PSI-3471: Fundamentos de Sistemas Eletrônicos Inteligentes
Primeiro semestre de 2024 Exercício-programa Prof. Hae
Data de entrega: 30/06/2024 (domingo) até 23:59 horas
Data limite de entrega (com desconto de 0,5 por dia de atraso): 04/07/2023 (quinta) até 23:59 horas

Obs. 1: Não posso aceitar entrega após data-limite, pois preciso fechar as notas.
Obs. 2: Este EP pode ser resolvido em dupla ou individualmente.

1. Introdução

Kaggle (www.kaggle.com) é uma plataforma de competição de ciência de dados e uma comunidade online de profissionais de aprendizado de máquina da Google. Kaggle permite que os usuários encontrem e publiquem conjuntos de dados, explorem e construam modelos em um ambiente baseado na web, trabalhem com outros cientistas de dados e participem de competições para resolver desafios de ciência de dados [Wikipedia]. Kaggle fornece um ambiente de programação semelhante ao Google Colab com GPU/TPU.

Em Kaggle, há o conjunto Pistachio Image Dataset:
https://www.kaggle.com/datasets/muratkokludataset/pistachio-image-dataset/data
Este conjunto contém 1232 imagens de pistache Kirmizi e 916 de pistache Siirt, totalizando 2148 imagens. A figura 1 mostra algumas dessas imagens.

Sinceramente, eu não consigo enxergar nenhuma diferença entre os dois tipos de pistache. Porém, parece que alguns modelos de rede convolucional apresentam taxas de acerto de até 98,84%! Isto é, deve haver diferenças visíveis entre os dois tipos de pistache. Vocês conseguem visualizar as diferenças? Seria interessante descobrir o que os modelos olham para classificar os pistaches.

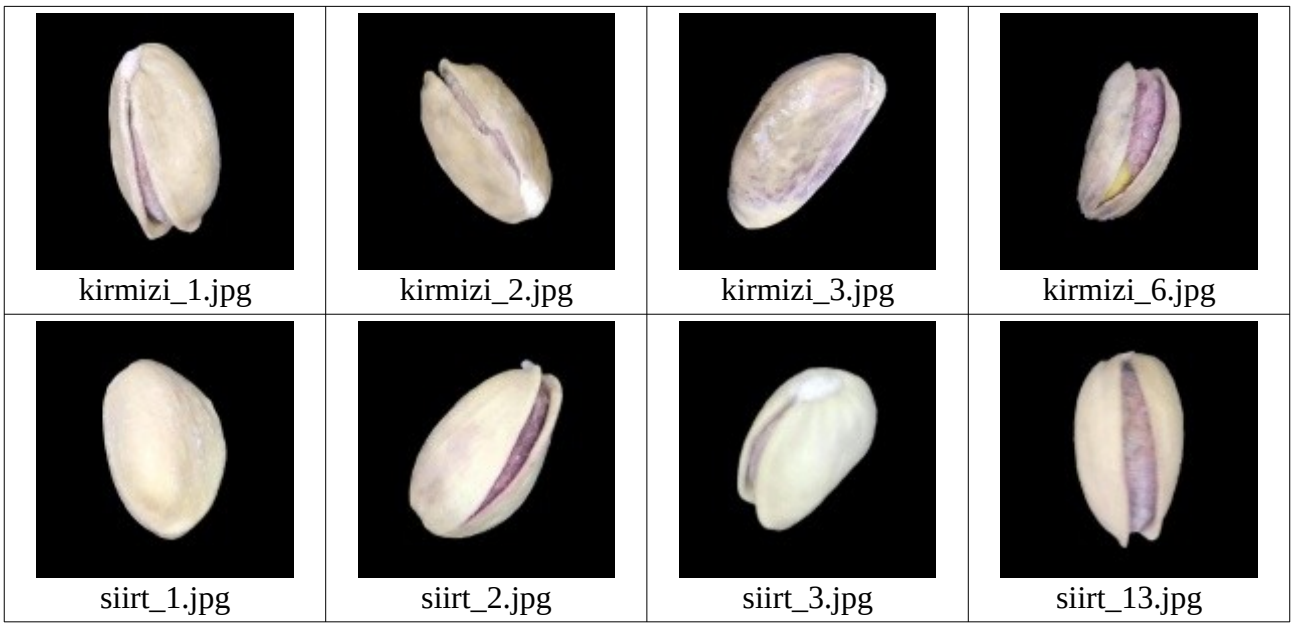

Figura 1: Algumas imagens de pistaches Kirmizi e Siirt.

2. Baixar conjunto de imagens

Se você quiser fazer o seu exercício em Kaggle, o conjunto de dados estará prontamente disponível. Porém, se quiser fazer em Colab ou em computador local, precisa primeiro baixar o conjunto.

O site abaixo descreve como baixar um conjunto de dados da Kaggle no Colab ou no seu computador local:

https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-google-colab/

Resumindo as informações desse site, você primeiro precisa criar uma conta na Kaggle:

https://www.kaggle.com/

Depois, na página principal da Kaggle, clique em “Account” que levará à página “Settings”. Nessa página, clique em “Create New Token”. Isso baixará um arquivo “kaggle.json”, onde você obterá o seu *username* e uma *key*, algo como:

```
{"username":"seunome","key":"9e9f84b521cd43657418455fce99120b"}
```

Agora, no seu ambiente Colab, copie o conteúdo do programa 1 numa célula. Você deve substituir o conteúdo em amarelo pelas suas informações obtidas de “kaggle.json”. A execução dessa célula irá baixar o conjunto de dados de pistache no seu diretório default Colab (/content) e o descompactará.

```
#https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-google-colab/
!pip3 install -q kaggle
!mkdir ~/.kaggle
!echo '{"username":"seunome","key":"9e9f84b521cd43657418455fce99120b"}' > ~/.kaggle/kaggle.json
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets muratkokludataset/pistachio-image-dataset
!unzip -u pistachio-image-dataset
```

Programa 1: Célula Colab para baixar o conjunto de pistache da Kaggle. Substitua as partes amarelas pelas suas informações.

Se quiser baixar as imagens no seu computador local Linux, use o programa 2. Grave o script como *download.sh*. Depois, torne-o executável e rode-o:

```
$ chmod +x download.sh
$ ./download.sh
```

```
set -v
#https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-google-colab/
pip3 install -q kaggle
mkdir ~/.kaggle
echo '{"username":"seunome","key":"9e9f84b521cd43657418455fce99120b"}' > ~/.kaggle/kaggle.json
chmod 600 ~/.kaggle/kaggle.json
kaggle datasets download muratkokludataset/pistachio-image-dataset
unzip -u pistachio-image-dataset
set +v
```

Programa 2: *Bash script* para baixar o conjunto de pistaches da Kaggle no computador local. Substituas partes amarelas pelas suas informações.

3. O exercício

3.1 - 3 pontos: Reduzir e alinhar os grãos horizontalmente

1) Todas as imagens originais estão no formato .JPG com 600×600 pixels. Provavelmente, essas imagens são grandes demais para alimentar os algoritmos de aprendizado de máquina. Escolha um valor *n* (que você julgar adequado) e reduza as imagens originais para a resolução *n*×*n*, usando algum método de interpolação (que você achar adequado). Além disso, vamos alinhar as imagens horizontalmente para verificar se esta operação ajuda a classificar os pistaches.

Nota: Contrariando a intuição, alinhar as imagens não ajuda muito na classificação.

Vamos criar o programa *alinha* que reduz as imagens e alinha os pistaches horizontalmente. Por exemplo:

```
$ alinha imagem.jpg      (OU $ python3 alinha.py imagem.jpg)
```

deve ler *imagem.jpg* e gerar duas imagens:

R_imagem.jpg: *imagem.jpg* reduzida para dimensão *n*×*n*.

A_imagem.jpg: *imagem.jpg* reduzida para dimensão *n*×*n* e alinhada horizontalmente.

O seguinte comando deve reduzir e alinhar todas as imagens *kirmizi*.jpg* do diretório atual, gerando duas imagens *R_kirmizi*.jpg* e *A_kirmizi*.jpg* para cada imagem *kirmizi*.jpg*:

```
$ alinha kirmizi*.jpg
```

Acredito que a forma mais fácil de fazer isso é usar “wildcard expansion”. Veja Anexo A para mais informações.

Nota: Evidentemente, é possível fazer estas operações dentro dos programas de aprendizado de máquina. Mas o processamento fica mais rápido se fizer estas operações uma única vez. Além disso, é possível verificar visualmente se o alinhamento foi feito com sucesso.

2) Para gerar as imagens alinhadas *A_*.jpg* de dimensão *n*×*n*, ache o centro de massa e a orientação da imagem original convertida numa imagem binária. Estou passando a função *findCenterAndOrientation* que acha o centro de massa e a orientação de uma imagem binária (uma imagem *uchar* ou *uint8* que só tem pixels brancos (255) ou pretos (0)). Para maiores detalhes, veja a discussão em:

https://stackoverflow.com/questions/14720722/binary-image-orientation

https://en.wikipedia.org/wiki/Image_moment

```
#include "procimagem.h"
(...)
Point3d findCenterAndOrientation(Mat_<uchar> src) {
  Moments m = moments(src, true);
  double cen_x = m.m10/m.m00;
  double cen_y = m.m01/m.m00;
  double theta = 0.5 * atan2( 2*m.mu11, m.mu20-m.mu02);
  return Point3d(cen_x, cen_y, theta);
}
```

Um exemplo de chamada desta função, para uma imagem binária *a*, é:

```
Point3d p=findCenterAndOrientation(a);
printf("%f %f %f\n",p.x,p.y,p.z);
```

Após a chamada, *p.x* e *p.y* conterão o centro de massa da imagem *a* e *p.z* representará o ângulo em radianos da inclinação da figura. Rotacione e centralize a imagem, para que a imagem esteja

centrada e a orientação dos pistaches seja horizontal, como mostra a figura 2. Você deve calcular a orientação usando imagem binária, mas a imagem alinhada deve ser colorida.

Nota: Parece que as imagens originais já estão centralizadas, de forma que não vai fazer muita diferença centralizar novamente ou não.

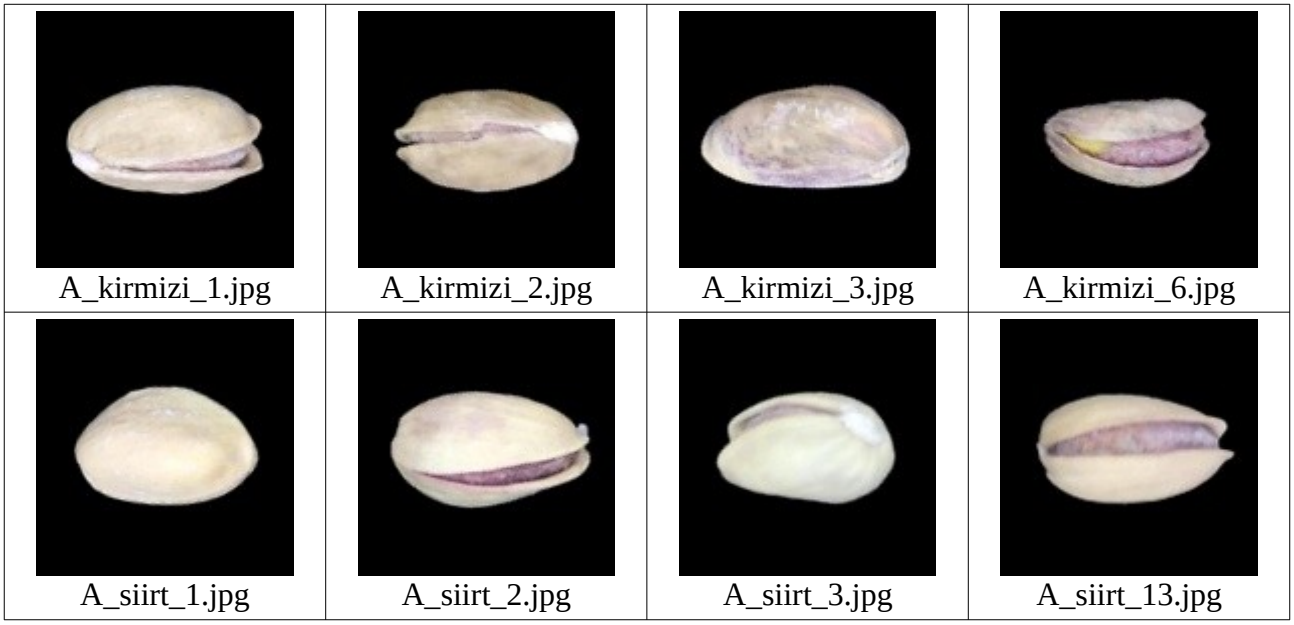

Figura 2: Pistaches da figura 1 alinhados horizontalmente.

3.2- 3 pontos: Classificar as imagens reduzidas e alinhadas usando um algoritmo clássico

O objetivo da parte 2 do exercício é classificar os tipos de pistaches usando algum método de aprendizado de máquina clássico, verificando se alinhar os pistaches horizontalmente ajuda (ou não) nessa tarefa. Para isso:

a) Leia todas as imagens reduzidas de pistaches *R_kirmizi* e *R_siirt*, embaralhe-as aleatoriamente, use a metade para o treino e a outra metade para o teste. Use algum método de aprendizado de máquina clássico para classificar as imagens de teste. Informe qual foi a taxa de erro obtida.
b) Leia todas as imagens alinhadas de pistaches *A_kirmizi* e *A_siirt*, embaralhe-as aleatoriamente, use a metade para o treino e a outra metade para o teste. Use o mesmo método de aprendizado de máquina clássico do item anterior para classificar as imagens de teste. Informe qual foi a taxa de erro.

Se quiser, execute algum pré-processamento (como extração de atributos) antes de fazer a classificação. Exercícios com taxas de acerto mais altas receberão notas maiores.

3.3 - 4 pontos: Classificar as imagens não-alinhadas e alinhadas usando a rede neural convolucional

O objetivo da parte 3 do exercício é classificar os tipos de pistaches usando alguma rede neural convolucional, verificando se alinhar os pistaches horizontalmente ajuda (ou não) nessa tarefa. Para isso:

a) Leia todas as imagens reduzidas de pistaches *R_kirmizi* e *R_siirt*, embaralhe-as aleatoriamente, use a metade para o treino e a outra metade para o teste. Use rede neural convolucional para classificar as imagens de teste. Informe qual foi a taxa de erro de teste.
b) Leia todas as imagens alinhadas de pistaches *A_kirmizi* e *A_siirt*, embaralhe-as aleatoriamente, use a metade para o treino e a outra metade para o teste. Use a mesma arquitetura do item (a) para classificar as imagens de teste. Informe qual foi a taxa de erro de teste.

Exercícios com taxas de acerto mais altas receberão notas maiores. Quem resolver o exercício corretamente usando somente as técnicas que vimos neste curso já receberá a nota máxima. Quem conseguir aumentar a taxa de acerto, mesmo usando técnicas que não estudamos, pode receber nota extra.

4. Observações

Obs. 1: Entregue os programas-fontes (alinha.*, classico.*, convolucional.*). Você pode enviar links para Colab ou Kaggle em vez dos programas.

Obs. 2: Entregue um documento PDF de no máximo 4 páginas (relatorio.pdf) descrevendo o funcionamento dos seus programas, os resultados obtidos e as suas conclusões. O envio do relatório é obrigatório (veja o anexo B). O relatório é um documento Word, LaTeX, LibreOffice ou GoogleDocs convertido para PDF. Um notebook Python anotado não será aceito como relatório.

Obs. 3: Entregue um vídeo de no máximo dois minutos explicando o funcionamento dos seus programas, os resultados obtidos e as suas conclusões. Se EP foi feito em dupla, podem enviar dois vídeos (a duração total não deve superar 2 minutos), cada membro explicando uma parte do EP. No vídeo deve aparecer em algum momento os rostos e os documentos do grupo. 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. Vocês podem 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, Renato e Wesley - hae.kim@usp.br, renatocan@gmail.com, wesleybeccaro@usp.br) tenham acesso ao seu vídeo.

Obs. 4: Não envie o conjunto de imagens. Se você quiser explicar algo sobre alguma imagem, coloque-a no relatório e mostre-a no vídeo.

Obs. 5: Não se esqueçam de escrever/falar os nomes da dupla (ou do único aluno) 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 os arquivos e/ou links em edisciplinas. Após professor/monitor ter corrigido EP, não serão aceitas novas entregas.

Anexo A: Argumentos e expansão de wildcard

A expansão de wildcard pode tornar mais simples executar um programa para vários arquivos de entrada, por exemplo, executar o programa *alinha* para os 50 arquivos ".jpg" do diretório, sem ter que chamar o programa *alinha* 50 vezes.

Um programa escrito em C++ ou Python, tanto em Linux como em Windows, pode aceitar argumentos. Os programas abaixo, em C++ e Python, imprimem o número de argumentos e cada um dos argumentos recebidos.

```cpp
//arg.cpp
#include <cstdio>
using namespace std;
int main(int argc, char** argv) {
  printf("Argc=%d\n",argc);
  for (int i=0; i<argc; i++)
    printf("%s\n",argv[i]);
}
```

```python
#arg.py
import sys
print("N arg=",len(sys.argv))
for i in range(len(sys.argv)):
  print(sys.argv[i])
```

Se executar esses programas, eles imprimem o número de argumentos seguido por cada um dos argumentos. O argumento índice zero é o nome do programa. Por exemplo:

```
Linux$ ./arg x y
Argc=3
./arg
x
y

Linux$ python3 arg.py x y
N arg= 3
arg.py
x
y
```

Linux possui um mecanismo chamado "wildcard expansion" que substitui argumentos com símbolo * ou ? pelos nomes dos arquivos que "casam" com o padrão. O símbolo * casa com um número arbitrário de caracteres e o símbolo ? casa com um único caractere. Digamos que seu diretório contenha os seguintes arquivos:

```
diretorio$ ls
Arborio1.jpg  Arborio2.jpg  arg  arg.cpp  arg.py
```

Se você executar nesse diretório:

```
Linux$ ./arg *.jpg
```

Linux irá expandir “*.jpg” para “Arborio1.jpg Arborio2.jpg”. Seria o mesmo que tivesse escrito:

```
$ ./arg Arborio1.jpg Arborio2.jpg
```

O programa irá imprimir:

```
Argc=3
./arg
Arborio1.jpg
Arborio2.jpg
```

Este mecanismo pode ajudar executar o programa *alinha* para todas as imagens de pistache de cada diretório, sem ter que chamar *alinha* 50 vezes. O mesmo vale para um programa Python em Linux:

```
Linux$ python3 arg.py *.jpg
N arg= 3
arg.py
Arborio1.jpg
Arborio2.jpg
```

Windows não faz expansão de wildcard. Porém, o compilador GCC que está incluso em Cekeikon faz expansão de wildcard, para emular esta funcionalidade de Linux. Assim, a expansão de wildcard só não funciona para Python em Windows ou se você usar um compilador de C++ diferente de GCC em Windows.

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 seu 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. Escreva usando menos palavras possível. Use o tempo verbal o mais simples possível. Evite “verbos ocultos” (por exemplo, substitua “precisamos realizar uma revisão das contas da Agência” para “precisamos rever as contas da Agência”; substitua “fiz o pagamento do seu salário” por “paguei o seu salário”). 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, pois 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

1 https://plainlanguage.gov/guidelines/

referência bibliográfica onde a técnica está descrita. Use elementos gráficos como imagens 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.