Exercício-programa de PSI3471

| **Nome Completo** | **Número USP** |
| --- | --- |
| Gustavo Henrique da Silva Amaral | 12551686 |
| Thiago da Rocha Calomino Gonçalves | 12554647 |

*Prof. Hae Yong Kim*

# Enunciado do problema

O objetivo deste exercício-programa é fazer uma classificação de grãos de pistache de dois tipos através das imagens. Os tipos existentes no Pistachio Image Dataset são Kirmizi e Siirt.

Para realizar este estudo, a primeira etapa consiste em processar as imagens de duas formas: redução e alinhamento dos grãos horizontal e centralizadamente em todas as imagens, para que o posicionamento deles nas imagens seja sempre uniforme.

A segunda etapa é verificar se o alinhamento dessas imagens melhora a taxa de acerto do algoritmo de classificação. Para que possamos comparar as taxas de acerto das imagens reduzidas e alinhadas serão implementadas duas abordagens: uma por SVM (Máquina de Vetores de Suporte) e outra por CNN (Rede Neural Convolucional).

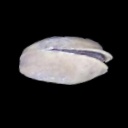
# Técnicas usadas

## Reduzir e alinhar os grãos horizontalmente

* + 1. Todas as imagens originais estão no formato JPG com 600×600 pixels. Dado que são grandes demais para alimentar os algoritmos de aprendizado de máquina, foi escolhido um valor n para reduzir as imagens originais para a resolução n×n
    2. Além disso, foi feito o seguinte:
       1. Conversão para escala de cinza e binária
          1. Para realizar essa conversão usamos a função cv2.cvtColor(). Em seguida, foram convertidas para binário pela função cv2.threshold(), transformando os pixels em preto e branco
       2. Cálculo do centro de massa e orientação
          1. Utilizamos os momentos da imagem binária, calculados pela função cv2.moments(), para determinar o centro de massa e a orientação das imagens.
          2. O centro de massa foi calculado utilizando as coordenadas m10/m00 e m01/m00.
          3. A orientação foi determinada pela fórmula

0.5 \* np.arctan2(2 \* moments['mu11'], moments['mu20'] - moments['mu02']).

* + - 1. Redução de tamanho
         1. As imagens foram redimensionadas para uma dimensão menor (128 x 128 pixels) utilizando a função cv2.resize().
      2. Alinhamento horizontal: foi aplicada uma rotação usando a matriz de rotação cv2.getRotationMatrix2D() e a função cv2.warpAffine()
      3. Exemplo de alinhamento com uma imagem de grão Kirmizi



*Fig.: Grão de Kirmizi (Da esquerda para a direita: imagem original, versão alinhada e versão reduzida)*

## Classificação das imagens reduzidas e alinhadas usando um algoritmo clássico (SVM)

* + 1. Utilizamos um classificador SVM com kernel linear, implementado pela biblioteca Scikit-Learn. Os dados foram divididos em conjuntos de treino e teste utilizando a função train\_test\_split(). O modelo foi treinado com o conjunto de treino e avaliou-se a acurácia com o conjunto de teste.

## Classificar as imagens não-alinhadas e alinhadas usando uma rede neural convolucional

* + 1. Foi implementada uma rede neural convolucional com uso da biblioteca TensorFlow e Keras. A arquitetura da CNN inclui camadas convolucionais, de pooling e totalmente conectadas. O modelo foi treinado com o conjunto de treino e avaliou-se a acurácia com o conjunto de teste.
    2. A CNN foi compilada com o otimizador Adam e a função de perda sparse\_categorical\_crossentropy.

# Instruções dos ambientes de execução

Este exercício-programa foi executado no ambiente Google Collab usando a linguagem de programação Python. Algumas bibliotecas usadas para simulação foram:

* **OpenCV**: Utilizada para processamento de imagens, incluindo conversão de cores, binarização, cálculo de momentos e transformação de imagens.
* **NumPy**: Utilizada para operações numéricas e manipulação de arrays.
* **Scikit-Learn**: Utilizada para implementar o classificador SVM e para funções de pré-processamento e avaliação.
* **TensorFlow e Keras**: Utilizadas para construir, treinar e avaliar a rede neural convolucional (CNN).
* **glob**: Utilizada para listar arquivos em diretórios.
* **os**: Utilizada para manipulação de caminhos de arquivos e diretórios.

Realizamos a importação do conjunto [Pistachio Image Dataset](https://www.kaggle.com/datasets/muratkokludataset/pistachio-image-dataset/data) disponível no Kaggle.

O conjunto de imagens foi importado do Kaggle diretamente para o ambiente de resolução do problema usando o API Token de um dos membros da dupla.

O ponto mais importante na execução deste exercício-programa é a importação dos dados, além da definição em código dos diretórios de entrada e saída para obtenção das imagens que serão alvo de processamento.

# Resultados

Para avaliar a eficácia dos métodos de classificação, foram comparadas as taxas de acerto das imagens reduzidas e alinhadas utilizando tanto o classificador SVM quanto a rede neural convolucional (CNN):

| **Tipo da imagem** | **Taxa de acerto (%) [SVM]** | **Taxa de acerto (%) [CNN]** |
| --- | --- | --- |
| Imagens reduzidas | 83.98 | 83.61 |
| Imagens alinhadas | 82.62 | 87.70 |

A taxa de acerto em ambos os métodos para imagens reduzidas e desalinhadas foi praticamente a mesma. Por outro lado, para imagens alinhadas a taxa de acerto foi maior pela implementação em rede neural convolucional e praticamente a mesma por SVM.

Em particular, observamos que a CNN teve um melhor desempenho em comparação com o SVM, tanto para imagens reduzidas quanto para imagens alinhadas.

# Conclusões

Realizar o alinhamento das imagens aumentou significativamente a acurácia final registrada, provando, portanto, que o modelo de grãos alinhados é mais robusto. A acurácia atinge praticamente 90% ao usar CNN, ao passo que mal passa de 85% para o modelo implemetado por SVM ou apenas com imagens desalinhadas.

# Referências

1. [PSI-3471: Fundamentos de Sistemas Eletrônicos Inteligentes (2024 - 1º semestre) Exercício programa prof. Hae](http://www.lps.usp.br/hae/psi3471/ep1-2024/ep.pdf)
2. [Binary Image Orientation - Stack Overflow](https://stackoverflow.com/questions/14720722/binary-image-orientation)
3. [Image Thresholding](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)
4. [Transformações geométricas](http://www.lps.usp.br/hae/apostila/transgeo-ead.pdf)
5. [One-hot-encoding, o que é? | Medium](https://arthurlambletvaz.medium.com/one-hot-encoding-o-que-%C3%A9-cd2e8d302ae0)
6. [Rede neural convolucional (CNN) em Tensorflow/Keras](http://www.lps.usp.br/hae/apostila/convkeras-ead.pdf)
7. [Cross Validation: Avaliando seu modelo de Machine Learning | by Eduardo Braz Rabello | Medium](http://www.lps.usp.br/hae/apostila/convkeras-ead.pdf)