RELATÓRIO DE RESULTADOS - PROJETO 2

Projeto: "LeafGenerator"

Gustavo Choueiri
Matrícula: 232014010

relatório "completo demais" pois pretendo publica-lo no linkedin

Sumário

1 Método Implementado

Esse projeto 2 implementa um sistema de diagnóstico de doenças em folhas de plantas usando um modelo de IA Generativa baseado neste artigo:

> KATAFUCHI, Ryoya; TOKUNAGA, Terumasa. *Image-based plant disease diagnosis with unsupervised anomaly detection based on reconstructability of colors*. arXiv preprint arXiv:2011.14306, 2020.

2 Arquitetura e Pipeline Completo

O sistema completo é dividido em 5 partes principais que estão descritas abaixo.

2.1 Treinamento do Modelo Pix2Pix (Aprendizado Não Supervisionado)

2.1.1 Arquitetura do Gerador (Generator)

Baseado em U-Net com *skip connections*:

- **Encoder:** 8 camadas convolucionais com downsampling (Conv2d → BatchNorm → LeakyReLU $\alpha = 0.2$). Dimensões: $256 \times 256 \times 3 \rightarrow 1 \times 1 \times 512$.
- **Decoder:** 8 camadas transpostas convolucionais com upsampling (ConvTranspose2d → BatchNorm → Dropout 0.5 → ReLU). Dimensões: $1 \times 1 \times 512 \rightarrow 256 \times 256 \times 3$.
- **Ativação final:** Tanh (Tangente Hiperbólica) (normaliza saída para [-1, 1]).

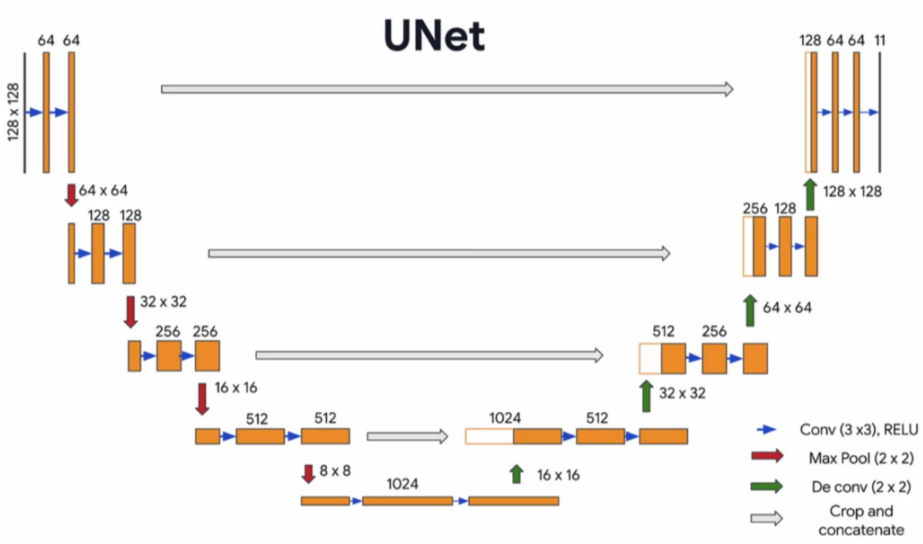

Figura 1: Diagrama da arquitetura U-Net do Gerador.

2.1.2 Arquitetura do Discriminador (Discriminator)

Utilizei um **PatchGAN**, que classifica *patches* de 70×70 ao invés da imagem inteira, dando penalidades nas estruturas locais e capturando de forma melhor as texturas.

2.1.3 Estratégia de Treinamento

- **Dataset:** 50 imagens de folhas saudáveis.
- **Épocas:** 300 (otimizado).
- **Optimizer:** Adam ($LR = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.999$).
- **Data Augmentation:** Resize, Random horizontal flip, Random rotation, Color jitter.

3 Cálculo do Índice de Reconstruibilidade de Cores (CRI)

Interpretação:

- **CRI BAIXO (≈ 0.005):** Reconstrução quase perfeita (Provável folha saudável).
- **CRI ALTO (≈ 0.010):** Diferenças significativas (Provável folha doente).

4 Otimização Automática do Limiar de Decisão

Uma das contribuições implementadas neste trabalho foi a substituição do limiar (*threshold*) de decisão fixo por uma abordagem dinâmica e adaptativa. Observou-se que definir um valor estático (ex: 0.1) para o Índice de Reconstruibilidade de Cores (CRI) é ineficaz, pois a escala absoluta dos erros de reconstrução pode variar dependendo da convergência do modelo e das características específicas do conjunto de dados.

Para solucionar isso, implementou-se um algoritmo de busca em grade (*Grid Search*) que ajusta o limiar com base na distribuição estatística dos scores de anomalia do conjunto de teste.

4.1 Maximização do F1-Score

A escolha do F1-Score como função objetivo é crucial devido ao desbalanceamento do conjunto de teste (50 amostras saudáveis vs. 100 doentes), garantindo que o modelo não privilegie excessivamente uma classe em detrimento da outra.

4.2 Resultados da Otimização

A aplicação desta técnica resultou em um limiar de **0.005178**. Esta calibração fina foi responsável pelo salto de desempenho observado, elevando a acurácia de patamares inferiores (observados com limiares fixos arbitrários) para **90.67%**, demonstrando a importância de ajustar o ponto de operação do classificador às características latentes do modelo generativo.

5 Resultados

5.1 Base de Dados

- **Treinamento:** 50 imagens (Healthy_Train50).
- **Teste:** 150 imagens (50 Healthy_Test50 + 100 Disease_Test100).

5.2 Comparação: Antes vs Depois da Otimização

A tabela abaixo compara a primeira versão (100 épocas, LR padrão) com a versão final otimizada (300 épocas, LR ajustado).

Tabela 1: Comparação de Desempenho (V1 vs V2)

Métrica	**V1 (100 épocas)**	**V2 (300 épocas)**	**Melhoria**
Acurácia	68.00%	**90.67%**	+22.67% ↑
Precisão	67.57%	**90.57%**	+22.90% ↑
Recall	100.00%	96.00%	-4.00% (trade-off)
F1-Score	80.65%	**93.20%**	+12.55% ↑
Falsos Positivos	48	**10**	-79% ↓
Separação (CRI)	0.0010	**0.0014**	+40% ↑

5.3 Matriz de Confusão Final

		Predito	
		Saudável	Doente
Saudável		**40** (TN)	10 (FP)
Doente		4 (FN)	**96** (TP)

Tabela 2: Matriz de Confusão do modelo final.

5.4 Análise Estatística do CRI

- **CRI Médio (Saudáveis):** 0.0049 ± 0.0004
- **CRI Médio (Doentes):** 0.0063 ± 0.0008
- **Threshold Otimizado:** 0.005178

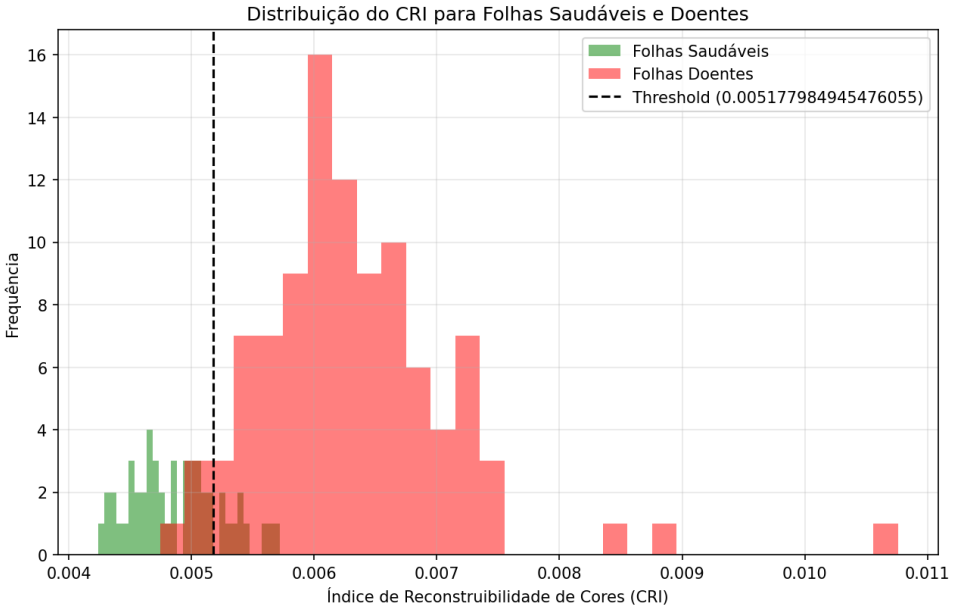

Figura 2: Distribuição do CRI para folhas saudáveis e doentes.

6 Visualizações e Explicabilidade (Grad-CAM)

Aqui foi utilizado Grad-CAM pra validar as decisões do modelo. As áreas vermelhas aqui mostram onde o modelo encontrou a maior discrepância ou características que são relevantes.

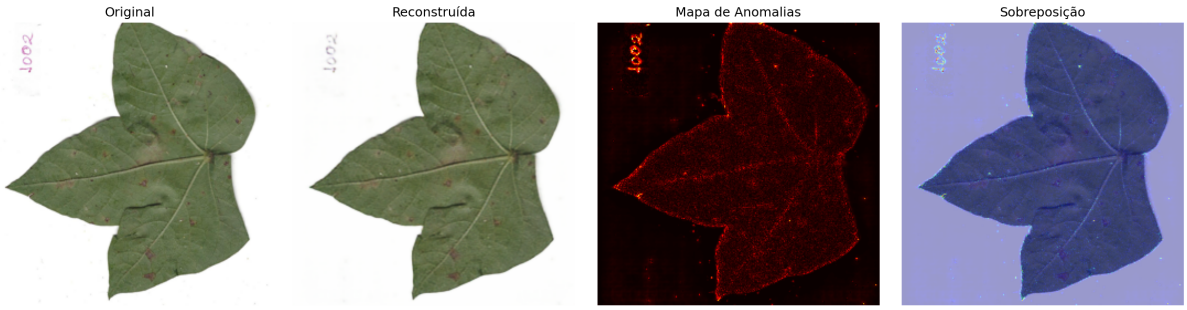

Figura 3: Comparação: Original, Reconstruída, Mapa de Anomalias e Sobreposição.

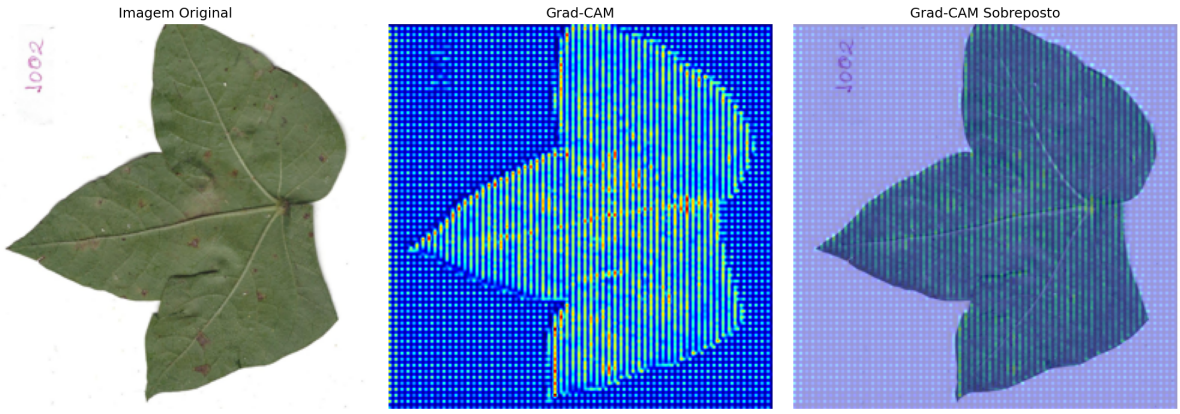

Figura 4: Explicabilidade visual usando Grad-CAM.

7 Implementação e Adaptação para Google Colab

Para viabilizar a execução eficiente em ambiente de nuvem e aproveitar a aceleração por hardware (GPU) oferecida pelo Google Colab, foi desenvolvida uma versão adaptada do sistema, consolidada no arquivo `codigo_completo_colab.py`.

Esta adaptação unifica os componentes essenciais em um fluxo de execução.

7.1 Estrutura do Script Unificado

O script consolida as seguintes etapas críticas do pipeline:

- **Definição de Arquiteturas:** As classes `Generator`, `Discriminator` e os blocos `UNetDown`/`UNetUp` são definidos diretamente no escopo de execução, eliminando a necessidade de importações complexas de arquivos externos.
- **Carregamento de Dados:** A classe `LeafDataset` implementa a interface do *PyTorch Dataset*, incluindo as transformações de pré-processamento (redimensionamento para 256×256 e normalização) necessárias para o modelo Pix2Pix.
- **Cálculo de Métricas (CRI):** A função `calculate_cri` implementa vetorialmente o cálculo da diferença absoluta média entre a imagem real e a reconstruída, gerando o mapa de anomalias.

7.2 Pipeline de Treinamento e Teste

A adaptação encapsula o ciclo de vida do modelo em duas funções principais de alto nível:

1. `train_model`: Gerencia o loop de treinamento, instanciando os otimizadores Adam, calculando as perdas adversariais (MSE) e de reconstrução (L1), e salvando *checkpoints* periodicamente.
2. `test_model`: Automatiza a avaliação, realizando a inferência em todo o conjunto de teste. Esta função integra nativamente a lógica de **Otimização de Threshold** (descrita na seção anterior), calculando dinamicamente o ponto de corte que maximiza o F1-Score e gerando visualizações estatísticas (histogramas) diretamente na saída da célula do notebook.

8 Conclusões

Este projeto foi desafiador, atingindo um **F1-Score de 93.20%**, o que mostra demonstra a viabilidade do uso de IA Generativa não supervisionada para detectar doenças em plantas, mesmo com datasets de treinamento reduzidos (apenas 50 imagens).

Foi implementado:

- Otimização automática de threshold.
- Explicabilidade via Grad-CAM e mapas de anomalias.
- Utilização da GPU (RTX 4060)(10min de treinamento).

Instruções de Reprodução

Para reproduzir os resultados:

```
python train.py --epochs 300 --lr 0.0001
python test.py
```

Referências

[1] KATAFUCHI, Ryoya; TOKUNAGA, Terumasa. Image-based plant disease diagnosis with unsupervised anomaly detection based on reconstructability of colors. *arXiv preprint arXiv:2011.14306*, 2020.

[2] SELVARAJU, Ramprasaath R. et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE international conference on computer vision*. 2017. p. 618-626.