Cats and Dogs Classification report

By Matthis Brocheton

1. Program Execution

Environment Setup:

Ensure you have Python 3 installed.
Install the necessary Python packages. You can install them via pip by running the following command:

```
pip install numpy opencv-python scikit-learn scikit-image
```

Execution:

To execute the program, ensure that you have your training and testing datasets saved in the appropriate directories. Specifically, create the following folder structure:

- **data/**
  - **train/** (contains subfolders for each class, such as **Cat/** and **Dog/**)
  - **test/** (contains subfolders for each class)

You can run the program from your terminal or command line by executing the following command:

```
python image_classification.py --train data/train --test data/test --n_components 50 --target_size 128 128
```

Command Explanation:

- --train: Specifies the path to the training dataset directory.
- --test: Specifies the path to the testing dataset directory.
- -- n_components: Sets the number of PCA components to use (default is 50).
- --target_size: Defines the size to which each image will be resized (default is 128x128).

After running the command, the program will process the images, perform feature extraction, and display the classification accuracy in the terminal.

2. Classification accuracy:

After running the program, the accuracy of the classification will be printed in the terminal in the following format:

```
Accuracy: 58.50%
```

3. Conclusion:

In this assignment, I successfully implemented feature extraction techniques such as **Sobel**, **PCA**, and **HOG**. These methods allowed me to capture relevant information from images, including edges, dimensionality reduction, and texture features. I then utilized their outputs as input for the **nearest neighbor** algorithm, applying **SAD (Sum of Absolute Differences)** to measure image similarity. Finally, I calculated the **accuracy** to evaluate the model's performance, and I was able to achieve an accuracy of **58.50%**.

This project helped me understand how these technologies work, which was highly interesting and educational, especially for their ability to efficiently process and analyze images in a classification context.

4. Discussion:

Flattening Feature Extractions: One of the primary difficulties was converting the outputs of various feature extraction methods, such as **Sobel,** into a flattened format that could be processed by the **nearest neighbor** algorithm. This required reshaping multidimensional data, which was not always straightforward.

Optimization of Algorithms: Initially, the algorithms, particularly the **PCA** and the **Sobel**, were running very slowly. I had to optimize the code to improve the efficiency of these algorithms.

Label Assignment for Each Image: Another question I encountered was how to assign labels to each image efficiently. I found a solution by integrating the label assignment directly during the image loading process. This approach simplified the workflow and ensured that each image was correctly labeled as it was loaded into the program.