Urban-Eye: Classificação de Problemas Urbanos Utilizando Deep Learning e Transfer learning

Vinícius Santos Monteiro

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
São Carlos – SP – Brazil

vini.mon@usp.br

Abstract. *The automated identification of urban issues, such as potholes, illegal garbage disposal, and damaged infrastructure, is crucial for efficient city management. This study details the development of "Urban-Eye," a Deep Learning model designed to classify citizen-provided images into nine distinct categories of urban problems. We utilize a public dataset from Kaggle, addressing the significant challenge of severe class imbalance. The proposed approach employs Transfer Learning, using a pre-trained MobileNetV2 architecture, combined with data augmentation and a weighted loss function (class_weight) to force the model to focus on rare classes. Experimental results demonstrate that this methodology provides a robust and accurate pipeline for automated urban issue classification.*

Resumo. *A identificação automatizada de problemas urbanos, como buracos, descarte ilegal de lixo e infraestrutura danificada, é crucial para a gestão eficiente das cidades. Este estudo detalha o desenvolvimento do "Urban-Eye", um modelo de Deep Learning projetado para classificar imagens fornecidas por cidadãos em nove categorias distintas de problemas urbanos. Utilizamos um dataset público do Kaggle, abordando o desafio significativo de um severo desbalanceamento de classes. A abordagem proposta emprega Aprendizado por Transferência (Transfer Learning), utilizando uma arquitetura MobileNetV2 pré-treinada, combinada com Data Augmentation e uma função de perda ponderada (class_weight) para forçar o modelo a focar em classes raras. Os resultados experimentais demonstram que esta metodologia provê um pipeline robusto e preciso para a classificação automatizada de problemas urbanos.*

1. Introdução, Motivação e Objetivos

As cidades enfrentam diariamente desafios que impactam a mobilidade, segurança e qualidade de vida. A detecção e classificação manual de ocorrências urbanas, como buracos, queda de árvores ou falhas na infraestrutura elétrica, é um processo lento e que consome muitos recursos dos órgãos públicos. A automação desse processo, através da análise de imagens capturadas por cidadãos, pode apoiar diretamente na priorização de ações de manutenção e limpeza.

A ascensão das Redes Neurais Convolucionais (*CNNs*) revolucionou a área de Visão Computacional, permitindo que máquinas alcancem desempenho em nível humano na classificação de imagens. No entanto, datasets do mundo real raramente são balanceados. Este projeto utiliza um dataset público do Kaggle que simula essa realidade, apresentando um severo desbalanceamento entre as classes.

O código-fonte e o notebook de treinamento podem ser encontrados nos seguintes links: [1] [2] [3]

O objetivo principal deste trabalho é desenvolver um pipeline completo de classificação de imagens de problemas urbanos, com foco principal na criação de um modelo robusto que supere o desafio do desbalanceamento de classes, garantindo que categorias raras (mas críticas), como "Estacionamento Ilegal" (*IllegalParking*) ou "Poluição por Animais Mortos" (*DeadAnimalsPollution*), sejam identificadas corretamente.

2. Abordagem Proposta

A solução foi desenvolvida como um pipeline de Deep Learning utilizando TensorFlow e Keras. A metodologia foi dividida em quatro etapas principais: (1) Análise Exploratória e Preparação dos Dados, (2) Definição do Pipeline de Pré-processamento, (3) Arquitetura do Modelo com Aprendizado por Transferência, e (4) Estratégia de Treinamento Ponderado.

2.1. Análise Exploratória e Pré-processamento

A primeira etapa consistiu em uma Análise Exploratória de Dados (EDA). Um *DataFrame* mestre foi criado para mapear o caminho de cada imagem, seu rótulo e sua divisão no conjunto.

A análise visual confirmou a qualidade e a variabilidade das imagens. A descoberta mais crítica foi o severo desbalanceamento de classes no conjunto de treino, como ilustrado na Figura 1. Classes como '*Damaged concrete structures*' possuíam mais de 9.000 amostras, enquanto '*IllegalParking*' possuía menos de 60. Este achado definiu a estratégia central do projeto.

[1] https://www.kaggle.com/code/viniciusjamal/urban-eye-an-urban-issue-ai-classifier
[2] https://github.com/vini-mon/Urban_Eye
[3] (https://www.kaggle.com/datasets/akinduhiman/urban-issues-dataset)

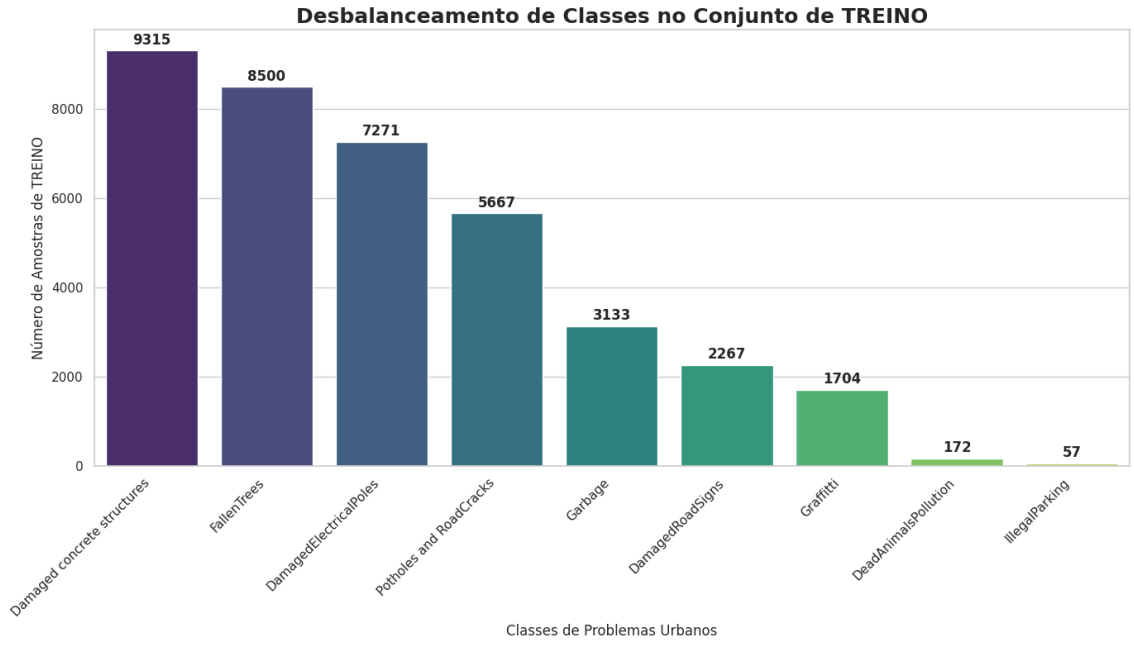

Figure 1. Distribuição e desbalanceamento das 9 classes no conjunto de treino.

Foi decidido não utilizar o *K-Fold Cross-Validation*, apesar de seus benefícios em robustez. Dado o alto custo computacional do treinamento de *CNNs* (especialmente em recursos limitados como os do Kaggle) e a existência de um conjunto de validação pré-definido, optou-se por uma abordagem de iteração mais rápida (treino/validação/teste) para estabelecer um *baseline* funcional.

2.2. Pipeline de Dados e Data Augmentation

Utilizamos a classe 'ImageDataGenerator' do Keras para criar dois pipelines de dados:

- **Gerador de Treino:** Aplicou normalização (um *rescale* com definição 1./255) e um conjunto de técnicas de *Data Augmentation* para combater o overfitting e gerar amostras sintéticas para as classes raras. As transformações incluíram rotação (de até 20 º), zoom (de até 0.15x), inversão horizontal e variação de brilho (de 0.8x até 1.2x).
- **Gerador de Validação/Teste:** Aplicou apenas a normalização (mesma *rescale* do conjunto de treino), garantindo que a avaliação fosse feita em dados "limpos", como seriam encontrados no mundo real.

2.3. Arquitetura do Modelo: Transfer Learning

Em vez de treinar uma CNN do zero, adotamos a abordagem de **Aprendizado por Transferência (*Transfer Learning*)**.

Foi escolhida a arquitetura *MobileNetV2* [1], pré-treinada no dataset *ImageNet*, devido ao seu excelente balanço entre eficiência computacional e poder de extração de características.

O modelo foi implementado da seguinte forma:

- O modelo base foi carregado com removendo a camada de classificação original.

- O modelo foi ”congelado”, preservando os pesos da *ImageNet* e evitando que fossem destruídos durante o início do treino.
- Uma ”cabeça” de classificação customizada foi adicionada ao topo para reduzir a dimensionalidade e a camada final foi fechada em 9 camadas, referente as 9 classes do conjunto.

2.4. Estratégia de Treinamento

O modelo foi compilado com o otimizador *Adam*. As métricas monitoradas foram *accuracy*, *precision* e *recall*, pois a acurácia sozinha é uma métrica enganosa em dados desbalanceados.

Utilizando a biblioteca *scikit-learn*, calculamos pesos no modo ”*balanced*”, que atribui um peso maior às classes sub-representadas (ex: ”*IllegalParking*”) e um peso menor às classes super-representadas (ex: ”*Damaged concrete structures*”). Isso força o otimizador a penalizar severamente os erros cometidos nas classes raras, obrigando o modelo a aprendê-las.

3. Resultados Experimentais

O modelo foi treinado em ambiente Kaggle (GPU Tesla T4). O *”EarlyStopping”* interrompeu o treinamento na época 26, restaurando os pesos da época 16 (a melhor época, com *‘val_loss‘* de 0.27853).

O modelo final foi então avaliado no conjunto de teste, um conjunto de dados completamente novo para o modelo.

3.1. Análise de Erros: Matriz de Confusão

A Matriz de Confusão (Figura 2) fornece uma visão qualitativa dos erros do modelo. A diagonal principal representa os acertos. Pode-se observar que o modelo obteve alta performance na maioria das classes.

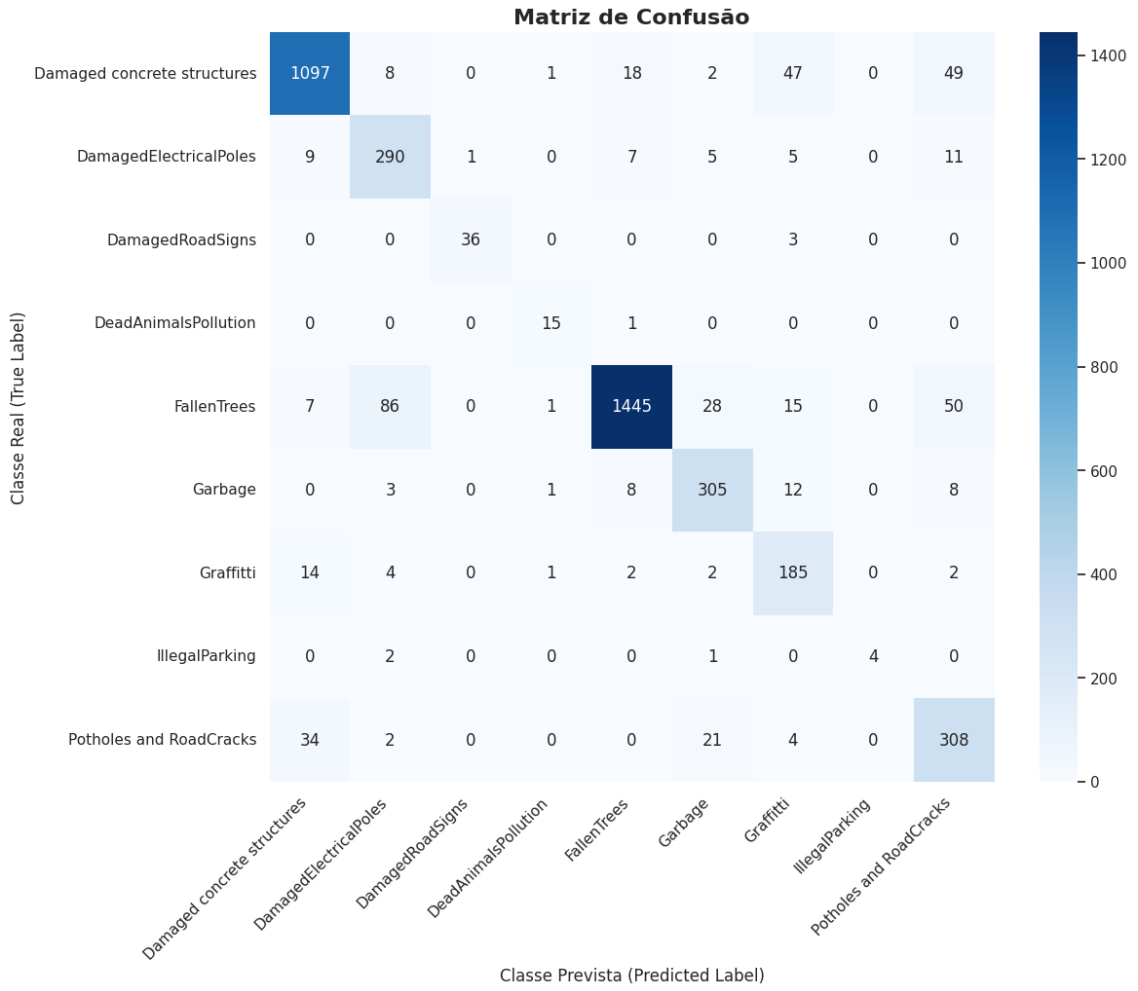

Figure 2. Matriz de Confusão no conjunto de teste.

3.2. Performance de Classificação: Curva ROC-AUC

Para uma avaliação robusta em cenários multi-classe e desbalanceados, calculamos a curva ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) na estratégia One-vs-Rest (OvR), ilustrada na Figura 3.

O modelo alcançou um *AUC Macro-average* de 0.99 e um *AUC Micro-average* também de 0.99. A média ”Macro” é particularmente importante, pois trata todas as classes com igual importância, demonstrando que o modelo foi capaz de distinguir efetivamente até mesmo as classes raras, validando nossa estratégia do peso de classes.

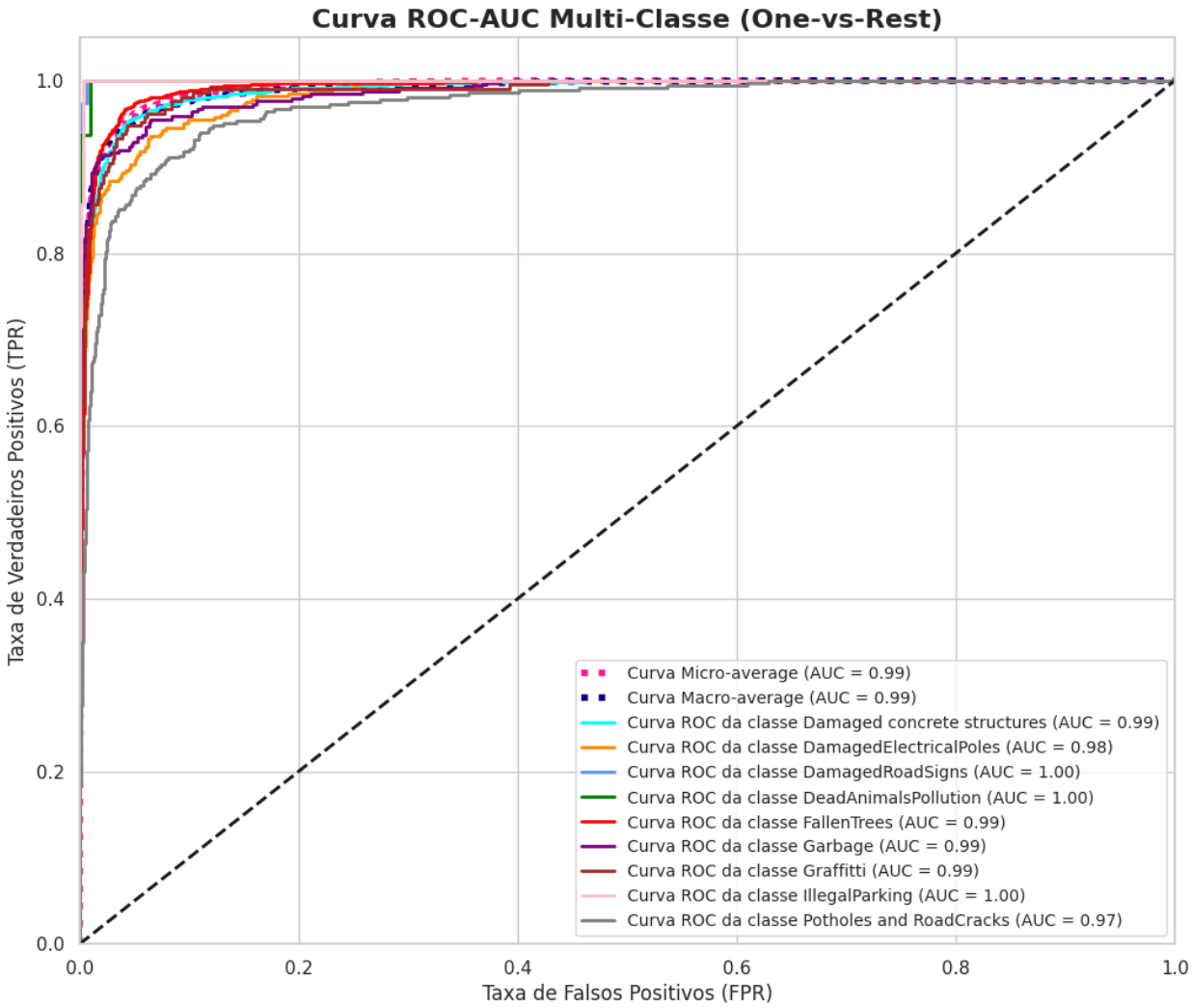

Figure 3. Curva ROC-AUC (One-vs-Rest) no conjunto de teste, com médias Micro e Macro.

4. Conclusões

Este trabalho demonstrou o desenvolvimento bem-sucedido de um pipeline de Deep Learning, o ***"Urban-Eye"***, para a classificação de problemas urbanos. O estudo comprovou que a principal dificuldade em datasets do mundo real, o desbalanceamento de classes, pode ser efetivamente mitigado através de uma metodologia combinada de Aprendizado por Transferência, Data Augmentation e, crucialmente, o uso de uma função de perda ponderada para o deesbalanceamento das classes.

O modelo final apresenta performance robusta, como evidenciado pela Matriz de Confusão e pela alta pontuação do *AUC Macro-average*.

References

[1] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Abadi, M., et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. arXiv preprint arXiv:1603.04467.