BRENO DA SILVA NOGUEIRA - 12400392

DAVIDSON VIEIRA DE SOUZA - 12611668

JOÃO VICTOR DIAS BANDINI - 12543061

JULIANA ALMEIDA SANTOS - 12566178

Exercício-Prático: Síntese de Imagem

Docente: João Bernardes

Disciplina: Computação Gráfica (ACH2117)

São Paulo/SP

2024

Sumário

1) Introdução

Dentro da Ciência da Computação, existe uma área vasta e dinâmica focada na manipulação e geração de imagens digitalmente, esse campo é a Computação Gráfica. Esse campo de estudo abrange desde a criação de gráficos 2D e 3D até a renderização de cenas complexas, o que garante à área uma aplicação em diferentes cenários, como entretenimento, medicina, engenharia e design. Abordando o contexto acadêmico, o processamento de imagens é uma disciplina fundamental dentro da computação gráfica, porque permite a análise e modificação de imagens digitais para diferentes finalidades.

Sendo assim, neste trabalho prático, o desenvolvimento será focado no processamento de imagens com o objetivo de desenvolver um modelo de reconhecimento facial. A proposta é interessante, pois o reconhecimento facial é uma técnica muito utilizada nas áreas de segurança, autenticação e organização de imagens, dentre outras aplicações. Porém, é de amplo conhecimento que a eficiência de um modelo de reconhecimento está totalmente atrelada à qualidade e diversidade dos dados utilizados em seu treinamento.

Então, a fim de aprimorar o treinamento do modelo proposto, foi utilizada a técnica de processamento de imagens Data Augmentantion. Essa técnica consiste em um conjunto de métodos usados com o intuito de aumentar a quantidade de dados de treinamento sem a necessidade de coletar novas informações, nesse caso, imagens. Essa técnica envolve a aplicação de diferentes transformações nas imagens, como rotações, espelhamentos, redimensionamentos e alterações de brilho, o que resulta em variações das imagens originais. A aplicação dessa técnica é interessante porque, teoricamente, ajuda a melhorar a generalização do modelo, tornando-o mais robusto e eficaz ao lidar com diferentes condições e variações de imagens de entrada.

Portanto, partindo dessa informação teórica, definiu-se que o principal objetivo do exercício prático é descobrir se o uso de Data Augmentation realmente influencia positivamente a performance do modelo de reconhecimento facial. Nas

seções a seguir, serão explorados o desenvolvimento do exercício e também a utilização do modelo obtido.

2) Desenvolvimento

Para criar um sistema de reconhecimento facial, poderiam ser utilizadas várias técnicas diferentes para chegar no mesmo resultado, e para este trabalho, foi decidido por utilizar *Aprendizado de Máquina por Similaridade*. Sendo esse um subtópico de Machine Learning, que como o nome indica, cria ligações entre imagens por medidas de similaridade.

A implementação do código do modelo de Machine Learning fica à cargo da biblioteca de criação de redes neurais criada pelo Google, o *Tensorflow*. Junto dela, foi usado também o *Tensorflow Similarity*, sendo este uma biblioteca derivada que foi utilizada para a implementação de algoritmos específicos de Similaridade.

Para realizar o processamento de imagens, foram utilizadas duas bibliotecas próprias para visão computacional e manipulação de imagens, sendo elas o *OpenCV* (uma biblioteca em C++ com bindings para Python) e o *PIL (Python Imaging Library)*. Com elas, foi feito o processamento (data augmentation) das imagens com bastante facilidade, além de ser uma ótima interface para manipular as imagens até que as mesmas cheguem no formato em que o modelo consumirá.

O Data Augmentation se dá por uma função de mesmo nome que vai aplicar nas imagens os mais diversos filtros e transformações com base uma chance aleatória, sendo esses filtros:

- Espelhamento vertical da Imagem - 50% chance;
- Rotação aleatória da imagem em um eixo de até 90º para esquerda ou direita - 40% chance;
- Mudança de escala de cores - 20% chance;
    - Escala de cinzas - 50% chance;
    - Preto e Branco - 50% chance;
- Filtro aleatório - 90% chance;
    - Blur Gaussiano;
    - Filtros de Mínima, Máxima, Mediana, Moda;
    - Dentre outros.

Usou-se como base para a implementação do modelo de Machine Learning um exemplo da documentação do próprio *Tensorflow Similarity*, em que é

apresentado o esqueleto de uma análise que devolve um modelo de similaridade, o qual pode ser encontrado aqui. E, para o tratamento da pipeline de imagens que foi utilizada, adotou-se outro tutorial, dessa vez do *Tensorflow* base, que ensina a como criar um modelo que é alimentado por um grande dataset.

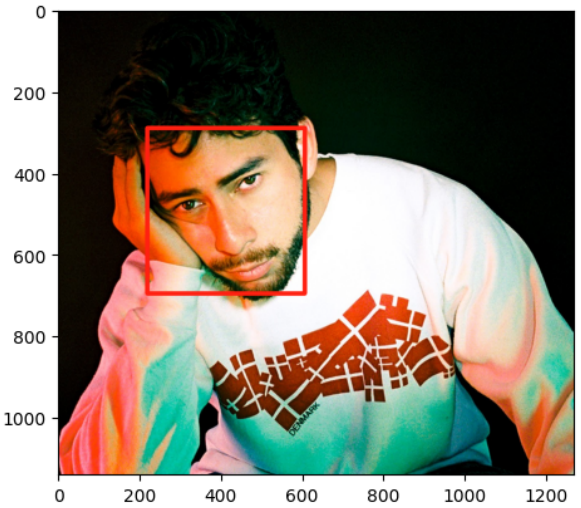
As imagens utilizadas aqui, a fim de poupar o trabalho de classificação manual de várias fotos pessoais, foram retiradas do Google Fotos ao procurar nomes famosos. Os famosos que foram escolhidos pela equipe arbitrariamente foram: *Billie Eilish*, *Childish Gambino*, *Bruno Mars*, *Taylor Swift*, *Kanye West*, *Tyler The Creator e The Weeknd*.

Todas as fotos usadas foram enviadas juntamente ao trabalho; foram mandadas as originais (não processadas a priori) para que o experimento possa ser replicado localmente. Contudo, é normal que os resultados finais sejam ligeiramente diferentes, pela natureza de modelos de machine learning e por questões de distribuição de dataset.

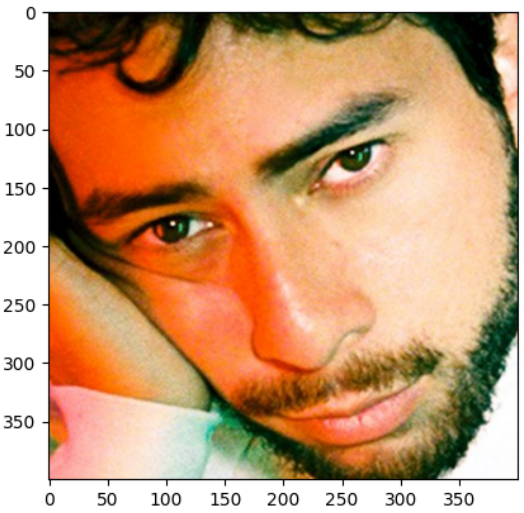
I. Descrição do Código-Fonte

O código segue um passo a passo bastante simples:

1) Limpar o dataset de imagens a fim de remover imagens corrompidas ou formatos não suportados;
2) Realizar um pré-processamento das imagens:
    - Foi utilizada uma biblioteca (mtcnn) para detectar se há rostos na imagem, e, se houver, quais são as suas coordenadas.



3) Em seguida, foi recortado o rosto encontrado em uma imagem de 400x400 pixels que foi salva em uma pasta separada.



4) Realização do Data Augmentation no Dataset
    - Selecionou-se uma imagem e aplicou-se um conjunto de filtros de forma randômica, sendo que uma imagem pode ter uma ou mais variantes melhoradas ou, até mesmo, nenhuma variante ainda.

Legenda: Uma foto com filtro de espelhamento, P&B e rotação.

    - Foi colocada a tag “-aug-” no nome das imagens melhoradas para depois ser possível aplicar um filtro facilmente do dataset.

5) Com os dados em mãos, primeiramente foi criado um modelo de Similaridade sem as imagens melhoradas, e adotou-se algumas métricas de treinamento e de validação.
    - Precisão, Recall, F1-Score, Perda de Treino, etc.

6) Por fim, foi treinado um segundo modelo com a adição das imagens melhoradas, e as métricas foram obtidas, também, para comparar com o modelo anterior.

O código foi escrito em formato de Notebook do Python, ou seja, deve ser lido como um script sequencial, sem um ponto de entrada “main”. Porém, o código foi pensado para que cada célula (separação de blocos do código) tenha um significado semântico, ou seja, cada bloco “faz um pedaço” do passo a passo descrito acima. Onde for conveniente, estará anotado com comentários o que cada parte do script faz.

Importante notar que o notebook deve ser lido como um “printscreen” do código como um todo, já que as saídas do código estão juntas com o código em si, como um documento mostrando o que foi feito para chegar nas análises aqui descritas. Portanto, é interessante analisar o notebook como foi entregue antes de rodá-lo pela primeira vez, e se for rodar, é interessante brincar com as constantes deixadas no começo do código para chegar em resultados diferentes.

Um último ponto a se considerar é que esse código é um tanto pesado de se rodar, tendo a duração de sua execução completa na casa de horas. E, dependendo da configuração do computador em que o código estiver rodando, é possível que o treinamento do modelo seja demorado demais para ser prático, ou, até mesmo, não tenha os recursos para treiná-lo, especialmente se não houver uma GPU. Porém, se houver uma, é necessário a instalação do toolkit CUDA e que seja rodado na plataforma linux, o que pode representar uma dor de cabeça.

Para expressar se não é possível rodar no computador, geralmente o Tensorflow irá subir um erro de “ResourceExhaustedError”, ou, no stack trace do erro, terá alguma menção de falta de memória. Para tentar rodar mesmo assim, é

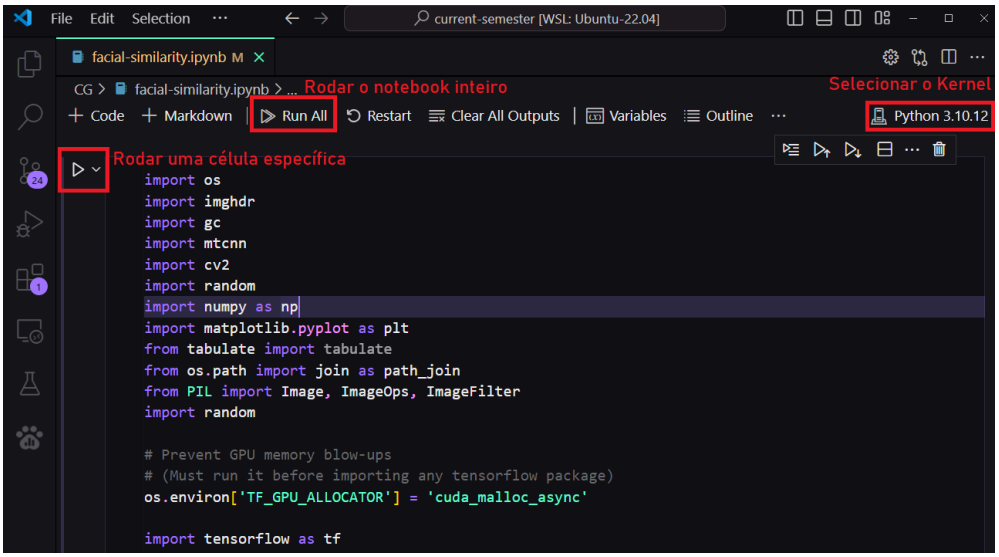
possível diminuir as dimensões da constante “INPUT_SIZE”, ou mexer nos pesos que compõem o modelo na função “bootstrap_model()”.

Se mesmo assim não for possível treinar o modelo localmente para a correção, será disponibilizado o modelo que foi treinado durante o desenvolvimento do projeto para que seja possível rodar o restante do notebook. Baixe o arquivo “pretrained_models” da pasta compartilhada, extraia na raiz do projeto e coloque a constante “PRETRAINED” como “True” (Atenção! A primeira letra deve ser maiúscula). Se essa opção for usada, não vai ser possível utilizar imagens próprias, uma vez que é um modelo treinado para reconhecer apenas um subconjunto de pessoas.

II. Instruções

Para executar o código localmente, é necessário seguir tais passos:

1) Interpretador Python instalado
    - O desenvolvimento foi feito e testado na versão Python 3.10.12, contudo, qualquer versão igual ou superior à 3.9 deve ser suficiente.
2) Instalação das dependências Python
    - Na raíz do projeto existe o arquivo “requirements.txt”, que dentro possui todas as bibliotecas necessárias e suas respectivas versões. Para instalar rode no terminal o seguinte comando:
        - pip install -r requirements.txt
3) Ambiente Jupyter Instalado
    - Nesse ponto, é recomendado que se utilize o Visual Studio Code e instale as duas extensões abaixo, uma vez que elas resolvem todos os requisitos necessários para rodar notebooks direto na IDE, sem ter que rodar manualmente um servidor Jupyter:
        - ms-toolsai.jupyter (ele pode abrir um modal quando rodar o notebook avisando que não tem o jupyter, clique em “sim” para instalar)
        - ms-python.python



Legenda: Como usar a interface do VsCode

- Contudo, se for um impedimento usar esse método, é possível instalar o servidor Jupyter manualmente:
    - Instalar: Installing the classic Jupyter Notebook interface
    - Rodar o Notebook: Running the Notebook

4) Obter imagens

- Na pasta compartilhada, estão sendo enviadas as fotos utilizadas no treinamento do modelo. Basta extrair a pasta “rawphotos” na raíz do projeto e tudo deve funcionar perfeitamente. Contudo, se quiser usar fotos personalizadas:
    - Crie uma pasta na raíz do projeto e dentro dela crie subpastas, cada uma representando uma pessoa (coloque o nome da pessoa como nome da subpasta) e dentro coloque várias imagens em que somente essa pessoa aparece;
    - Mude a string na constante “IMG_PATH” para o nome da pasta que você criou;
    - Alternativamente, você pode inserir novas subpastas dentro da pasta “rawphotos” e assim utilizará as fotos enviadas na pasta compartilhada e as customizadas.

3) Resultados

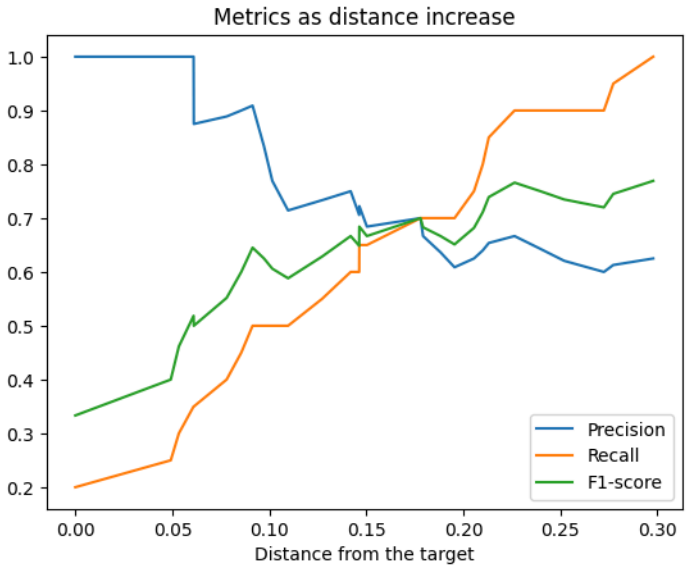
Para medir o sucesso de ambos os modelos, serão checadas quatro métricas de qualidade, sendo elas:

- **Precisão**: Métrica que mede a razão entre Positivos Verdadeiros e Falsos Positivos nas predições;
- **Revocação**: Métrica que mede a razão entre Positivos Verdadeiros e Falsos Negativos nas predições;
- **Acurácia Binária**: Mede quantas predições foram corretas do total de predições (no caso, se é ou não a pessoa certa) ;
- **F1-Score**: Média Harmônica entre precisão e revocação .

Rodando o modelo sem Data Augmentation, foram obtidas as seguintes métricas provindas do passo de calibração do modelo:

Precisão	Revocação	Acurácia Binária	F1-Score
62,5%	100%	62,5%	77%

Ou seja, ele sabe reconhecer quando uma pessoa não é outra, porém, sofre um pouco para entender qual é a pessoa certa.



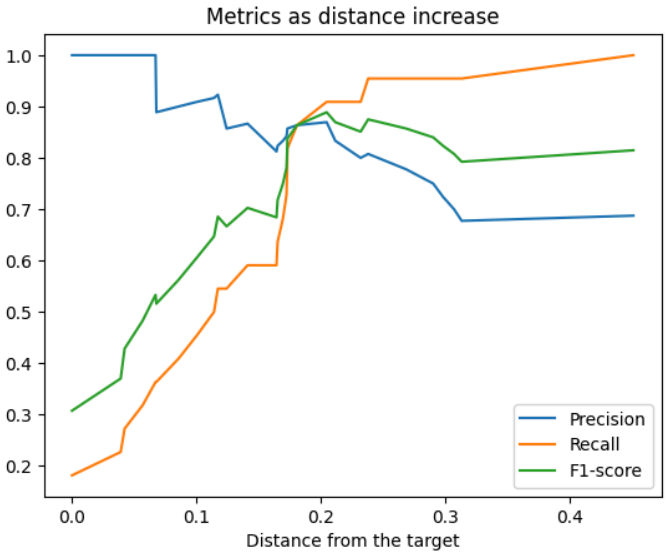
Esse gráfico deve ser lido como: Quanto as métricas variam quando a distância da medida de similaridade entre duas imagens aumenta? Pode-se ver que quando a distância de similaridade se aproxima de 0, a precisão é de 100%, o que é

de se esperar, contudo, quanto mais longe uma imagem é da outra, menos precisa a predição é. Nessa situação, a revocação aumenta, como esperado.

Rodando o modelo com data augmentation, obteve-se o seguinte:

Precisão	Revocação	Acurácia Binária	F1-Score
87%	90%	62,5%	88%

Agora o modelo reconhece de uma maneira mais apurada quem é a pessoa certa quando realiza a classificação, porém em detrimento de um pouco da revocação. Nesse caso, não é muito relevante ter uma revocação tão certeira, pois o maior interesse é em saber quem é a pessoa, e não em quem ela não é.



Na curva de métricas, pode-se ver que a precisão não tem um declínio tão acentuado, e seu platô (em torno de 70%) acontece ligeiramente mais para cima do que o modelo anterior (cerca de 60% de precisão).

4) Conclusões e considerações finais

Comparando tanto as métricas quanto as curvas em relação à distância das métricas entre duas imagens, foi possível observar uma melhora notável utilizando Data Augmentation. Contudo, ainda assim trata-se de um modelo com uma performance geral um tanto medíocre (coincidentemente 62,5% nos dois casos), o que pode acontecer por diversos motivos, como:

- O dataset utilizado não é o mais refinado possível, visto que foi retirado do Google Fotos, e com qualidades de imagens bastante variadas;
- Por ser um modelo que necessita de bastante poder computacional para criar um modelo de pesos grandes o suficiente para ter uma acurácia interessante, foi necessário reduzi-lo para criar um modelo mais compacto que pudesse ser mais facilmente executado;
- O algoritmo de criação de novas imagens poderia ser mais refinado para criar ainda mais distorções nas fotos, porém de maneira mais natural, para imitar como uma foto real seria inserida no modelo

Em suma, seria interessante analisar mais a fundo como os parâmetros do aprendizado por similaridade funcionam para criar um modelo mais robusto e mais específico para o caso de uso apresentado, e, para isso, vir acompanhado de um dataset com uma curadoria maior.

No mais, notou-se no decorrer desse trabalho a importância que o estudo de processamento de imagens representa dentro da visão computacional no âmbito de machine learning, corroborando a dinamicidade da Computação Gráfica em diferentes áreas da Computação.

5) Bibliografia e Código-grafia

Carregar e pré-processar imagens. TensorFlow. Disponível em: < https://www.tensorflow.org/tutorials/load_data/images?hl=pt-br#create_a_dataset>.

Similarity. GitHub. Disponível em: <https://github.com/tensorflow/similarity/blob/master/examples/supervised_hello_world.ipynb>.

TensorFlow Similarity Supervised Learning Hello World. GitHub. Disponível em: <https://github.com/tensorflow/similarity/blob/master/examples/supervised_hello_world.ipynb>.

Pillow (PIL Fork) 10.3.0 documentation. Pillow. Disponível em: <https://pillow.readthedocs.io/en/stable/reference/index.html>.

Métricas de Avaliação: acurácia, precisão, recall… quais as diferenças? Medium. Disponível em: <https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>.