Documentação Projeto de Colorização de Rostos com GANs

---

1. Objetivo

Este projeto visa realizar a **colorização automática de imagens em preto e branco de rostos humanos** utilizando redes neurais generativas adversariais (**GANs**). O foco está em reconstruir imagens realistas a partir de fotos em tons de cinza, melhorando a nitidez e aplicando filtros estéticos ao resultado final.

---

2. Dataset

- **Fonte**: UTKFace Dataset
- **Link**: https://www.kaggle.com/datasets/jangedoo/utkface-new
- **Conteúdo**: Imagens de rostos humanos com variações de idade, etnia e sexo
- **Formato**: .jpg
- **Quantidade utilizada**: 10.000 imagens
- **Quantidade disponível**: 23.000 imagens
- **Pasta após extração**: /content/UTKFace/crop_part1
- **Pré-processamento**:
    - Redimensionamento para 128x128
    - Conversão para escala de cinza (entrada)
    - Normalização de pixel (0–1)

---

3. Estrutura do Projeto

a) Gerador — GeneratorUNet

- Arquitetura baseada em **U-Net** com blocos *Encoder-Decoder*
- Entrada: imagem 1 canal (P&B)
- Saída: imagem 3 canais (RGB)
- Função de ativação final: Sigmoid

b) Discriminador

- Recebe pares (imagem P&B, imagem RGB)
- Tenta distinguir se a imagem colorida é real ou gerada

- Arquitetura CNN com 3 camadas

---

4. Hiperparâmetros

Parâmetro	Valor
Tamanho imagem	128 x 128
Batch size	16
Épocas	120
Otimizador	Adam
Learning Rate	0.0002
Betas (Adam)	(0.5, 0.999)
Função GAN	BCEWithLogits
Função Recon	L1Loss
Peso da L1	100.0

---

5. Processo de Treinamento

Para cada batch de imagens:

1. **Discriminador (D)**:
    - Recebe imagem real → rotula como 1
    - Recebe imagem gerada → rotula como 0
    - Calcula Loss_D = média(real, fake)
    - Otimiza parâmetros
2. **Gerador (G)**:
    - Gera imagem fake
    - É avaliado pelo Discriminador
    - Calcula Loss_G = Loss_GAN + 100 × Loss_L1
    - Otimiza parâmetros

A combinação adversarial + reconstrução garante fidelidade na cor e na estrutura facial.

---

6. Pós-Processamento

- Filtros aplicados com OpenCV:
    - cv2.detailEnhance (realce de detalhes)
    - cv2.bilateralFilter (suavização sem perda de bordas)
- Conversão RGB para BGR para salvar com cv2.imwrite

---

7. Resultados

Saídas geradas

- gan_colorida.png: imagem colorizada pela GAN
- gan_filtrada.png: imagem com filtro estético
- loss_gan.png: curva de perdas

8. Avaliação

Curva de Perda

- Loss_G tende a estabilizar ao longo das épocas
- Loss_D oscila moderadamente, indicando equilíbrio adversarial

Qualitativo

- Resultados visualmente coerentes
- Boa distribuição de tons de pele e fundo
- Pequenos artefatos em bordas ou regiões com baixa iluminação

---

9. Limitações

- Resolução limitada (128x128)
- Tempo de treino elevado em CPU
- Modelo restrito a rostos humanos
- Não utiliza **perceptual loss** ou **modelos pré-treinados**
- Resultado pode ter tons irreais em expressões exageradas

---

10. Melhorias Futuras

- Aumentar resolução para 160×160 ou 256×256

- Substituir L1Loss por **Perceptual Loss (VGG16)**
- Usar **GANs condicionais** com rótulos (sexo, idade)
- Aplicar **data augmentation** no dataset

---

11. Execução e Reprodutibilidade

Pré-requisitos:

- Python 3.9+
- PyTorch 2.x
- OpenCV
- Matplotlib
- PIL (Pillow)

Execução no Google Colab:

1. Baixe o dataset no kaggle, extraia o arquivo e renomeie para utk.zip
2. Suba utk.zip no Colab
3. Rode o script completo em uma célula
4. Aguarde o treinamento (~1h30 em GPU)
5. Verifique /content/resultados_gan_color/