

# Processamento de Imagens

## OCR Simples

**Heleny Maria Diniz Bessa<sup>1</sup>, Rithie Natan Carvalhaes Prado<sup>1</sup>**

<sup>1</sup>Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas Gerais (PUC Minas)  
Caixa Postal 1.686 – 30535.901 – Belo Horizonte – MG – Brasil

`heleny.bessa@sga.pucminas.br, rithie.prado@sga.pucminas.br`

**Abstract.** *Optical Character Recognition (OCR) is an implementation that unites Digital Image Processing and Machine Learning in order to get the computer to understand and analyze the characters present in an image, extracting data from them. The use cases are plenty and varied, which makes this field attractive and constantly full of innovations. Some applications to it are in Assistive Technology for the visually impaired and Word Processing. In our project, we implemented an OCR using Python that realizes basic processing operations on the input image, as well as feeds the received data to classification algorithms.*

**Resumo.** *Reconhecimento Ótico de Caracteres (OCR) é uma implementação que une Processamento de Imagens Digitais e Machine Learning com o objetivo de fazer com que o computador entenda e analise caracteres presentes em uma imagem, extraíndo dados desta. O caso de uso são muitos e variados, o que faz este campo atraente e constantemente cheio de inovações. Algumas de suas aplicações são em Tecnologia Assistiva para deficientes visuais e processamento de palavras. Neste trabalho, implementamos um OCR em Python que realiza operações simples de processamento em cima de uma imagem, assim como alimenta o seu conteúdo para algoritmos de classificação.*

### 1. Descrição do Problema

Embora o computador atual consiga produzir e reproduzir imagens, não consegue entender e criar conclusões a partir de seus conteúdos. Uma frase em um fundo branco não passa de *pixels*, seu significado incompreendido pela máquina. Para suprir esta deficiência, Visão Computacional estuda como fazer com que estes conjuntos de *pixels* tenham seu significado intrínseco traduzido para uma linguagem que o computador entenda.

Uma das subáreas é a de processamento que busca a compreensão de caracteres em imagens, o processador de texto. Para tal, criamos um software de Reconhecimento Ótico de Caracteres usando a linguagem *Python* que recebe uma imagem, realiza operações de edição sobre ela para aumentar sua qualidade e de reconhecimento do que está escrito.

### 2. Técnicas Implementadas

OpenCV é uma biblioteca *Python* desenvolvida pela Intel para programas na área de Visão Computacional. Esta biblioteca nos permite a manipulação de imagens, como redimensionamento, suavização e conversão para tons de cinza e detecção de bordas. Utilizamos a versão 4.5.4.

Outra biblioteca utilizada é o mahotas, utilizada para binarização da imagem.

Para o processamento e consequente obtenção de informação, utilizamos de 3 algoritmos de classificação cujas teorias serão brevemente explicadas.

## 2.1. Distância de Mahalanobis

É uma medida da distância que se baseia nas correlações entre variáveis com os quais distintos padrões podem ser identificados e analisados.

## 2.2. Rede Neural Simples

Redes Neurais são modelos computacionais inspirados pela forma com a que nosso próprio cérebro funciona.

## 2.3. Support Vector Machine (SVM)

SVM é um conjunto de métodos usados para classificação, regressão e detecção de *outliers*. Ele busca uma linha de separação entre dois grupos para classificar suas instâncias.

## 3. Métodos Implementados

Foi implementado em um sistema de cinco arquivos: main.py, mnist.py, svm.py, MahalanobisClassifier.py e NonNeuralNetwork.py. Falaremos destes arquivos na lista abaixo.

- main.py - Arquivo principal de execução do programa cujo não há nenhum método implementado. Aqui, apenas fazemos o tratamento da imagem do usuário e utilizamos os classificadores para obter uma resposta.
- mnist.py - Arquivo para importar as informações do dataset e plotar imagens com matplotlib. Os métodos neste arquivo são:
  - readidx(filename) - Recebe os caminhos para os arquivos do dataset. - ordem de complexidade:  $O(n)$
  - loaddataset() - Carrega dataset da keras.database. - ordem de complexidade:  $O(n)$
  - preppixels() - pré formata a database do keras.database - ordem de complexidade:  $O(n)$
- svm.py - Arquivo de execução do Classificador SVM.
  - svmmodel() - treina a base de dados com classificador SVM e predita a imagem do usuário - ordem de complexidade:  $O(n^3)$
- MahalanobisClassifier.py - Arquivo de execução do Classificador de KNeighborClassifier com Distancia de Mahalanobis.
  - mahalanobismodel() - treina a base de dados com classificador de Mahalanobis e predita a imagem do usuário - Classifica ordem de complexidade:  $O(k * n * d)$
- NonNeuralNetwork.py - Arquivo de execução do Classificador de Rede Neural Simples.
  - NeuralNetwork - Classe de rede neural simples
  - neuralnetmodel() - treina a base de dados com classificador de Rede Neural e predita a imagem do usuário - ordem de complexidade:  $O(nt(ij+jk+kl))$

## 4. Bibliotecas Utilizadas

- OpenCV
- sklearn
- tensorflow
- mahotas
- numpy
- scikit-learn
- struct
- matplotlib


## 5. Instalação e Uso

### 5.1. Instalação

Para a instalação basta ir até o diretório do projeto e executar o comando **pip install -r requirements.txt**.

### 5.2. Uso

Para usar basta executar o comando **python main** dentro do diretório do projeto. Durante a execução do programa será requisitado que o usuário insira o caminho para imagem, assim como a imagem abaixo:



```
Insira o caminho para a imagem:  
exemplos/imagem.png
```

Figure 1. Caminho da imagem

## 6. Tempo de Execução

Tempo de execução dos classificadores em treinamento:

- SVM - 40.40 segundos
- Mahalanobis - 7.80 segundos
- Rede Neural Simples - — 21.53 segundos

## 7. Resultados

SVM obteve melhores resultados com 95% de acurácia.

## 8. Conclusão

Neste trabalho, conseguimos modificar a imagem inserida de forma que ela foi entendida pelo algoritmo, suas regiões que continham números foram identificadas e conseguimos passar as informações recolhidas para algoritmos de classificação.

## References

Carregando e manipulando imagens em python - akira - ciência da computação.

Python machine learning tutorial. (n.d.).

(2020). Classificação de dígitos com machine learning.

(2021). Opencv python tutorial.

Antonello, R. *Introdução a Visão Computacional com Python e OpenCV*, volume 1.

Mallick, S. (2021). Feature based image alignment using opencv (c /python).