PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Segundo Semestre de 2022 Exercício-programa
Data de entrega: 23/10/2022 (domingo) até 23:59 horas

O objetivo deste exercício é colorir automaticamente imagens em níveis de cinza. Como o esforço computacional necessário para treinar uma rede capaz de colorir uma imagem genérica está fora das possibilidades deste exercício, vamos limitar o domínio do problema, colorindo apenas as faces frontais de um banco de dados.

O site:
https://fei.edu.br/~cet/facedatabase.html
contém bancos de dados de faces humanas. Baixei 400 imagens JPG coloridas, numeradas de 001 a 200, com 360×260 pixels, de rostos frontais com expressões neutras (*???ac.jpg*) e sorridentes (*???bc.jpg*). Redimensionei as imagens originais para 384×256 pixels, para que sejam divisíveis por 2 várias vezes, facilitando processá-las com redes convolucionais (coluna *a* da figura 1).

No problema de colorir, normalmente o sistema de cores CieLAB é utilizado (veja o anexo A). Nesse sistema, o componente *L* é o nível de cinza do pixel e os componentes *A* e *B* especificam a coloração do pixel. Assim, converti as imagens do sistema RGB para sistema CieLAB e extraí o componente *L*, obtendo as imagens em níveis de cinzas (*al.jpg e *bl.jpg, coluna do meio da figura 1).

As 800 imagens assim obtidas (400 coloridas *????c.jpg* e 400 imagens em níveis de cinza *????l.jpg*) estão zipadas como *feiFrontG2C.zip* e armazenadas em:
WEB: http://www.lps.usp.br/hae/psi3472/ep-2020/feiFrontG2C.zip OU
Google Drive: https://drive.google.com/file/d/1Y183sWVCao0-zgp1_k6dhGJZbj9MjuOg/view?usp=sharing

Vamos usar os 300 pares de imagens de entrada-saída numeradas de 001 a 150 como exemplos de treino. Vamos usar os 100 pares de imagens numeradas de 151 a 200 para teste.

Por exemplo, vamos usar o par de imagens (*001al.jpg*, *001ac.jpg*) como exemplos treino de entrada-saída. Dada a imagem em níveis de cinza *001al.jpg* (componente *L* do CieLAB de *001ac.jpg*) precisamos colori-la, isto é, achar os componentes *A* e *B* do CieLAB de *001ac.jpg* que especificam as cores dos pixels. Os componentes *A* e *B* de *001ac.jpg* podem ser obtidas convertendo a imagem do sistema RGB para CieLAB e pegando as duas últimas bandas de cores (veja anexo A).

Depois de terminar o treino, gerando o *modelo.h5*, vamos aplicar o modelo obtido nas imagens de teste. Vamos colorir a imagem em níveis de cinza ????*l.jpg* obtendo a imagem colorida processada ????*p.jpg*. Se o modelo for bom, a imagem ????*p.jpg* (processada) deve ser semelhante à imagem colorida original ????*c.jpg*.

A coluna *c* da figura 1 mostra algumas imagens de teste coloridas desta forma. As imagens coloridas pela minha rede estão com qualidade visual bastante boa, exceto pela cor do cachecol verde que a rede coloriu como marrom. Note que não há como rede neural descobrir qual era a cor original do cachecol olhando somente a imagem em níveis de cinza. No site:
deepai.org/machine-learning-model/colorizer

tem um aplicativo que colore imagens genéricas em níveis de cinza. A coluna *d* mostra as imagens coloridas por esse site.

(a) Imagens coloridas originais QY.	(b) Imagens em níveis de cinza QX.	(c) Imagens QP coloridas pela minha rede.	(d) Imagens coloridas pelo site deepai.org.

Figura 1: (a) Imagens coloridas originais; (b) conversão para níveis de cinza (componente *L* do sistema CieLAB); (c) imagens coloridas pela minha rede; e (d) imagens coloridas pelo site deepai.org/machine-learning-model/colorizer. Note que não há como rede descobrir que a cor original do cachecol da imagem da terceira linha era verde.

Usei MSE (loss='mean_squared_error') para medir a diferença entre os componentes AB das imagens originais e os das imagens coloridas artificialmente, com os valores de AB indo de -0,5 a +0,5. Obtive erro MSE dos componentes AB de treino de 0,000171 de teste 0,000199. O seu programa deve obter obrigatoriamente erros MSE_AB menores que 0,00023 e 0,00026 para treino e teste. Porém, evidentemente, quanto menor for o erro obtido, melhor.

Você deve criar um notebook Colab com quatro células descritas abaixo. Se você for resolver o exercício num computador local, pode escrever quatro programas Python para serem executados num mesmo diretório. As células 1 e 4 estão dadas – a primeira faz download do BD e a quarta calcula o erro de saída. Você só precisa escrever as células 2 e 3.

Use obrigatoriamente a convenção de nomes de arquivos adotado neste enunciado:
a) Use as imagens de 001 a 150 para treino e 151 a 200 para teste.
b) Processe todas as imagens em níveis de cinza *????l.jpg* do diretório default (tanto os de treino como de teste) e escreva as imagens coloridas processadas *????p.jpg* correspondentes também no diretório default. Caso não siga esta convenção, a célula 4 não conseguirá calcular corretamente o erro MSE do seu programa.

Célula 1: Download) Pega o BD da internet e o descompacta no diretório default (tanto no Colab quanto no computador local). O código está abaixo.

```
#get_bd.py
url="http://www.lps.usp.br/hae/psi3472/ep-2020/feiFrontG2C.zip"
import os; nomeArq=os.path.split(url)[1]
if not os.path.exists(nomeArq):
  print("Baixando o arquivo",nomeArq,"para diretorio default",os.getcwd())
  os.system("wget -nc -U 'Firefox/50.0' "+url)
else:
  print("O arquivo",nomeArq,"ja existe no diretorio default",os.getcwd())
print("Descompactando arquivos novos de",nomeArq)
os.system("unzip -u "+nomeArq)
```

Célula 2: Treino) Você deve escrever esta célula que treina o modelo a partir das imagens de treino entrada-saída de 001 a 150 e salvá-lo como *modelo.h5*. Quando o treino terminar, sugiro que faça download do *modelo.h5* no computador local, pois Colab irá apagá-lo quando terminar a seção.

Célula 3: Predição) Você deve escrever a célula que aplica o modelo treinado em todas as imagens em níveis de cinza *????l.jpg* do diretório default. O seu programa deve gerar 400 imagens coloridas *????p.jpg* processando as 400 imagens em níveis de cinza *????l.jpg* (gere saídas tanto para as imagens de treino como para as imagens de teste). Por exemplo, processando imagem em níveis de cinza *151al.jpg* pela rede, deve ser gerada imagem colorida *151ap.jpg*.

Célula 4: Cálculo de erro) Calcula a média de MSE entre as imagens *????c.jpg* e *????p.jpg* para os conjuntos de treino e de teste. Rode este código para verificar o desempenho do seu programa (MSE de treino e de teste, medidos no sistema de cores RGB e nos componentes *AB* do sistema CieLAB). A métrica principal para medir o desempenho do seu modelo é MSE_AB calculado nos dados de teste.

```python
#mse.py
import cv2
import numpy as np; np.random.seed(7)
import sys

def MSE_RGB(indice):
  ay=cv2.imread("%03dac.jpg"%indice,1)
  if ay is None: sys.exit("Erro leitura %03dac.jpg"%indice)
  ap=cv2.imread("%03dap.jpg"%indice,1)
  if ap is None: sys.exit("Erro leitura %03dap.jpg"%indice)
  ay = ay/255.0-0.5; ap = ap/255.0-0.5
  mse1 = np.square(np.subtract(ay,ap)).mean()

  ay=cv2.imread("%03dbc.jpg"%indice,1)
  if ay is None: sys.exit("Erro leitura","%03dbc.jpg"%indice)
  ap=cv2.imread("%03dbp.jpg"%indice,1)
  if ap is None: sys.exit("Erro leitura","%03dbp.jpg"%indice)
  ay = ay/255.0-0.5; ap = ap/255.0-0.5
  mse2 = np.square(np.subtract(ay,ap)).mean()

  mse = (mse1+mse2)/2
  return mse

def MSE_AB(indice):
  ay=cv2.imread("%03dac.jpg"%indice,1)
  if ay is None: sys.exit("Erro leitura %03dac.jpg"%indice)
  ay=cv2.cvtColor(ay,cv2.COLOR_BGR2LAB);
  ap=cv2.imread("%03dap.jpg"%indice,1)
  if ap is None: sys.exit("Erro leitura %03dap.jpg"%indice)
  ap=cv2.cvtColor(ap,cv2.COLOR_BGR2LAB);
  ay = ay/255.0-0.5; ap = ap/255.0-0.5
  mse1 = np.square(np.subtract(ay[:,:,1:3],ap[:,:,1:3])).mean()

  ay=cv2.imread("%03dbc.jpg"%indice,1)
  if ay is None: sys.exit("Erro leitura","%03dbc.jpg"%indice)
  ay=cv2.cvtColor(ay,cv2.COLOR_BGR2LAB);
  ap=cv2.imread("%03dbp.jpg"%indice,1)
  if ap is None: sys.exit("Erro leitura","%03dbp.jpg"%indice)
  ap=cv2.cvtColor(ap,cv2.COLOR_BGR2LAB);
  ay = ay/255.0-0.5; ap = ap/255.0-0.5
  mse2 = np.square(np.subtract(ay[:,:,1:3],ap[:,:,1:3])).mean()

  mse = (mse1+mse2)/2
  return mse

#<<<<<<<<< main <<<<<<<<<<<<
soma=0.0
for i in range(1,151): soma += MSE_RGB(i)
print("Training MSE_RGB=",soma/150)

soma=0.0
for i in range(151,201): soma += MSE_RGB(i)
print("Test MSE_RGB=",soma/50)

soma=0.0
for i in range(1,151): soma += MSE_AB(i)
print("Training MSE_AB=",soma/150)

soma=0.0
for i in range(151,201): soma += MSE_AB(i)
print("Test MSE_AB=",soma/50)
```

Saída do meu programa:

```
Training MSE_RGB= 0.000325
Test MSE_RGB= 0.000366
Training MSE_AB= 0.000170
Test MSE_AB= 0.000198
```

Obs. 1: Este exercício deve ser resolvido preferencialmente em duplas ou individualmente.

Obs. 2: Em princípio, deve usar o mesmo ambiente de programação utilizado em aula (Python, Keras, OpenCV, Colab). Se quiser utilizar outro ambiente, converse antes com o professor.

Obs. 3: Entregue um vídeo de no máximo três minutos explicando o funcionamento dos seus programas e os resultados obtidos. No vídeo deve aparecer em algum momento o seu rosto e um documento seu. É necessário que o vídeo contenha áudio, pois é muito difícil entender um vídeo sem a explicação falada. O vídeo não pode ter sido acelerado para diminuir a duração, pois dificulta entender a fala. Você pode enviar o vídeo em si (.mkv, .mp4, .avi, etc.) ou um link para o vídeo (youtube, google drive, etc). No segundo caso, deve assegurar que todos os professores (Hae e Maurício - hae.kim@usp.br, mlisboa@usp.br) tenham acesso ao seu vídeo.

Obs. 4: Entregue os programas-fontes ou link para notebook Google Colab. Armazene as imagens de saída *????p.jpg* e o modelo *modelo.h5* obtidos no Google Drive e envie o link. Você deve assegurar que todos os professores tenham acesso a esse material.

Obs. 5: Não se esqueçam de escrever os nomes da dupla (ou do único aluno, escrevendo: "exercício feito individualmente") em três lugares diferentes: no campo "comentários sobre o envio", no início do vídeo e no início dos programas-fontes. Caso contrário, um dos alunos da dupla pode ficar sem a nota.

Obs. 6: Envie o material através de edisciplinas. Dentro do prazo, você pode substituir o material anterior por um novo.

Nota para mim: A solução privada está em https://colab.research.google.com/drive/1A3w9G5sulAs-o5Qy2yGSDhplckwcImTx?usp=sharing

Anexo A: Sistema de cores CieLAB

CieLAB (também conhecido como CIE L*a*b*) é um espaço de cores projetado para que uma certa quantidade de mudança nesse espaço corresponda aproximadamente à mesma quantidade de mudança na percepção visual humana. Por ter sido projetado para se aproximar à visão humana, CieLAB é utilizado nos problemas onde é importante que o erro de coloração medido no espaço das cores seja equivalente ao erro de coloração percebido por seres humanos [wikiLAB].

Esse sistema expressa a cor como três valores L, A e B. OpenCV utiliza lightness *L* de preto (0) a branco (100); componente *A* de verde (-127) a vermelho (+127); e componente *B* de azul (-127) a amarelo (+127) [OpenCV]. Nas matrizes em float32, OpenCV armazena um pixel em CieLAB nesse formato ($0 \le L \le 100$, $-127 \le A \le +127$, $-127 \le B \le +127$). Nas matrizes em uint8, OpenCV faz a conversão abaixo para que todos os valores estejam entre 0 e 255:

$L_8 = L*255/100$, $A_8 = A+128$, $B_8 = B+128$.

As imagens ????*l.jpg* correspondem ao componente L_8 (de 0 a 255).

Para alimentar rede neural e medir o erro MSE, estou trabalhando com imagens CieLAB armazenadas em matrizes float32 no intervalo de -0.5 a +0.5:

$L_5 = L_8/255-0.5$, $A_5 = A_8/255-0.5$, $B_5 = B_8/255-0.5$.

Assim, todos os componentes L_5, A_5 e B_5 estarão no intervalo de -0.5 a +0.5 para alimentar a rede e para medir o erro MSE.

OpenCV possui a função cvtColor que converte imagem RGB (ou BGR) de/para CieLAB:

```
lab=cv2.cvtColor(bgr,COLOR_BGR2LAB)
bgr=cv2.cvtColor(lab,COLOR_LAB2BGR)
lab=cv2.cvtColor(rgb,COLOR_RGB2LAB)
rgb=cv2.cvtColor(lab,COLOR_LAB2RGB)
```

A imagem de saída será do mesmo tipo que a imagem de entrada (float32 ou uint8).

Tome cuidado que OpenCV trabalha com sistema BGR enquanto que Python/PyPlot trabalha com sistema RGB. As seguintes funções fazem as conversões entre os dois sistemas:

```
bgr=cv2.cvtColor(rgb,COLOR_RGB2BGR)
rgb=cv2.cvtColor(bgr,COLOR_BGR2RGB)
```

[wikiLAB] https://en.wikipedia.org/wiki/CIELAB_color_space
[OpenCV] https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

Anexo B: Relatórios dos exercícios programas

O mais importante numa comunicação escrita é que o leitor entenda, sem esforço e inequivocamente, o que o escritor quis dizer. O texto ficar "bonito" é um aspecto secundário. Se uma (pseudo) regra de escrita dificultar o entendimento do leitor, essa regra está indo contra a finalidade primária da comunicação. No site do governo americano [1], há regras denominadas de "plain language" para que comunicações governamentais sejam escritas de forma clara. As ideias por trás dessas regras podem ser usadas em outros domínios, como na escrita científica. Resumo abaixo algumas dessas ideias.

(1) Escreva para a sua audiência. No caso do relatório, a sua audiência será o professor ou o monitor que irá corrigir o seu exercício. Você deve enfocar na informação que o seu leitor quer conhecer. Não precisa escrever informações que são inúteis ou óbvias para o seu leitor.

(2) Organize a informação. Você é livre para organizar o relatório como achar melhor, porém sempre procurando facilitar o entendimento do leitor. Seja breve. Quebre o texto em seções com títulos claros. Use sentenças curtas. Elimine as frases e palavras que podem ser retiradas sem prejudicar o entendimento. Use sentenças em ordem direta (sujeito-verbo-predicado).

(3) Use palavras simples. Use o tempo verbal o mais simples possível. Evite cadeia longa de nomes, substituindo-os por verbos (em vez de "desenvolvimento de procedimento de proteção de segurança de trabalhadores de minas subterrâneas" escreva "desenvolvendo procedimentos para proteger a segurança dos trabalhadores em minas subterrâneas"). Minimize o uso de abreviações (para que o leitor não tenha que decorá-las). Use sempre o mesmo termo para se referir à mesma realidade (pode confundir o leitor se usar termos diferentes para se referir a uma mesma coisa). O relatório não é obra literária, não tem problema repetir várias vezes a mesma palavra.

(4) Use voz ativa. Deixe claro quem fez o quê. Se você utilizar oração com sujeito indeterminado ou na voz passiva, o leitor pode não entender quem foi o responsável (Ex: "Criou-se um novo algoritmo" - Quem criou? Você? Ou algum autor da literatura científica?). O site diz: "Passive voice obscures who is responsible for what and is one of the biggest problems with government writing."

(5) Use exemplos, diagramas, tabelas, figuras e listas. Ajudam bastante o entendimento.

O relatório deve conter pelo menos as seguintes informações:

Identificação

Nomes dos integrantes do grupo, números USP, nome da disciplina, etc.

Breve enunciado do problema

Apesar do enunciado do problema ser conhecido ao professor/monitor, descreva brevemente o problema que está resolvendo. Isto tornará o documento compreensível para alguma pessoa que não tem o enunciado do EP à mão.

Técnica(s) utilizada(s) para resolver o problema

Descreva quais técnicas você usou para resolver o problema. Se você mesmo inventou a técnica, descreva a sua ideia, deixando claro que a ideia foi sua. Se você utilizou alguma técnica já conhecida, utilize o nome próprio da técnica (por exemplo, filtragem Gaussiana, algoritmo SIFT, etc.) juntamente com alguma referência bibliográfica onde a técnica está descrita. Use elementos gráficos como ima-

1 https://plainlanguage.gov/guidelines/

gens intermediárias e diagramas, pois ajudam muito a compreensão. Não “copie-e-cole” código-fonte, a não ser que seja relevante. Use preferencialmente o pseudo-código.

Ambiente de desenvolvimento utilizado

Em qual plataforma você desenvolveu o programa? Como o professor/monitor pode compilar o programa? Você utilizou que bibliotecas?

Operação

Como o professor/monitor pode executar o programa? Que argumentos são necessários para a execução do programa? Há parâmetros que devem ser configurados? Quais arquivos de entrada são necessários? Quais arquivos de saída são gerados?

Resultados Obtidos

Descreva os resultados obtidos. Qual é o tempo de processamento típico? O problema foi resolvido de forma satisfatória?

Referências

Descreva o material externo utilizado, como livros/artigos consultados, websites visitados, etc.