

Trabalho 5 - MC920

GUSTAVO HENRIQUE STORTI SALIBI

RA: 174135 / Ciência da Computação - Graduação

E-mail: gustavohstorti@gmail.com

I. INTRODUÇÃO

Agrupamento k-means é um método de Clustering que objetiva particionar n observações dentre k grupos onde cada observação pertence ao grupo mais próximo da média.

O objetivo deste trabalho é aplicar técnicas de agrupamento de dados (ou seja, técnicas de aprendizado de máquina não supervisionado) para reduzir (quantizar) o número de cores de uma imagem colorida, procurando manter a qualidade da aparência geral da imagem de entrada.

A seção seguinte, II, explica de forma geral o problema a ser tratado neste trabalho. A seção III fala sobre como o programa lida com as entradas de dados e como ele é construído. A seção IV mostra os resultados obtidos e os descreve. A seção V, por sua vez, trata das limitações encontradas. A seção VI explica como e onde as saídas são salvas. A seção VII, por fim, apresenta mais alguns exemplos de resultados obtidos.

II. ESPECIFICAÇÃO DO PROBLEMA

Dada uma imagem colorida de entrada e um número n de cores, deve-se produzir uma imagem de saída referente à entrada com a quantidade de cores reduzida a n .

Para isso, 4 etapas serão realizadas:

- 1) Ler a imagem colorida de entrada;
- 2) Aplicar a técnica k-means de agrupamento de dados para encontrar grupos de cores mais representativas;
- 3) Salvar dicionário (codebook) gerado pela técnica de agrupamento, ou seja, os centros dos grupos e os rótulos correspondentes a cada pixel da imagem;
- 4) Reconstruir a imagem com cores reduzidas a partir do dicionário armazenado.

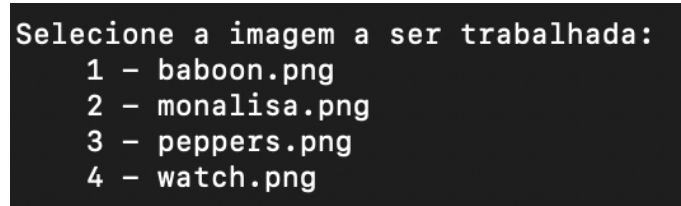
III. ENTRADA DE DADOS

O programa foi desenvolvido na linguagem Python e testado com o interpretador Python 3.7. Foram também utilizadas as seguintes bibliotecas extras: OpenCV 4.1, Scikit-Learn 0.20.3 e Numpy 1.12.1.

Para executar o programa, basta utilizar o seguinte comando:

```
python3 programa.py
```

Após isso, um menu irá aparecer com as seguintes opções de imagens contidas na pasta "imagem/entrada":



```
Selecione a imagem a ser trabalhada:
1 - baboon.png
2 - monalisa.png
3 - peppers.png
4 - watch.png
```

Figura 1. Menu exibido ao rodar programa.

Em seguida, o programa pedirá a quantidade de cores para qual a imagem será reduzida.

Uma vez selecionadas a imagem desejada e a quantidade de cores, o programa irá processá-la e salvará a saída no diretório imagens/saída.

A. Código

O código começa por importar as bibliotecas necessárias. Depois disso, o menu é exibido junto de um código para capturar a opção desejada.

Então, o programa segue de forma sequencial e é dividido em 4 partes correspondentes às etapas descritas na seção anterior. Cada uma delas é facilmente encontrável por conta dos comentários separando o código.

IV. SOLUÇÃO

Tomaremos como base para comparação a seguinte imagem original:



Figura 2. Imagem original - watch.png

As imagens a seguir mostram a redução de cores para 128, 64, 32, 16, 8 e 2, respectivamente:



Figura 3. watch.png - 128 cores



Figura 4. watch.png - 64 cores



Figura 5. watch.png - 32 cores



Figura 6. watch.png - 16 cores



Figura 8. watch.png - 2 cores



Figura 7. watch.png - 8 cores

V. LIMITAÇÕES

Dependendo do tamanho da imagem de entrada, o tempo de processamento pode ser bem demorado. Se utilizado o algoritmo padrão de k-means, isso pode se tornar inviável.

Além disso, diminuir o número de cores faz com que detalhes sejam perdidos. Muitas vezes estes detalhes não são perceptíveis e não causam problemas, mas outras, como em uso médico, eles podem ser essenciais.

VI. SAÍDA DE DADOS

O programa gera, para cada execução, uma imagem de saída com o nome da imagem de entrada acrescido do número de cores. Ela é salva no diretório imagem/saída.

VII. MAIS EXEMPLOS

Abaixo estão mais alguns exemplos obtidos.



Figura 9. Imagem original - monalisa.png



Figura 11. monalisa.png - 64 cores



Figura 10. monalisa.png - 128 cores



Figura 12. monalisa.png - 32 cores



Figura 13. monalisa.png - 16 cores



Figura 15. monalisa.png - 2 cores

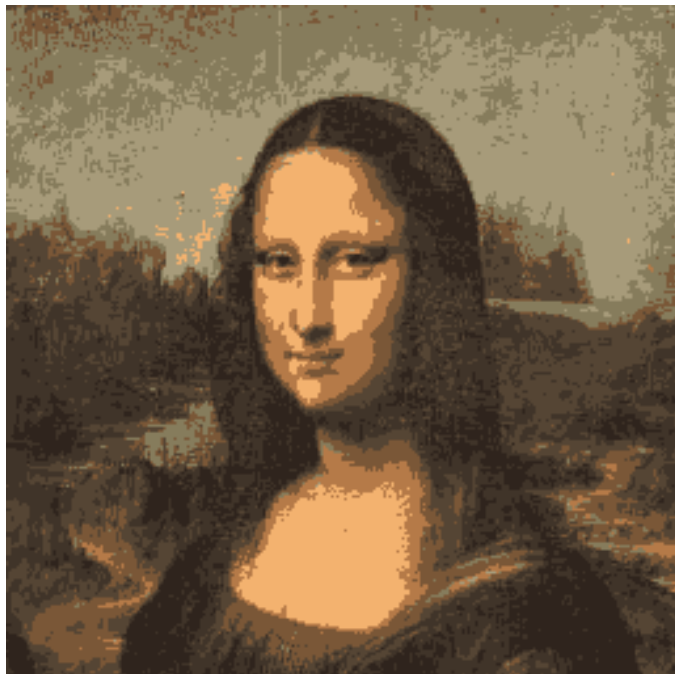


Figura 14. monalisa.png - 8 cores