectrais mais explícitos, reduzindo a influência das variações estocásticas no domínio espacial.

Como consequência, a rede neural passa a aprender principalmente as propriedades estruturantes que caracterizam cada textura, em vez de depender apenas de padrões locais de intensidade no domínio espacial. Esse comportamento contribui para melhorar a capacidade de generalização do modelo e para aumentar a robustez e a precisão das estimativas obtidas, indicando que a representação no domínio da frequência pode ser uma estratégia promissora para aplicações de análise e caracterização de texturas em imagens digitais.

REFERÊNCIAS

HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-3, n. 6, p. 610–621, nov. 1973.

SCHWARTZ, W. R.; PEDRINI, H. Método para classificação de imagens baseada em matrizes de coocorrência utilizando características de textura. *Revista Brasileira de Processamento de Imagens*, v. 16, n. 2, p. 15–24, 2003.

CONCI, A. D. *et al.* Segmentação por textura e localização do contorno de regiões em imagens multibandas. *Revista Brasileira de Cartografia*, v. 58, n. 2, p. 123–135, 2006. Disponível em: <http://www.ic.uff.br/~aconci/pub2006.html>. Acesso em: 7 jan. 2025.

HOSNY, K. M. *et al.* Improved color texture recognition using multi-channel orthogonal moments and local binary pattern. *Multimedia Tools and Applications*, v. 80, n. 9, p. 13179–13194, 13 jan. 2021.

MUKUNDAN, R.; ONG, S.H. A Discrete Orthognal Moment Features Using Chebyshev Polynomials. Disponível em: <*https://www.researchgate.net/publication/29487108_Discrete_Orthognal_Moment_Features_Using_Chebyshev_Polynomials*>. Acesso em: 18 ago. 2025.

OJALA, T. *et al.* Outex - new framework for empirical evaluation of texture analysis algorithms. 11 ago. 2002.

GEUSEBROEK, J.M.; SMEULDERS, A.W.M. Amsterdam Library of Textures (ALOT) - Homepage. Disponível em: <*https://aloi.science.uva.nl/public_alot/*>. Acesso em: 18 ago. 2025.

ZHOU, H. *et al.* A hybrid approach of combining Random Forest with texture analysis and VDVI for desert vegetation mapping based on UAV RGB data. *Remote Sensing*, v. 13, n. 10, p. 1891, 12 maio 2021.

AMORIM, L.; FILHO, M. B. Análise de lacunaridade em imagens urbanas: um estudo de caso em dois bairros de Córdoba, Espanha. *Revista de Morfologia Urbana*, v. 5, n. 2, p. 65–81, 2017. ISSN 2182-7214.

SARKAR, N.; CHAUDHURI, B. B. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 1, p. 115–120, 1994.

SOUZA, J.; PEREIRA, M. Fractal measures of image local features: an application to texture recognition. *Journal of Texture Research*, v. 20, n. 4, p. 123–134, 2021. DOI: 10.48550/arXiv.2108.12491.

TIWARI, S. *et al.* Investigations on segmentation-based fractal texture for texture classification in the presence of Gaussian noise. *PLOS ONE*, v. 20, n. 1, p. e0315135, 2025. Acesso em: 10 jan. 2025.

HUANG, Y. *et al.* Review of wavelet-based unsupervised texture segmentation, advantage of adaptive wavelets. *IET Image Processing*, v. 12, n. 9, p. 1626–1638, 19 abr. 2018.

MALLAT, S. G. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 11, n. 7, p. 674–693, 1989.

YESILLI, M. C. *et al.* Automated surface texture analysis via Discrete Cosine Transform and Discrete Wavelet Transform. *Precision Engineering*, v. 77, p. 141–152, set. 2022.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 4. ed. Nova Jersey: Pearson, 2018. p. 215–255.

JÚNIOR, C. R. S.; MONTE, Á. F. G.; CUNHA, D. M. Utilização de imagens no domínio da frequência espacial e redes neurais artificiais para determinação de propriedades ópticas de tecidos. *Revista Brasileira de Física Médica*, v. 15, p. 635, 2021. DOI: 10.29384/rbfm.2021.v15.19849001635. Disponível em: <https://www.rbfm.org.br/rbfm/article/view/635>. Acesso em: 8 ago. 2025.

WIJK, J. J. Spot noise - texture synthesis for data visualization. *Computer Graphics*, v. 25, n. 4, p. 309–318, 1991.

MARTINS, A. C. G.; SIMÕES, A. S.; PRADO, G. I. Uso de redes neurais artificiais e o modelo de Spot Noise para classificação de imagens de cereais. In: *National Conference on Artificial Intelligence*, 2007, Rio de Janeiro. Anais [...]. Rio de Janeiro: IME, 2007.

Tikhonov, A. N.; Arsenin, V. Y. *Solutions of Ill-posed Problems*. Washington: Winston, 1977.

WANG, Y. *et al.* An image-based data-driven model for texture inspection of ground workpieces. *Sensors*, v. 22, n. 14, p. 5192, 11 jul. 2022.

ABDALLAH, S. E. *et al.* Deep learning model based on ResNet-50 for beef quality classification. *Information Sciences Letters*, v. 12, n. 1, art. 24, jan. 2023.

LIU, X.; ALDRICH, C. Deep learning approaches to image texture analysis in material processing. *Metals*, v. 12, n. 2, p. 355, 18 fev. 2022.

HUBER, N. R. *et al.* Evaluating a convolutional neural network noise reduction method when applied to CT images reconstructed differently than training data. *Journal of Computer Assisted Tomography*, v. 45, n. 4, p. 544–551, jul. 2021.

SCABINI, L. *et al.* A comparative survey of Vision Transformers for feature extraction in texture analysis. *arXiv preprint*, arXiv:2406.06136. Disponível em: <https://arxiv.org/abs/2406.06136>. Acesso em: 28 out. 2024.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, Massachusetts: The MIT Press, 2016.

GÉRON, A. *Mãos à obra: aprendizado de máquina com Scikit-Learn, Keras e TensorFlow: conceitos, ferramentas e técnicas para a construção de sistemas inteligentes*. 2. ed. São Paulo: Novatec, 2020.

IOFFE, S.; SZEGEDY, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*, 32., 2015, Lille. Anais [...]. Lille: [s.n.], 2015. p. 448–456.

NIELSEN, M. A. *Neural Networks and Deep Learning*. 2015. Disponível em: <http://neuralnetworksanddeeplearning.com>. Acesso em: 7 ago. 2025.

HUBER, P. J. *Robust estimation of a location parameter.* Annals of Mathematical Statistics, v. 35, n. 1, p. 73–101, 1964.

CONGALTON, R. G. A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data. *Remote Sensing of Environment*, vol. 37, n. 1, p. 35-46, 1991.

PANTALEÃO, E.; SCOFIELD, G. Comparação entre medidas de acurácia de classificação para imagens do satélite ALOS. *Simpósio Brasileiro de Sensoriamento Remoto, 14*. (SBSR), 2009, Natal. Disponível em: <http://marte.sid.inpe.br/col/dpi.inpe.br/sbsr@80/2008/11.17.20.26/doc/7039-7046.pdf>. Acesso em: 27 mar. 2024.

APÊNDICE A — CÓDIGO DE TREINAMENTO

O código abaixo apresenta a estrutura utilizada para o treinamento de ambos os modelos (Domínio Espacial e Domínio de Fourier). A única alteração realizada entre os experimentos foi o apontamento dos diretórios das imagens de treinamento (data_dir), conforme as especificações de cada experimento.

```
import os
import csv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger

drive.mount('/content/drive')

# Parâmetros de execução
TREINAR_DO_ZERO = True
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
EPOCHS = 250

# Dicionário de labels (normalização entre -1 e 1)
class_mapping = {
    'circulo_raio20': -1.0,
    'circulo_raio15': -0.75,
    'circulo_raio10': -0.5,
    'circulo_raio5': -0.25,
    'quadrado_lado10': 0.25,
    'quadrado_lado20': 0.5,
    'quadrado_lado30': 0.75,
    'quadrado_lado40': 1.0
}

# Caminhos de arquivos
base_dir = '/content/drive/MyDrive/Projeto Mestrado/imagens/transformadas_fourier'
checkpoint_path =
'/content/drive/MyDrive/Defesa/CNN_Dom_Fourier/CNN_8Classes_128px_Dom_Fourier.keras
'
log_csv_path =
'/content/drive/MyDrive/Defesa/CNN_Dom_Fourier/log_treinamento_CNN_8Classes.csv'

class ManualLogger(keras.callbacks.Callback):
    def __init__(self, path):
        self.path = path
        if not os.path.exists(self.path):
            with open(self.path, 'w', newline='') as f:
                csv.writer(f).writerow(['epoch','loss','mae','val_loss','val_mae'])

    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        with open(self.path, 'a', newline='') as f:
            csv.writer(f).writerow([
```

```
                epoch,
                f"{logs.get('loss',0):.6f}",
                f"{logs.get('mae',0):.6f}",
                f"{logs.get('val_loss',0):.6f}",
                f"{logs.get('val_mae',0):.6f}",
            ])

def load_data(directory):
    imgs, labels = [], []
    valid_ext = ('.jpg', '.jpeg', '.png')

    print("Iniciando carregamento do dataset...")
    for folder, target in class_mapping.items():
        path = os.path.join(directory, folder)
        if not os.path.exists(path):
            print(f"Alerta: Pasta {folder} não encontrada.")
            continue

        files = [f for f in os.listdir(path) if f.lower().endswith(valid_ext)]
        count = 0
        for fname in sorted(files):
            try:
                img = load_img(os.path.join(path, fname), color_mode='grayscale',
target_size=IMG_SIZE)
                img = img_to_array(img) / 255.0
                imgs.append(img)
                labels.append(target)
                count += 1
            except:
                print(f"Erro ao carregar {fname}")
        print(f"Classe {folder}: {count} imagens carregadas.")

    return np.array(imgs), np.array(labels)

X, y = load_data(base_dir)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, shuffle=True
)

def build_cnn():
    model = models.Sequential([
        layers.Input(shape=(128,128,1)),

        layers.Conv2D(32,(5,5), strides=2, padding='same',
kernel_regularizer=keras.regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D((2,2)),

        layers.Conv2D(64,(3,3), strides=2, padding='same',
kernel_regularizer=keras.regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D((2,2)),

        layers.Conv2D(128,(3,3), padding='same',
kernel_regularizer=keras.regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.LeakyReLU(),

        layers.GlobalAveragePooling2D(),
        layers.Dense(32, activation='relu'),
        layers.Dense(1)
    ])
    return model

# Lógica de carregamento/reinicio
```

```
if TREINAR_DO_ZERO or not os.path.exists(checkpoint_path):
    print("Criando modelo novo.")
    model = build_cnn()
else:
    print("Carregando modelo do checkpoint.")
    model = keras.models.load_model(checkpoint_path)

model.compile(
    optimizer=optimizers.Adam(learning_rate=2e-4),
    loss=tf.keras.losses.Huber(delta=1.0),
    metrics=['mae']
)

callbacks = [
    ManualLogger(log_csv_path),
    ModelCheckpoint(checkpoint_path, monitor='val_loss', save_best_only=True,
verbose=1),
    EarlyStopping(monitor='val_loss', patience=25, restore_best_weights=True,
verbose=1),
    CSVLogger(log_csv_path.replace('.csv', '_keras.csv'), append=True)
]

history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    callbacks=callbacks,
    verbose=1
)

# Visualização de resultados
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label='Treino')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Histórico de Treinamento - Domínio de Fourier')
plt.ylabel('Huber Loss')
plt.xlabel('Época')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

APÊNDICE B — CÓDIGO DE TESTE

O código abaixo apresenta a estrutura utilizada para o teste da rede. A única alteração realizada entre os experimentos foi o apontamento dos diretórios das imagens de teste (data_dir), conforme as especificações de cada experimento.

```
import os
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.metrics import mean_absolute_error, r2_score

# Configurações de ambiente
checkpoint_path =
'/content/drive/MyDrive/Defesa/CNN_Dom_Fourier/CNN_8Classes_128px_Dom_Fourier.keras
'
test_dir = '/content/drive/MyDrive/Projeto Mestrado/imagens/transformadas_fourier'
IMG_SIZE = (128, 128)

# Classes de teste (inclui as 8 de treino + as 6 de generalização)
class_mapping = {
    'circulo_raio20': -1.0, 'circulo_raio17_5': -0.875, 'circulo_raio15': -0.75,
    'circulo_raio12_5': -0.625, 'circulo_raio10': -0.5, 'circulo_raio7_5': -0.375,
    'circulo_raio5': -0.25, 'quadrado_lado10': 0.25, 'quadrado_lado15': 0.375,
    'quadrado_lado20': 0.5, 'quadrado_lado25': 0.625, 'quadrado_lado30': 0.75,
    'quadrado_lado35': 0.875, 'quadrado_lado40': 1.0
}

def load_test_set(directory):
    x_val, y_val, names = [], [], []
    for folder, target in class_mapping.items():
        path = os.path.join(directory, folder)
        if not os.path.exists(path): continue

        files = [f for f in os.listdir(path) if f.lower().endswith(('.png',
'.jpg'))]
        for fname in files:
            img = load_img(os.path.join(path, fname), color_mode='grayscale',
target_size=IMG_SIZE)
            x_val.append(img_to_array(img) / 255.0)
            y_val.append(target)
            names.append(folder)
    return np.array(x_val), np.array(y_val), names

# Carregamento do modelo e dados
model = keras.models.load_model(checkpoint_path)
X_test, y_true, class_names = load_test_set(test_dir)

# Predições
y_pred = model.predict(X_test).flatten()

# Cálculo de métricas globais
mae = mean_absolute_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

print(f"Resultados Finais - MAE: {mae:.4f} | R2: {r2:.4f}")

# Visualização residual
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_true, y_pred, alpha=0.4, color='blue')
plt.plot([min(y_true), max(y_true)], [min(y_true), max(y_true)], '--r',
linewidth=2)
plt.title('Validação da Regressão: Predito vs Real')
plt.xlabel('Valores Reais (Ground Truth)')
plt.ylabel('Predições do Modelo')
plt.grid(True, linestyle=':', alpha=0.6)
plt.show()
```

APÊNDICE C — CÓDIGOS DA MATRIZ CONFUSÃO

Este apêndice apresenta o algoritmo utilizado para a discretização das predições contínuas da rede neural em classes nominais, permitindo a avaliação do desempenho do modelo por meio de matrizes de confusão. O método baseia-se na busca pelo vizinho mais próximo no espaço euclidiano para associar o valor escalar predito à classe correspondente. Além da matriz, o script calcula a Acurácia Global (AG), a Acurácia do Produtor (AP) e a Acurácia do Usuário (AU) para todas as 14 categorias analisadas.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Definição dos alvos conforme o mapeamento do Spot Noise
class_values = mapping_y_true
class_names = list(class_values.keys())
class_targets = np.array(list(class_values.values()))

def get_nearest_class(value, targets):
    """Encontra o índice da classe com menor distância absoluta."""
    return np.argmin(np.abs(targets - value))

# Conversão dos resultados de regressão para labels discretos
y_true_class = [get_nearest_class(v, class_targets) for v in
df_resultados['y_true_valor']]
y_pred_class = [get_nearest_class(v, class_targets) for v in
df_resultados['previsao_valor']]

# Geração da matriz de confusão
cm = confusion_matrix(y_true_class, y_pred_class, labels=range(len(class_names)))
df_cm = pd.DataFrame(cm, index=class_names, columns=class_names)

# Visualização da matriz
plt.figure(figsize=(12, 8))
sns.heatmap(df_cm, annot=True, fmt='d', cmap='viridis')
plt.xlabel("Classe Predita")
plt.ylabel("Classe Real")
plt.title("Matriz de Confusão - Desempenho do Modelo")
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()

# Cálculo das métricas de acurácia
diag = np.diag(cm)
ag = np.sum(diag) / np.sum(cm)
ap = diag / cm.sum(axis=1) # Recall
au = diag / cm.sum(axis=0) # Precision

metrics = pd.DataFrame({
    "Classe": class_names,
    "AP_produtor": ap,
    "AU_usuario": au
})

print(f"Acurácia Global (AG): {ag:.4f}")
print("\nMétricas por Classe:")
print(metrics)
```