A Comparative Study of Content Based Image Retrieval

Candidate Number:

Department of Informatics & Engineering
University of Sussex
Supervised by
April 2023

Statement of Originality

This report is submitted as part requirement for the degree of Computer Science with A.I. BSc at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years.

Abstract

This paper explores the topic of Content Based Image Retrieval (CBIR). More specifically it explores a range of techniques used to retrieve images based on their visual content. 3 CBIR models are built with different approaches to cover a variety of methods used within the field, they range from early computer vision approaches to modern pre-trained neural networks to see an increase in performance based on the strengths/ weaknesses of the previous models. Hyperparamter tuning and optimisation will be take place to further find performance and the effects of changing parameters within the system will be evaluated to gain an understanding of the workings of the models.

Contents

1 Introduction

1.1 Project Overview

The aim of this project is to develop a Content Based Image Retrieval (CBIR) system which can be used as a platform to adjust parameters and assess performance. This report presents an iterative approach to building the system, starting with low-level mathematics based models, then moving onto to more complex approaches using machine learning and neural networks. The aim of this approach is to cover the topic of CBIR as a whole to gain a wide understanding of its development starting with early techniques such as colour histograms then moving to using modern approaches such as Convolutional Neural Networks (CNNs) to extract image features.

1.2 Motivation

Content Based Image Retrieval (CBIR) involves the extraction of features from raw images and calculation of an associative measure of similarity between a query image and database images based on those features or classifying images into classes. These features include colour, texture and shape. This project involves the development of a CBIR system that explores the benefits and drawbacks associated with different approaches, traditional methods which utilise a more mathematical approach with use of colour histograms that represent the colour distribution in an image, Gabor filters and Haralick textures that describe the texture of an image, and then move onto a more complex, modern approaches with use of a pre-trained convolutional neural network for high - level feature extraction. Low- level feature extraction methods fall short on bridging the semantic gap that exists between low- level image pixels captured by machines and high- level semantic concepts perceived by humans [7]. To fill this gap the use of machine learning for image classification and CNNs for image feature extraction is explored. CBIR systems can be used in a variety of applications. For example, they can be used to find a particular object or scene such as a yellow flower or a beach scene.

1.3 Project Aims

The main aim of this project is to research and develop a CBIR system using different models to assess performance. I will focus on a machine learning approach for reasons outlined in **section 3**.

1.4 Core Goals

- Implement a CBIR system which can perform accurate image retrieval for a query image on an image database using traditional feature extraction techniques create a similarity measure.
- Implement a CBIR system which can perform accurate image retrieval for a query image on an image database using machine learning techniques.
- Implement a CBIR system which can perform feature extraction using a CNN.
- Produce a graphical user interface that allows for image storage, analysis and retrieval.

1.5 Stretch Goals

- Determine which image features provide the best classification or categorisation.
- Determine which techniques to extract image features deem most effective.
- Determine which techniques of classifying image classes deem most effective.
- Experiment with parameter optimisation via machine learning.
- Provide more complex interactivity between the user and the model(s).
- Implement a text based image retrieval system.

1.6 Project Planning

Planning for this project is done through the software 'Notion' [18], the software is used primarily for scheduling and checklists of progress. It is also cloud based so can be accessed on all devices. The planning methods for the project include a GANTT chart and Kanban board which are linked together so changes on one are reflected on the other, this is supplemented with a timetable made in Excel. Note taking for research and general project notes are taken on OneNote which again is cloud based. Figure 1 shows the GANTT chart made for this project so enough time was allocated to ensure smooth development of the project, most of the main development stage was left open as the project was mainly research-based.

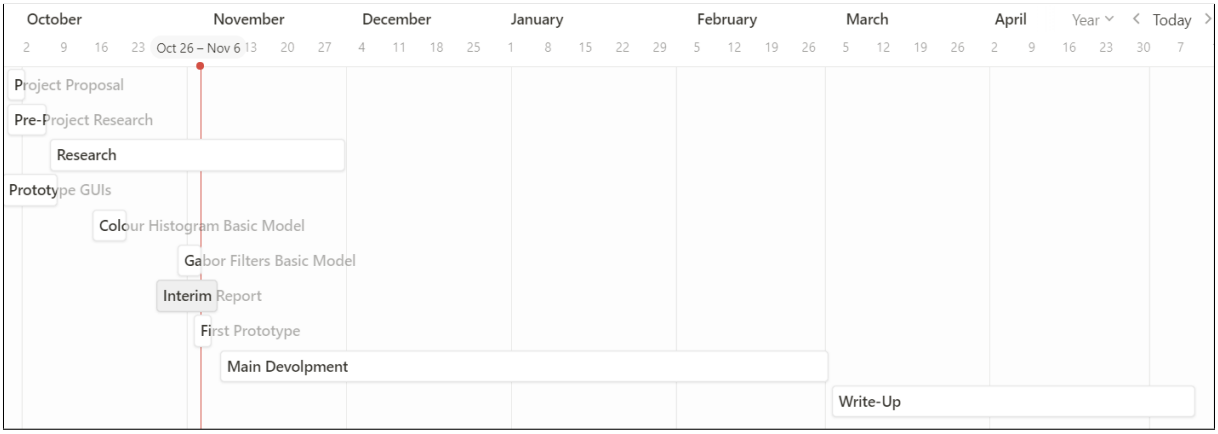


Figure 1: A GANTT chart built for this project.

1.7 Project Relevance

Development of the internet, and availability of image capturing devices have caused the size of digital image collections to increase rapidly [14]. Content based image systems solve this by indexing images by their visual content to retrieve images. CBIR will only become more relevant as these collections increase in size. As will be explored in later sections, there is limitless approaches involving mathematical techniques as well as machine learning approaches, which all lend well with computer science and artificial intelligence.

2 Professional Considerations

In this section I will discuss how the project relates to each of the four key principles of the BCS code of conduct. It should be stated that there are minimal ethical considerations for this project as it is mainly focused on research and development however in later stages of development I will conduct user testing and review.

2.1 Public Interest

Privacy, security and well-being of the public is unaffected by this project. The final application will only store media locally and the media stored will already be on the users device, it will also not pose a security risk. Third party code, software or ideas will be credited and licensed appropriately. The application developed from this project will be accessible to all individuals regardless of their background or any protected characteristics.

2.2 Professional Competence and Integrity

I am undertaking this project with it being completable in mind, I know this because my development process is scalable and with many iterations of different models to find a solution to the problem. Having a basic working prototype demonstrates this. I have and will continue to expand this because of the broad natural of CBIR as a field. Illegality or injury is of no concern for this project due to it being entirely virtual with no physical or social effects.

2.3 Duty to Relevant Authority

This project will be undertaken with due care and diligence in accordance with the University requirements, developing it as far as possible and creating a comprehensive application. I will seek to avoid conflicts of interest and will accept responsibility for any that arise. I will not disclose any confidential information without the permission of the University or as required by legislation. I will not misrepresent the functionality and performance of my project or take advantage of lack of relevant knowledge or experience of others.

2.4 Duty to the Profession

In undertaking this project I accept my personal duty to uphold the reputation of the BCS and the profession, acting with integrity in all professional relationships. I aim to encourage and support fellow members with their professional development wherever possible, without collusion. I will seek to improve my professional standards by aiding in their development.

3 CBIR Overview - Research & Related Works

This section will outline existing solutions to the CBIR problem and provide context to an overview of the field.

3.1 Feature Extraction Methods

Low-Level Image features extraction is the basis of CBIR systems. Image features can be extracted either globally for an entire image, or from image regions as it has been found that human perceptions tends to focus on specific regions rather than the entire image [14]. Low-level image features provide a method to represent the visual content of an image that can be used to make comparisons to another to determine how similar they are. Feature can include colour, texture, shape and spatial location. The following sections will outline a overview on these features types and their role in a practical CBIR system.

3.1.1 Colour Features

It is well understood in the field of CBIR that different feature extraction methods have differing importance. Colour features are considered as one of the most important low-level visual features as the human eye can differentiate between visuals on the basis of colour [12]. Colour features are against changes due to image translation, scale and rotation cause. Colours are defined over a selected colour space which all serve for different applications, for the CBIR, a colour space that is close to human perception is necessary to decrease the semantic gap between low - level features and high - level concepts. The Hue/ Saturation/ Value (HSV) colour model is a non - linear transformation of the RGB colour space and based on intuitive colour characteristics as tint, shade and tone. The coordinate system is cylindrical and colours are defined inside a hexcone [5], this can be visualised in figure 2.

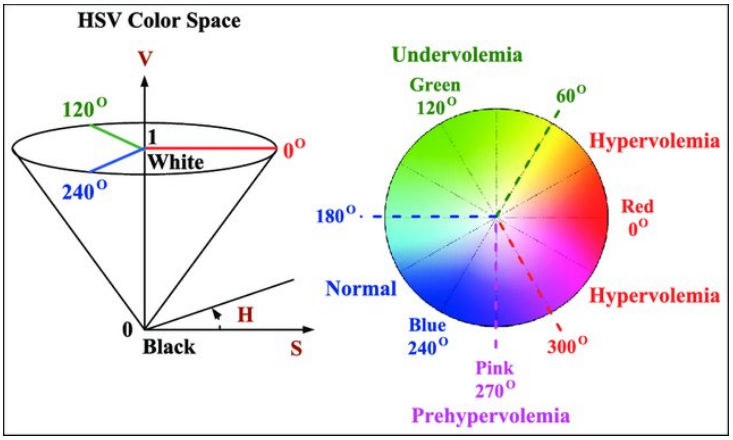


Figure 2: HSV color space and RGB color transformation.

HSV is often used in CBIR as it provides a more intuitive representation of colour over other colour spaces. It is robust to lighting changes over a colour space such as RGB as the Value (V) component

captures the brightness of a colour. An example of this is a search for an image with red flowers, the HSV colour space will select images with a red hue regardless of saturation or value. This then makes it easier to compare colour combinations and similarities, therefore is very important for maximising relevant information when extracting colour features in an image and will be used in all mathematics based models created for this project.

3.1.2 Texture Features

Texture is an important feature in CBIR as it can be used to provide information for classification because it describes the content of the image such as skin, clouds, trees and fabric. Texture can be used to add context to an image and define high - level semantics for image retrieval. They are also applicable to a wide range of images type such as natural scenes, buildings and objects. There are however some drawbacks to consider, Texture feature extraction can be computationally intensive to process. They are also sensitive to noise in the image, this can lead to false matches and inaccurate retrievals so it is important to provide image context with other features too. Texture features commonly used in CBIR include wavelet features such as Gabor filtering where features are based on the decomposition of an image into a set of wavelet coefficients to represent to frequency content of an image.

3.1.3 Shape Features

Shape features are important image features though not as widely used as colour and texture features and have shown to be useful in many domain specific images such as man-made objects [14]. Possible shape features include moment invariants, Fourier descriptors, consecutive boundary segments, aspect ratio and more [17]. They can be more robust to changes in lighting than colour and texture and are used to classify objects in an image. Shape features can be extracted from an image in various ways but a common approach is identifying edges in an image using an approach such as using edge orientation histograms (HoG). This approach first detects the edges in an image which in its most simple form considers edges as locations where there is a maximal change in pixel intensity. Edges are then classified into one of a set of orientations, this is done by calculating the gradient of an image at an edge point and assigning the edge to the orientation of the gradient. Edge orientations can be used to create a histogram by counting the number of edges in each orientation, this histogram can be used to represent the shape of an object in an image.

3.2 Related Works

3.2.1 Content-based image retrieval using colour histogram

As mentioned in the previous sections, colour is an important feature in CBIR. One method commonly used for colour feature extraction is colour histograms. Young-jun Song's paper [24] proposes a new method of colour representation and overcomes some disadvantages of colour histograms. Existing methods use only the frequency values of the same colour after colour quantisation which can result in quantisation errors. This disadvantage is further backed up in Ying Lius's survey of CBIR [14] where a dominant colour of an image is calculated using HSV (hue, saturation, value) histograms of regions of the image, the bin with the maximum size is selected and the average HSV value of all pixels is defined as the dominant colour for that region. In some examples the dominate colour can appear very different than the average colour of an image, as shown in figure 8.

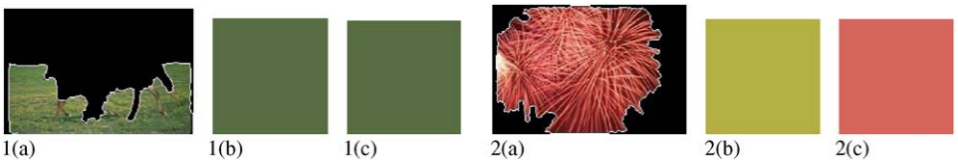


Figure 3: Average colour and dominant colour: (a) original region; (b) average colour; (c) dominant colour.

Young-jun Song proposes a new method. First of all, it divided HSV colour space into 16 regions for retrieving the colour features of image; then retrieved representative colour and frequency in each region as features. Representative colour in each region had the R, G, B mean value there. The queried image and the target image are then measured for similarity using a fuzzy colour function that takes advantage of colour R, G, B value and frequency [24].

3.2.2 CBIR based on CNN and SVM

Ruigang Fu and others in their paper [7] explain how machine learning can be used to bridge the semantic gap between low-level image pixels captured by machines and high-level semantic concepts perceived by humans. The method proposed generates deep features based on a pre-trained CNN and then uses distance metric learning (DML) to learn a similarly measure between the query image and the database images in the form of a SVM (Support vector machine). The outline of this system is showed in figure 4.

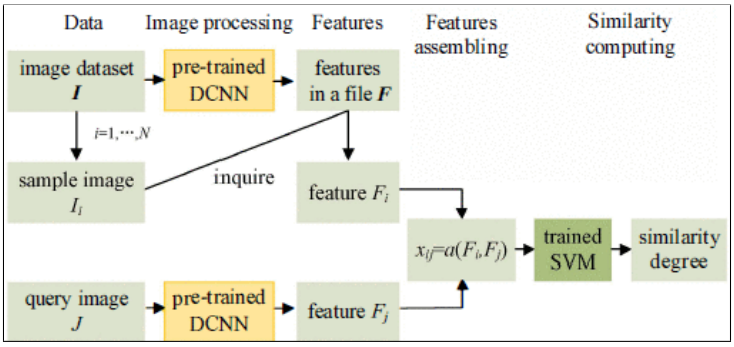


Figure 4: Ruigang Fu and others [7]'s proposed CBIR system

From their evaluation they found their system performed better than classical methods such as bag-of-words and also shows some interesting insight with Principle Component Analysis (PCA) to reduce dimensionality where compression to 512 dimensions shows no loss of performance. They however found that data-sets such as ImageNet used during training leads to poor training and fails to bridge the semantic gap for retrieval of images such as similar landmarks. For this reason models in this projects are trained on a varied dataset of image classes.

The model used used in this evaluation utilises a SVM to classify an image based on two classes, similarity and dissimilarity in respect to the query image and a given database image. A SVM works by mapping the input feature vector into a high dimensional space with kernel functions, and generating an optimal hyperplane which makes margin largest [8]. Figure 5 shows this margin in a linear and 2 dimensional conditions.

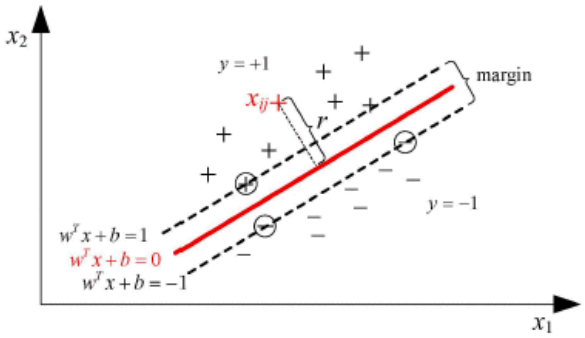


Figure 5: Optimal hyperplane and margin of a SVM under 2-dimensional and linear conditions [8].

Also in this evalutation, they found a linear SVM provided better performance over a RBF SVM, they don't explain why this is in the paper but from research [27] the decision boundary for this problem is mainly linearly separable, either the query image is similar to the database image or it is not. A linear SVM attempts to find a linear line which maximises the margin to the closest point to it therefore is more adapted for this problem. Figure 6 demonstrates this where a decision boundary for the data-points that are linearly separable is more accurately placed using linear SVM, and non-linearly separable points a non-linear or RBF SVM. A linear SVM also has the benefit of faster training and testing times, which is an important takeaway if experimenting with this approach of distance metric learning.

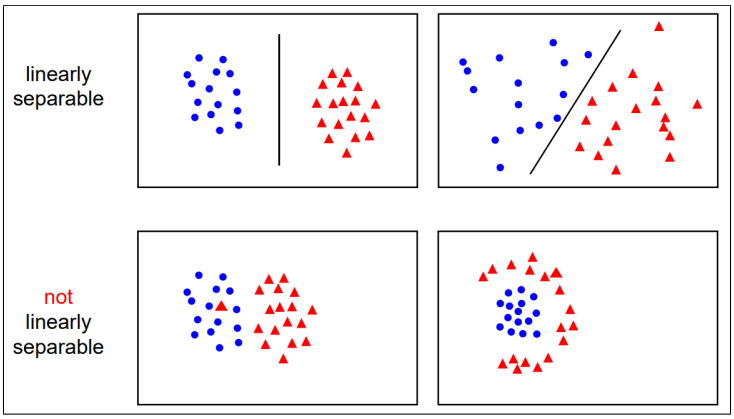


Figure 6: Linear and non-linear separability for a decision boundary [27]

3.2.3 Relevance Feedback (RF)

RF model intend to learn the users intentions in real time of processing, it is used to put the user in a retrieval loop to reduce the 'semantic gap' between feature extraction (low-level feature) and what the user wishes to retrieve (high-level semantic concepts) by continuous learning through interaction with the end user. User decisions can provide feedback to the system on relevance of images retrieved in response to a query and used to update weights in the system, such as similarly metrics and model parameters to improve the accuracy of retrieval results.

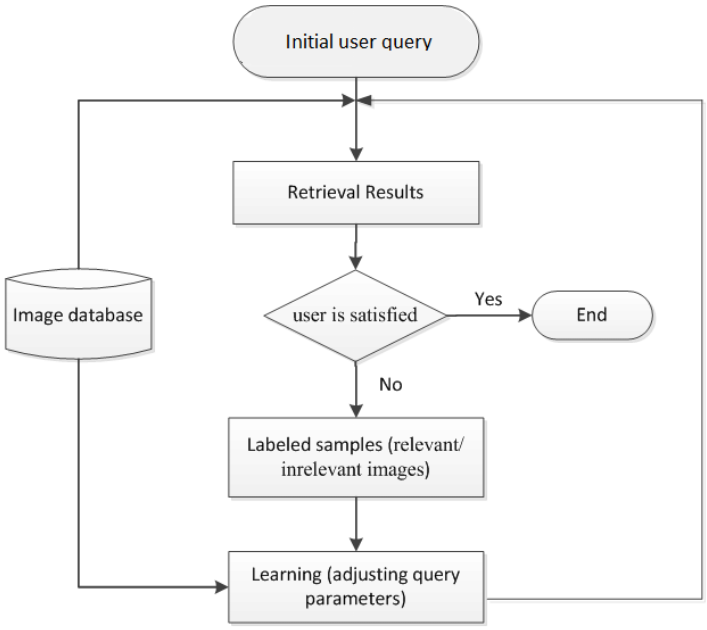


Figure 7: A CBIR system with Relevance Feedback [9]

Figure 7 shows a simple diagram of a CBIR with RF. A typical approach, shown as the learning step, involves adjusting the weights of low-level features to accommodate for user decisions. From a general machine learning view, RF essentially is a binary classification problem, where either an image is relevant to the query image or it is not. Because of the limited feedback from the user in real world application, RF systems are limited to a small sample learning methods. SVMs are popular for this application which has good performance for pattern classification patterns, fast learning and flexibility. [9]'s method utilises a batch mode for SVM active possible similar images are filtered from the database via a decision boundary learnt via SVM. A ranking function then orders the images of likelihood of being the best candidates using both the scores of SVM and similarity measure between query image and the database. This reduces the effect of an inaccurate decision boundary. Using a priority coefficient in the ranking function creates a batch of feedback examples which may be informative enough to improve the retrieval results. This model proved more effective over traditional schemes especially when the number

of initially labelled samples is small. It is important to note that an RF approach is time-consuming and difficult for a user to accurately access the relevance of images. This could lead to ambiguity and user bias, for these reasons this project will not pursue a RF approach in implementation.

3.3 Colour Feature Extraction

Color method	Pros.	Cons.
Histogram	Simple to compute, intuitive	High dimension, no spatial info, sensitive to noise
CM	Compact, robust	Not enough to describe all colors, no spatial info
CCV	Spatial info	High dimension, high computation cost
Correlogram	Spatial info	Very high computation cost, sensitive to noise, rotation and scale
DCD	Compact, robust, perceptual meaning	Need post-processing for spatial info
CSD	Spatial info	Sensitive to noise, rotation and scale
SCD	Compact on need, scalability	No spatial info, less accurate if compact

Figure 8: [20]

One method commonly used for colour feature extraction is Colour Histograms. The image is represented using a frequency distribution of colour bins; it counts similar pixels and stores it therefore acting as a global colour descriptor [25]. Colour histograms offer advantages such as low computational cost and insensitive to small variations in images (such as rotation). HSV is often used for colour feature extraction because the colour information is separated into three distinct components which are more intuitive with human perception[26].

4 Implementation

The programming language used to carry out this project is Python programming using the Visual Studio Code environment. Python provides provides many pre-existing relevant libraries to utilise within the project.

4.1 Python Libraries

- **Scikit-learn** [19] is used extensively throughout this project. It features various classification algorithms, pre-proccessing techniques and model selection methods to carryout a wide variety of machine learning tasks. This library is mainly used after feature extraction to classify images into classes and assess performance.

- **Keras** [3] is a high-level neural networks API running on top of TensorFlow which is used in Model 3 to carry out accurate feature exaction through the VGG16 model.

- **PySimpleGUI** [21] is a Python package for building GUIs - it is used to build the interface for the system.

- **OpenCV** [2] is a Python package used for real time computer vision, in this project it is used for preprocessing images and image operations.

4.2 Dataset Review

For this project, the Caltech101 [6] dataset will be used for a majority of testing on the models. The Caltech101 dataset contains 101 object categories, each with around 100 images which are of high quality and well labelled making it a valuable resource for training and evaluating CBIR systems. It is relatively small therefore making it quick and low computational costs for evaluating models which will be a large part of this project, it is also well - studied meaning research for optimal solutions are available for comparisons. It however is large enough to be representative of diverse image categories that CBIR need to be able to handle, A subset of the Caltech101 dataset can be seen in figure 9.

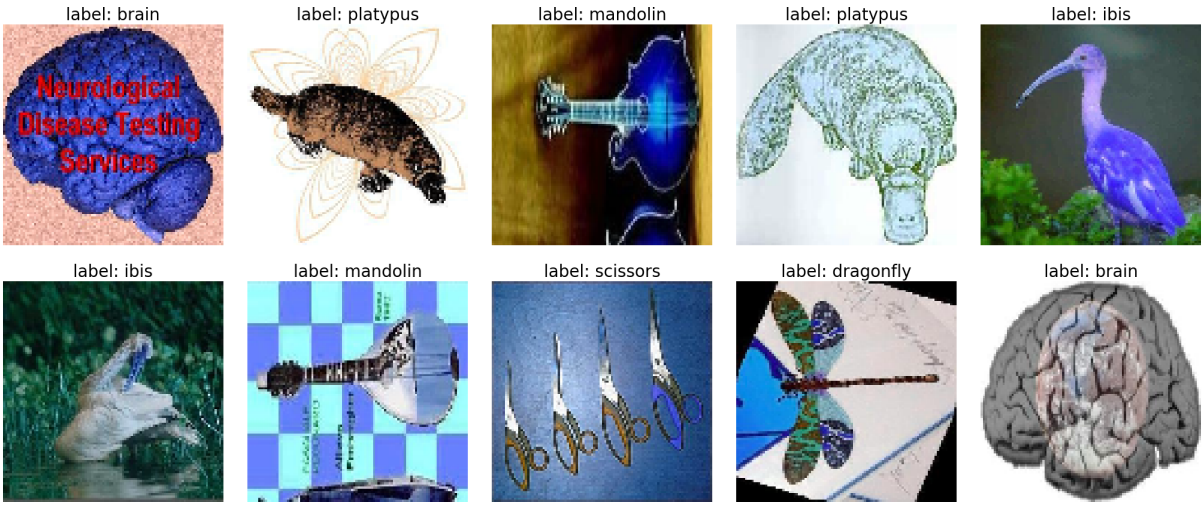
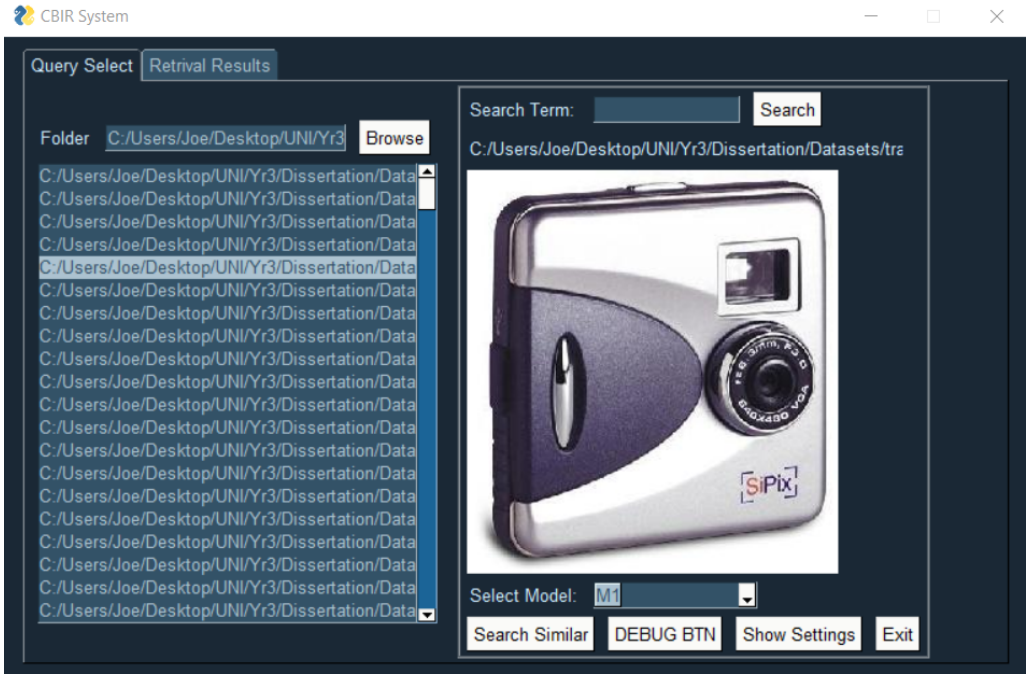


Figure 9: 10 random images from a subset of the Caltech101 dataset with corresponding labels.

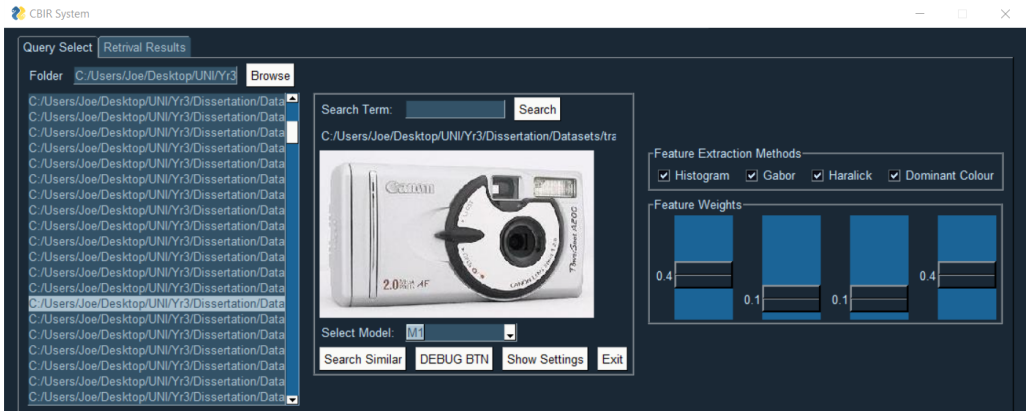
Figure 10: 10 images labelled 'buddha' in the caltech101 dataset

4.3 GUI

A GUI or graphical user interface was developed for this project as a visual way to interact with the system in a simple and intuitive manner. As this is a research based project, it was not a key priority and therefore is only functional on Model 1. Figure 11 shows the simple GUI made for model 1.



(a) GUI Default Display.



(b) GUI with model1 parameter settings displayed.

Figure 11: GUI display layouts.

5 Methods

5.1 Model 1: Low-Level Feature Extraction

The first model created aims to optimise a solution for the CBIR problem using traditional mathematics techniques to form low- level feature extraction methods. The model works by extracting features from image pixels to form colour, texture and shape feature vectors, which can be used to calculate a distance metric between a query image and a database image.

5.1.1 HSV Colour Histogram Features

Before performing colour feature extraction, an image is converted from RGB to HSV for reasons stated in section **3.1**. The global colour histogram is calculated as follows:
Step 1: Convert images from RGB space to HSV space.
Step 2: Create a histogram for each of the 3 components of HSV using cv2.calcHist [2].
Step 3: Flatten histogram to single feature vector.

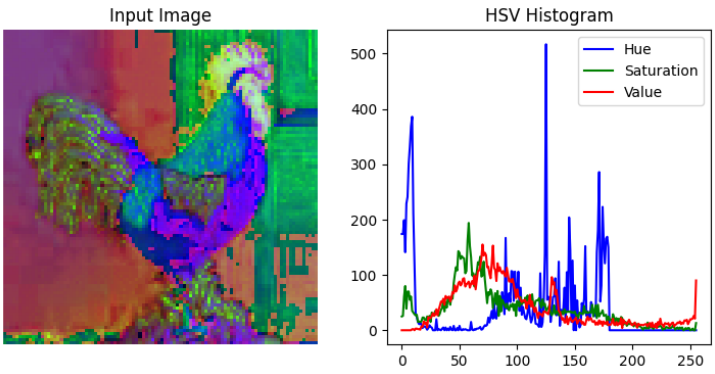


Figure 12: HSV histogram produced from input image using cv2.calcHist [2]

Figure 12 shows a visualisation of creating a histogram through the method stated for a given input image which has been converted to HSV colour space. The method of cv2.calcHist divides the colour space into a number of bins and counts the number of pixels in each bin, in this implementation 256 bins are used to create a more accurate feature vector at the cost of slower computation. This global colour histogram technique allows for efficient computation invariant to rotation and translation, however spatial distribution of colours is lost.

5.1.2 Dominant Colour Features

Dominant colour feature extraction is used to provide further colour context to the model. It is used over a average colour approach due to inaccuracies and can provide irrelevant results. For some images, its average colour could be different to how the image appears (see figure 8 for context). This approach would only further increase the semantic gap created between the pixels in the images and a high level concept of image similarity. It differs from the HSV colour feature extraction method in that representative colours are selected from each image instead of being fixed in the colour space [16]. This approach aims to provide a detailed image descriptor whilst keeping feature vector dimensionality low - each element in a feature vector will be a dominant colour found in the image.

In this approach to dominant colour feature extraction, the K-Means Algorithm [15] is used to extract dominant colour components from a HSV histogram using clustering. The aim of K- Means is the minimisation of an objective function that samples the distances between data points and the cluster centres and is calculated using the following equation:

$$J = \sum_{j=1} k \sum_{i=1} n \parallel x_i^{(j)} - c_j \parallel^2 \tag{1}$$

Where $\parallel x_i^{(j)} - c_j \parallel^2$ is the Euclidean Distance between data point $x_i^{(j)}$ and the cluster centre c_j [11], J represents the global objective function. The following outlines the steps to produce a dominant colour feature vector from a given image:

Step 1: Convert images from RGB space to HSV space.
Step 2: Transpose HSV image into shape 2.
Step 3: Initialise K Means with amount of clusters (Feature Vector Length)
Step 4: Generate a new partition by assigning data points to nearest cluster centre.
Step 5: Recalculate the centres for clusters receiving new data points and for clusters losing data points.
Step 6: Repeat steps 4, 5 until distance convergence criterion is met.
Step 7: Return clusters centres as a flattened feature vector.

$$transpose(img[H, W, 3]) = [H \cdot \frac{W}{2}, 3] \tag{2}$$

Where H is the image height, W the width, and the 3 colour channels of HSV. After step 7 a flattened feature vector is returned of length $3k$ where k is the number of clusters selected for K Means or the number of dominant colours extracted, for each value within the HSV colour space. For example, if initialised with 8 dominant colours, the feature vector returned will be of length 24.
Dominant colour feature extraction has several advantages for CBIR. Colour is an important cue for

human visual perception. Therefore extracting dominant colours within an image can create a more accurate representation of the colour context of an image. A major drawback of this approach is the algorithm will converge to local minima depending on if the initial clusters centres are initialised on outliers, it also can return non meaningful results if used on a noisy/ complex textured image and is sensitive to lighting changes.

5.1.3 Gabor Texture Features

Texture feature extraction comprises of two methods, Gabor filter features and Haralick features. Combining these two methods provides a robust description of texture for a given image. Before performing Gabor filter extraction, a filter bank needs to be created. Gabor filters allow for a description of textures localised in frequency and orientation, therefore, to form a detailed description of texture, 48 filters at at 8 different orientations and 6 different frequencies are created using a function written as follows:

$$g(x, y; \lambda, \theta, \sigma, \gamma) = \exp(-\frac{x^2 + \gamma^2 y^2}{2\sigma^2}) \ \exp(i(2\pi \frac{x^{'}}{\lambda})) \tag{3}$$

Where θ is the orientation, λ is the Frequency, σ is the standard deviation of the Gaussian, γ is the spatial aspect ratio and x, y represent the size of the filter returned. A subset of the filter bank can can be visualised in figure 13. Gabor filters are created at varying orientations and frequencies because texture in an image can be of different orientations and frequencies. Texture can be described by their spatial frequency being the number of cycles of a repeating pattern per unit distance, they can also be describe by their orientation which is the direction of the repeating pattern.

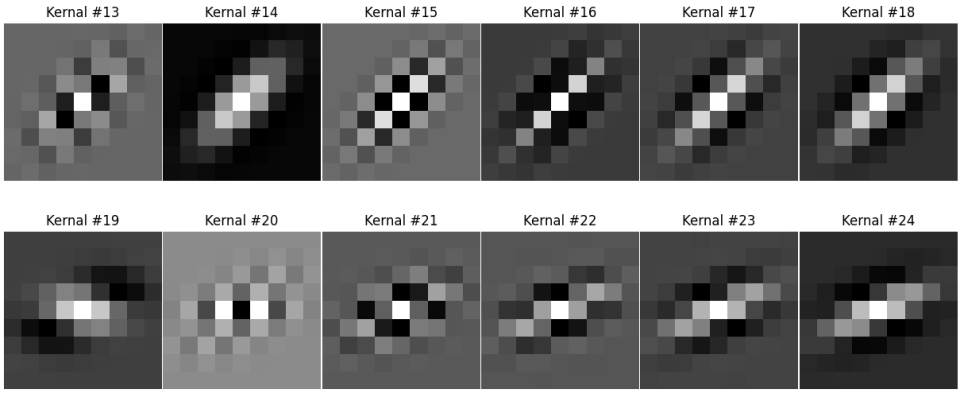


Figure 13: 12 Gabor filters at 2 orientations and 6 frequencies.

After constructing the Gabor filter bank, they are applied onto a MxN image matrix to extract the variance and average of filtered image by each Gabor.

$$av_m = \frac{\sum_x \sum_y \mid G_{m,n}(x, y) \mid}{MxN} \tag{4}$$

$$variance_m = \frac{\sum_x \sum_y (\mid G_{m,n}(x, y) - av_m \mid)^2}{MxN} \tag{5}$$

After these computations, the mean and variance of filter image compared to the input image matrix are concatenated into a single feature vector. This feature vector describes the spatial and frequency information of an image. This approach is effective to describe texture however has several drawbacks. Gabor filters convolutions with the input image are sensitive to noise in the image, this can result in inaccurate features being extracted and inaccuracies when classifying feature vectors.

5.2 Haralick Texture Features

The second method of texture feature extraction uses Haralick texture features. Haralick features can be used to distinguish between rough and smooth surfaces, for example rocks, sand and bricks in nature scenes. Haralick features are processed from the Gray Level Co-occurrence Matrix (GLCM) which calculates how many times two gray-level pixels adjacent to each other appear in an image [1]. The Haralick feature vector consists of 13 descriptors which are described in figure 14.

Number	Feature
1	Energy
2	Contrast
3	Correlation
4	Sum of Variance
5	Inverse Difference Moment
6	Sum Average
7	Sum Variance
8	Sum Entropy
9	Entropy
10	Difference Variance
11	Difference Entropy
12	Info. Measure of Correlation 1
13	Info. Measure of Correlation 2

Figure 14: Haralick Features used for feature extraction [4]

These 13 descriptors combine together to represent the texture of an image in lower-dimensional space. Similar to Gabor features they are also sensitive to noise and lighting changes in an input image therefore it is important to combine these feature extraction methods to create a more robust image descriptor.

5.2.1 Similarity Measurement

The Euclidean Distance is used to calculate a similarity measurement between a query image and a database image because of its higher accuracy and effectiveness. To measure the distance between two feature vectors of images, the Euclidean calculates the square root of the sum of the squared absolute differences shown in equation 6.

$$D(A) = \sqrt{\sum_{i=1}^{n}(Q_i - A_i)^2} \tag{6}$$

Where Q is a feature vector from the query image from an initial point i and A is a feature vector from a database image from an initial point i.
Model 1 is comprised of 4 feature vectors being histogram, Gabor, Haralick and dominant colours which for each database image a distance metric is calculated for a query image using equation 6. This process creates 4 distance vectors each with a corresponding weight to further tune feature importance and model accuracy. Database images with the lowest distance metric are deemed most similar to the query image of which the ten lowest are selected for retrieval. Results for this model can be seen in **Section 6.4.1**. Figure 15 shows the CBIR system architecture for this model.

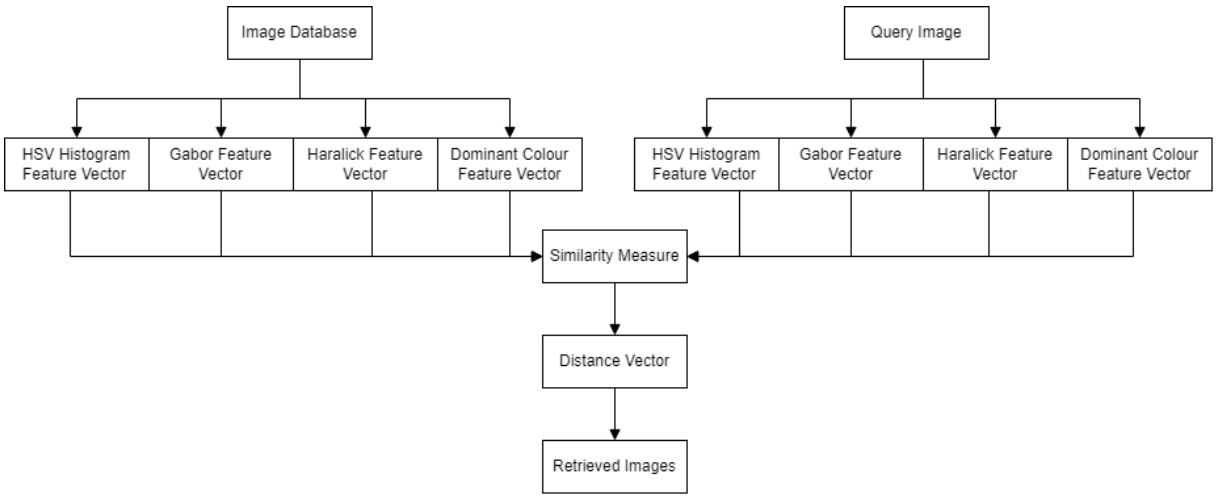


Figure 15: Model 1 system architecture

5.3 Model 2: Region Based Feature Extraction + Linear SVM

The second approach to the CBIR problem introduces a region based image segmentation to improve performance over model 1 to further reduce the semantic gap between the low - level image pixels and high - level human concept. It has been found that users are usually more interested in specific images regions over the global image, therefore this approach intends to mimic human perception to reduce the semantic gap. Image regions are created using a simple approach of slicing the image into different sections depending on the number of slices selected, this can be visualised in figure 16.

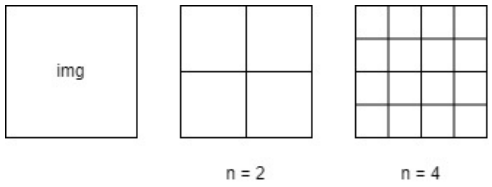


Figure 16: Image regions example with 2 and 4 slice number as parameters.

5.4 Shape Features

Model 2 introduces an image shape descriptor to extract spatial information of objects. Shape features are especially useful for many domain dependent images such as identifying similar man - made objects. For this reason shape features are included in module 2 to pair well with image segmentation.
The proposed shape image descriptor uses Histogram of Oriented Gradients (HOG) to identify object structure and edges. After an image is sliced into regular regions, The image gradient is computed in both the x and y direction. This is computed using a convolutional operation to obtain the gradient images as follows:

$$G_x = R \cdot D_x \tag{7}$$

$$G_y = R \cdot D_y \tag{8}$$

Where R is the input region, D_x and D_y are the filters in the x and y direction respectfully. This computation can be visualised in figure 17 on an image, note that in the implementation it will be computed in regions.

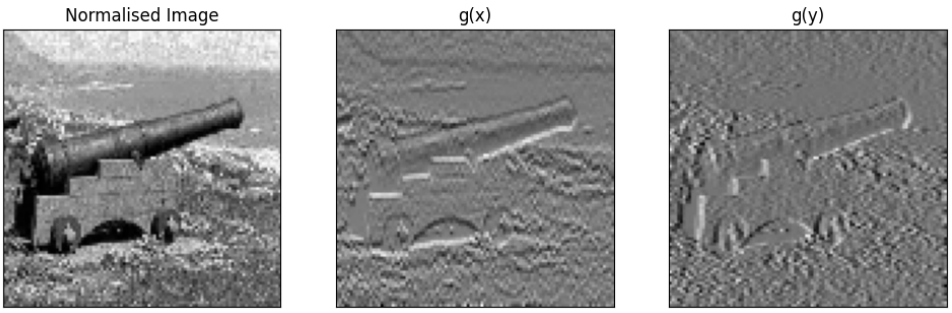


Figure 17: Normalised Grayscale Image, x and y derivatives

Gradient images can be used to compute gradient magnitude of the image using equation 9, then compute the orientation with equation 10.

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{9}$$

$$\theta = arctan2(G_y, G_x) \tag{10}$$

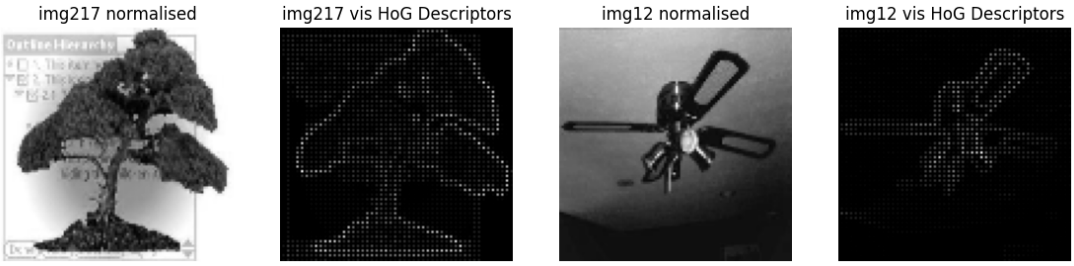


Figure 18: Visualisation of HoG Descriptors produced from function [19]

Figure 18 shows a visualisation of HoG feature extraction using sk.image.feature.hog [19]. In the figure the image regions can be seen as the grid of cells where a gradient is calculated for each. After all gradients are calculated, a histogram is produced which is concatenated into a single feature vector. This feature vector captures the spatial and frequency information of an input image. HoG features are robust to changes in illumination, contrast and scale for a given image, they are also effective at discriminating different objects which will work well with the classification stage of this model. They however are sensitive to noise so will be important to combine with a robust texture descriptor.

5.5 Feature Fusion

Feature Fusion is a process used to combine two or more features into a single feature vector to create a more robust and discriminative solution over a single input feature vector. Model 2 uses the same colour HSV histogram, Gabor and Haralick feature extraction methods used in model 1. The colour features are globally extracted and the Gabor, Haralick and HoG features are extracted regionally. Feature fusion can take place a multiple stages of CBIR; Matching - Level, feature-level or decision-level. In this approach the feature fusion takes place at the feature-level where after computing the individual feature vectors for colour, texture and shape they are combined to a single feature vector for classification. The feature fusion process is outlined in the following:

Step 1: Scale the normalised feature vectors using equation 11.
Step 2: Concatenate scaled feature vectors using equation 12.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{11}$$

Where x is the individual feature vector, min and max is the range which the features will be transformed into. In this implementation the range of $[0, 1]$ is used, meaning the minimum and maximum feature values will be 0 and 1 respectively.

$$F_{image} = [F_{colour}\ F_{texture}\ F_{shape}] \tag{12}$$

By the end of the feature fusion process, a feature dataset is formed. A row within the dataset represents an single image ($1x1052$).

5.5.1 Classification Stage

Support vector machines (SVMs) are learning methods used for classification. In the case of the CBIR system, it deals with multi-class classification problem of predicting image classes. Model 2 uses a SVM for classification over an Artificial Neural Network (ANN) for several reasons. High dimensionality after feature fusion results in high memory and computational costs for predictions, whilst SVMs offer effectiveness in high dimensional spaces with relatively low computational cost. An ANN classifier is also prone to over-fitting due to inadequate or unimportant training samples [13]. Once features have been extracted from the image database, the SVM is trained on a dataset of labelled image classes corresponding to image features. Once the SVM is trained, it can be used to classify new images in those classes, the SVM then outputs its predicted class. Results for this models performance can be seen in **section 6.4.2**. Figure 21 shows the system architecture for this model.

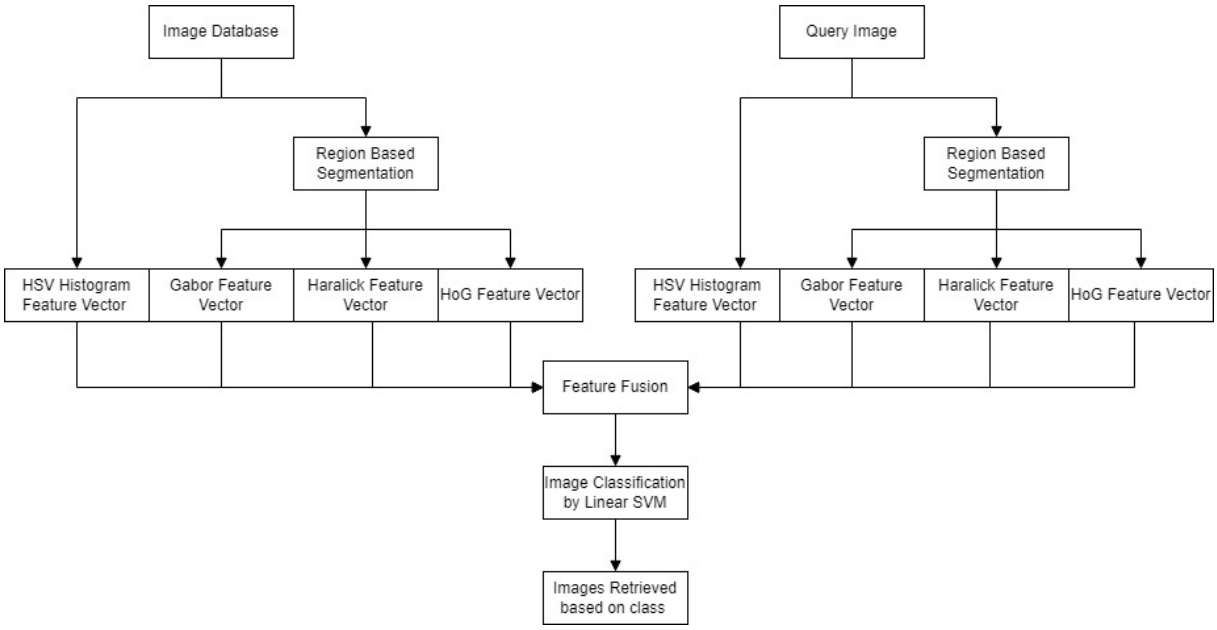


Figure 19: Model 2 system architecture

5.6 Model 3: CNN Based Feature Extraction + Linear SVM

The third approach to creating a CBIR model is with the use of a convolutional neural network to serve as an accurate feature extractor. A CNN is a deep artificial neural network comprised of linear layers including convolutional layers, or a fully-connected layer to perform convolution operations on the input data. Convolution operations are a mathematical operation which involves two functions combining to produce a third function which describes how one of original functions is modified by the other. This operation typically involves sliding a filter over an input data array and computing the dot product of the filter with the values in each window position. Convolution operations are able to capture local patterns in data therefore are a efficient technique to extract features, they are also invariant to translations so objects can be identified at different positions. A fully-connected layer is another linear layer in a CNN which connects every node in one layer to the next layer, in contrast to a convolutional layer which connects nodes in a local neighbourhood. Fully-connected layers are typically used in the last layers in a CNN to classify data which they are suited for because of their ability to learn complex relationships between the input data. The CNN is also made up of non-linear layers which apply a non-linear function called an activation function, this function is applied to the output of a layer used to introduce non-linearity into the network which is necessary to learn complex patterns. Another non-linear layer used is a pooling function which is a down-sampling operation to reduce the size of the feature vector while preserving important information. This reduces the computational complexity of the network.

Models 1 and 2 both use a traditional approach to feature extraction in the form of low - level feature extracted from the pixels in the image itself, model 3 however makes use of the VGG16 model which is a 16 - layer deep CNN developed by the Visual Geometry Group (VGG) [23], the models architecture can be seen in figure 21. The input of the VGG-16 network is of fixed size (244 x 244 x 3) therefore the Caltech101 dataset which will be used for demonstration is resized and preprocessed to fit the input requirements.

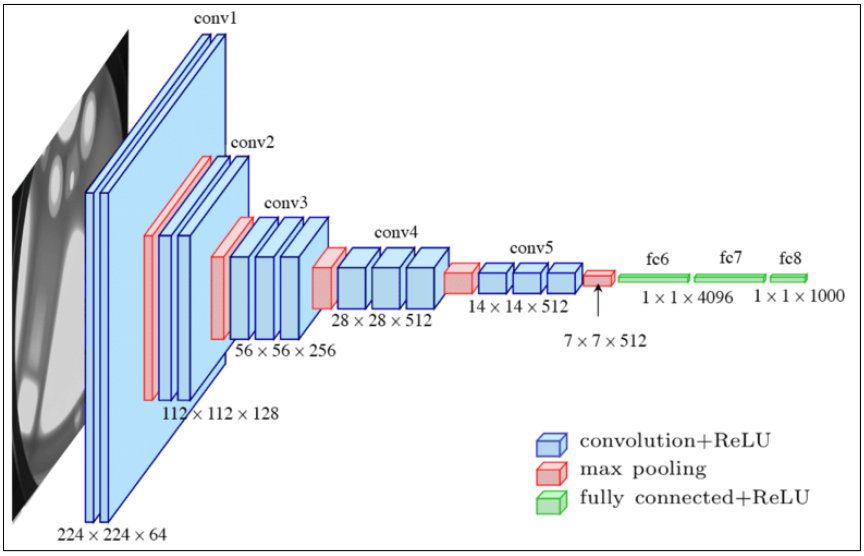


Figure 20: VGG-16 network architecture for feature extraction [22].

Preprocessing of images involves converting the images from RGB to BGR colour space. This is done because the BGR colour space is more efficient for CNNs as BGR is a linear transformation of the RGB colour space and linear transformations are faster to compute than non linear transformations. BGR is also more robust to noise in the image therefore increasing model performance. After converting colour space each colour channel is zero-centred with respect to the ImageNet dataset which is what the VGG16 model is then trained on. After preprocessing the images are passed into a various convolutional layers of its different receptive fields. The receptive field of a convolutional layer is the region of the input data array that affects the output of a neuron in the layer, they are used to determine what parts of the input array are relevant for the output of a particular neuron. The stride rate and max pooling layers are same throughout the VGG16 network being 3 on stride 1 in the convolutional layer and 2 with stride 2 in the max pooling layer. Stride rate is the number of pixels that a convolution filter moves over the input image, with a stride rate of 3 the convolution filter moves over 3 pixels at a time. A small stride rate like 3 can be used to control the down-sampling that occurs in the network, creating smaller feature vectors. In the VGG16 network the convolutional layers contain 64, 125, 256, 512 and 512 filters respectfully. The filters as mentioned is what is convolved with the window of input image to produce a feature map. Border pixels are padded before each convolutional layer to prevent the loss of information at the edges of the input image by adding zeros to the border of the image before each convolution. The final 3 layers of the VGG16 network contains 3 fully-connected layers where the first contains 1 x 1 x 4096 neurons followed by the final layer containing 1 x 1 x 1000 neurons to downsample the feature vector to 1000 features.
After building features vectors for a given image database, the images are classified using a optimised Linear SVM into image classes similar to the technique used in model 2. The system architecture can be seen in figure 21.

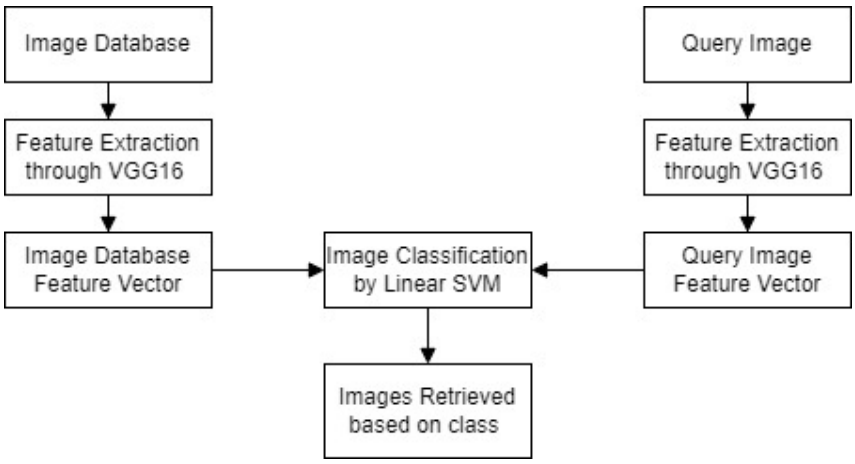


Figure 21: Model 3 system architecture

6 Experimental Results

In this section, the quantitative data of different approaches is tested to better optimise model performance in both classification and feature extraction. We evaluate into the effect of different parameters in the CBIR system and assess performance on changing them. To ensure validity of results, each test is repeated 3 times with a different random subset of the Caltech101 dataset being 8 unbalanced image classes containing from 36 - 150 images, this is done to reduce the effect of different image classes being biased by certain feature extraction methods, for example 'sunflower' labelled images may be biased towards shape features because of their characteristic petal shape. To provide a accurate estimate of model performance, models are trained using cross-validation. Cross-validation is a technique used to evaluate performance of a model on unseen data. The training data is split into multiple folds, training the model on a subset of the folds and testing the model on the remaining folds. This process is repeated multiple times with different folds used for training and testing, the average performance of the model across all folds is then used to evaluate model performance. 10 folds of the data and 3 repeats are used in these tests. To evaluate model performance, the F1 score is calculated which is a performance metric for classification and is defined as the harmonic mean of the precision and recall of a model. Precision in CBIR can be defined as the ratio of relevant images retrieved to all images retrieved. In the case of these tests, relevant images are ones with the same label.

$$Precision = \frac{A}{C} \tag{13}$$

Where A is the number of relevant images retrieved (True Positive) and C is all images retrieved (True Positive + False Positive).

Recall in CBIR is a measure of how many relevant images are correctly classified by the model in respect to a given image class.

$$Recall = \frac{A}{B} \tag{14}$$

Where A is the number of relevant images retrieved (True Positive) and B is the number of true positives and false negative images retrieved. The precision is also known as a positive predictive value, while the recall is also known as the sensitivity. As stated, the F1 score is a single number that summarises both the precision and the recall and is calculated as follows:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{15}$$

The F1 score ranges from 0 to 1. A score of 1 indicates perfect precision and recall, while a score of 0 indicates no precision and no recall. All these steps result in an accurate evaluation metric and results are consistent on multiple runs to ensure validity of results. This evaluation metric is used on all tests unless stated otherwise.

6.1 Machine Learning

In this study, a range of different machine learning techniques where tested to predict image classes from a feature dataset. The motivation for the study is achieving as much performance from model 2's feature extraction methods, creating more accurate predictions and to better understand the robustness of each feature extraction method for later tests. 4 machine learning techniques are tested which where selected based on CBIR surveys in papers [14] and [10]. Naives Bayes classification isn't featured in these papers however with its relatively simple implementation and low computational cost its included to as a baseline to compare to other methods, though good performance wasn't expected. K- Nearest Neighbour algorithm is a supervised learning classifier used to make predictions based on proximity to other data points. In this case it is selected to find the similarity between features within the feature database, to predict similar images and images classes. Support Vector Machines (SVM) are often used within CBIR to learn high - level semantics from low - level images features so are considered to be a good candidate for learning in CBIR. Multinomial Logistic Regression model is also used in this test as it makes no assumptions about distributions of classes in the feature space, it provides a measure of how appropriate the prediction labels is as well as the direction of association. The 4 supervised machine learning models are trained and evaluated using the method as stated at the start of this section.

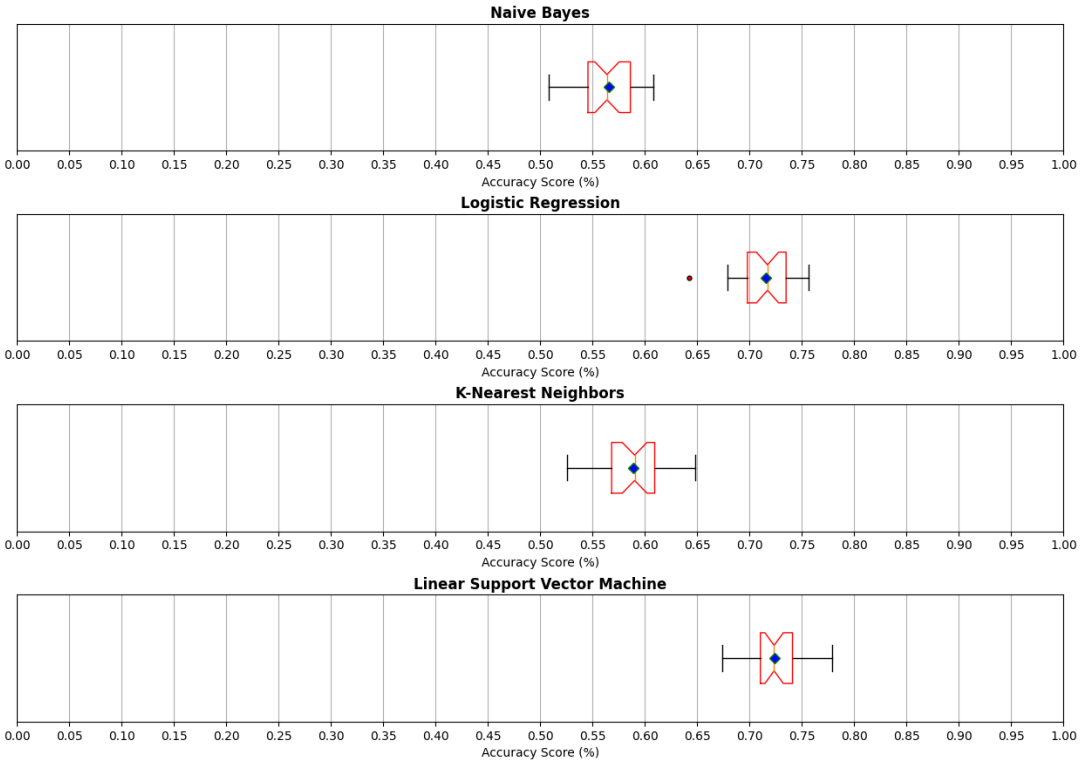


Figure 22: Box-plots of model accuracy for study 1

Figure 22 show the results from this test and note that Accuracy Score (%) denotes the F1 score. The boxplots represent the range of results carried out by cross validation where in a given run of image data, the data is folded 10 times and 3 repeats of the training, resulting in 30 F1 scores. For each F1 score, the mean is calculated for all runs of the image data (3 sets of image data). The Naive Bayes and KNN learning techniques perform predictably average, while the LR and Linear SVM perform the best. The KNN algorithms performance could be explained by the high dimensional feature vectors produced by the feature extraction process which is not suited for a KNN algorithm. A KNN algoithm is also sensitive to irrelevant features which may be produced by model 2s feature extraction methods, for example colour features of the same class could be dissimilar because of a different background colour. A Naive Bayes classifiers performance could be explained by several reasons. The Naive Bayes algorithm assumes that all features in a feature vector are independent of each other given an image class, for the case of this system, this assumption may not hold true as features correlate to each other. For example a petal shape features correlates to its texture features being of a teardrop like shape and being a smooth texture. A SVM and LR model perform well for this task as they are suited to train with large feature vectors, SVMs are able to learn nonlinear relationships between features which as discussed is prevalent in this

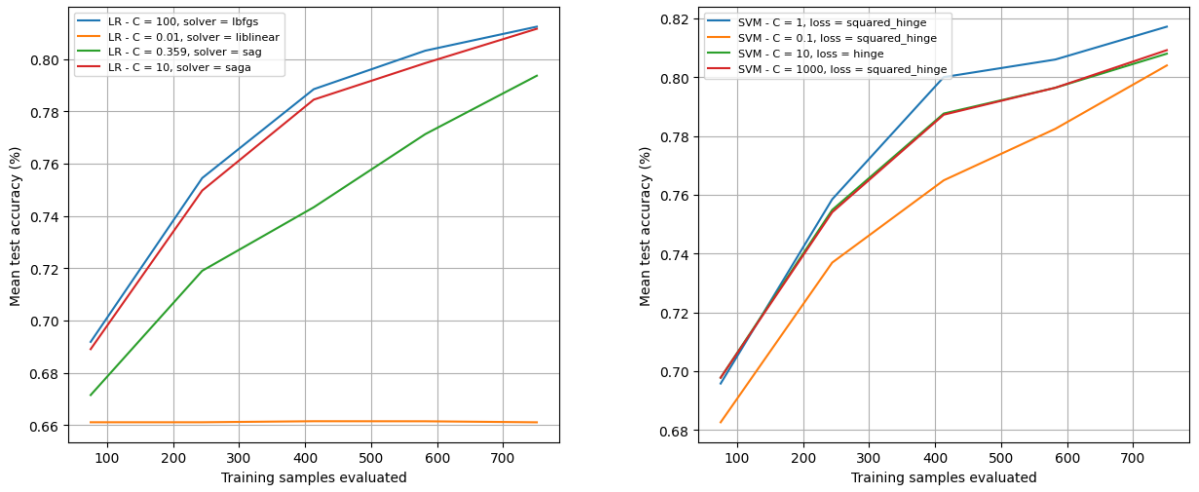
system and are robust to noise because of the versatile decision boundary.

6.1.1 Hyperparameter Tuning on best models

Hyperparameter tuning in ML is the process of finding the best combination of hyperparameters for a given ML model. Hyperparameters control how a model learns from data, therefore adjusting these settings can improve model performance. The method used to optimise hyperparameters for this project is Grid Search, this is a brute-force approach that tries all possible combinations of hyperparameter values to find an optimal solution. In this test, the two best models from the previous test are selected for tuning this being a SVM and Logistic Regression model. Both models regularisation parameter is tuned which controls the amount of regularisation applied to a model. This can be used to prevent over-fitting which occurs when a model trains too well on a dataset and does not generalise to new data well. For the SVM the loss function is also tuned which measures the error of a model, it is used to penalise the model for misclassifying data points. The SVM is tuned for hinge and squared-hinge loss, squared- hinge is a quadratic and is more likely to converge to a global minimum, while hinge-loss is a piecewise linear function but more efficient to compute. Another hyperparameter tuned for the linear SVM is penalty parameters. There are two penalty parameters being L1 and L2 that are used to control the complexity of the model. It should also be noted that hyperparameter tuning was carried out on a single subset of the Caltech101 dataset instead of 3 to reduce computational time, results (hyperparameter combination rankings) are consistent through multiple runs however accuracy values may be different.

$$L(y_i, \hat{y}_i) = (1 - y_i\hat{y}_i)^2 \tag{16}$$

$$L(y_i, \hat{y}_i) = max(0, 1 - y_i\hat{y}_i) \tag{17}$$



(a) Logistic Regression

(b) SVM

Figure 23: Learning Curves of the 2 models with 4 combinations of different hyperparameters.

Figure 23 shows the hyperparameter tuning of A linear SVM and Logistic Regression model with a combination of 4 different sets of parameters in the form of a learning curve. The different sets of parameters where selected to have a good range of performance from best to worst in the entire list of parameters used in the grid Search. Overall, this test showed that the 'lbfgs' solver for an LR model was the most effective solver on the feature database and 'squared hinge' was the most effective loss function for the SVM.

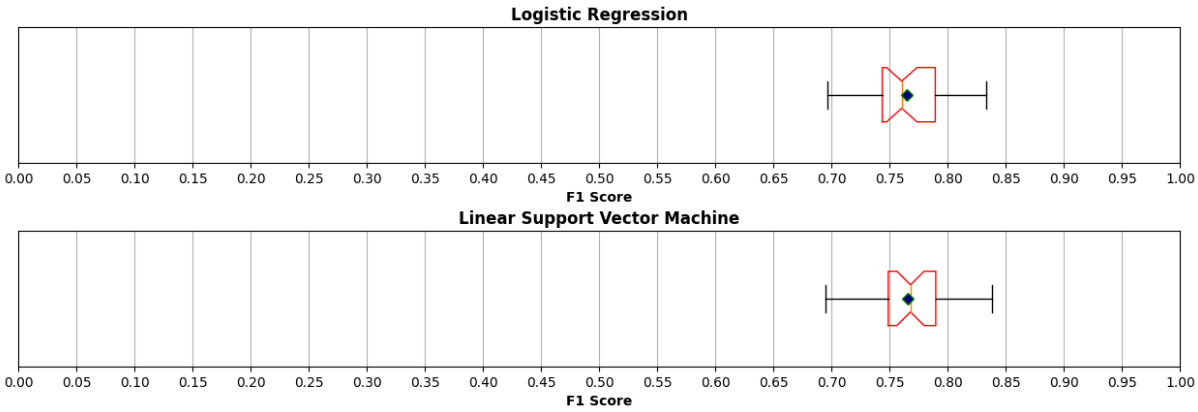


Figure 24: Normalised Grayscale Image, x and y derivatives

Figure 24 shows the boxplots of the ML models with optimised hyperparameters. It shows a far more consistent model over the unoptimised models shown in figure 22 as the boxplots are shorter. Shorter boxploxs can also indicate a less complex model as there is less variation in the models predictions.

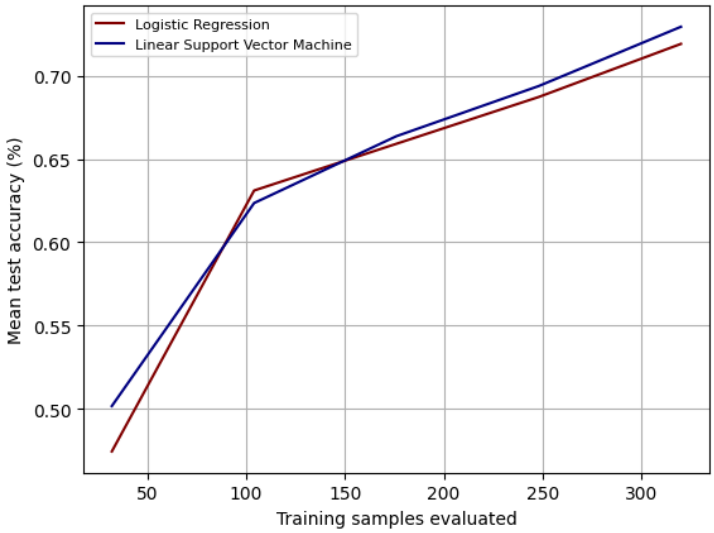


Figure 25: Learning Curve of Linear SVM and Logistic Regression Models with optimal hyperparameters

Figure 25 shows the learning curve of the 2 models tested with optimal hyperparameter combinations found using gridsearch. After training, the Linear SVM provides more accurate predictions over LR model which could be attributed to several reasons. First, Linear SVMs are deigned to be discriminative, as discussed in previous sections they are trained to find a hyperplane that separates the data into classes where possible, LR is a probabilistic model trained to predict the likelihood of an image belonging to a particular class. This could imply that the feature database produced from feature extraction is better suited to be linearly separable as different image classes have consistent features which can be attributed to that image class, for example a flowers petals shape or a saxophones colour being similar in all saxophone images. Another reason for this better accuracy is that SVMs are more robust to noise in the data over LR, this is because SVMs use a margin-based approach to classification where they only focus on data-points that are most difficult to classify into a class. SVMs can be more efficient to train than LR, especially when the amount of features is large. Each image in the feature database having 1024 different features and LR trains using a iterative algorithm which could be slow to converge. For the reasons stated a Linear SVM is used for both models 2 and 3 and used for further evaluation in the next tests.

6.2 Feature Importance

To accurately assess performance based on feature extraction methods, a test is carried that creates 4 feature vectors made up of feature fusion from individual feature extraction methods but missing a

single feature. The aim of this experiment is to find what feature extraction method creates the best performance by comparing the performance of the other feature vectors which all contain that single feature with the feature vector missing that feature being tested. The feature vectors are created as the following:

$$\begin{aligned} Colour &= [F_{HOG}\ F_{Gabor}\ F_{Haralick}] \\ Gabor &= [F_{colour}\ F_{Haralick}\ F_{HOG}] \\ Haralick &= [F_{colour}\ F_{Gabor}\ F_{HOG}] \\ HoG &= [F_{colour}\ F_{Gabor}\ F_{Haralick}] \end{aligned} \tag{18}$$

As with the other tests it is carried out on 3 random subsets of Caltech101 dataset with 8 image classes and classified with a Linear SVM with optimal hyperparameter settings and cross validated. An F1 score is used as the performance indicator.

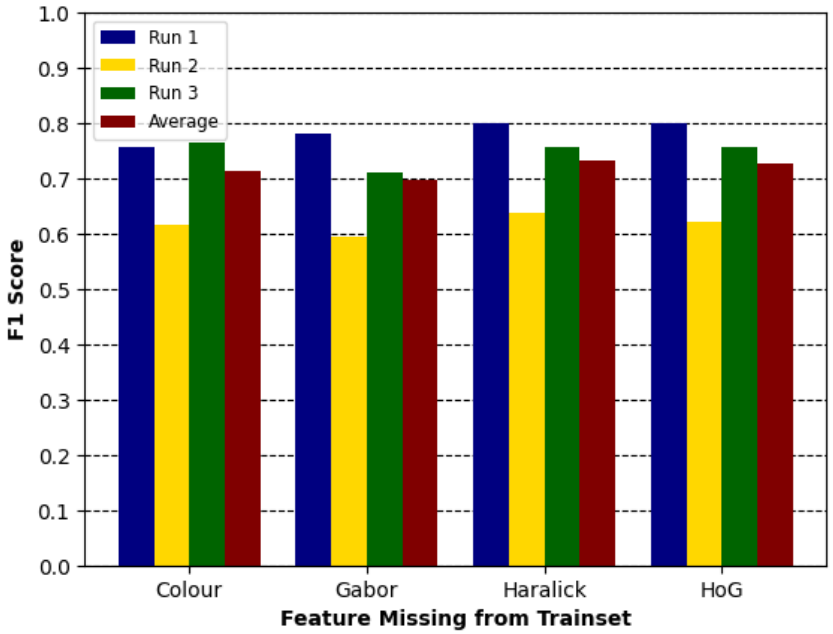


Figure 26: F1 Scores for feature vectors in figure 18 for 3 runs and average

Figure 26 shows the results from this test. Surprisingly colour features do not show the highest performance as which is what was expected from research. A reason for this could be the use of the Caltech101 dataset where shown in figure 9 and figure 10 images often are recoloured and inconsistent throughout a given image class. This could explain the reduced accuracy as the model is more likely to predict false positives as colour histograms are less similar to those in the same image class. The feature vector containing no Gabor features is the least accurate out of all others signifying they are the most important for create better performance in the model, Gabor filters extract texture features which are not dependent on colour which as discussed could lead to inaccuracy in the model. The Caltech101 dataset also contains a majority of object based images where texture features can be used to identify materials/ surfaces of the objects which are invariant to a given image class, this can also explain Haralick features importance however they aren't as important as Gabor features. This could be because Gabor features are more sensitive to the spacial frequency of texture, meaning they are better at distinguishing between different types of textures such as smooth or rough. Gabor features are also more robust to noise in the image, the caltech101 dataset is notably noisy and of low resolution which could be a factor in Gabor performing better than Haralick features. A technique to improve Haralick features could be to apply a Gaussian filter on the images for preprocessing to smooth the image. HoG features in this test show that they do not improve performance as much as other features, this could be because again the images are noisy where HoG features are sensitive. Combining all feature vectors results in an average F1 score

of 73.7% for this test which is more accurate than the best performing subset feature vector being the feature vector without Haralick features of 73.1% indicating that together all feature extraction methods work together to provide better performance in the model.

6.3 Feature Fusion Optimisation

Leading on from the previous section, a test was carried out to see the effects of different feature fusion combinations to see if using all feature extraction techniques are actually worthwhile or if there is another optimal solution. It also give insight into what feature extraction techniques work well together over some other combinations. 4 feature fusion vectors are created for this test and shown in figure 19, these four combinations are selected from results of the test in section **6.2** where colour and Gabor features seem to be the most effective therefore used in all vectors. As with the other tests it is carried out on 3 random subsets of Caltech101 dataset with 8 image classes and classified with a Linear SVM with optimal hyperparameter settings and cross validated. An F1 score is used as the performance indicator.

$$\begin{aligned} db_1 &= [F_{colour}\ F_{Gabor}] \\ db_2 &= [F_{colour}\ F_{HOG}] \\ db_3 &= [F_{Gabor}\ F_{HOG}] \\ db_4 &= [F_{colour}\ F_{HOG}\ F_{Gabor}\ F_{Haralick}] \end{aligned} \tag{19}$$

Where *colour* is colour histogram features, *HOG* is HOG shape features, *Gabor* is Gabor texture feature and *Haralick* is Haralick texture features. This test aims to find optimal feature combinations to create a more accurate model.

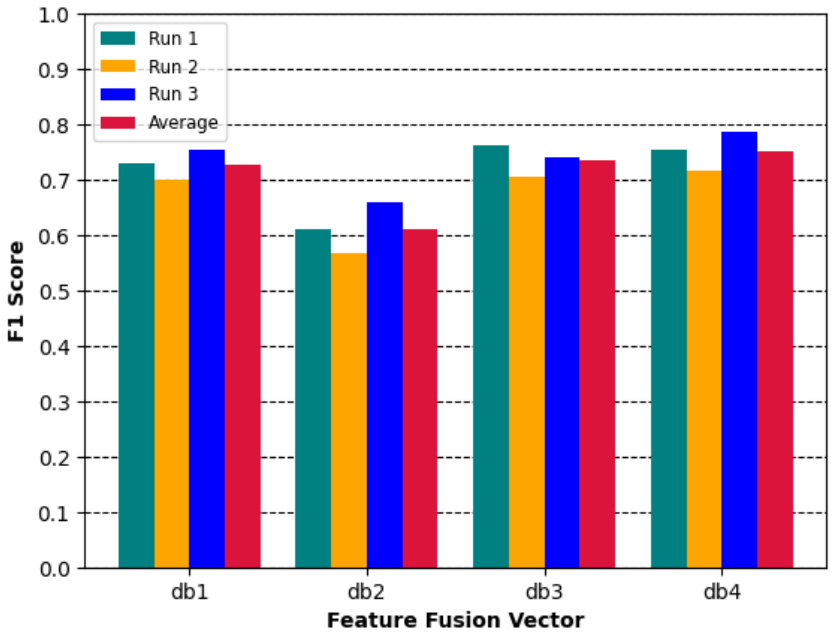


Figure 27: F1 Scores for feature vectors in figure 19 for 3 runs and average

Figure 27 shows the results from this test. The best performing feature vector is db_4 which contains all feature extraction techniques which is good indicator that the best performing solution utilises all feature extraction techniques working together to produce an accurate image descriptor with an average F1 score of 75.192% across these 3 runs. The worst performing feature vector is db_2 which contains only colour and HoG features, looking at the results for the test in section **6.2** HoG features (figure 26) showed high importance however it was found that Gabor features where the most important features which this feature vector is missing. The Caltech101 as discussed relies quite heavily on these features which explains the poor performance of this feature vector, this is further backed up by the other 3 feature vectors which contain Gabor features performing better. Feature vector db_3 performs similar to db_4 indicating that

HoG and Gabor are the most descriptive image feature extraction methods for this dataset, as discussed the dataset contains many object based images with varying colours explaining their high performance of 73.61% but still not as effective over all features of 75.192% for these runs.

6.4 Model Evaluation

6.4.1 Model 1

Model 1 works with a similarity distance metric an opposed to a image label classifier used for models 2 and 3. This can result in some cases where images with the lowest distance metric are not in the same class, but appear visually similar. This has several advantages and disadvantages. Using a similarity distance metric for retrieval allows much more flexibility in the system as a user can simply manually go through the top 'n' images retrieved until a suitable candidate(s) has been selected, it also has a different purpose of finding images that appear visually which all depend on the application in which the CBIR system is designed to work in. The following figures show 3 retrieval results using model 1, The image dataset used is again 8 image classes from Caltech101 as with all tests.

Figure 28: A given run of model 1 with query image (top left) and 9 retrieved images

Figure 29: A given run of model 1 with query image (top left) and 9 retrieved images

Figure 30: A given run of model 1 with query image (top left) and 9 retrieved images

Figures 28, 29 and 30 show the results of these image retrieval runs and show the strengths and limitations of the model. All retrievals show the strong colour weighting of feature extraction method (60% of feature weighting) which results in visually similar images however the context of the images is often incorrect. The model over 10 runs performed 38.88% accuracy of predicting the image class of a query image in the 10 lowest distance metric retrieved images. The most accurate run is shown in Figure 29, the 'ibis' image class contains mainly nature context images which could favour colour features with more green biased histograms produced. To better the approach of this model, an image segmentation approach could be used to identify objects in the image and remove the background/ irrelevant information. This model also shows how depending on a given query image, it may favour certain image features over others. For example, figure 28 shows a query image of a beaver with a white background which would favour shape/ texture features over colour features to identify other images with a 'beaver like' shape.

6.4.2 Model 2

Model 2 builds off model 1s shortcomings to improve performance. As mentioned, the model uses a image class classifier after feature extraction which has its own set of advantages and drawbacks over the distance metric system used in model 1. An image class classifier allows for training a model on looking for specific characteristics/ patterns of a given image class to make better predictions. However the main disadvantage of this approach is when the model gets a prediction wrong, the images retrieved are from a different class as the query image is classified at retrieval time. The following figures show 3 confusion matrices to evaluate the performance of the classification over 3 runs of different subsets of the Caltech101 dataset using 8 image classes using validation data.

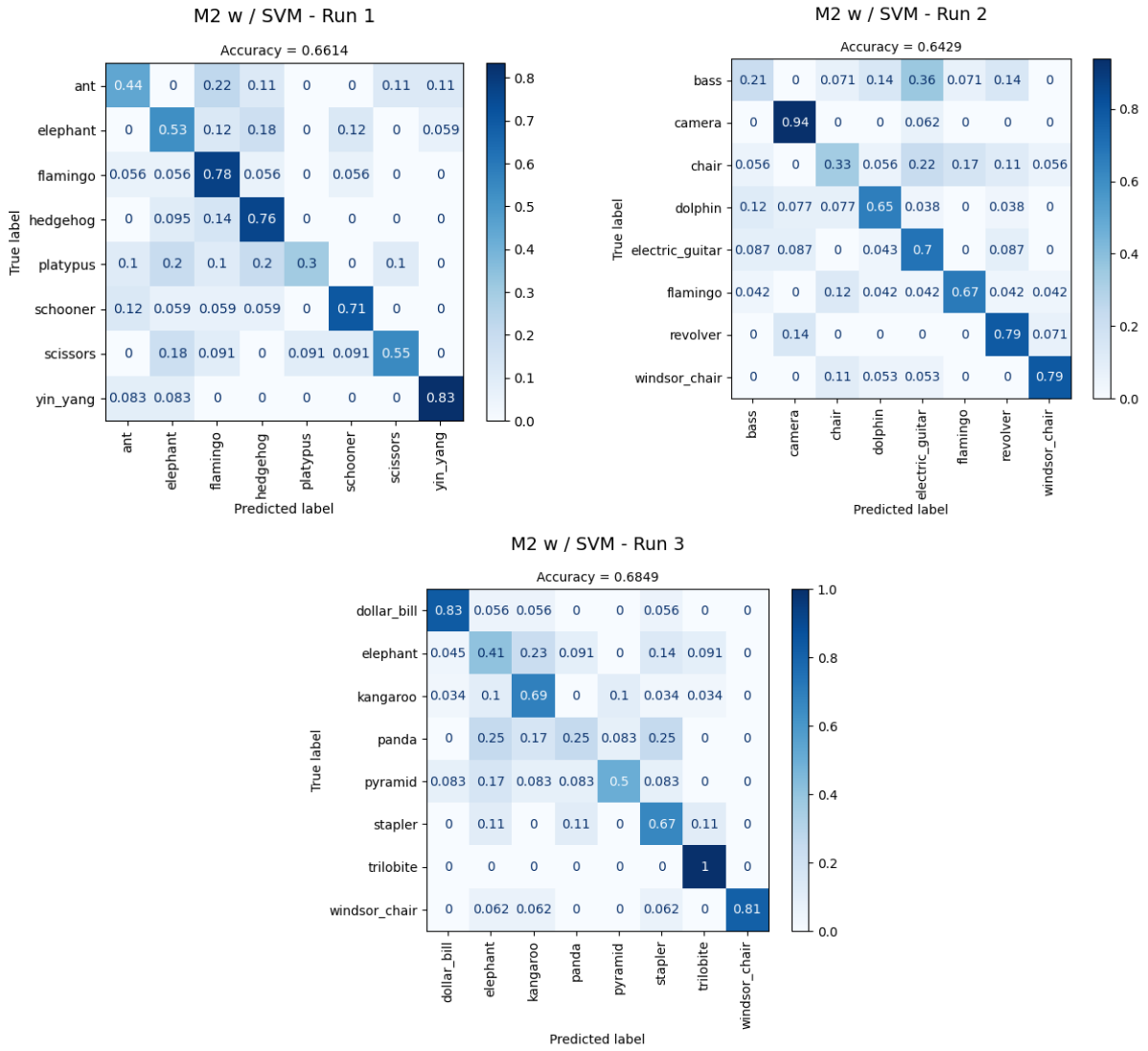


Figure 31: Confusion matrices of the results from model 2s feature extraction and classification using a linear SVM across 3 runs.

Figure 31 shows the results from this experiment. For some image classes, the model performs exceptionally well in labels such as 'trilobite' in run 3, 'yin_yang' in run 1 and 'camera' in run 2. the 'trilobite' image class contains images of trilobites which are extinct marine arthropods, all images contains very similar shape features being a fossil shape and are surrounded or embedded in rock therefore edges can be identified easily because of the contrasting colours. The 'yin_yang' class performs well with this model for similar reasons, the images in the class contain the same symbol so shape features perform well. Most images in the class are clip art style images so texture features should be consistent with a smooth surface with little noise, there are however 17% inaccurate predictions for this class in the 'ant' and 'elephant' classes which may be because of the large number of white pixels within the images which some images in these classes contain. A way to negate this inaccuracy is to perform image segmentation on key objects within the image to remove irrelevant information instead of the region based approach used. A image class where this model performed especially poorly is the 'bass' image class which contains image of bass fish. The model predicts a higher amount of bass images to be classed 'electric_guitar' over its correct class. Looking at the image classes, images often depict a person holding the fish in a similar way to that of holding a guitar, this could explain the high number of false negatives. The shape features extracted could be very similar, to combat these inaccuracies A CNN could be used like that in model 3 to learn high-level semantic concepts of differentiating a fish from a guitar.

6.4.3 Model 3

Model 3 provides the most accurate prediction out of all models created in this report. It incorporates the same classification process as used in model 2 including hyperparameters, so therefore can act as a good comparison of feature extraction techniques comparing the low- level extraction of image pixels in model 2 and high-level semantic ideas capable of model 3. Another key advantage of model 3's approach over the other 2 models approach is computational performance, an average run of feature extraction for 800 images by model 2 takes 430 seconds on average while model 3 takes just 55.

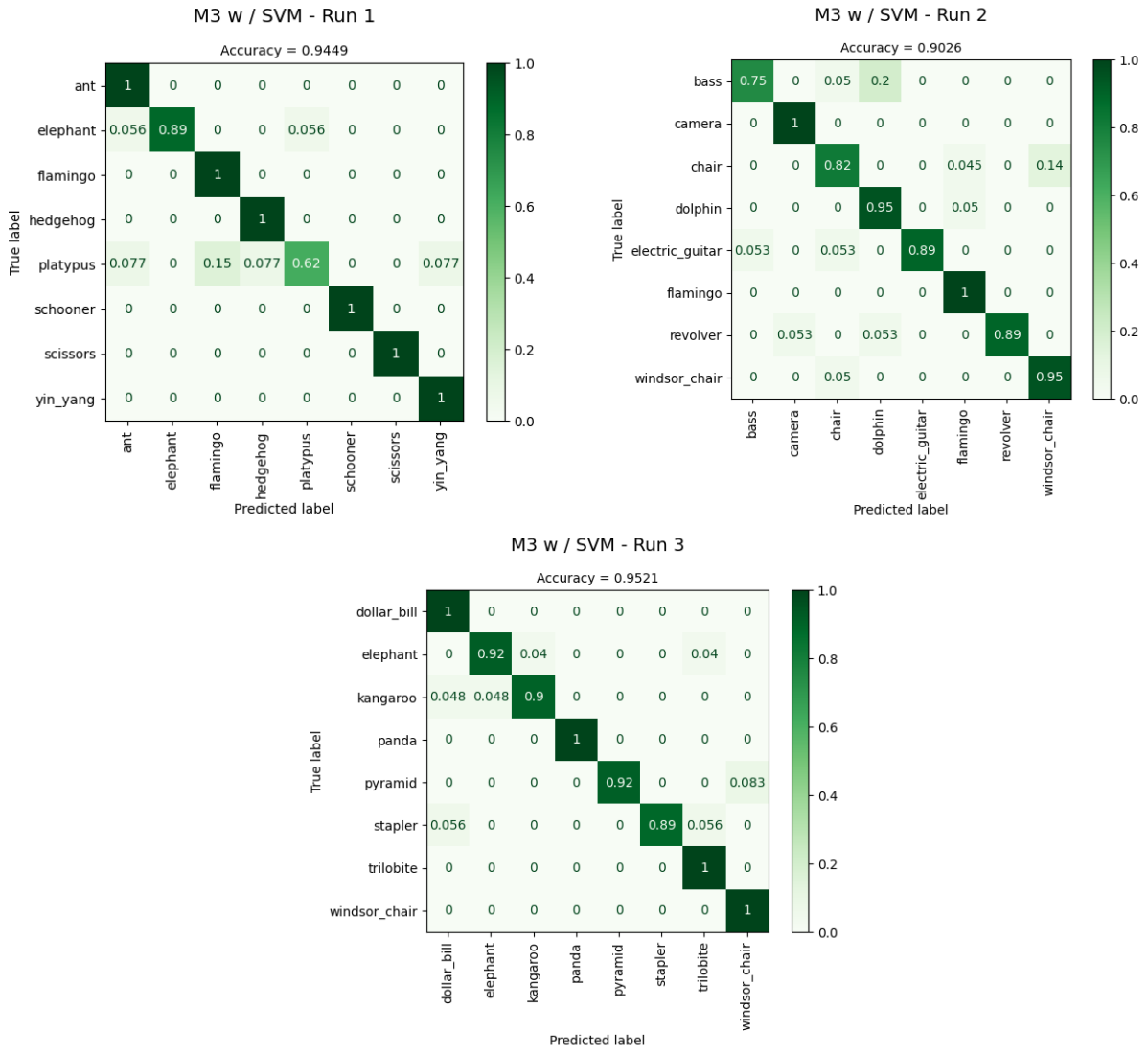


Figure 32: Confusion matrices of the results from model 3s feature extraction and classification using a linear SVM across 3 runs.

Figure 32 shows the confusion matrices of 3 runs of CBIR from model 3, it uses the same subsets of the datasets used in figure 31s test. Model 3 achieves 93.29% prediction accuracy across 10 runs of this experiment. Comparing the results of figure 31 and figure 32 shows a clear improvement of performance over model 2 but areas of weakness in both models appear in similar areas. An example of this is in run 1 where images labelled 'platypus' are in both tests the least accurately predicted class, a reason for this could be the dataset used for the run contains other 'animal' like creatures such as 'hedgehog' which could be classed as similar image types even from a human prediction. Model 2 also falsely predicts 'platypus' images as 'scissors' images which could be explained by the low-level feature extraction not being able to capture complex features to differentiate these classes while model 3 does not make any predictions in this class. Model 3 scores 62% prediction accuracy while model 2 scores 30% prediction accuracy for this class which shows a clear improvement in performance. An example of both models performing well in predictions is in run 2 for image class 'camera'. The 'camera' contains images of very similar shapes and

colours being a rectangular shape and grey-like colours allowing the shape and colour feature extraction techniques of model 2 to make consistent and accurate predictions.

7 Discussions

7.1 Conclusions

7.1.1 CBIR Models

From the results generated through the evaluation section I would suggest that the models are capable of accurate image retrieval based on a query image. Model 1 from figures 28, 29 and 30 show similarity between query and retrieved images but models 2 and 3 are very effective at identifying image classes with model 3 being the most capable model produced. The Caltech101 dataset provides many challenging and varied image classes ranging from nature to human faces which can be identified by the model. From the stretch goals set out in section **1.5** goals not achieved include implementing a text based image retrieval system which could be easily implemented with more time spent. Another stretch goal not achieved is providing complex interactivity between the user and the models, this project started to create a system designed for an end user however quickly evolved to become more of a research project to allow more time to be spent on optimising models and exploring more techniques. Model 1 can be interacted with a GUI which allows for somewhat complexity interaction but its not system wide as first desired. Lastly, more techniques of feature extraction would allow for a more detailed evaluation and allow for more flexibility in the system. Some techniques where experimented with including Daisy and SiFT shape features however lacked performance over HoG features so where removed.

7.1.2 Machine Learning

The most time in development for this project was spent on machine learning, optimising models and identifying the best models/ algorithms for classification. This was worthwhile as it provided the biggest performance improvements within the models and all stretch goals concerning machine learning have been met. Improvements could be made after feature extraction to have a model to learn a distance metric instead of classifying models based on classes, as stated this has its own benefits and drawbacks but for this project the main reason would be ambiguity for evaluation of the models as performance would be based on human perception instead of a accuracy score. Lastly an implementation of a CNN built by myself instead of having a pretrained model for model 3 would have been ideal, however difficulty in setting up GPU based computations for training the model resulted in absurd training times and was scrapped for the VGG16 model.

7.1.3 Self Evaluation

Despite some setbacks I believe this project was carried out well. The tools all performed well with an exception of the combination of windows 10 and visual studio making it very difficult to setup Keras GPU model training. To truly complete this project, implementation of a GUI working with all models would be ideal, but the final version of models I believe effectively carry out CBIR.

References

[1] "10.6 Haralick texture". In: (). URL: https://cvexplained.wordpress.com/2020/07/22/10-6-haralick-texture/.

[2] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[3] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[4] Filipe R Cordeiro, Wellington P Santos, and Abel G Silva-Filho. "An adaptive semi-supervised Fuzzy GrowCut algorithm to segment masses of regions of interest of mammographic images". In: *Applied Soft Computing* 46 (2016), pp. 613–628.

[5] Umur Erkut et al. "HSV Color Histogram Based Image Retrieval with Background Elimination". In: *2019 1st International Informatics and Software Engineering Conference (UBMYK)*. 2019, pp. 1–5. DOI: 10.1109/UBMYK48245.2019.8965513.

[6] Li Fei-Fei, Rob Fergus, and Pietro Perona. "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *Computer Vision and Pattern Recognition Workshop* (2004).

[7] Ruigang Fu et al. “Content-based image retrieval based on CNN and SVM”. In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. 2016 2nd IEEE International Conference on Computer and Communications (ICCC). 2016, pp. 638–642. DOI: 10.1109/CompComm.2016.7924779. URL: https://doi.org/10.1109/CompComm.2016.7924779.

[8] Ruigang Fu et al. “Content-based image retrieval based on CNN and SVM”. In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. 2016, pp. 638–642. DOI: 10.1109/CompComm.2016.7924779.

[9] Ngo Giang, Tao Ngo, and Dung Nguyen. “Image Retrieval with Relevance Feedback using SVM Active Learning”. In: *International Journal of Electrical and Computer Engineering (IJECE)* 6 (Dec. 2016), pp. 3238–3246.

[10] Ibtihaal M. Hameed, Sadiq H. Abdulhussain, and Basheera M. Mahmmod. “Content-based image retrieval: A review of recent trends”. In: *Cogent Engineering* 8.1 (2021). Ed. by D T Pham, p. 1927469. DOI: 10.1080/23311916.2021.1927469. eprint: https://doi.org/10.1080/23311916.2021.1927469. URL: https://doi.org/10.1080/23311916.2021.1927469.

[11] Dana E Ilea and Paul F Whelan. “Color image segmentation using a spatial k-means clustering algorithm”. In: (2006).

[12] Afshan Latif et al. “Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review”. In: *Mathematical Problems in Engineering* 2019 (Aug. 2019), p. 9658350. ISSN: 1024-123X. DOI: 10.1155/2019/9658350. URL: https://doi.org/10.1155/2019/9658350.

[13] Peng Liu et al. “SVM or deep learning? A comparative study on remote sensing image classification”. In: *Soft Computing* 21.23 (Dec. 2017), pp. 7053–7065. ISSN: 1433-7479. DOI: 10.1007/s00500-016-2247-2. URL: https://doi.org/10.1007/s00500-016-2247-2.

[14] Ying Liu et al. “A survey of content-based image retrieval with high-level semantics”. In: *Pattern Recognition* 40.1 (Jan. 2007), pp. 262–282. ISSN: 0031-3203. URL: https://www.sciencedirect.com/science/article/pii/S0031320306002184.

[15] J MacQueen. “Classification and analysis of multivariate observations”. In: *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA. 1967, pp. 281–297.

[16] B.S. Manjunath et al. “Color and texture descriptors”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 11.6 (2001), pp. 703–715. DOI: 10.1109/76.927424.

[17] R. Mehrotra and J.E. Gary. “Similar-shape retrieval in shape data management”. In: *Computer* 28.9 (1995), pp. 57–62. DOI: 10.1109/2.410154.

[18] Inc. Notion Labs. “Notion”. In: URL: https://www.notion.so.

[19] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Dong Ping Tian et al. “A review on image feature extraction and representation techniques”. In: *International Journal of Multimedia and Ubiquitous Engineering* 8.4 (2013), pp. 385–396.

[21] *PySimpleGUI*. 2015. URL: https://github.com/PySimpleGUI/PySimpleGUI.

[22] Malusi Sibiya and Mbuyu Sumbwanyambe. “Automatic Fuzzy Logic-Based Maize Common Rust Disease Severity Predictions with Thresholding and Deep Learning”. In: *Pathogens* 10.2 (2021). ISSN: 2076-0817. DOI: 10.3390/pathogens10020131. URL: https://www.mdpi.com/2076-0817/10/2/131.

[23] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[24] Young-jun Song et al. “Content-based image retrieval using new color histogram”. In: *Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004.* Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004. 2004, pp. 609–611. DOI: 10.1109/ISPACS.2004.1439129. URL: https://doi.org/10.1109/ISPACS.2004.1439129.

[25] Divya Srivastava, Rajesh Wadhvani, and Manasi Gyanchandani. “A review: color feature extraction methods for content based image retrieval”. In: *International Journal of Computational Engineering & Management* 18.3 (2015), pp. 9–13.

[26] Jun Yue et al. “Content-based image retrieval using color and texture fused features”. In: *Mathematical and Computer Modelling* 54.3 (2011). Mathematical and Computer Modeling in agriculture (CCTA 2010), pp. 1121–1127. ISSN: 0895-7177. DOI: https://doi.org/10.1016/j.mcm.2010.11.044. URL: https://www.sciencedirect.com/science/article/pii/S0895717710005352.

[27] A. Zisserman. “Lecture 2: The SVM classifier”. In: 2015. URL: https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf.