الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

Université Badji Mokhtar - Annaba

Badji Mokhtar – Annaba University

جامعة باجي مختار –عنابــة

Faculté : Technologie

Département : Informatique

Domaine : Mathématique-Informatique

Filière : Informatique

Spécialité : Systèmes Informatiques et Décision

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème

Genetic Algorithm-Driven Image Characteristic Extraction Through Color Shifting, masking and Thresholding Method

Présenté par : Chelghoum Mohammed Ouassim

Encadrant : Pr. Seridi Hacina Professeur Université Badaj Mukhtar Annaba

Jury de Soutenance :

Dr. Hamadeche Billel	MCB	Université Badaj Mukhtar Annaba	Président
Pr. Seridi Hacina	Professeur	Université Badaj Mukhtar Annaba	Encadrant
Dr. Hariri Walid	MCA	Université Badaj Mukhtar Annaba	Examinateur

Année Universitaire : 2023/2024

TABLE OF CONTENTS

ABSTRACT

One of the most popular machine learning models for classifying images is neural networks. But because neural networks are so dependent on the input data, even a small change in the input can have a significant impact on the output.

In this master thes is, we present a novel approach for image preprocessing and classification optimization using genetic algorithms in conjunction with HSV color thresholding, channel shifting, and image masking techniques. Our goal is to build a highly efficient model with optimal architecture and classification accuracy. We tested our approach on 12 diverse datasets collected from Kaggle archives, which vary in data types and classification complexities.Our experimental results demonstrated that the proposed approach achieved more efficient results on 10 out of the 12 datasets compared to traditional methods. This highlights the effectiveness of our genetic algorithm-based optimization in improving image classification accuracy across various types of data.

RÉSUMÉ

L'un des modèles d'apprentissage automatique les plus populaires pour la classification des images est les réseaux neuronaux. Cependant, en raison de la forte dépendance des réseaux neuronaux aux données d'entrée, même un petit changement dans l'entrée peut avoir un impact significatif sur la sortie. Dans cette thèse de master, nous présentons une approche novatrice pour le prétraitement des images et l'optimisation de la classification en utilisant des algorithmes génétiques en conjonction avec le seuillage de couleur HSV, le décalage de canal et les techniques de masquage d'image. Notre objectif est de construire un modèle très efficace avec une architecture et une précision de classification optimales. Nous avons testé notre approche sur 12 ensembles de données divers collectés dans les archives de Kaggle, qui varient en types de données et en complexité de classification. Nos résultats expérimentaux ont démontré que l'approche proposée a obtenu des résultats plus efficaces sur 10 des 12 ensembles de données par rapport aux méthodes traditionnelles. Cela souligne l'efficacité de notre optimisation basée sur les algorithmes génétiques pour améliorer la précision de la classification des images à travers divers types de données.

ملخص

يعد نموذج الشبكات العصبية من أكثر نماذج التعلم الآلي شيوعاً لتصنيف الصور .ولكن بسبب اعتماد الشبكات العصبية الكبير على بيانات الإدخال، يمكن أن يكون لتغيير طفيف في الإدخال تأثير كبير على المخرجات .في هذه الرسالة للماجستير، نقدم نهجاً جديداً لمعالجة الصور وتحسين التصنيف باستخدام الخوارزميات الجينية بالتزامن مع تحديد عتبة الألوان في نظامHSV ، وتحويل القنوات، وتقنيات قناع الصورة .هدفنا هو بناء نموذج عالي الكفاءة يتمتع بهيكلية ودقة تصنيف مثالية .لقد اختبرنا نهجنا على 12مجموعة بيانات متنوعة تم جمعها من أرشيفاتKaggle ، والتي تتنوع في أنواع البيانات وتعقيدات التصنيف .أظهرت نتائجنا التجريبية أن النهج المقترح حقق نتائج أكثر كفاءة في 10 من أصل 12 مجموعة بيانات مقارنة بالطرق التقليدية .وهذا يبرز فعالية تحسيننا المستند إلى الخوارزميات الجينية في تحسين دقة تصنيف الصور عبر أنواع البيانات المختلفة.

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The world has witnessed incredible changes over the years in every aspect of life, but the field of technology deserves special attention since we now have incredibly strong machines that can handle vast amounts of data. Not only have we eliminated all traditional paper-based work, but we have also succeeded in surpassing previously unheard-of platforms thanks to storage space for storing these data and speed and connectivity to transfer data worldwide.

Classification is one of these ideas that shows a lot of promise, and over the past few decades, many models and innovative methods have been developed in it. However, as with all human endeavors, There is always a chance of optimization, where one would reach a point and the other would continue from there and so on, with knowledge that evolves gradually.neural networks are the ones we are interested in because they are straightforward to operate but produce incredibly effective results. However, there is a small drawback to neural networks the set attributes that are used to construct a neural network model may not always be the best ones, and if we change those attributes, the classification results will change with each new set. Therefore, finding the optimal set has always been the best case scenario when using neural networks. To solve this issue, various strategies were tested in an effort to find a better model each time.

With feature extraction, Classification can take advantage of the underlying relationships and structures that are concealed within the data; varying representations result in varying degrees of images data complexity and can influence the overall performance of the classification. There are currently many different types of feature extraction on algorithms available; they all offer unique data transformation and preprocessing pipelines that effectively support classification performance .

Noisy data are labour-intensive and require a deep perception of the used classification algorithm to acquire convenient outcomes for a specific task, therefor improved performance is likely to be achieved through fine-tuning the chosen algorithm's parameters to the problem at hand. Furthermore, because of the links between the features that are concealed in the images data, these parameters can be quite complex and difficult to adjust. This leads inevitable huge characteristic loss.

This thesis introduces a novel autonomous approach that terminates the reduction and alteration process in response to the provided data and its hidden linkages. For improved feature extraction, combining image processing methods with deep learning models is a viable approach in addition to conventional feature extraction strategies. We can combine techniques like edge detection, color filtering, and genetic algorithms assessed by neural network designs to take advantage of each method's advantages and capture complex visual data. By making it easier to extract rich and meaningful representations from images, this fusion improves the performance of classification models on a variety of tasks. The combination of these methods yields more accurate and dependable results by strengthening the model's resilience and generalization abilities in addition to improving feature extraction.

This work will be structured in the following way:

- **Chapter 1:** present at first glance some concepts related to the subject of the project, and talk about some existing work on the same problem, and finally we expose the remaining problems.
- **Chapter 2:** we talk about the objectives of our solution and the algorithm

that we built to obtain (or converge) those results.

- **Chapter 3:** we talk about the data sets used, the method of evaluation and the results obtained with a discussion on these results.

The master thesis ends with a general conclusion, what’s done and what remains to be found.

CHAPTER 1 Feature Extraction

The process of extracting features from images entails locating and expressing unique structures inside the image. Analyzing an image's pixel composition is undoubtedly one. However, this is a basic functionality. Edges, corners, and even more intricate textures shapes and color all be considered high-level features in a picture.

Features are characteristics of an image. With these unique characteristics, you may be able to distinguish one image from another. This is the first step in computer vision. By extracting these features, you can create representations that are more compact and meaningful than merely the pixels of the image. It helps further analysis and processing.

In this chapter, we will introduce the concept of feature extraction such as a central principle concept of Deep learning, and finally The importance of feature extraction in Images preprossing and Predictive Modeling.

1.1 Representation learning

In machine learning, feature learning or representation learning is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data. This replaces manual feature engineering and allows a machine to both learn the features and use them to perform a specific task. [8]

The need for input that is easy to process mathematically and computationally drives feature learning in machine learning tasks like classification. Nevertheless, attempts to define certain features algorithmically have not succeeded when applied to real-world data, such as images, videos, and sensor data. An alternative to using explicit algorithms is to find these features or representations through inspection.

Unlabeled input data is used to learn features in unsupervised feature learning. Dictionary learning, independent component analysis, auto encoders, matrix factorization, and various clustering techniques are a few instances.

1.2 Images Definition

An image is rather difficult to describe in a general way. An image is a representation of the world. In image processing, most of the time, it is considered to be a mathematical function from RxR to R, where the input pair is considered as a spatial position and the output singleton as the intensity (color or grayscale levels) of the physical phenomenon. However, it happens that the image is called "3D," so the function is from RxRxR to R. Color images can be represented either by three images representing the three fundamental colors, or by an image from RxR to RxRxR. A digital image is an image whose surface is divided into elements of fixed sizes called cells or pixels, each with a characteristic grayscale or color level taken from the corresponding location in the real image, or calculated from an internal description of the scene to be represented.

1.3 What is a Feature?

An individual measurable property in a recorded dataset is called a feature. In statistics and machine learning, features are frequently referred to as "variables" or "attributes." The use case of a model is correlated with or affected by relevant features. Features in a patient medical dataset may include blood pressure, cholesterol, age, gender, and other pertinent observed characteristics.

1.4 Images Features Types

1.4.1 Spectral features

The image is processed as a matrix of pixels when spectral characteristics are used. Each of these represents the color and brightness levels that are present at the respective spots in the image [1]. For this field, any relationship between pixels is meaningless because forms and contours are not of importance to us. The fact that these attributes are independent of image size and coding deformations is one benefit of using them in processes.

Color space, in which the distribution of colors throughout the image is of interest, is the most fundamental kind of spectral feature.

1.4.2 Shape Features

Shape recognition is among the most fundamental image processing issues. While it is fairly simple for a human to identify shapes in an image, existing algorithms require a great deal of effort to do the same task. As a result, numerous features based on shape were created to aid in this procedure. It was demonstrated that it is crucial to separate these features into two groups: the first group consists of features that are translation, rotation, and scaling invariant, while the other group consists of features that lack these attributes. Typically, features from the first category are easier to extract and call for less complicated methods [2].

1.4.3 Texture features

Texture features are patterns in images that are homogeneous in ways other than arising from the existence of a single color or intensity. These characteristics include crucial details on the surface's structural arrangement and interaction with the environment [1].

1.5 Why is Feature Extraction Important?

Feature extraction plays an important role in image processing. This technique is used to detect features in digital images such as edges, shapes, or motion. Once these are identified, the data can be processed to perform various tasks related to analyzing an image.

1.6 Common Feature Extraction Techniques

1.6.1 Local Binary Pattern

LBP works by comparing the intensity of a central pixel in a small neighborhood with the intensity of its surrounding pixels. Each pixel in the neighborhood is assigned a binary value based on whether its intensity is greater than or less than the intensity of the central pixel (threshold). These binary values are then concatenated into a binary number, which represents the texture of that neighborhood.[10]

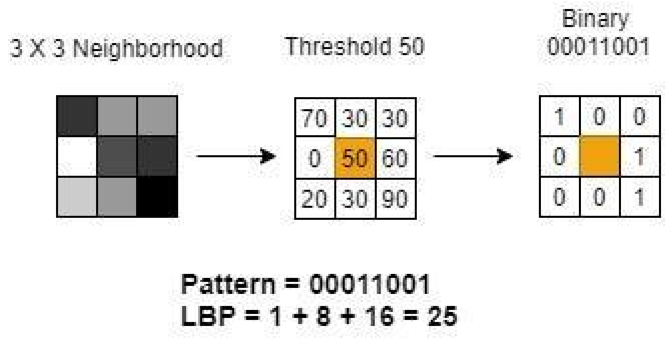


Figure 1: How LBP works

1.6.2 Histogram of Oriented Gradients

Histogram of oriented gradients method is also quite similar to edge orientation histograms. The HOG descriptor focuses on the structure or the shape of an object. It's better than any edge descriptor as it uses magnitude as well as the angle of the gradient to compute the features. For the regions of the image, it generates histograms using the magnitude and orientations of the gradient.[11]

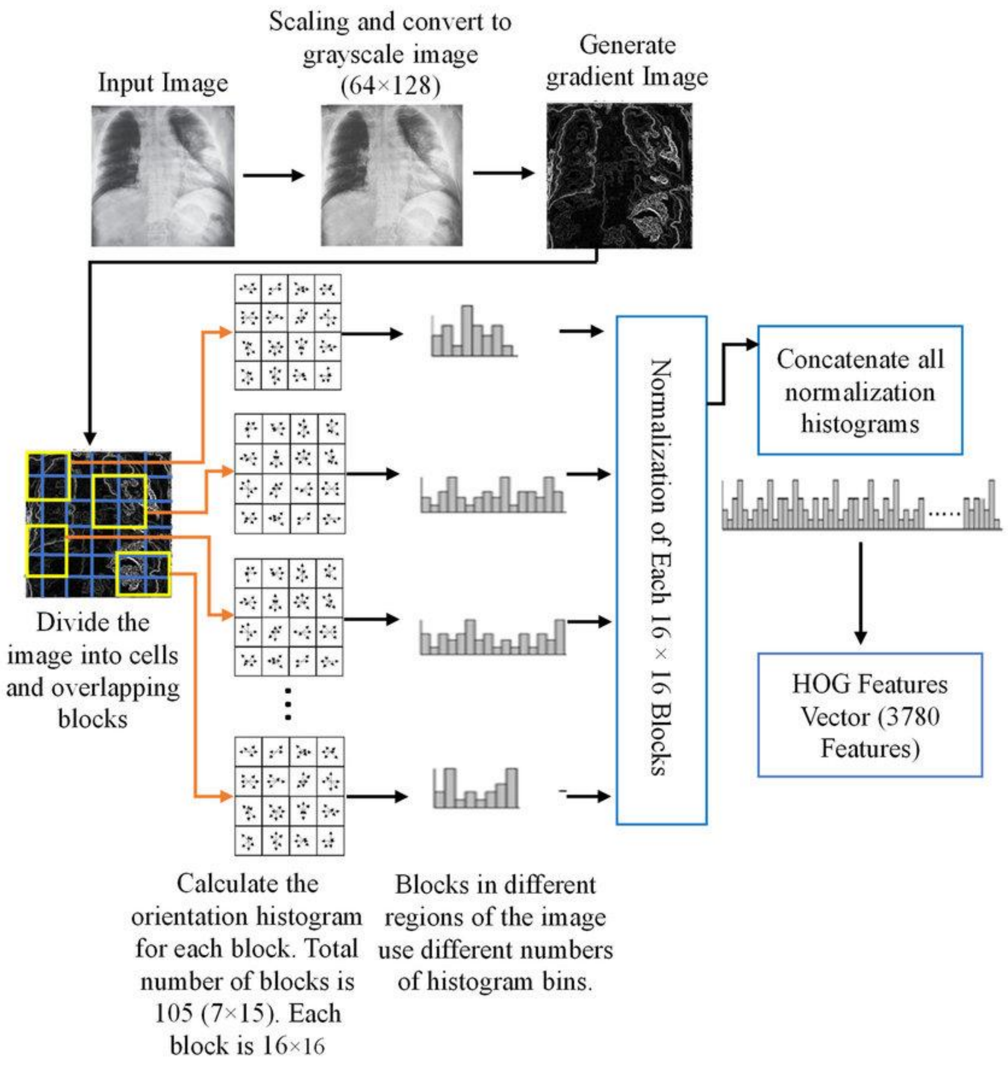

Input Image
Scaling and convert to grayscale image (64×128)
Generate gradient Image
Divide the image into cells and overlapping blocks
Calculate the orientation histogram for each block. Total number of blocks is 105 (7×15). Each block is 16×16
Blocks in different regions of the image use different numbers of histogram bins.
Normalization of Each 16 × 16 Blocks
Concatenate all normalization histograms
HOG Features Vector (3780 Features)

Figure 2:COVID-19 Detection from Chest X-ray Images Using Feature Fusion and Deep Learning Md. Abdul Based and all (2021)

1.6.3 Autoencoders

Autoencoders can identify key data features. The autoencoder concept hinges on learning from the coding of the original data sets to derive new, more potent features. It achieves this by training a neural network to recreate its input, which forces it to discover and exploit structures in the data. Through this process, autoencoders reduce dimensionality and extract significant features from the data, contributing to more effective machine-learning models.[3]

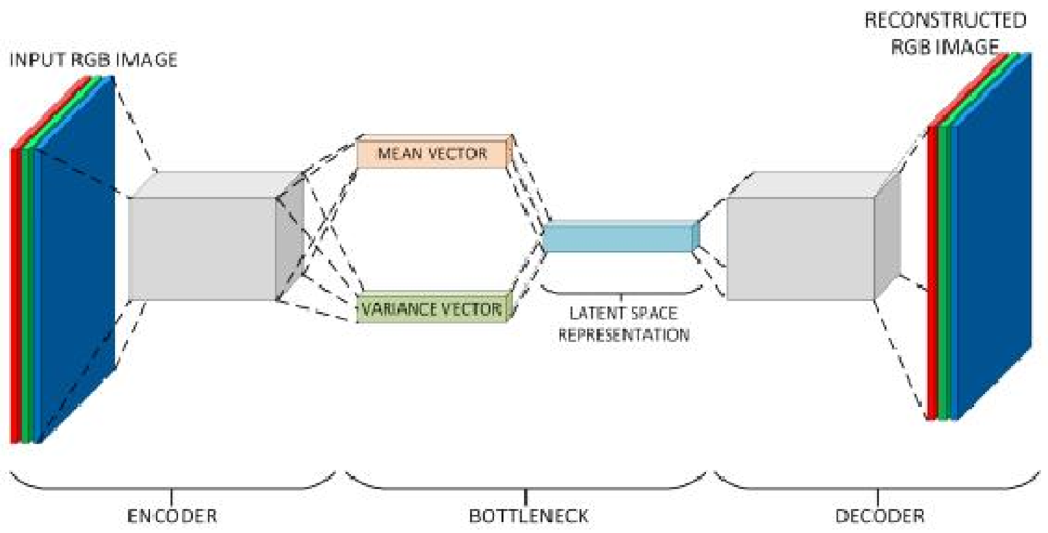


Figure 3: The Basic Concept of Autoencoder

1.6.4 Image Processing Techniques

Image processing techniques involve raw data analysis to identify and isolate significant characteristics or patterns in an image. This could involve identifying edges and corners or extracting features like color, texture, and shape. These features can then be used for tasks such as image classification, object detection, and image segmentation.[3]

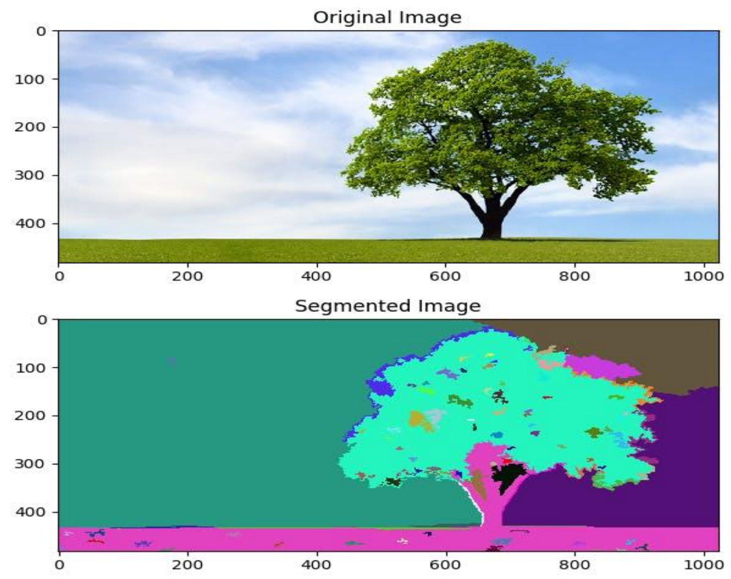


Figure 4: Segmented image example

1.7 Color spaces

Although color is perceived as a feeling rather than a characteristic measurement like length or temperature, an electromagnetic beam with a noticeable frequency can be quantified as a characteristic amount, so there is a suitable way to depict numerical requirements.

1.7.1 RGB color space

RGB color space may be a broadly utilized color space for image description. It includes three color elements red, green, and blue. Its representation consists of 24-bit usage whereby 8 bits are allocated to each filter R, G, and B respectively. This means that each channel incorporates an extension of values between 0 to 255 RGB representation. One remarkable shortcoming of the RGB color space is that it isn't perceptually uniform implying that the determined separation in RGB space does not genuinely relate to the perceptual color distinction [4].

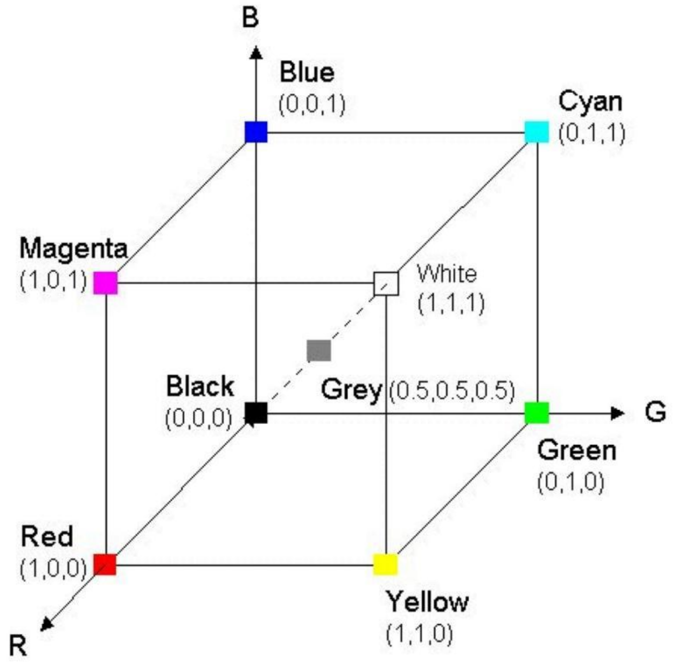


Figure 5: RGB representation.

1.7.2 HSV color space

HSV stands for hue, saturation, and value and it was developed with the aim of determining the human perspective of colors.It relates color (hue) ranging from 0 to 360, saturation (shade) ranging between 0–100 percent and sometimes is referred to “purity” and brightness (value) ranging between 0–100 percent. HSV is cylindrical geometry, with hue, their precise measurements, beginning at the red key at 0°, going through the green essential at 120° and the blue essential at 240°, and afterward to red at 360° [5].

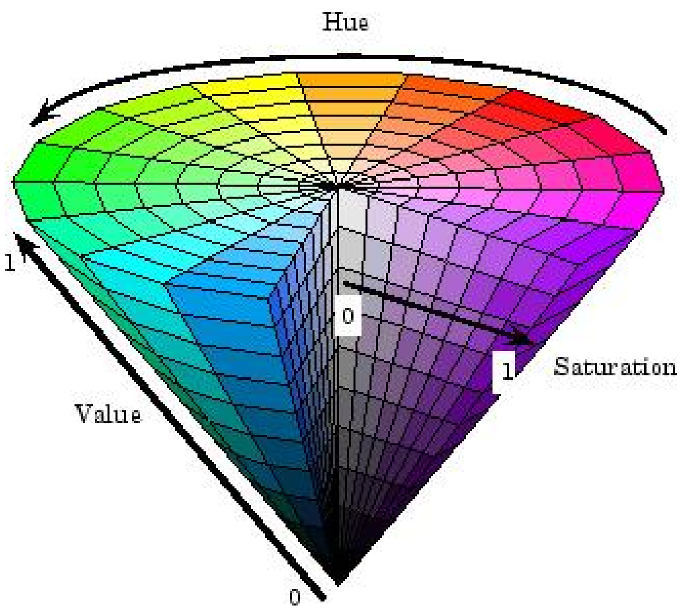


Figure 6: HSV representation.

The formula Eq. (1) transform from RGB to HSV:

$$H = \begin{cases} \theta & B \le G \\ 360^\circ & B > G \end{cases}$$

Where :

$$\theta = \cos^{-1} \frac{0.5 \, * \, [(R-G) + (R-B)]}{\sqrt{(R-G)^2 + (R-G)} \, * (G-B)}$$

$$S = 1 - \frac{3}{R + G + B}(\min(R, G, B))$$

$$V = \frac{1}{3}(R + G + B)$$

1.7.3 LAB color space

The CIE (Commission Internationale de lclairage) LAB color space is an additional widely used one. The International Commission on Illumination, or CIE, defined the CIELAB color space in 1976. It is also known as CIE L*a*b* or sometimes just "Lab" color space.

The first color models based on the physical properties of light were developed by the

CIE. The CIE RGB, CIE XYZ, and CIELab are the most prominent models. They all explain how every color can be created on a plane at a specific illumination level. While CIELab is derived indirectly from the CIE XYZ, the CIE XYZ and CIE RGB are computed using the light wavelength from the physic representation of the color (Sonka & Hlavac, 2008) [6].

1.8 Color Thresholding

Use color thresholding to specify a color range and return a black and white image. All colors between the start and stop colors (inclusively) become white and the rest of the image pixels become black. The two colors are separated with a hyphen between them. Thresholding, by default, take place in the sRGB colorspace. Use the -colorspace to perform the thresholding in an alternative colorspaces.(currently limited to sRGB, Gray, HSV, HSL, HCL, HSB, and HSW).

1.9 Genetic Algorithms

By definition, a genetic algorithm is a search heuristic that draws inspiration from Charles Darwin's theory of natural selection. The process of natural selection, in which the most fit individuals are chosen for reproduction in order to create offspring of the following generation, is reflected in this algorithm.[23]

To put it simply, a genetic algorithm (GA) ensures both the complete exploration of the search space and the convergence towards a global solution. It achieves this by simulating naive evolution. The basic concept involves starting with an initial population of solutions, which is generated randomly, and evaluating each solution in the population using a fitness function. The most fit individuals are then more likely to be selected to move on to the next generation, with some of the best fit individuals already set to be in that generation. For the remaining solutions, we apply genetic operations like crossover and mutation to produce new solutions.

These individuals (selected AND genetically modified) form the next generation, this process is repeated until a given criterion is met, as it is presented in the figure below.

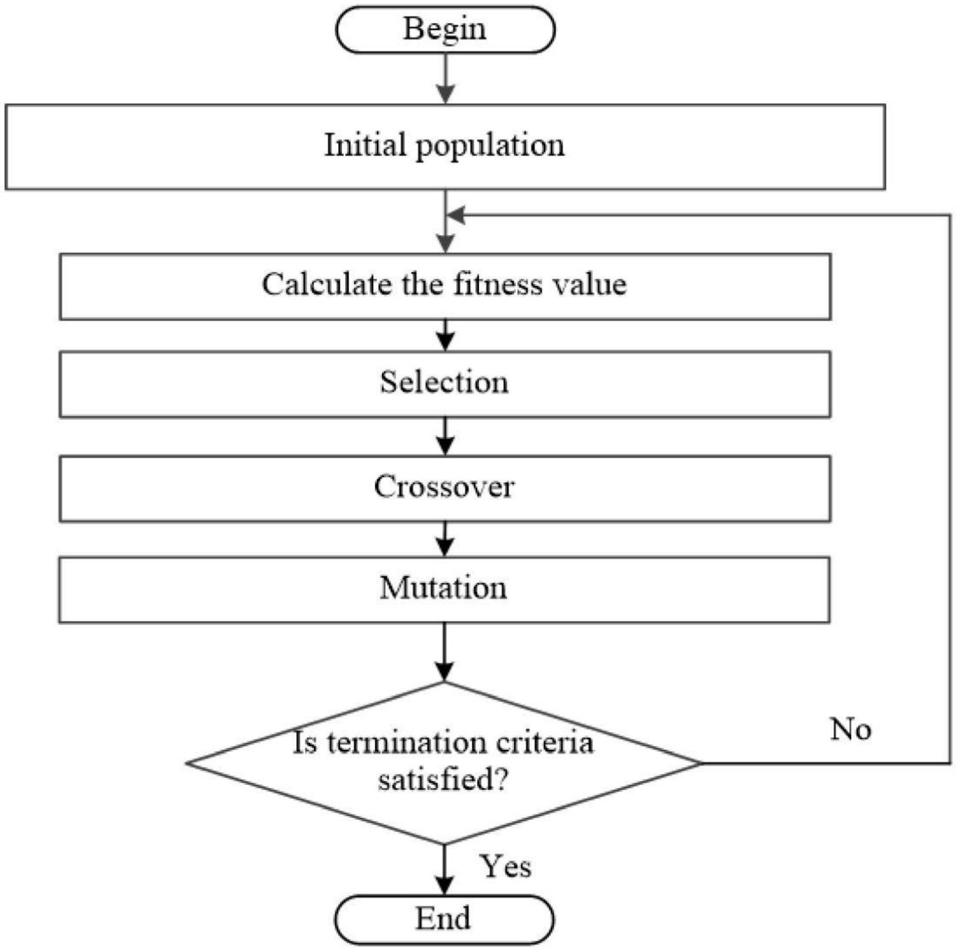

Begin
Initial population
Calculate the fitness value
Selection
Crossover
Mutation
Is termination criteria satisfied?
No
Yes
End

Figure 7: Genetic algorithm flowchart

A genetic algorithm can be simplified into four main parts: a population of individuals, each of which represents a potential solution. An assessment technique known as the fitness function is used to determine whether or not if an individual is a good solution or not. a mechanism for selection that selects the most promising members of the population to produce the next generation. Crossover and mutation are genetic operators that guarantee search space navigation.

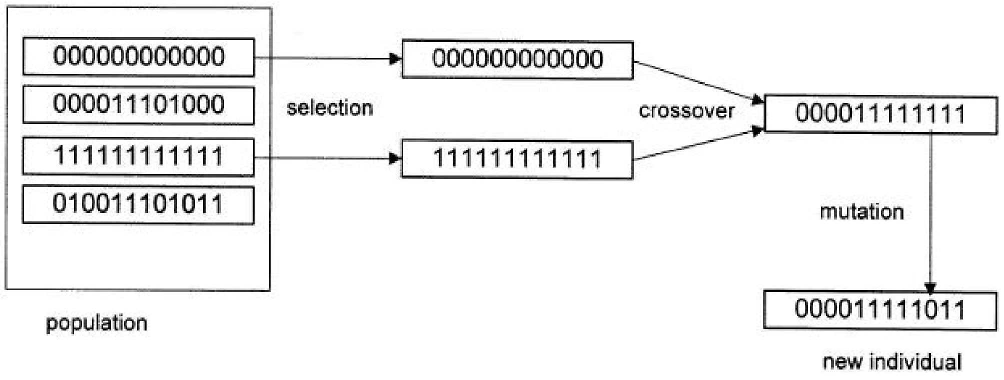


Figure 8: Genetic operations mutation and crossover

1.10 What is deep learning?

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning models can be taught to perform classification tasks and recognize patterns in photos, text, audio and other various data. It is also used to automate tasks that would normally need human intelligence, such as describing images or transcribing audio files.

Deep learning is an important element of data science, including statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier.

Where human brains have millions of interconnected neurons that work together to learn information, deep learning features neural networks constructed from multiple layers of software nodes that work together. Deep learning models are trained using a large set of labeled data and neural network architectures.

Deep learning enables a computer to learn by example. To understand deep learning, imagine a toddler whose first word is dog. The toddler learns what a dog is -- and is not -- by pointing to objects and saying the word dog. The parent says, "Yes, that is a dog," or, "No, that is not a dog." As the toddler continues to point to objects, he becomes more aware of the features that all dogs possess. What the toddler is doing, without knowing it, is clarifying a complex abstraction: the concept of dog. They are doing this by building a hierarchy in which each level of abstraction is created with knowledge that was gained from the preceding layer of the hierarchy [7].

1.11 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The architecture of CNNs is inspired by the visual processing in the human brain, and they are well-suited for capturing hierarchical patterns and spatial dependencies within images.[9]

Key components of a Convolutional Neural Network include:

Convolutional Layers: These layers use filters, also called kernels, to apply convolutional operations to input images in order to detect features like textures, edges, and more intricate patterns. The spatial relationships between pixels are preserved with the aid of convolutional operations.

Pooling Layers: By downsampling the spatial dimensions of the input, pooling layers lower the network's parameter count and computational complexity. A typical pooling operation called "max pooling" chooses the maximum value among a set of adjacent

Fully Connected Layers: Predictions are made by these layers using the high-level features that the preceding layers have learned. Every neuron in one layer is connected to every other layer's neuron through them.

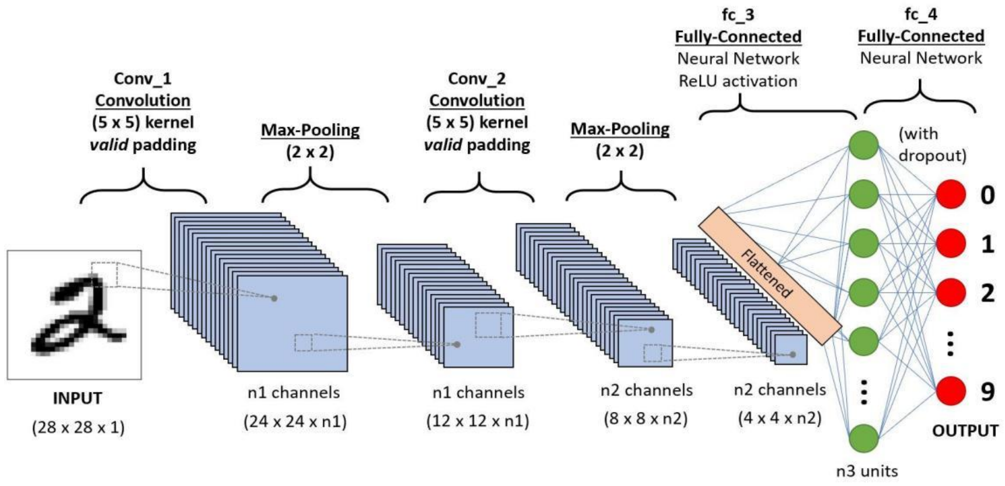


Figure 9: Convolutional Neural Network (CNN)

1.11.1 Advantages of CNN

- CNNs can achieve state-of-the-art accuracy on a variety of image recognition tasks, such as image classification, object detection, and image segmentation.
- CNNs can be very efficient, especially when implemented on specialized hardware such as GPUs.
- CNNs are relatively robust to noise and variations in the input data.
- CNNs can be adapted to a variety of different tasks by simply changing the architecture of the network.[9]

1.11.2 Disadvantages of CNN

- CNNs can be complex and difficult to train, especially for large datasets.
- CNNs can require a lot of computational resources to train and deploy.
- CNNs require a large amount of labeled data to train.
- CNNs can be difficult to interpret, making it difficult to understand why they make the predictions they do.[9]

1.12 Metrics:

1.12.1 Cross Validation

Cross validation is a technique for assessing how the statistical analysis generalizes to an independent data set.It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.

Using cross-validation, there are high chances that we can detect overfitting with ease. [15]

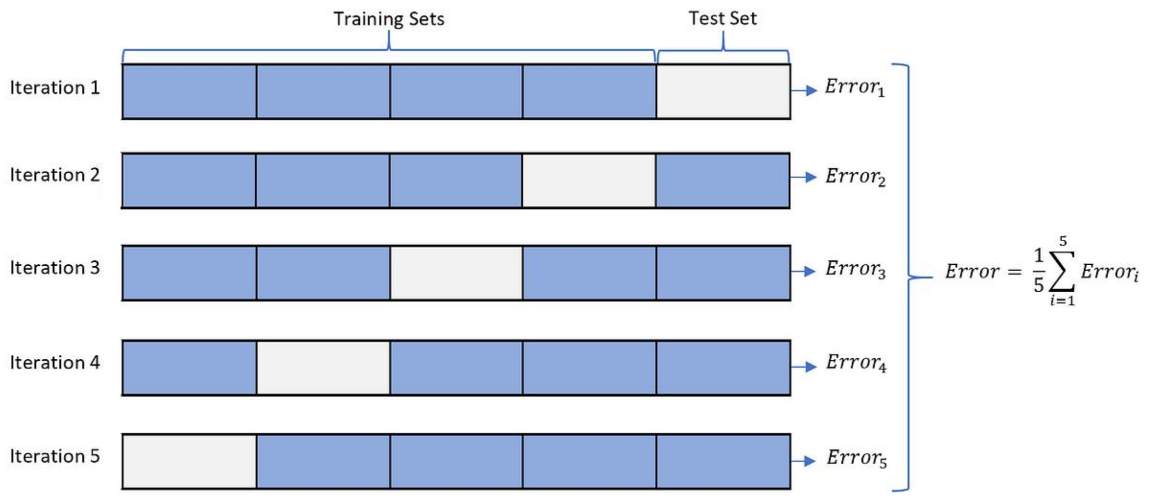


Figure 18: K folds Cross validation

1.12.2 Accuracy

Accuracy classification score.

In multi-label classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

1.13 Related works:

A Susanto, Z H Dewantoro, C A Sari3, D R I M Setiadi, E H Rachmawanto (2020) proposes a method of classification of shallots based on quality using the Naïve Bayes Classifier. The Naïve Bayes classifier is a classifier based on statistics and simple probabilities that are widely used in many sophisticated learning methods today. Shallots are classified into three classes, namely good, medium, and poor quality. To get good classification results, appropriate extraction of features is needed, in this case, color feature extraction is selected with the hue saturation value (HSV) model and to identify the size the area, metric and perimeter calculations are used. To get the right size, some morphological operations such as filling holes and opening are performed. Based on the experimental results on 60 training data and 60 testing data classification shallot quality using Naïve Bayes Classifier produces an accuracy of up to 91.67%.[12]

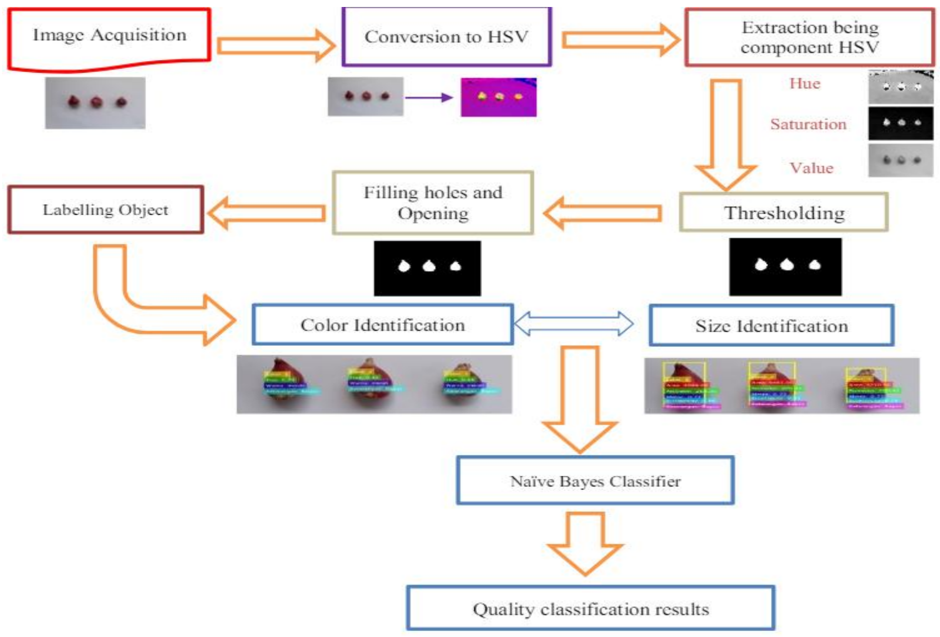
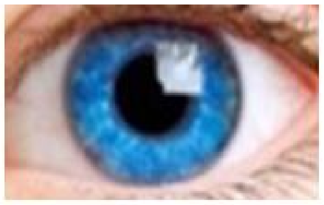
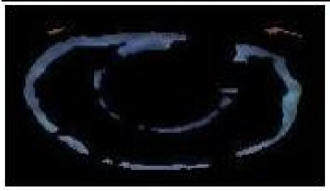


Figure 10: Susanto, Z H Dewantoro, C A Sari3, D R I M Setiadi, E H Rachmawanto Proposed Method.

Fajrul Islami (2021) In this study, the threshold method used will be combined with the HSV mode for color detection. The threshold method will separate the object and the image background, while HSV will help improve the segmentation results based on the Hue, Saturation, Value values to be able to detect objects more accurately. Segmentation is carried out using the original input image without preprocessing or direct segmentation. As we know that in digital image processing, there are steps that are usually done to get a good input image, namely preprocessing. In this preprocessing stage, processes such as image conversion and image intensity changes are carried out so that the input image is better. Therefore, even though the input image is used without going through the preprocessing stage, the object can be segmented properly based on the color type of the object. The results of this segmentation can later be used for recognition and identification of image objects. The results of the test method for object segmentation achieved a color similarity level of 25%, with an accuracy rate of 75% in detecting uniform color objects. So that this method can be one of the most effective methods in segmenting image objects without pre-processing or direct thresholding.[21]





Threshold Value	0.2
Hue	0.6098
Saturation	0.3188
Value	0.5412

Figure 11: Shows the data segmentation on the image of the original iris eye 1 with the similarity of colors 0.3102,Hue 0.6098, Saturation 0.3188, Value 0.5412

Naman Karanth, Arya Adesh , Chandan Kasamsetty, Abhinav Samaga, G. Shobha, Minal Moharir (2024) This paper presents a tool that smartly coordinates different deep-learning techniques to modify the color patterns found on different parts of a saree. MODNet is applied for background removal and custom-trained Mask R-CNN models are utilized to precisely segment the saree body and border. The subsequent application of a color-changing algorithm in the HSV color space facilitates independent color modification for the saree border and body. The methodology proposed in this paper can be extended to any kind of clothing such as shirts, trousers, kurtas, kimonos, etc. An accuracy of 93.01% was achieved for the saree border segmentation, and an accuracy of 89.23% was achieved for the saree body segmentation when tested on a set of 50 test images.[13]

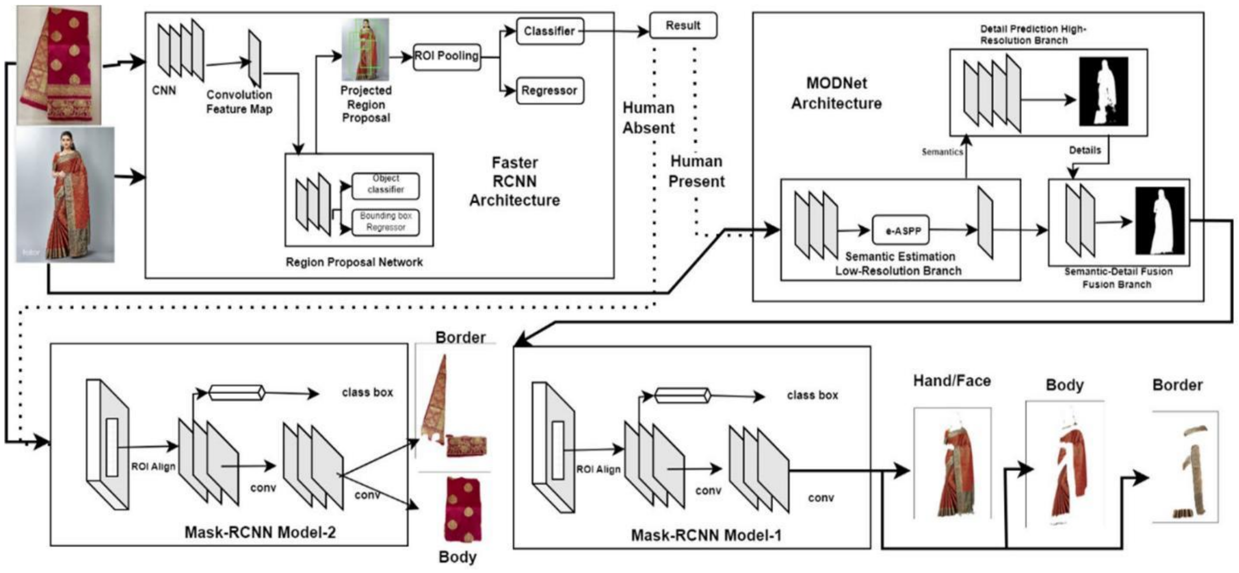


Figure 12: Block Diagram for Automatic Classification and Color Changing of Saree Components Using Deep Learning Techniques

Conclusion

Data is the fundamental component of all data science and machine learning projects and research, as this chapter's content makes clear. The format in which the information is provided, the amount, quality, and quantity of missing data, and the existence of misleading information. The data that is processed by different algorithms is influenced by all of these variables and more.

Machine learning still relies heavily on feature extraction, which is essential to allowing machines to understand their environment. The development of feature extraction techniques will remain crucial to the creation of intelligent systems as long as technology progresses. The difficulty is in creating extractors that are not just practical and efficient but also morally and fairly constructed, guaranteeing that everyone can benefit equally from machine learning.

CHAPTER 2 Contribution

2.1 Introduction

This chapter will introduce the primary techniques used for this master thesis, with the goal of identifying the best CNN Model with the least amount of architectur or coming very close to it. Including the problem's genetic representation and the various algorithmic components.

In this chapter we are going to talk about the objectives of our solution and explain step by step the algorithm that we built to obtain those results.

2.1 Objectives

Being able to consistently extract the best features from a data set is the aim of this work. in an effort to construct an ideal model or, at the very least, approach an ideal outcome.

As demonstrated in the previous chapter, the remaining challenge in building models is limiting the set of features to work with. To put it another way, we must choose a good set of features, those that work better and can most effectively divide the data, and preserve them for use. Only then will we be able to create models that are either optimal or very close to optimal.

2.2 Algorithme

As we covered in the previous chapter, genetic algorithms are powerful tools that can be used to explore the search space and solve heuristic problems. additionally the advantageous effects of color thresholds and image masking.

Our goal in this work is to build a good model that will be ideal in terms of architecture and classification accuracy. This can be accomplished by extracting attributes from the data and preprocessing it in the right way.The method involves using color treshold in conjunction with channel shifting color and image masking to maximize noise reduction and clarify the best characteristics. In this section, we provide examples of how the suggested genetic algorithm approach was utilized to accomplish the goal that permits creating the best possible data representation. The suggested plan appears to adhere to the subsequent steps. First, the steps for encoding and decoding are explained in detail. Next, steps for image processing are demonstrated.

Finally, in order to process the data and build an improved model, a fitness function is defined and genetic optimization is carried out.

2.3.1 First part: Encoding

We represent the attribute values for the shifting applied mask and Canny threshold as a single chromosome. The chromosome's integer sequence length is 16, with the first 6 elements dedicated to mask creation, elements 6 to 10 for shifting, elements 11 to 15 for Canny detection values, and the last element determining how the process is applied. For example:

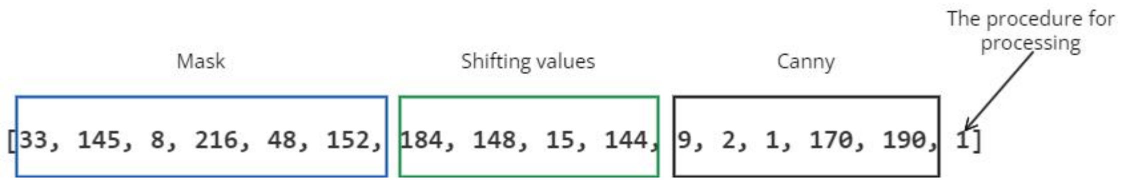
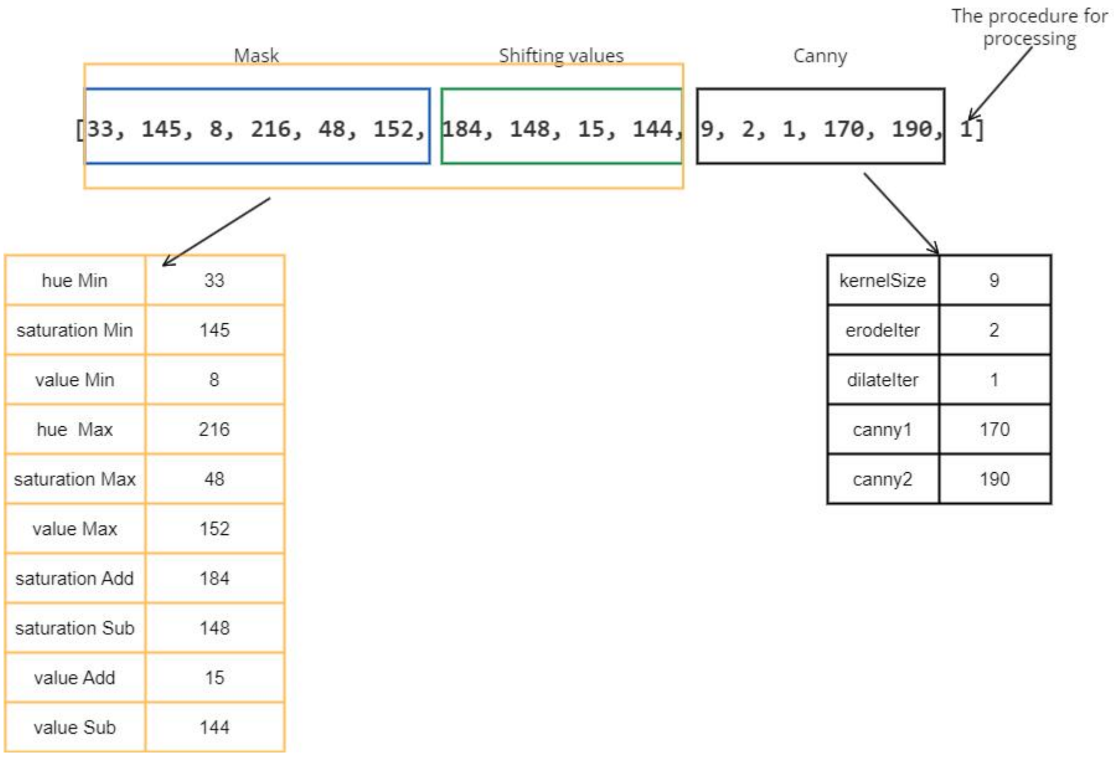


Figure 13: Chromosome representation

2.3.2 Part two: Decoding and construction processing objects

We construct two objects when we decode the chromosome: the first is called hsvFilter, and the second is the edge detection.



hue Min	33
saturation Min	145
value Min	8
hue Max	216
saturation Max	48
value Max	152
saturation Add	184
saturation Sub	148
value Add	15
value Sub	144

kernelSize	9
erodeIter	2
dilateIter	1
canny1	170
canny2	190

Figure 14: Decoding and construction processing objects

In this work, we created an image preprocessing algorithm for advanced image manipulation based on the HSV (Hue, Saturation, Value) color space representation.

HSVFilter

We did this by using a specially designed HSVFilter class.The parameters that define the minimum and maximum thresholds for every HSV channel (hMin, sMin, vMin, hMax, sMax, vMax) and the adjustment values for the saturation and value channels (sAdd, sSub, vAdd, vSub) are used to initialize the HSVFilter class.

This class's main function, shift_channel, modifies a channel's values by a given amount while making sure the final values stay inside the allowed range (0-255). The method raises channel values below a threshold when the adjustment amount is positive, while capping values that reach or surpass the threshold at 255. On the other hand, when making negative adjustments, the procedure sets values that are at or below the threshold to 0 and reduces channel values that are above a specific threshold.

$$\text{If amount } > 0$$

$$\begin{cases} \text{ci} \geq \text{lim} & \text{ci} = 255 \\ \text{ci} < \text{lim} & \text{ci} = \text{ci} + \text{amount} \end{cases}$$

$$\text{If amount } < 0$$

$$\begin{cases} \text{ci} \leq \text{lim} & \text{ci} = 255 \\ \text{ci} > \text{lim} & \text{ci} = \text{ci} + \text{amount} \end{cases}$$

The algorithm proceeds as follows for each image in the dataset:

Color Space Conversion: The input image is converted from the RGB to the HSV color space.

Channel Splitting: The HSV image is decomposed into its individual H (Hue), S (Saturation), and V (Value) channels.

Channel Adjustment: The shift_channel method is applied to the S and V channels using the specified sAdd, sSub, vAdd, and vSub values to adjust their intensity levels.

Thresholding: Two masks are created using the specified minimum and maximum thresholds for the H, S, and V channels. The lower mask is defined by hMin, sMin, and vMin, while the upper mask is defined by hMax, sMax, and vMax.

Mask Application: Utilizing OpenCV's cv.inRange function, a binary mask is generated to identify pixels within the specified HSV range. This mask is then applied to the shifted HSV image using the cv.bitwise_and function, isolating the desired pixel values.

Final Conversion: The resulting image, containing only the pixels within the specified

HSV range, is converted back from the HSV to the BGR color space, and subsequently to the RGB color space for further processing or analysis.

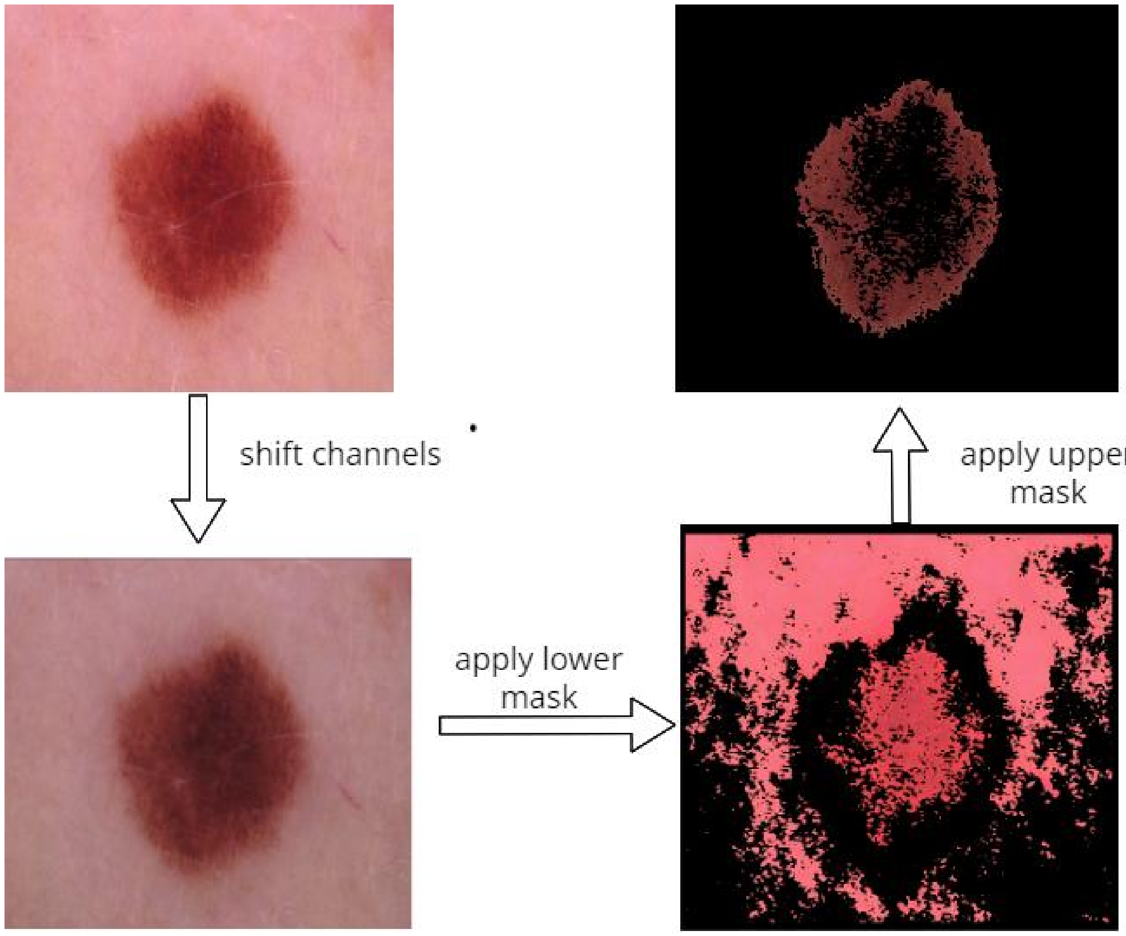


Figure 15: Channel shifting and masking treatment

EdgeFilter

We introduce a class called EdgeFilter that is designed to extract and improve edges in images by utilizing techniques derived from computational image processing. The kernel size (kernelSize), erosion iterations (erodeIter), dilation iterations (dilateIter), and parameters for Canny edge detection (canny1, canny2) are specified when the EdgeFilter class is instantiated.

Apply_edge_filter, the class's main method, uses a sequence of morphological operations to process the input image before performing Canny edge detection. The following is how the algorithm works:

Kernel Definition: A kernel of size (kernelSize x kernelSize) consisting of ones is defined. This kernel serves as the structuring element for morphological operations.

Erosion: The original image undergoes erosion using the defined kernel, with the number of iterations specified by erodeIter. Erosion reduces the size of bright regions and enhances edges.

Dilation: The eroded image is then subjected to dilation using the same kernel, with the number of iterations specified by dilateIter. Dilation expands the regions of interest and fills in gaps in the edges.

Canny Edge Detection: The resultant image from the morphological operations serves as the input for Canny edge detection. Canny edge detection is applied using parameters canny1 and canny2, which represent the lower and upper thresholds for edge detection.

Conversion: The resulting single-channel edge image is converted back to a 3-channel BGR image to maintain consistency with the original image format.

Lastly, the enhanced edge-containing processed image is sent back for additional study or display.

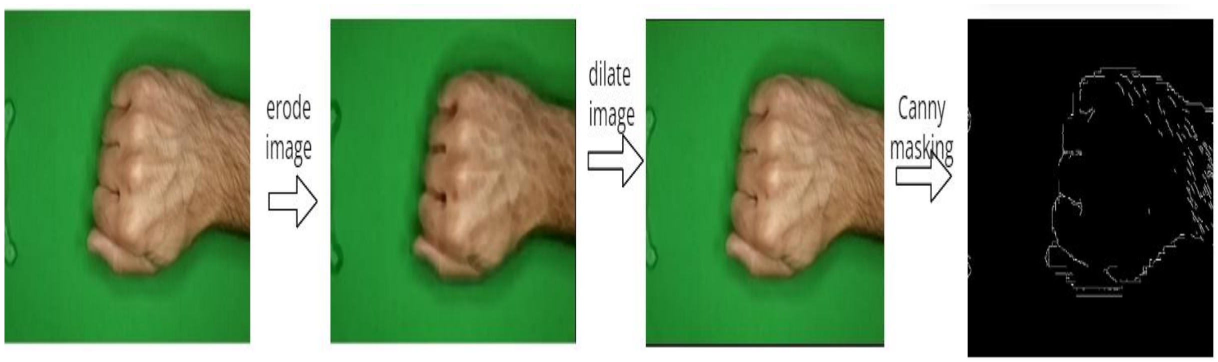


Figure 16: Edge filtering treatment

The procedure for processing

The chromosome contains a value determining the processing procedure for image datasets:

if it's 0, only HSV filtering is applied;

if it's 1, only edge detection is performed;

and if it's 2, both HSV filtering and edge detection are applied to the images.

2.3.3 Part three:Evalutions

In order to evaluate the set of images created with chromosome and see if they are the best combination, we build a Cnn classifier from this set and evaluate it using a 10 folds cross validation.

2.3.4 Part four: Exploitation

Selection Pair

The selection_pair method is used to select two genomes from the population based on their fitness. This method ensures that genomes with higher fitness values have a higher probability of being selected.

Single Point Crossover

The single point crossover method performs a crossover operation between two parent genomes, producing two offspring.

Mutation

The mutation method introduces random variations into a genome's filter with a specified probability, which is essential for maintaining genetic diversity.

$$f(x) = \begin{cases} ind \notin \{1, 3, 5\},\ new\_filter[i] = |filter[i] - R(minRange, maxRange)| \\ ind \in \{1, 3, 5\}, new\_filter[i] = R(new\_filter[i-1], maxRange) \end{cases}$$

2.3.5 Part five:

Following the completion of the genetic algorithm's (GA) search for the ideal combination, we extract the chromosome that best represents the solution and go through the previous steps again. This entails creating a fresh test dataset and applying 10-fold cross-validation to assess it. It's crucial to remember that the test set was kept apart from the dataset before to the GA starting, so it wasn't visible to the GA while it was optimizing.

The above algorithm is represented by the diagram below:

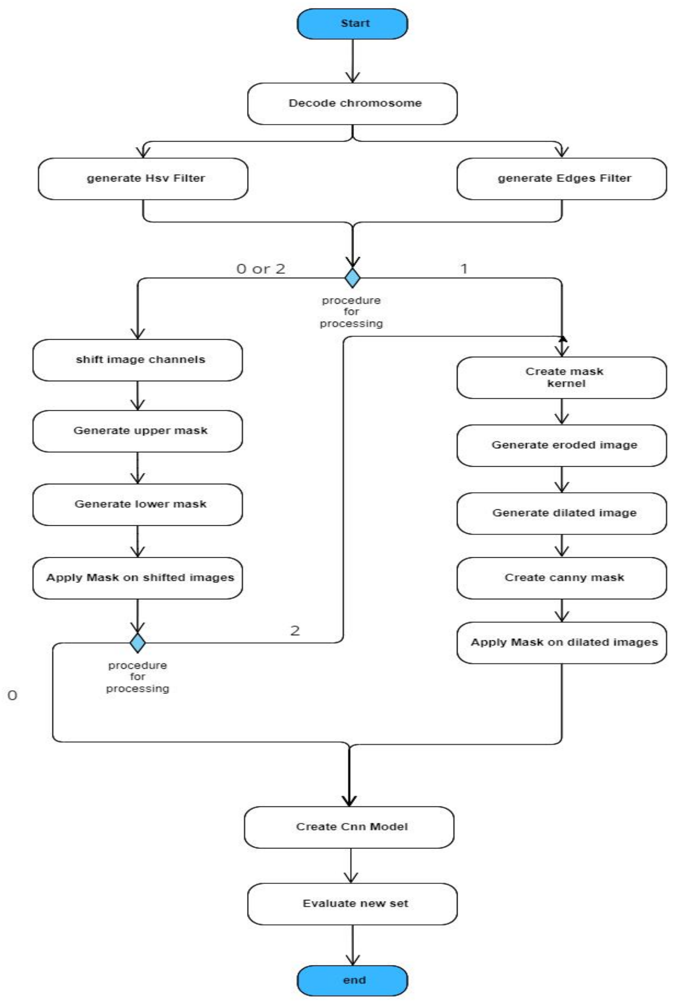


Figure 17: Flow chart of the approach algorithm

2.3 Parameters and algorithm procedures

Certainly the methodology we employed incorporated a multitude of functions and parameters that we refined through experience; primarily, the parameters of the Genetic Algorithm are those that, when altered, impact the outcomes, and there are many of those:

2.4.1 Termination criterion

A genetic algorithm must search for a long time in order to explore the entire search space and find an optimal solution, or approach one. However, we must set a termination criterion to ensure that the algorithm doesn't run indefinitely or overfit into a solution. There are many termination criteria that a genetic algorithm can be controlled with; the one we have selected is max evaluations.

2.4.2 Max evaluation

The Number of generations to be evaluated.

and the one we found fits our problem is between 3 to 20 iterations.

2.4.3 Population size

This represents the initial population size.

and we have put it equal between 20 to 24.

2.4.4 Offspring population size

This is the offspring population size.

we have set its value equel to population size.

2.4.5 Mutation rate

the possibility of a chromosome to be mutated.

its value is equal to 0.05.

Conclusion

This chapter contains a demonstration of our approach for extracting features from data sets. We have listed the various stages of our algorithm along with the parameters we used and fixed their values during the method's testing and experience.

CHAPTER 3 Experimentation

3.1 Introduction

In this chapter, we will discuss our tests and experiments, as well as the data sets we used and the assessment methodologies. Alongside the different metrics we used to evaluate our model. Second, we will look at the results acquired and quickly discuss them.

3.2 Data sets

In this work we have used 12 datasets collected from Kaggle archives, for the experimental part of our work, in order to increase the variety of the data types, and the classification complexity.

Kaggle however is a multipurpose web platform, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges. [14]

the table below summarizes the characteristics of the chosen data sets:

-N° Classes: number of classes

-Number° Instances: number of instances

index	**data sets**	**class**	**samples**
1	Bean Leaf Lesions	3	1167
2	Chicken disease	4	6508
3	Covid19	3	272

4	Dog or cat	2	697
5	dog-breeds	8	541
6	flowers	5	4317
7	hand-sign	10	1431
8	Lemon	2	2076
9	Lung X-Ray Image	3	3475
10	Monkeypox Skin Lesion	2	228
11	R-P-S	3	2188
12	SkinCancer	2	3297

Table 1: Datasets description

3.3 Materials

3.3.1 Hardware

In order to put this strategy into practice, the algorithm's coding and model construction were implemented on a local desktop using the following specifications:

- Memory: 16GB DDR4 2600 MHz
- Processor: 8th Gen Intel® Core™ i5-8600K
- Graphics: Nvidia Gtx 1050 ti 4Gb VRam

3.3.2 Software

We will walk through the software configuration in this section, which enabled us to implement and build up the entire framework. We'll be discussing the configuration of the local machine. The operating system on the local PC is Windows 10 Pro.

3.3.2.1 Coding language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.[16]

3.3.2.2 Masking construction package

OpenCV short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.[17]

3.3.2.3 Data visualization packages

Matplotlib is an open-source Python library originally developed by neurobiologist John Hunter in 2002. Its initial purpose was to visualize the brain signals of epileptic individuals. To achieve this, Hunter aimed to replicate the graphic creation capabilities of MATLAB using Python.[22]

3.3.2.4 Machine learning packages

Scikit-learn is a key library for the Python programming language that is typically used in machine learning projects. Scikit-learn is focused on machine learning tools including mathematical, statistical and general purpose algorithms that form the basis for many machine learning technologies. As a free tool, Scikit-learn is tremendously important in many different types of algorithm development for machine learning and related technologies.[18]

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.[19]

TensorFlow is an open-sourced package designed for applications involving deep learning. Additionally, it supports conventional machine learning. TensorFlow was initially created without considering deep learning for large numerical calculations. However, it has also proven valuable for deep learning development, so Google made it available to the public.[20]

3.4 Evaluation

To evaluate the efficacy of a genetic algorithm (GA) designed to identify the optimal filter for a dataset, we implemented a systematic approach incorporating both training and testing phases, coupled with cross-validation (CV). The evaluation methodology is delineated as follows:

Data Splitting: Initially, the dataset is partitioned into training and testing subsets. This ensures that the model's performance can be independently assessed on unseen data, which is critical for evaluating its generalizability and robustness.

Filter Optimization via Genetic Algorithm: The GA is applied exclusively to the training subset. The goal here is to evolve and identify the most effective filter configuration for the data. The GA iteratively refines the filter parameters by simulating natural selection processes, such as mutation, crossover, and selection, to converge on an optimal solution.

Application of Optimal Filter: Once the GA determines the best filter, this optimal configuration is applied to the test subset. This step is crucial as it allows us to evaluate how well the filter generalizes to new, unseen data, ensuring that the filter's efficacy is not limited to the training data alone.

Evaluation with Cross-Validation: The filtered test subset is subjected to a 10-fold cross-validation (10-CV). This involves partitioning the test data into ten equal parts, where each part is iteratively used as a validation set while the remaining nine parts constitute the training set. This process is repeated ten times to ensure that every data point in the test set is used for validation exactly once. The results from each fold are averaged to provide a robust estimate of the filter's performance.

Comparative Analysis: The performance of the GA-optimized filter is benchmarked against several predefined filters, as well as Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG). These comparisons provide a context for assessing the relative effectiveness of the GA-optimized filter. Metrics such as accuracys are employed to quantify performance differences, thereby offering a comprehensive evaluation.

This multi-step evaluation framework ensures that the GA-optimized filter is rigorously tested and compared, providing insights into its potential advantages over traditional filtering techniques. By incorporating cross-validation and benchmarking against established methods, we can confidently ascertain the value of our GA-based approach in enhancing data filtering and feature extraction processes.

Model:

```
Model: "sequential"
________________________________________________________________
 Layer (type)                Output Shape              Param #
================================================================
 batch_normalization_2 (Batc  (None, 224, 224, 3)       12
 hNormalization)

 conv2d_4 (Conv2D)            (None, 222, 222, 16)      448

 max_pooling2d_4 (MaxPooling  (None, 111, 111, 16)      0
 2D)

 conv2d_5 (Conv2D)            (None, 109, 109, 32)      4640

 max_pooling2d_5 (MaxPooling  (None, 54, 54, 32)        0
 2D)

 dense_2 (Dense)              (None, 54, 54, 64)        2112

 flatten_2 (Flatten)          (None, 186624)            0

 dense_3 (Dense)              (None, Number of class)             933125

================================================================
Total params: 940,337
Trainable params: 940,331
Non-trainable params: 6
________________________________________________________________
```

3.5 Results

The different Results obtained by this work could be listed briefly in the table below:

			Filters					Feature extractors		
index	data sets	original data	gaussian blur	Median Blur	Laplacian Filter	unsharp mask	Bilateral Filter	hog	LBP	approach
1	Bean Leaf Lesions	33.9	39.9	42.29	31.64	36.48	40.42	37.61	34.12	**49.6**
2	Chicken disease	48.73	53.44	47.46	43.32	53.69	51.82	47.01	53.41	**72.7**
3	Covid19	43.61	46.25	47.5	45.27	44.02	35.27	36.66	40	**78.05**

4	Dog or cat	49.04	47.14	50.95	51.42	49.52	45.23	48.57	50.95	**55.71**
5	dog-breeds	12.75	17.68	12.9	18.45	17.24	16.61	15.90	10.2	**43.12**
6	flowers	34.64	39.82	**43.68**	35.65	41.12	41.14	34.41	26.9	34.01
7	hand-sign	11.39	14.65	12.79	74.18	16.04	14.41	79.30	15.58	**92.79**
8	Lemon	62.92	62.47	61.79	77.02	59.17	57.61	82.69	50.72	**89.56**
9	Lung X-Ray Image	46.69	56.76	54.2	73.23	55.99	50.74	73.61	41.97	**76.68**
10	Monkeypox Skin Lesion	**60.95**	53.57	56.42	53.75	56.42	57.85	45.00	52.14	53.81
11	R-P-S	38.47	40.65	43.37	48.84	37.28	36.51	76.25	39.42	**76.41**
12	SkinCancer	63.23	60.70	59.09	62.12	53.13	57.27	62.22	52.52	**74.04**

Table 2: Obtained results summary

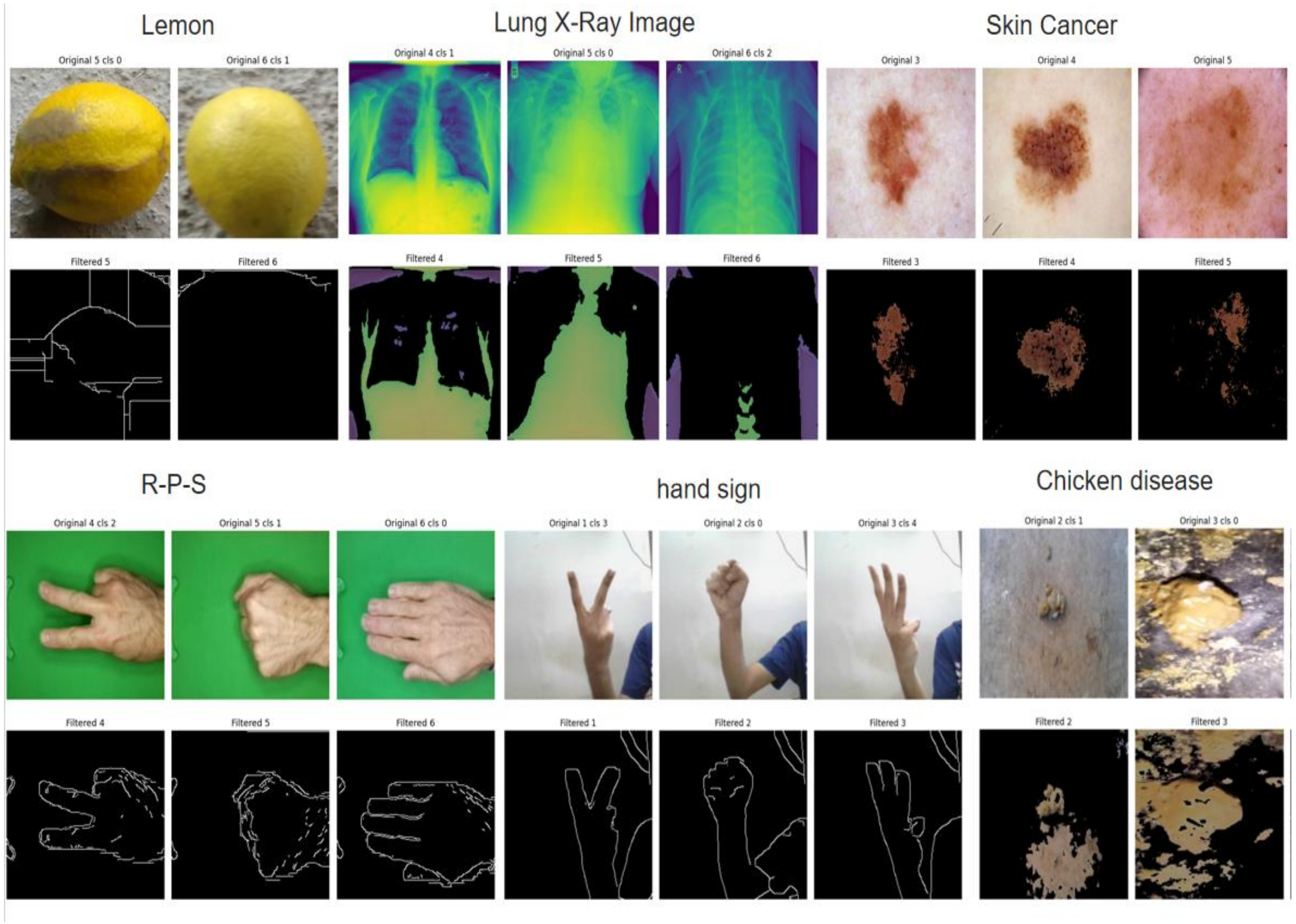


Figure 19 Some data sets before and after

3.6 Discussion

It is clear from the results shown in the above table that our genetic algorithm (GA)-optimized filter method performed better on most of the datasets that were tested. To be more precise, our approach worked in 10 of the 12 datasets, with the least successful performance still being over 83%. This demonstrates the stability and effectiveness of our method on a variety of datasets.

Strong generalizability of the GA-optimized filter is demonstrated by its high success rate across multiple datasets. This implies that rather than being overfit to particular datasets, the GA is good at finding filter configurations that are widely applicable. This flexibility is a big benefit, particularly for real-world applications where data characteristics can change drastically.

Our GA-optimized filters generally outperformed predefined filters and well-known feature extraction methods like Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG). This comparative advantage shows how genetic algorithms can potentially find more efficient filtering strategies that conventional methods might overlook.

Even though the GA-optimized filter performed significantly better than the other approaches, it fell short in two datasets. This implies that more specialized or refined methods might be needed for some data types. It is recommended that future research concentrate on examining these particular cases in order to comprehend the shortcomings of the current GA implementation and create focused enhancements.

Conclusion

In conclusion, our GA-based filter optimization approach outperforms well-established methods like LBP and HOG and achieves high success rates across most tested datasets. The overall results confirm that genetic algorithms have the potential to be effective tools for feature extraction and data filtering, even though there is still room for improvement and investigation. Future research should aim to enhance the efficiency and adaptability of this approach, ensuring it remains effective as data landscapes continue to evolve.

GENERAL CONCLUSION

Every machine learning and artificial intelligence project that succeeds needs well-represented and organized data. Some researchers even contend that in order to increase the caliber of machine learning outcomes, we should put less effort into creating new frameworks and architectures and more into creating methods and algorithms that work with the data itself and reveal its hidden characteristics. More precisely, they contend that artificial intelligence frameworks and architectures play a smaller role than the data and everything else related to it, including its form, internal structure, and feature relationships.

Although the aforementioned claim cannot yet be verified, a number of studies focused on data preprocessing, transfer learning, representation learning, and feature extraction have been published recently. As a result, the research community's interests have shifted, with a greater emphasis now being placed on data manipulation and restructuring. Later, the researcher's interest was realized in a number of works where developers began to apply machine learning models in previously unheard-of ways.

As a conclusion to this master's thesis, we started by introducing data filtering and well-known feature extraction techniques, like Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG), in order to solve our problem of optimizing data . Next, we presented method. By using the GA's evolutionary processes to choose the optimal filter configurations, our approach aims to improve data preprocessing. To increase classification performance, the GA iteratively looked for the best set of filter parameters.we tested our approach on 12 datasets and compared the results with predefined filters as well as LBP and HOG techniques. Our GA-optimized filters demonstrated superior performance in 10 out of the 12 datasets.

Overall, our research shows how feature extraction and data filtering can be optimized with genetic algorithms, offering a useful tool for enhancing data analysis outcomes even with simple model architecture.

Reference

[1] .M. J. Swain; D. H. Ballard; Indexing via Color Histograms; University of Rochester; Rochester

[2] .Matevž Kunaver; Jurij Tasic;Image feature extraction - an overview; University of Ljubljana

[3].https://domino.ai/data-science-dictionary/feature-extraction

[4] Edgar Chavolla, Daniel Zaldivar, Erik Cuevas & Marco A. Perez Color Spaces Advantages and Disadvantages in Image Color Clustering Segmentation

[5] GF Shidik, FN Adnan, C Supriyanto, RA Pramunendar, PN AndonoMulti color feature, background subtraction and time frame selection for fire detection

[6] Riyadh M. Al-saleem Baraa M. Al-Hilali and Izz K. Abboud Mathematical Representation of Color Spaces and Its Role in Communication Systems

[7] Linda Tucci What is machine learning and how does it work? In-depth guide

[8]: https://en.wikipedia.org/wiki/Feature_learning

[9]https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

[10] https://aihalapathirana.medium.com/understanding-the-local-binary-pattern-lbp-a-powerful-method-for-texture-analysis-in-computer-4fb55b3ed8b8

[11] COVID-19 Detection from Chest X-ray Images Using Feature Fusion and Deep Learning Md. Abdul Based and all (2021)

[12] A Susanto, Z H Dewantoro, C A Sari3, D R I M Setiadi, E H Rachmawanto Shallot Quality Classification using HSV Color Models and Size Identification based on Naive Bayes Classifier (2020)

[13] Naman Karanth, Arya Adesh , Chandan Kasamsetty, Abhinav Samaga, G. Shobha, Minal Moharir Automatic Classification and Color Changing of Saree Components Using Deep Learning Techniques(2024)

[14] https://en.wikipedia.org/wiki/Kaggle

[15] https://towardsdatascience.com/cross-validation-430d9a5fee22

[16] https://www.python.org/doc/essays/blurb/

[17] https://www.geeksforgeeks.org/opencv-overview/

[18] https://www.geeksforgeeks.org/opencv-overview/

[19] https://numpy.org/doc/stable/user/whatisnumpy.html

[20] https://www.spiceworks.com/tech/devops/articles/what-is-tensorflow/

[21] Fajrul Islami. Implementation of HSV- based Thresholding Method for Iris Detection

[22] https://datascientest.com/en/matplotlib-master-data-visualization-in-python#:~:text=Matplotlib%20is%20an%20open%2Dsource%20Python%20library%20originally%20developed%20by,capabilities%20of%20MATLAB%20using%20Python.

[23] https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3