Relatório Projeto 1 de Visão Computacional

1st Thiago Matheus de Oliveira Costa
Universidade Federal de Viçosa
Rio Paranaíba, Minas Gerais
thiago.costa@ufv.br

2nd Pedro Lucas de Oliveira Costa
Universidade Federal de Viçosa
Rio Paranaíba, Minas Gerais
pedro.l.costa@ufv.br

***Abstract*—In this paper, we present the strategies and tools employed to classify 600 images distributed across six classes: apple, bat, beetle, bell, bird, and bone. First, we conducted the segmentation process using the OpenCV library. The "findContours" function was used to identify the contours of the dataset images, while the "BoundingRect" function was applied to extract the bounding box for each image segment.**

Next, feature extraction was performed using the scikit-image library. The "regionprops" function was utilized to compute features such as area, circularity, eccentricity, perimeter, major axis length, and minor axis length.

The dataset was then divided into training, testing, and validation subsets, with 70% of the data allocated for training, 30% for testing, and 20% of the training set reserved for validation.

Subsequently, we normalized the features using the Normal Feature Transform.

Finally, we implemented three classification models: K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). Each model was evaluated using traditional metrics, including accuracy, F1-score, precision, recall, and support.

***Index Terms*—Image classification, K-NN neighbors, Computer Vision.**

I. Introduction

In this paper, we worked with a dataset consisting of 600 images evenly distributed across six classes: apple, bat, beetle, bell, bird, and bone, with 100 images per class. The objective was to classify these images and achieve the best possible results.

II. Experiment Design

A. Image Segmentation

For this dataset, we selected the OpenCV library, as it is well-suited for working with the simplicity of the images. The dataset features binary images where the region of interest is displayed in white, while the background is black. By applying the "findContours" and "BoundingRect" functions, we obtained highly effective results in image processing and segmentation. As shows the figures 1 and 2

B. Feature Extraction

For feature extraction, we chose to use the scikit-image library, specifically the measure.regionprops function. Given the simplicity of the images, this function effectively extracted the features for each image with ease.

Identify applicable funding agency here. If none, delete this.

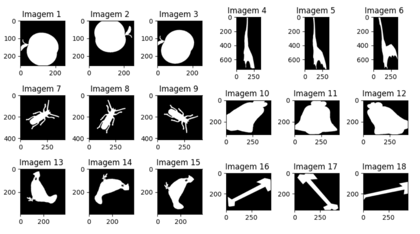


Fig. 1. Original Images.

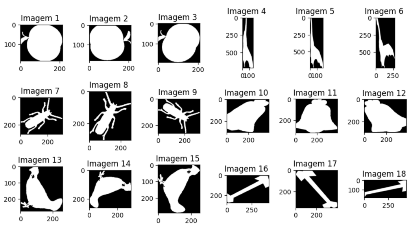


Fig. 2. Segmented Images

We selected the following features: area, circularity, eccentricity, perimeter, major axis length, and minor axis length. This choice was driven by the observation that the classes primarily differ in terms of circularity, size, and shape. By leveraging these features, we were able to effectively distinguish and separate the classes.

C. Dataset Spli

For the dataset split, we allocated 70% of the images for training, 30% for testing, and 20% of the training set for validation. As a result, out of the 600 images, 336 were used for training, 84 for validation, and 180 for testing. The images from each class were evenly distributed across these subsets.

D. Data Normalizing

To normalize the feature data for effective use in classification strategies, we first calculated the mean and standard deviation of the training set using the NumPy library. Then, we applied the normalization process to each dataset split (train, validation, and test) using the Scaler function from the scikit-learn library to ensure consistency and proper scaling across all splits.

E. Classification

To classify the images we trained and tested three strategies:

K-Nearest Neighbors (K-NN) - The K-NN algorithm classifies data points based on their proximity to others in the feature space. The number of neighbors (k) is a hyperparameter, and different values were tested to identify the one that yielded the best performance. We utilized the scikit-learn library for implementation, ensuring an efficient and seamless process for both training and testing. In this study, we experimented with the following values for k: [1, 3, 5, 7, 9].

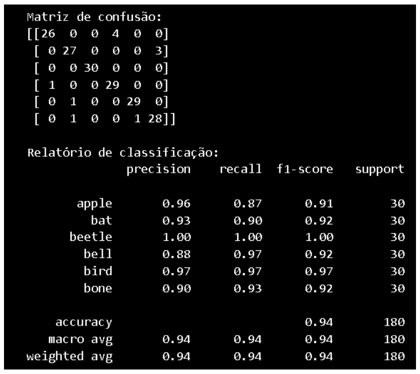
Support Vector Machine (SVM) - Support Vector Machine (SVM) - The SVM algorithm classifies data by finding the optimal hyperplane that separates data points of different classes in the feature space. This method is particularly effective for high-dimensional data and works well with both linear and non-linear boundaries. We used the scikit-learn library to implement the SVM, experimenting with different kernel functions, including linear and radial basis function (RBF), to determine the best configuration for our dataset. The hyperparameters, such as the regularization parameter (C) and kernel coefficient (), were tuned using the gridsearch to achieve optimal performance.

Multilayer Perceptron (MLP) - The MLP is a type of neural network composed of an input layer, one or more hidden layers, and an output layer, using backpropagation for training. It is particularly effective for learning complex patterns in the data. For this study, we implemented the MLP classifier using the scikit-learn library. Key hyperparameters such as the number of hidden layers, the number of neurons per layer, the activation function (e.g., ReLU), and the learning rate were tuned to optimize performance. The algorithm was trained on the normalized features, and early stopping was used to prevent overfitting during training.

The primary reasons for choosing these algorithms were their computational efficiency, the simplicity of the binary images, they are easily implemenetd and the fact that we manually extracted each feature. These factors made the selected methods well-suited for our dataset and classification task.

F. Evaluation

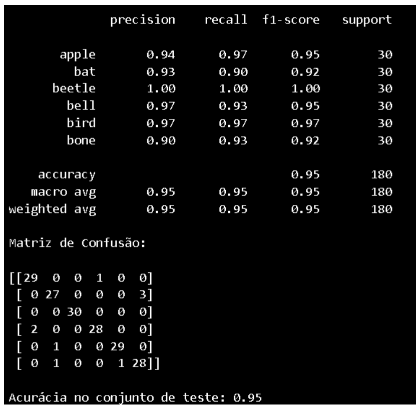
For the K-nn neighbors we got the results presented in the figure 3.



```
Matriz de confusão:
[[26  0  0  4  0  0]
 [ 0 27  0  0  0  3]
 [ 0  0 30  0  0  0]
 [ 1  0  0 29  0  0]
 [ 0  1  0  0 29  0]
 [ 0  1  0  0  1 28]]

Relatório de classificação:
              precision    recall  f1-score   support

       apple       0.96      0.87      0.91        30
         bat       0.93      0.90      0.92        30
      beetle       1.00      1.00      1.00        30
        bell       0.88      0.97      0.92        30
        bird       0.97      0.97      0.97        30
        bone       0.90      0.93      0.92        30

    accuracy                           0.94       180
   macro avg       0.94      0.94      0.94       180
weighted avg       0.94      0.94      0.94       180
```

Fig. 3. K-NN Neighbors

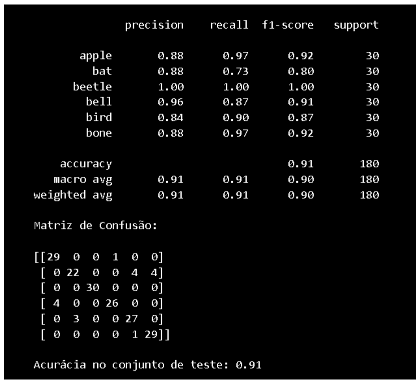
For the Multi Layer Perceptron we got the results presented in the figure 4.



```
              precision    recall  f1-score   support

       apple       0.94      0.97      0.95        30
         bat       0.93      0.90      0.92        30
      beetle       1.00      1.00      1.00        30
        bell       0.97      0.93      0.95        30
        bird       0.97      0.97      0.97        30
        bone       0.90      0.93      0.92        30

    accuracy                           0.95       180
   macro avg       0.95      0.95      0.95       180
weighted avg       0.95      0.95      0.95       180

Matriz de Confusão:

[[29  0  0  1  0  0]
 [ 0 27  0  0  0  3]
 [ 0  0 30  0  0  0]
 [ 2  0  0 28  0  0]
 [ 0  1  0  0 29  0]
 [ 0  1  0  0  1 28]]

Acurácia no conjunto de teste: 0.95
```

Fig. 4. Multi Layer Perceptron

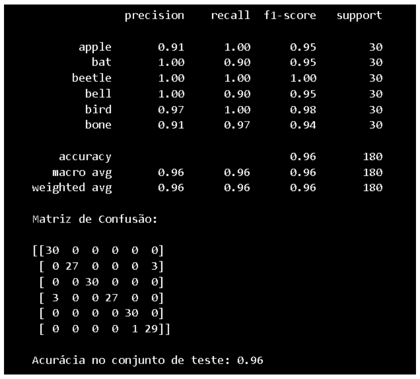
For the SVM we got the results presented in the figure 5.



```
              precision    recall  f1-score   support

       apple       0.88      0.97      0.92        30
         bat       0.88      0.73      0.80        30
      beetle       1.00      1.00      1.00        30
        bell       0.96      0.87      0.91        30
        bird       0.84      0.90      0.87        30
        bone       0.88      0.97      0.92        30

    accuracy                           0.91       180
   macro avg       0.91      0.91      0.90       180
weighted avg       0.91      0.91      0.90       180

Matriz de Confusão:

[[29  0  0  1  0  0]
 [ 0 22  0  0  4  4]
 [ 0  0 30  0  0  0]
 [ 4  0  0 26  0  0]
 [ 0  3  0  0 27  0]
 [ 0  0  0  0  1 29]]

Acurácia no conjunto de teste: 0.91
```

Fig. 5. SVM

For the SVM using the grid search we got the results presented in the figure 6.



```
              precision    recall  f1-score   support

       apple       0.91      1.00      0.95        30
         bat       1.00      0.90      0.95        30
      beetle       1.00      1.00      1.00        30
        bell       1.00      0.90      0.95        30
        bird       0.97      1.00      0.98        30
        bone       0.91      0.97      0.94        30

    accuracy                           0.96       180
   macro avg       0.96      0.96      0.96       180
weighted avg       0.96      0.96      0.96       180

Matriz de Confusão:

[[30  0  0  0  0  0]
 [ 0 27  0  0  0  3]
 [ 0  0 30  0  0  0]
 [ 3  0  0 27  0  0]
 [ 0  0  0  0 30  0]
 [ 0  0  0  0  1 29]]

Acurácia no conjunto de teste: 0.96
```

Fig. 6. SVM with the grid search