

Ecole Nationale Polytechnique
Département d'**Electronique**
DIGIUS LINK ALGERIA

Internship Report

Image Processing-Based Classification of
USB Drive Boxes Using K-Nearest Neighbors

Written by: Ahmed Assem **MEDDAH**
Louai Abdellah **BENAISSA**
Mohamed Khaled **FERROUKHI**

Supervised by: Abderraouf **ADJAL**

| | |
|---|-----------|
| 1 General Introduction | 1 |
| 1.1 About the Theme | 2 |
| 1.2 About the receiving company - ASA | 2 |
| 1.2.1 Company Description | 2 |
| 1.2.2 Products and Services | 2 |
| 1.2.3 Contacts & Location | 3 |
| 2 Image Processing | 5 |
| 2.1 Introduction | 6 |
| 2.1.1 What's image proceesing | 6 |
| 2.1.2 Why do we need it? | 6 |
| 2.2 Image Scanning | 7 |
| 2.2.1 Direct Contour Detection | 7 |
| 2.2.2 Masking | 7 |
| 2.3 Warp Transformation | 9 |
| 2.3.1 Polygon approximation | 9 |
| 2.3.2 Auto-rotation | 10 |
| 2.3.3 CLAHE Application | 10 |
| 3 Extracting the Data | 13 |
| 3.1 Optical Character Recognition | 14 |
| 3.2 K-Nearest Neighbors | 14 |
| 3.2.1 Histogram of Oriented Gradients (HOG) | 15 |
| 3.2.2 Labeling & KNN Training | 16 |
| 3.2.3 Results | 17 |
| 4 Conclusion | 19 |

Summary

Chapter 1

General Introduction

1.1 About the Theme

In today's digital era, USB flash disks have become ubiquitous tools for data storage and transfer. With an array of storage capacities available in the market, determining the capacity of a USB flash disk quickly and accurately is crucial for both consumers and retailers alike. However, manually extracting this information from the packaging can be time-consuming and error-prone.

To address this challenge, our internship project focuses on leveraging the power of image processing and machine learning techniques to automate the extraction of capacity numbers from USB flash disk boxes. By developing innovative algorithms and models, we aim to create a streamlined solution that enhances efficiency and accuracy in identifying USB flash disk capacities directly from their packaging images.

1.2 About the receiving company - ASA

1.2.1 Company Description

ASA is a company specializing in the distribution and marketing of IT and electronic products in Algeria. They are partnered with DIGIUS LINK ALGERIA, a prominent figure in the country's electronic manufacturing industry.



1.2.2 Products and Services

ASA prides itself on offering a wide range of high-quality IT and electronics. Their primary focus is on customer satisfaction, and they achieve this through exceptional support services designed to enhance the customer experience.

Products categories:

1. Mice
2. Chargers
3. USB Cables
4. RAM Modules
5. Flash Disks
6. Power Cables



Figure 1.1: ASA's Products

1.2.3 Contacts & Location

The commercial headquarters of ASA is situated in Cité Krim Belkacem Grp 04 Villa N°02 Dar-El-Beida - Alger, serving as the central hub for all commercial and administrative operations.

However, the manufacturing facility is divided into two sections, both located in Baraki - Alger.

One section specializes in bulk manufacturing of items such as mice, cables, and chargers. The other section is dedicated to the production of more complex products, including RAM modules, flash disks, and general PCB manufacturing.

Contact informations:

- **Phone number:** +213 (0)23 816 652
- **Mobile number:** +213 (0) 550 902 702
- **Email Adress:** contact@asafash.com

Chapter 2

Image Processing

2.1 Introduction

2.1.1 What's image processing

Image processing is the process of transforming an image into a digital form and performing certain operations to extract useful information from it. In this field, all images are treated as 2D signals, and predetermined signal processing methods are applied to enhance their quality, recognize objects, and perform other tasks.

2.1.2 Why do we need it?

Our job is to build a system capable of recognising a USB box's size just by seeing it, the input is a picture of a rotated and out of perspective BOX , and the output should be a number specifying the size of the USB.

Example : the input is as follows :



Figure 2.1: ASA's Manhattan 16 GB USB box

The output in this case should be : **16 GB** Therefore a bunch of image processing techniques are used to reach the desired result.

2.2 Image Scanning

Image scanning is a modern technique used by new camera systems to digitally alter images to improve the readability of documents contained in images, and it contains a few steps which are **Contour detection**, **Warp Transformation**, **Auto Rotation** and **CLAHE application**.

2.2.1 Direct Contour Detection

Contour detection is the task where the program would need to detect the box's edges, identify them and draw them using open-cv functions like **findContours** and **drawContours** but in the case of our study, it wasn't a very straight forward job as the complexity of the image made it impossible for a correct contour detection.



Figure 2.2: RAW findContours output

The details in the USB Box need to be eliminated or separated from the box itself for the contour detection to occur.

2.2.2 Masking

HSV color space:

We can use the **HSV Color space** to filter out the components of the USB box before trying the contour detection.

The HSV color space stands for hue, saturation, and value. It is a cylindrical-coordinate representation of points in an RGB color model. Here's what each component means:

- **Hue:** The angle around the central vertical axis, starting at the red primary (0°), passing through the green primary (120°), and the blue primary (240°), then wrapping back to red (360°).
- **Saturation:** The distance from the central axis, representing color intensity.
- **Value:** The distance along the central axis, indicating brightness or lightness.

By converting the image from **RGB** to **HSV** and using an HSV mask we can filter out all the contents of the USB box and make it ready for contour detection, we do that by specifying a range of colors that we want (upper and lower bounds) and then we create a mask with only colors contained in that specific range, and then use the **drawContours** and **cv.FILLED** method to fill the mask with the same white as the USB box.

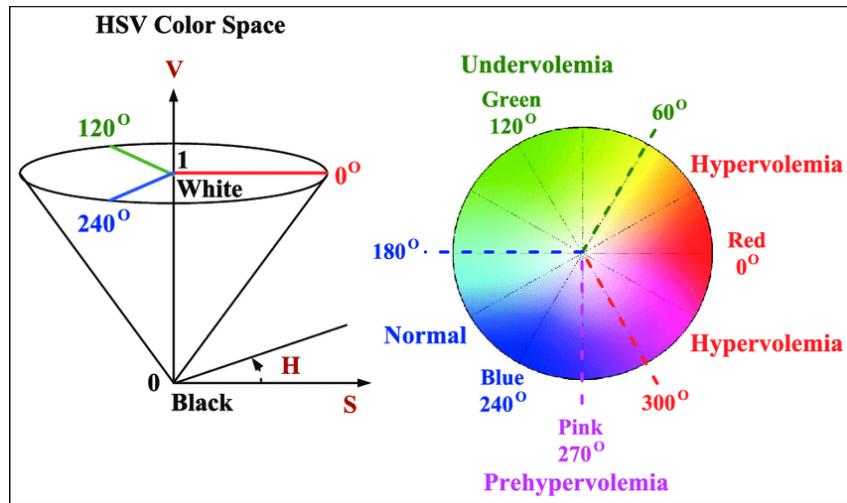


Figure 2.3: HSV Color Space

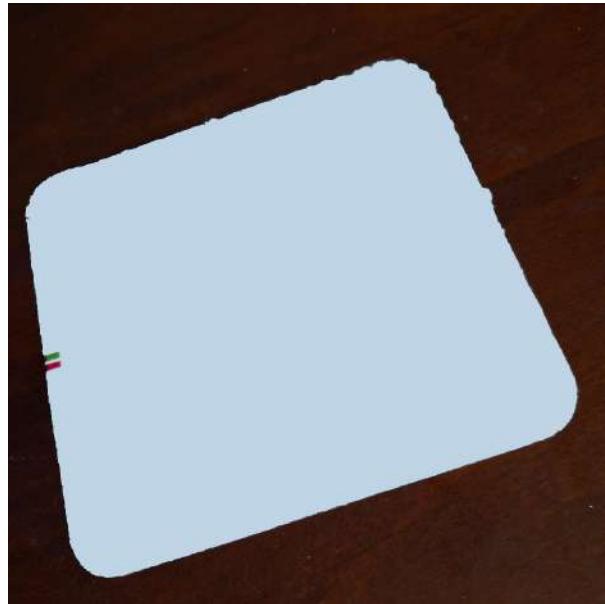


Figure 2.4: USB Box with filtered details

We obtain the following result :

Now that we have filtered the image of all unnecessary details (for the contour detection) we can detect edges accurately using the `findContours` function, we can then take the contour points coordinates and place them on the original non-filtered image to obtain the desired result at the end.

Here is the result :



(a) Old contour detection



(b) Contour detection with masking

Figure 2.5: Comparison between the two methods

2.3 Warp Transformation

2.3.1 Polygon approximation

Now that we have an accurate contour around the USB box, we need to approximate it with a quadrilateral (a 4 points polygon), and we can do that by calculating the convex hull of the contour. The **convex hull** of any given set of points S is the minimum polygon that contains all of those points S denoted by $\text{conv } S$.

By using the convex hull and drawing out the polygon, we obtain the following contour that results in the warp transformation.

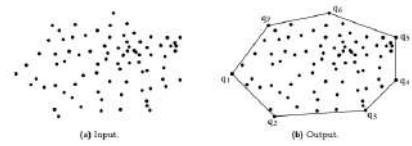


Figure 2.6: Convex Hull of points



(a) Quadrilateral points



(b) Warp into perspective

Figure 2.7: Warp Transformation

2.3.2 Auto-rotation

In the last example the picture was already correctly oriented but that is generally NOT the case as it can be rotated in multiples of 90 degrees thus we need a system that can detect the angle of rotation and rotate the image correctly.

To do that we thought of finding a reference in the image that we use to know the angle of rotation, and we observed that the **ASA** logo is situated on the top right of the image when it is correctly oriented, but also the logo is the only part of the image that contains a bright red color, making it the perfect choice for our system to detect the orientation and correct it.

If we detect that the red presence is on the **top right** of the image then it is correctly oriented, if we detect it on the **bottom right** then it needs a **90°** degrees rotation , if we detect it on the **bottom left** of the image then it needs a **180°** degrees rotation and if it's on the **top left** of the image then it needs a **-90°** degrees rotation.

To detect the red and orange-ish hues of the **ASA** logo, we used another mask with different range values, to only detect bright red hues, and then used numpy functions to calculate the average red intensity in the 4 150x150 pixel corners of the image.

Here is the mask that we used :

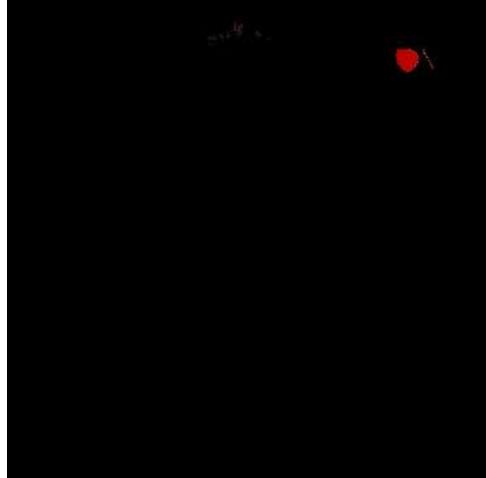


Figure 2.8: Red Hues Mask

2.3.3 CLAHE Application

Contrast Limited Adaptive Histogram Equalization (CLAHE) is a variant of adaptive histogram equalization (AHE) used to enhance contrast in images.

AHE computes histograms for distinct sections of an image and redistributes lightness values to improve local contrast and edge definitions. However, **AHE** can overamplify noise in homogeneous regions.

CLAHE limits contrast amplification to prevent noise issues.

Here is the CLAHE histogram :

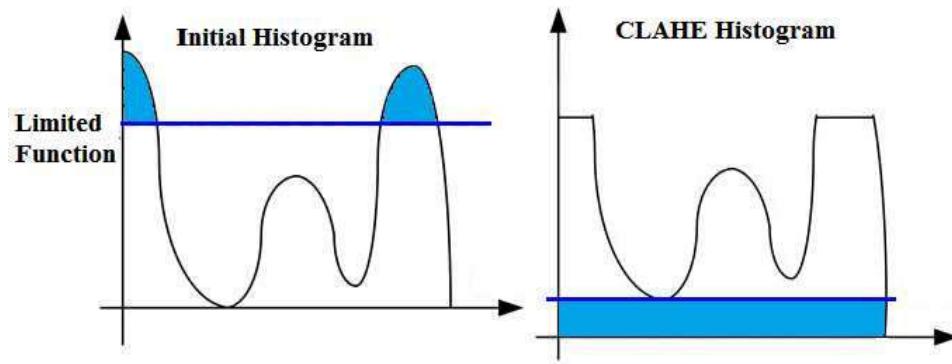


Figure 2.9: CLAHE Histogram

The result on the image of the USB box:



Figure 2.10: CLAHE Application

What remains is determining the exact coordinates of the numbers on the USB box which is an easy task to do, we binarize and apply a Gaussian blur to the cropped image to get better results.

16

Figure 2.11: Cropped image

Chapter 3

Extracting the Data

3.1 Optical Character Recognition

OCR (Optical Character Recognition) is a technology that converts images of typed, handwritten, or printed text into machine-encoded text. This process enables the extraction of text from images, making it searchable and editable.

Pytesseract is a Python library that facilitates the use of OCR by serving as a wrapper for Google's Tesseract-OCR Engine. The Tesseract-OCR Engine is a robust and widely-used tool for text recognition, known for its accuracy and support for numerous languages and character sets.



Despite Pytesseract's capabilities, we chose not to use it because the font on our USB devices is incompatible with Tesseract's OCR methods, leading to inaccurate text extraction. Our application demands high precision, so we developed a custom KNN model instead.

3.2 K-Nearest Neighbors

We crafted a custom OCR solution with a KNN model, tailored for our USB fonts, ensuring higher accuracy. This model adeptly handles text nuances, guaranteeing precise OCR outcomes aligned with our requirements.

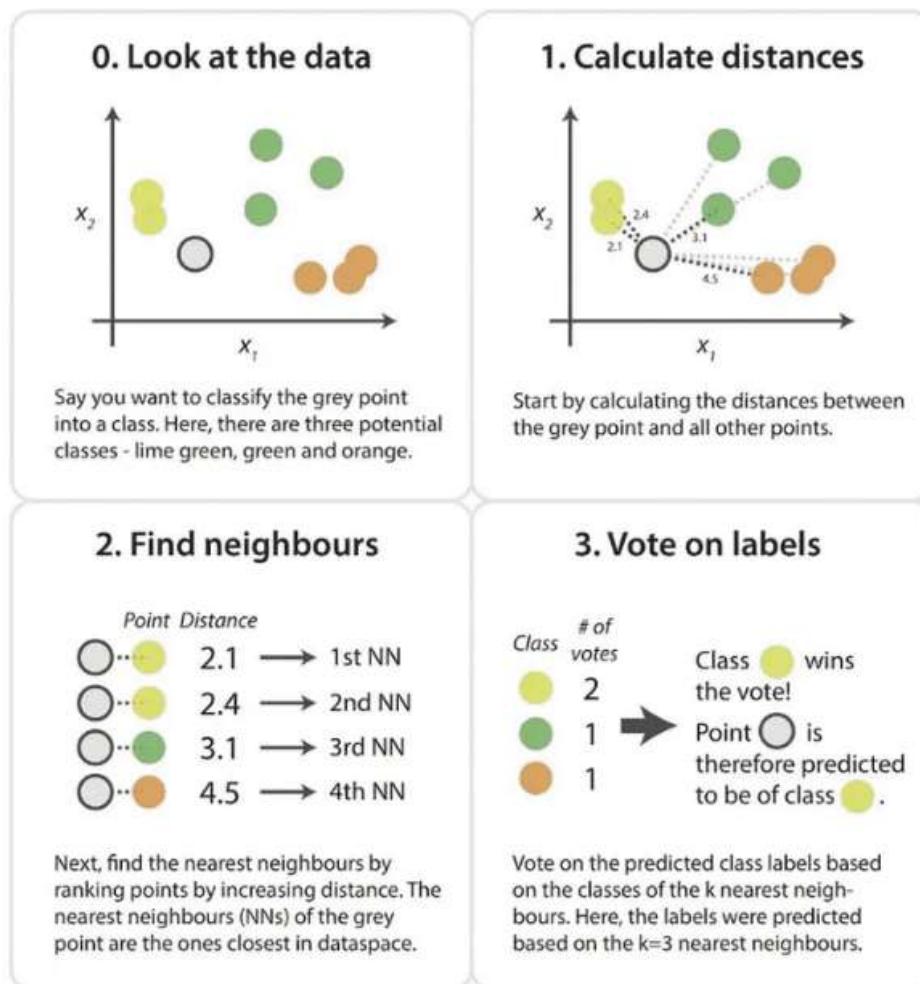


Figure 3.1: KNN Functionality

K-nearest neighbors (KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data.

3.2.1 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) is a method used to understand the shape and structure of images. It works by looking at the direction and strength of edges in different parts of the image. This information is then used to create a kind of "map" that represents the overall texture and pattern of the image. HOG is often used in tasks like recognizing objects in pictures.

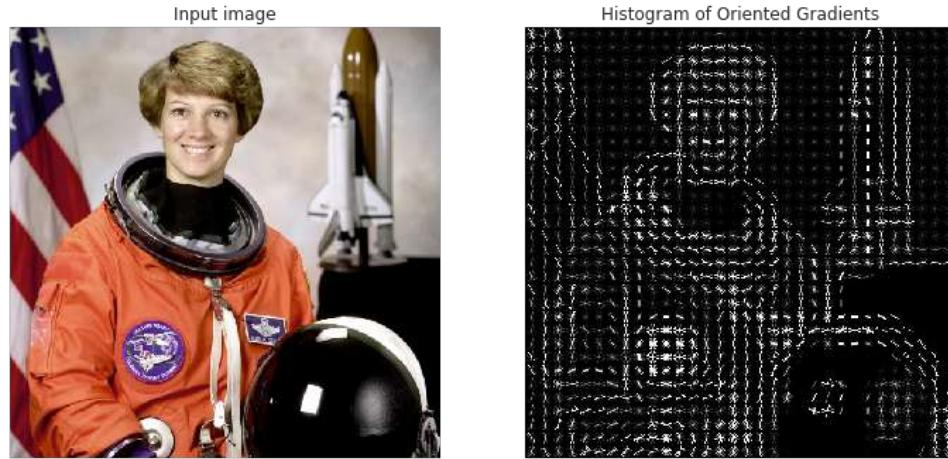


Figure 3.2: HOG Result

HOG excels in capturing the unique patterns of digits, enabling precise number extraction from images, even amidst varying backgrounds or lighting. Its emphasis on edge and gradient features enhances accuracy, ensuring reliable digit recognition for our application.

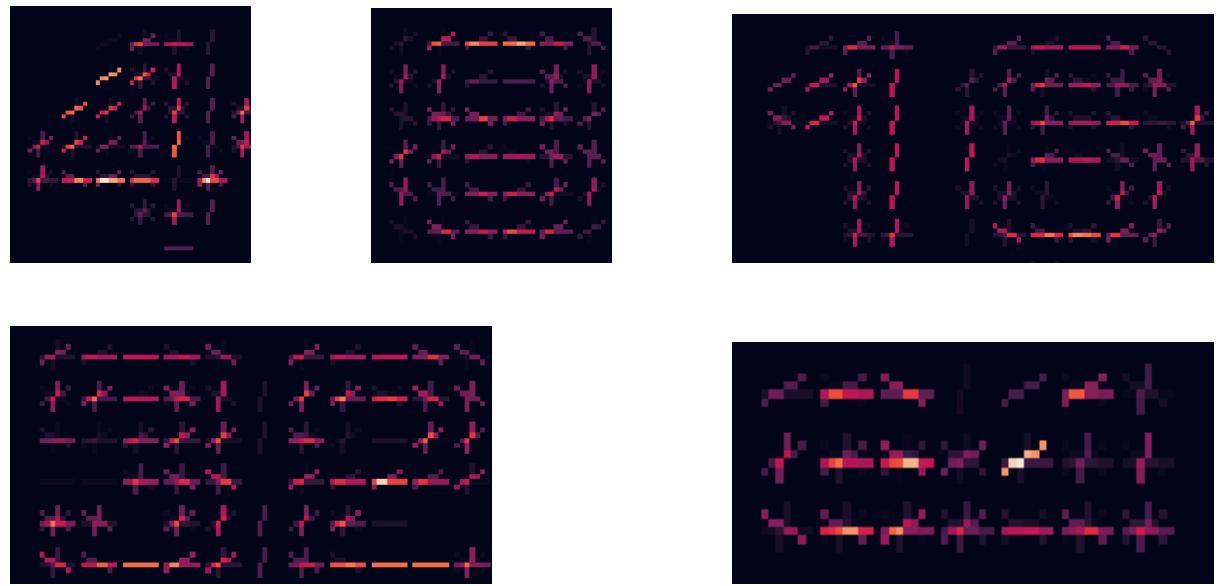


Figure 3.3: HOG Image of the numbers

3.2.2 Labeling & KNN Training

The result of our HOG function is a tuple that contains the image showed before as well as the features of our image. We'll use Pandas dataframe to manipulate that 2 dimensional data easily.

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.



We got about 553536 feature on our dataset which is huge and hard to manage but we can see that we have a lot of NaN columns that we should get rid of

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 553526 | 553527 | 553528 | 553529 | 553530 | 553531 | 553532 | 553533 | 553534 | 553535 |
|---|----------|-----|-----|-----|----------|-----|----------|-----|-----|----------|-----|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 0.433861 | 0.0 | 0.0 | 0.0 | 0.433861 | 0.0 | 0.433861 | 0.0 | 0.0 | 0.433861 | ... | NaN |
| 1 | 0.408248 | 0.0 | 0.0 | 0.0 | 0.408248 | 0.0 | 0.408248 | 0.0 | 0.0 | 0.408248 | ... | NaN |
| 2 | 0.427526 | 0.0 | 0.0 | 0.0 | 0.516188 | 0.0 | 0.201538 | 0.0 | 0.0 | 0.516188 | ... | NaN |
| 3 | 0.267548 | 0.0 | 0.0 | 0.0 | 0.542404 | 0.0 | 0.189185 | 0.0 | 0.0 | 0.542404 | ... | NaN |
| 4 | 0.155568 | 0.0 | 0.0 | 0.0 | 0.526178 | 0.0 | 0.220006 | 0.0 | 0.0 | 0.526178 | ... | NaN |

Figure 3.4: Results of Pandas' Head

Using dropNaNs function of Pandas on the axis 1, we got about 3888 features which is just good to operate with

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 3878 | 3879 | 3880 | 3881 | 3882 | 3883 | 3884 | 3885 | 3886 | 3887 |
|---|----------|-----|-----|-----|----------|-----|----------|-----|-----|----------|-----|------|------|------|------|------|------|------|------|------|------|
| 0 | 0.433861 | 0.0 | 0.0 | 0.0 | 0.433861 | 0.0 | 0.433861 | 0.0 | 0.0 | 0.433861 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 0.408248 | 0.0 | 0.0 | 0.0 | 0.408248 | 0.0 | 0.408248 | 0.0 | 0.0 | 0.408248 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.427526 | 0.0 | 0.0 | 0.0 | 0.516188 | 0.0 | 0.201538 | 0.0 | 0.0 | 0.516188 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.267548 | 0.0 | 0.0 | 0.0 | 0.542404 | 0.0 | 0.189185 | 0.0 | 0.0 | 0.542404 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 0.155568 | 0.0 | 0.0 | 0.0 | 0.526178 | 0.0 | 0.220006 | 0.0 | 0.0 | 0.526178 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Figure 3.5: Results of Pandas' Head after features drop

We opted for a 60/40 split of our dataset for training and testing. Despite its small size, allocating 60% for training allows our model to learn more effectively. However, with only 40% reserved for testing, evaluation may be less robust. Despite this, the balance ensures reasonable performance assessment. Cross-validation will further validate our model's performance.

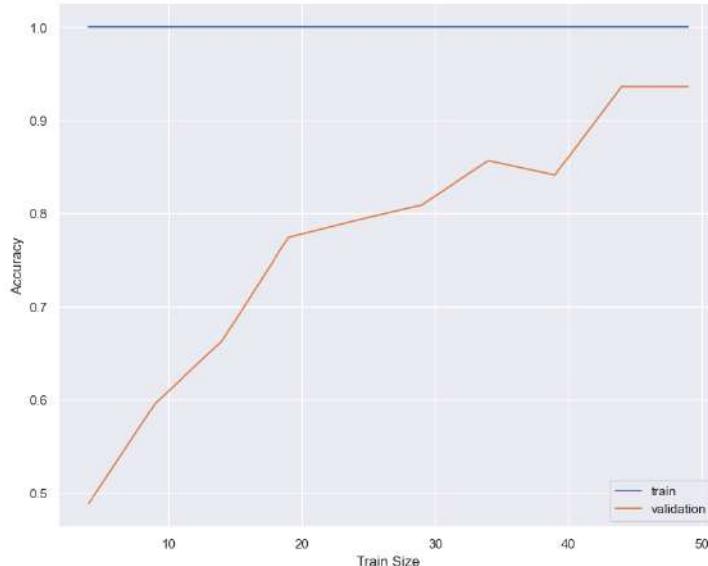


Figure 3.6: The learning curve

3.2.3 Results

With a training model accuracy of 87.10%, our results are promising given the small dataset. To improve accuracy, we can gather more image data. A larger dataset offers greater diversity for the model to learn from, potentially boosting performance without altering the model architecture.

Confusion matrix:

In machine learning, a confusion matrix is a visualization tool used to evaluate the performance of a classification model. It provides insights into how well the model is classifying data points by comparing the predicted labels with the actual labels.

We used the Seaborn library for visualizing the confusion matrix. All predictions were close except for



Figure 3.7: Confusion matrix

the third label (32GB), which lacks sufficient data. Adding more images for this label can improve the model's understanding and accuracy.

Chapter 4

Conclusion

In our internship, the primary objective was to develop a USB box classifier system capable of determining the box size solely from an image, regardless of camera orientation or perspective. To achieve this, we broke down the main task into smaller components, addressing specific challenges. We delved into techniques like optimal contour detection and utilized the HSV color space for efficient filtering and automatic segmentation.

Regarding classification, we explored various methods to identify USB sizes from cropped images. Ultimately, we opted to create our own KNN-based model for accurate classification. The internship was both interesting and enjoyable, presenting a raw problem that demanded improvisation and strong problem-solving skills to devise effective strategies for each aspect.

References

- [1] Cornell Computer Science Departement. “Transformation & Wrapping”. In: (2012).
- [2] OpenCV Documentation. “Adaptive Histogram Equalization”. In: (2017).
- [3] ASA FLASH. “ASA’s General Informations”. In: (2019).
- [4] Built in. “Histogram of Oriented Gradients: An Overview”. In: (2023).
- [5] pip. “Pandas”. In: (2024).
- [6] pip. “Seaborn”. In: (2024).
- [7] Dennis T. “Confusion Matrix Visualization”. In: (2019).
- [8] Analytics Vidhya. “A Complete Guide to K-Nearest Neighbors”. In: (2024).
- [9] Wikipedia. “HSV & HSL”. In: (2020).
- [10] Wikipedia. “Optical Character Recognition”. In: (2023).