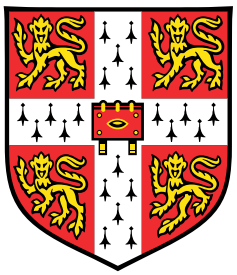
Identifying Looted Artefacts



Sedinam Simpson

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Murray Edwards College September 2024

I would like to dedicate this thesis to my family - Edward Simpson, Gertrude Simpson and Senyo Simpson.

Declaration

I, Sedinam Simpson of Murray Edwards College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. This report contains 12,970 words, excluding declarations, bibliography, photographs and diagrams, but including tables, footnotes, figure captions and appendices. The software used for this report was written by the author, however the code for LSH was adapted from https://github.com/dougian/lsh/blob/master/src/lsh.py. All code is publicly available and can be found on GitHub at https://github.com/sediesimpson/looted-artifacts-project.

Sedinam Simpson
September 2024

Acknowledgements

I would like to thank my supervisors Dr Matthew Johnson and Dr Christos Tsirogiannis - without their constant support and guidance, from a technical standpoint, this would not have been possible. I would also like to thank Dr John Dudley for his support as a course advisor. I would like to thank my family for their unwavering support, love and grace during this period. Whenever I was worried or stressed they stood by me, and helped me through it. Finally, I would like to thank the “best study group to ever exist" - Andrej Jovanović, Clare Heinbaugh, Darius Feher and Elijah Gutierrez. The numerous study and writing sessions carried me through this time. I am grateful for the friendship formed in these moments.

Abstract

There is a need for rapid detection and classification of illicit artefacts within forensic archaeology in order to discover illicit trades. Today this process is slower than necessary, with an expert archaeologist having to manually check if an artefact appears in catalogues more than once lacking a legal provenance. This work is concerned with the design of a duplication detection system to aid in the identification of the provenance of artefacts. We investigate if it is possible to develop a system to identify potential duplicate images in an artefact image database given a new artefact image. We explore avenues of improvement, including computational efficiency of the system and the enhancement of the feature extractor. Exhaustive and approximate search algorithms are employed, specifically radius-based nearest neighbours (RBNN) search and locality sensitive hashing (LSH). A pre-trained ResNet-50 is initially used as the feature extractor. Recall is the prioritised metric of evaluation as the system will be used by an expert who will be able to filter through images and confirm duplicates. It was hypothesised that fine-tuning the ResNet-50 would assist in the performance of the RBNN and LSH, however, this did not prove to be the case. The best performing permutation of the system was the RBNN with a pre-trained ResNet-50. It achieved an average per-class precision of 0.44, an average per-class recall of 0.84 and a mean average precision (mAP) of 0.71.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

Adam Adaptive Moment Estimation

BCE Binary Cross-Entropy

c-ANN c-Approximate Nearest Neighbour

CBIR Content-Based Image Retrieval

CNN Convolutional Neural Network

kNN k-nearest neighbour

LSH Locality Sensitivity Hashing

mAP Mean Average Precision

RBNN Radius-based Nearest Neighbour

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

Chapter 1

Introduction

There is a need for rapid detection and classification of illicit artefacts within forensic archaeology. An estimated 140,000 - 700,000 antiquities, with a monetary value ranging between €64-318 million annually, are traded within Europe, North Africa and West Asia (Brodie et al., 2019). There is however no clear way of determining whether these are legally obtained (licit) or illegally obtained (illicit) antiquities.

The manual task of searching for illicitly traded items takes forensic archaeologists on the order of 1-4 hours per item (Raja and Tsirogiannis, 2023). This laborious work motivates the development of a potential automated solution (Winterbottom et al., 2022). One notes however that trafficked items are normally undocumented, which in turn makes it difficult to employ technological solutions in this domain. A human-in-the-loop is often necessary to determine whether something has been trafficked, and authorities can only intervene once the proof of illicit origin is established (Winterbottom et al., 2022).

A key method that experts use to identify trafficked artefacts is to compare catalogues produced by auction houses and private galleries with images seized from traffickers. This involves manually cropping images from catalogues and storing them into folders labelled as the category to which the items belong. Once these are categorised, they search within the categorised folders to determine if a specific artefact has appeared in a catalogue more than once within a certain time-frame lacking a legal provenance[1]. If the artefact is lacking legal provenance, it is cause for concern and the item is “flagged" as suspicious. Further investigation into seized documents is then conducted to prove its illicit origin. With the aid of machine learning, we can aim to make this process efficient. Thus, the objective of this work is the identification of duplicate images with the same object within a dataset.

Note that from this point onward when referring to the “expert" or “forensic archaeologist", we are referring to Dr Christos Tsirogiannis. Dr Tsirogiannis is a Senior Researcher,

[1] Provenance is the record of ownership of the artefact.

the Head of Research on Illicit Antiquities Trafficking of the UNESCO Chair of Threats to Cultural Heritage and Cultural Heritage related Activities in Ionian University in Greece, and Expert at the Swiss Federal Office for Culture. He will be the main user of the solution developed in this project.

1.1 Research Focus

This work is concerned with the design of a duplication detection system contributing towards the identification of the provenance of artefacts within auction houses and private collection sales. The duplication detection system should be able to identify images of the same object within different catalogues. This is distinct from a general duplication detection system as it is specialised towards the artefacts within a database developed by the expert.

The primary objective is to determine a set of techniques that are well suited to the duplication detection problem and to investigate the characteristics of these techniques, including but not exclusive to recall, precision and query speed. A comparative analysis of different methods will be performed. Three research questions are put forward to formalise the nature of this work:

- **Research Question 1**: Can we develop a system for the task of identifying potential duplicate images in an artefact image database given a new artefact image query?
- **Research Question 2**: Is it possible to determine duplicate images in a computationally efficient way?
- **Research Question 3**: Can the performance of the system be improved through a more powerful feature extractor?

1.2 Motivation

There are multiple efforts that are currently employed by different organisations, such as UNESCO and the International Criminal Police Organisation (INTERPOL) which aid in combating the trafficking of artefacts. INTERPOL specifically has a database of Stolen Works of Art (Patias and Georgiadis, 2023). However, there are few efforts that aid in searching large corpora of artefact images within legal and publicly available catalogues and databases of galleries, museums and auction houses. This presents the opportunity to produce a body of work that analyses different methods of doing so automatically with performance and efficiency in mind.

1.3 Scope

This work is bound such that it has a specific focus. Given more time and computation, one might be able to extend this work beyond the scope. I have made several key scoping decisions in order to ensure the problem remains tractable. First, the exploration and comparison of one exact matching method and one approximate matching method of query images to images within a database will be performed. Performance metrics have been bound to recall, precision and query time and the matching will be performed only on the duplicate dataset provided by the expert.

1.4 Overview of Dissertation

The remaining chapters of this dissertation include:

- **Chapter 2** outlines the problem context at hand, and provides a deep-dive into the data and how it is structured.
- **Chapter 3** is a literature review covering the field of forensic archaeology in this context and the efforts uncovered to tackle repatriation of looted artefacts. The field of image search and applications of image search, alongside a breakdown of feature extraction methods is also discussed. The aim of this is to understand the context in which these methods have been used in the past and how they are applicable to the current domain.
- **Chapter 4** outlines the duplicate identification problem specifically looking at image retrieval, search algorithms, the requirements for the solution, feature extraction and similarity measures.
- **Chapter 5** outlines the step-by-step approach to solving the problem, specifically: the duplicate identification system.
- **Chapter 6** outlines the experiments performed, looking at parameter optimisation and query time experiments.
- **Chapter 7** shows the results and discussion using the methods chosen.
- **Chapter 8** discusses the implementation of fine-tuning a CNN to use as the feature extractor.

- **Chapter 9** discusses the results of the duplication identification system using the fine-tuned feature extractor.
- **Chapter 10** provides a conclusion and motivation for future work.

Chapter 2

Problem Context and Data

We will focus on a categorisation system that an expert in the field of forensic archaeology has adopted in order to track the provenance of artefacts. We looked at this specific system because it was the one at hand, an exemplar in the field, and contains duplicates of the kind we wanted to find. Figure 2.1 below depicts the pipeline or workflow that the expert forensic archaeologist undertakes when uncovering the provenance of an artefact. We seek to understand how one goes from identifying an artefact in a catalogue as one that has been looted.

A Polaroid of a looted vase from South Italy is within the seized images that the expert has on hand as seen in Figure 2.1(a). Figure 2.1(b) is a dealers' image that was captured after the vase was smuggled to Switzerland – again found within the seized images. The vase is then pictured within a Sotheby's auction catalogue on 23 June 1989 in New York without any provenance, as seen in Figure 2.1(c). The same vase is depicted without any provenance on the website of the Metropolitan Museum of Art in New York, immediately after it was acquired at the 1989 Sotheby's auction, as seen in Figure 2.1(d). Finally, the vase is pictured after its return to Italy, as seen in Figure 2.1(e), repatriated to its country of origin.

To aid in identifying artefacts in auction, gallery and museum catalogues, the expert has devised a categorisation system. He and his volunteers manually crop artefacts within the catalogues and label them. They place each captured artefact into a category manually. When looking at new catalogues, they refer to these cropped images to match current catalogue artefacts to artefacts they may have seen in the past. As a result, they have a dataset consisting of roughly 63,500 images of artefacts from 2011-2020, collected from hundreds of catalogues produced by 12 different auction houses and galleries (Raja and Tsirogiannis, 2023). An example of a categorisation is: Vases $\rightarrow$ Italian Vases $\rightarrow$ Kraters [1]. The same categorisation takes place with images seized in the hands of traffickers. Once an object lands in certain

[1] A krater is a vase or jar that has a large round body and a wide mouth.

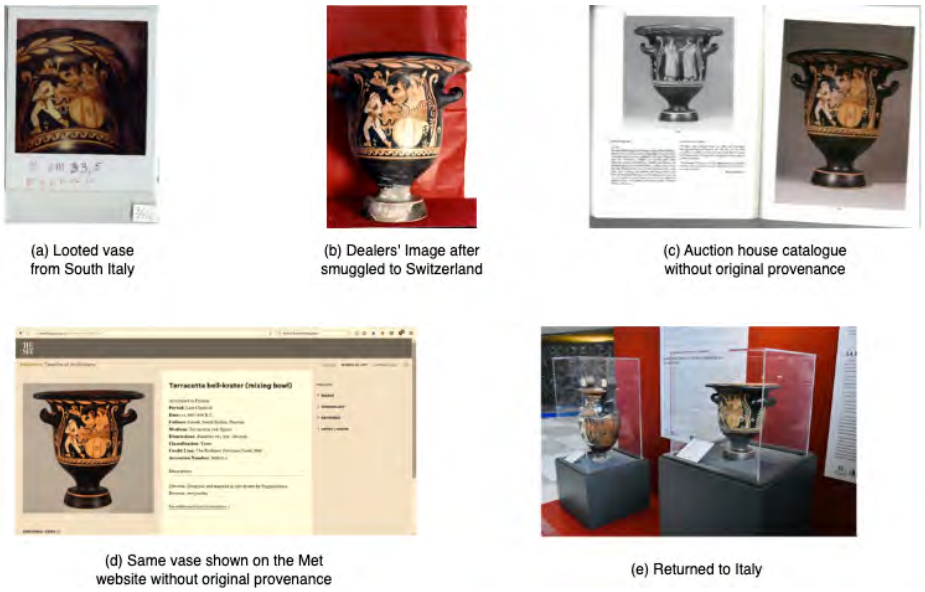


Fig. 2.1 An example workflow of the expert forensic archaeologist.

category or sub-category, the expert searches within the category in order to determine if there are matches to any of the images from the traffickers. If the image of the trafficked artefact is found to match one of the catalogue images, this can trigger the process of querying its provenance.

Recall the categorisation: Vases $\rightarrow$ Italian Vases $\rightarrow$ Kraters. The hierarchy of the categories is important. Since the same object can qualify for two or more categories or subcategories, objects may exist in many permutations of the hierarchy.

- The same item appears within the same category and subcategories.
- The same item appears within the same category but a different subcategory.
- The same item appears in entirely different categories and subcategories.

In Figure 2.2 the item appears in the same (sub)category. In Figure 2.3, the item shares the same subcategory but have a different higher-level category.

This poses a challenge. Take the performance evaluation of a solution as an example. The accuracy of the model is dependent on the hierarchy of the category. In Figure 2.3, if we choose the subcategory as the ground truth, it is straightforward to compare. However, if we choose the higher-level category, they will be assigned different labels and therefore assumed to be different objects. While the underlying performance of the solution is in reality the same, the choice of category level can have an impact on the results. Formally, this is a multi-label problem – objects can be assigned many labels. This is particularly relevant as

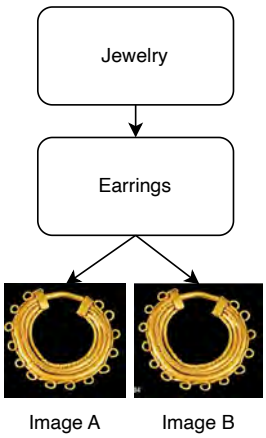


Fig. 2.2 The same object can exist within the same category and sub-categories, and can be named differently.

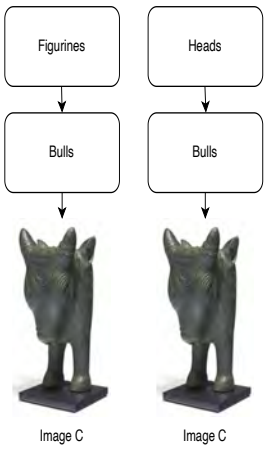


Fig. 2.3 The same image of an object can be reproduced within completely different categories.

we use a CNN as a feature extractor in our solutions. The multi-label nature of the data has to be taken into account in order to train the model. This is discussed in depth in Section 8.

Another challenge in the dataset is images with multiple artefacts. An example is shown in Figure 2.4. While these have an official categorisation, it is ambiguous in nature. As part of data cleaning, we remove all such cases from the dataset.

Fig. 2.4 Multiple tools captured within a single image.

2.1 Ground truth determination

A subset of the dataset has been crafted consisting of the duplicate images within said dataset. It was constructed by the forensic archaeologist who went through the manual exercise of identifying duplicates and organising them. There are 301 objects that are duplicated and the number of duplicated images for each object ranges from 1 to 14. We have two definitions for duplicate:

1. **Image Duplication**: The image itself is an exact copy such that more than one of the same image exists within the dataset. That is to say that the same image was used in different catalogues by the same seller.

2. **Object Duplication**: The same object is depicted in a different images. This is the common case as the object appears in catalogues of different sellers (auction houses and/or dealers' galleries).

Image duplication can be considered a subset of object duplication, where the object is captured under the exact same set of conditions. In the case of object duplication as defined above, images will vary. The photograph of the object may be taken with different angles, distances, perspectives and be taken under different lighting. The typical image attributes such as brightness, contrast and blur may vary too. Additionally, as we mentioned earlier, the resolutions will vary as well. All of these contribute to the challenge of object duplication.

An example of image duplication is shown in Figure 2.2. Figure 2.5 shows an example of object duplication. The same statue is depicted in catalogues of different sellers. There are visual difficulties with the statue being captured from different angles. Other examples exist such that only portions of the artefact are duplicated as seen in Figure 2.6. In the first three images from the left, the top part of the Candelabrum is duplicated, and in the last three images that the whole Candelabrum is duplicated. Our hypothesis is that the first three images will be matched together, and the last three image will be matched together. However, matching across the first three and last three may prove to be difficult. Recall that this work is motivated by the problem of legal provenance. It is to our benefit to identify every instance of a duplicate, as it gives a fuller picture of the history of the object. Therefore, the ideal solution would be able to match the first three images with the last three, despite the difficulty in doing so.

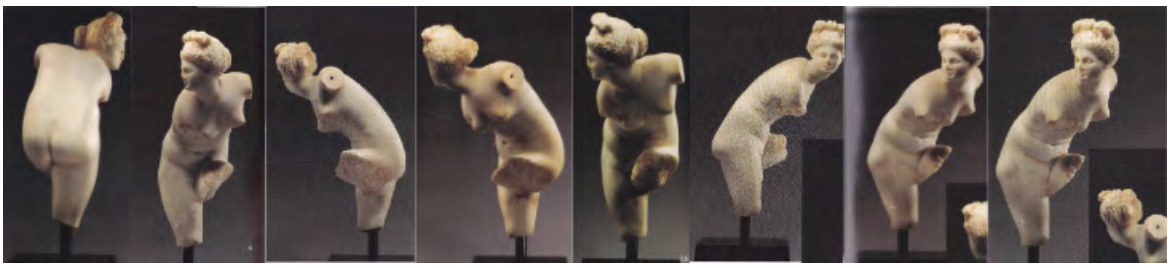


Fig. 2.5 Eight instances of the same female statue in different images in the dataset i.e. it has been duplicated.

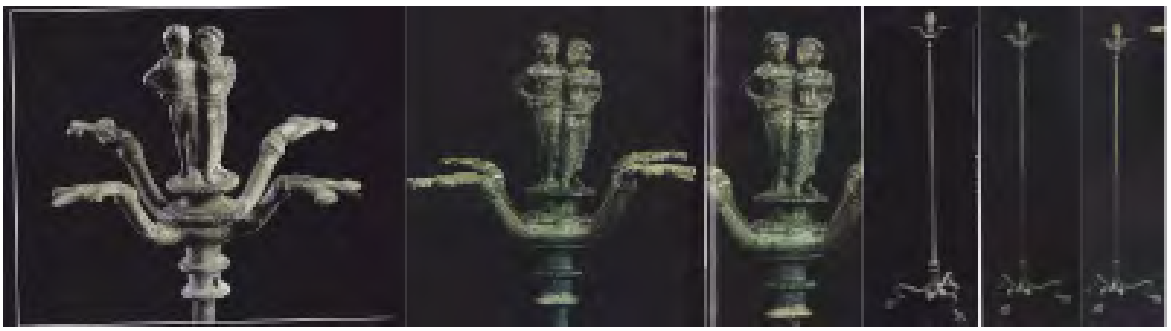


Fig. 2.6 Five instances of the same Candelabrum within the dataset.

Chapter 3

Background

This chapter reviews literature pertaining to image search. It focuses on four main topics: exact and approximate matching methods, background subtraction, feature extraction, and the state of computer vision in the field of forensic archaeology.

3.1 Image Search

There are multiple methods and applications of image search. Here, we describe two approaches - an exact matching approach and an approximate matching approach.

3.1.1 Exact Matching

Chen et al. (2020) describe a k-Nearest Neighbours (kNN) search method where a distance metric is defined, a similarity metric is computed between a query image and every other image in the database, and a query image is accepted as being the same as an image in the database if said distance metric is below a certain, pre-defined threshold. This is therefore known as an exact matching approach. There are many distance methods that can be used - for example, Euclidean, Cosine and Minkowski to name a few. kNNs are widely used and have seen applications within facial recognition systems (Parveen and Thuraisingham, 2006), biometric authentication (Hu et al., 2008) and have been used in stock market predictions (Chen and Hao, 2017). Since exact matching occurs, kNNs suffer from the curse of dimensionality - this is the inability of the kNN to distinguish between candidate points since the distances between near and far candidate points from the query images become approximately equal as dimensionality increases (Kouiroukidis and Evangelidis, 2011). One way to combat this is to use a c-approximate nearest neighbour search (c-ANN) method.

3.1.2 Approximate Matching

c-ANN methods result in points where the distance between the candidate point and the query point is at most c times the distance of the query point to its nearest neighbors. Kd-Trees, ball-trees and locality sensitivity hashing (LSH) (Shakhnarovich et al., 2006) are examples of c-ANN methods. Specifically considering LSH, we note that it maps high-dimensional data to low-dimensional representations by using random hash functions. Within each hash function, a point is assigned to a hash bucket. The closer data points in high-dimensional space are mapped to the same hash bucket within low-dimensional space with a high probability. The converse also applies i.e. the farther data points in high-dimensional space are mapped to the same hash bucket within low-dimensional space with a low probability.

LSH is known for its sub-linear query performance with respect to dataset size as well as its relatively good query accuracy. LSH has been used for music retrieval (Yu et al., 2009), document processing (Bagban and Kulkarni, 2020), image and video processing (K et al., 2023) and to detect malware (Yousefi-Azar et al., 2018). It is also used in plagiarism and near-duplicate detection which is similar to the application presented in this work (Jafari et al., 2021).

3.2 Background Subtraction

A common problem that occurs within computer vision systems is the tendency to learn or use non-essential information to match images. Within the context at hand, the images have plain backgrounds with similar colouring - red, white and black. As there are similar backgrounds in images, the algorithm at hand could match on this and not the object itself. To combat this, background subtraction is applied to separate the foreground and background. We see manual methods of background subtraction using masking which is the method employed in this work, alongside more complex methods such as Mixture of Gaussian subtraction (Zivkovic, 2004). There are many other methods of background subtraction such as mean filtering, frame differencing, and using a running Gaussian average (Sobral and Vacavant, 2014).

3.3 Feature Extraction

For an image search or content-based information retrieval approach, it is important to have good feature vector representations of your images. More recent years have seen the rise of Convolutional Neural Networks (CNNs) for feature extraction. CNNs are good feature

extractors as they are able to exploit the spatial relationships within images. Additionally, they create translation-invariant features since they use shared weights across the whole input space. Although CNNs are largely used for image classification, we see them used in a variety of applications such as object detection, facial recognition, speech recognition and document analysis (Ersavas et al., 2024). Specifically working with pre-trained CNNs, we note that they have the ability to capture low-level image information before any learning (Ulyanov et al., 2020). There have been many developments in CNNs over the years and the Residual Network (ResNet) is particularly desirable due to its ability to learn significantly deeper architectures than its counterparts. The architecture up until prior to the last fully connected layer can also be used to create feature embeddings that represent the images in some meaningful vector form. These in turn can be used for a variety of tasks such as image search tasks. The idea here is that semantically close images usually have close representation on the last layers.

3.4 Computer Vision in Forensic Archaeology

The use of machine learning methods within forensic archaeology is extremely limited. Winterbottom et al. (2022) has used it to classify artefacts into type, specifically classifying multiple instances of the same object as the object itself. They achieve an accuracy of 72% and note an improvement on classification accuracy up to 83% when the number of instances per object is increased. The authors run further experiments on classifying the images into four different subcategories: Oriental, Egyptian, Fulling Mill and Castle. Maaten et al. (2006) create a content-based information retrieval (CBIR) system to find drawings of artefacts that are most similar to drawings or photographs of excavated historical glass. To do this, the authors use a k-nearest neighbour search to match the shape descriptors extracted from the artefacts to the drawings or photographs of the excavated glass. The authors do not quote any success metrics of this approach but note a shortfall of the system is its inability to identify glasses which are damaged and have missing parts as a result. Secondary to this, the authors create a coin classification system to classify Dutch early-medieval coins, and quote a test accuracy of 78%. Boon et al. (2009) describe a coin recognition system where a similarity measure between coins that is invariant to rotations is computed and subsequently perform 1-nearest neighbour classification to match a query coin to an already existing coin. They report a high accuracy of 92% on a modern coin dataset and a relatively low accuracy of 43% on an archaeological dataset. This difference in accuracy is described by wear-and-tear as well as morphological variations of these ancient coins.

Chapter 4

Duplicate Identification

This work is motivated by the desire to aid in the determination of legal provenance of artefacts. The nature of this problem requires finding duplicates in a large database of known images. Formally, this is a problem of image retrieval. This chapter provides a detailed discussion of the subject matter and the challenges faced therein.

In Section 4.2, we propose a set of requirements that our search algorithm must fulfil. These are based on the characteristics of the problem at hand, discussed further in the section. The requirements motivated the use of two solutions: radius-based nearest neighbours and locality sensitive hashing. These are covered extensively in Chapter 5. The remainder of the chapter pays attention to feature extraction, an important step in our image search solution.

4.1 Image Retrieval

Image retrieval is the process of retrieving images from a large collection of images based on an input query. This input may be an image itself but may also be metadata such as keywords, captions or titles describing the image. In the context of this work, an image is used as the query and the retrieval is successful if duplicate images are returned. This constitutes a search problem.

Search algorithms are broadly categorised as either exhaustive or approximate. In an exhaustive search, the input image is compared against every other image in the database. This method may provide good results but is not well suited for larger dataset sizes (Makkar et al., 2017). For a single input image, the time complexity is linear (or in big O notation, $O(n)$). Approximate search aims to improve on this, through an approximation: for a given input image, which images are the most likely to be similar to it. The idea is that we can reduce the number of comparisons that need to be made by partitioning the search space and only comparing within the partition. As a trivial example, imagine we had various images

with different backgrounds: pink, red and blue. Our dataset is then partitioned into these categories. If we had an input image with a pink background, we can limit the search to only the images with pink backgrounds. In this work, we develop solutions using both methods. The exact method aims to answer Research Question 1 while the approximate method aims to answer Research Question 2.

Another dimension of the search problem is the representation of the data. In the most trivial case, when searching images, the images themselves can be compared directly. Another option is to use a feature embedding. A feature embedding maps a high dimensional input into a lower dimensional output, with the output dimensions representing some features of the input. For example, assume an image can be mapped into a four dimensional embedding with the features: shape, colour, brightness, blur. Instead of comparing the images, these four dimensional embeddings are compared. This not only helps with processing speed but can be more robust than comparing images directly (Jiang, 2009).

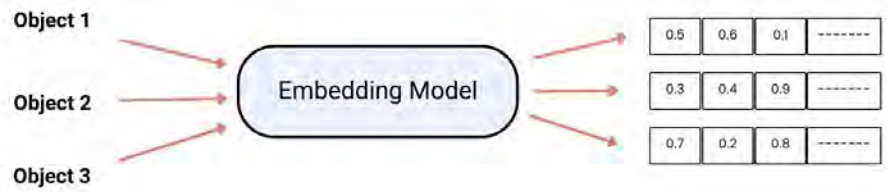


Fig. 4.1 A high level overview of objects transformed into a feature embedding.

Feature extraction is a common part of the processing pipeline in computer vision tasks. Traditionally, methods such as scale-invariant feature transform (SIFT) are used to do feature extraction (Lowe, 2004). Increasingly, convolutional neural networks (CNN) are used to do feature extraction (Li et al., 2022). They have proven themselves capable (Krizhevsky et al., 2017), being used in a plethora of computer vision tasks, from image classification (Krizhevsky et al., 2017) to self-driving cars (Bojarski et al., 2016). Given their capability, we use a CNN to create embeddings. This is discussed further in Section 4.3

Throughout we've mentioned that image retrieval returns similar images but have taken no measure to define similarity. Similarity is dependent on the problem at hand. There are many measures of similarity. For an image, a simple measure is a pixel-wise difference

such as the mean squared error (MSE). The more pixels that are the same at their respective locations, the smaller the difference. The structural similarity index measure (SSIM) attempts to measure the similarity of two images based on their perceptual difference. That takes into account attributes such as contrast and texture.

Above, we mentioned that our solution uses embeddings of the images. One way to measure the similarity of embeddings is the cosine similarity. It measures the similarity between two vectors by measuring the angle between them. Ideally, duplicate images will have vectors that lie on top of each other, or marginally close to each other, whilst those that are different from each other lie in completely different spaces and/or directions. Other measures exist such as the Euclidean distance. Similarity measures pertaining to our problem are discussed in Section 4.4.

4.2 Search Algorithms

We are concerned with the provenance of an object. Thus, it is beneficial to retrieve every instance of the object so that an accurate record of its movement can be established. For example, if an object has moved through galleries A, B, C and D, it would be to our advantage to know about every instance, assuming we have them all captured in our dataset. If we only retrieve images A and B, we would not be able to construct an accurate history. This may lead to missing illicit trade activity. Therefore, we are concerned with the retrieval of all images, at the expense of retrieving additional incorrect images. In other words, we care about having high recall (finding every instance of a given image) at the expense of high precision (all the found images are a match to the query image).

To decide which exhaustive and approximate search methods to use, a number of requirements were proposed and different methods were evaluated against these requirements. For exhaustive matching, we have one requirement:

Requirements	Description
RQ1	Able to capture any number of duplicate images

Table 4.1 Requirements checklist for exhaustive search methods.

Exhaustive search requires a comparison of the input to every other image in the database. This method guarantees finding the exact nearest neighbours. A common way of approaching this would be to use k-nearest neighbour (kNN) (Rottok et al., 2018). However, requirement RQ1 states that we need to be able to capture any number of duplicate images. This is necessary as we do not know how many duplicates there are for a given query image.

Additionally, it will only increase over time. Since k-nearest neighbours only returns at most k images, we can only guarantee to ever find k duplicates. Thus, it does not fulfil our criteria and is excluded.

Another approach is to return all the nearest neighbours within a radius. Assume we have a sufficiently good feature extractor that maps duplicate objects on top of each other. No matter how many new duplicates we add to the dataset, they will be mapped onto the others. With an optimised radius, we are able to retrieve all duplicate images. This fits our criteria. We use radius-based nearest neighbour (RBNN) search as the algorithm for exhaustive search. It is discussed further in Section 5.2.

The requirements for approximate search are shown in Table 4.2.

Requirements	Description
RQ1	Able to capture any number of duplicate images
RQ2	Light on computational resources
RQ3	Work well with high dimensional data
RQ4	Work well with large dataset sizes

Table 4.2 Requirements checklist for approximate search methods.

Requirement RQ1 is shared with approximate search for the same reasons discussed earlier. Requirement RQ2 is qualitative in nature. It should be an algorithm that can be run on a moderately powerful computer, making it easy to use in any environment. Additionally, it removes the need for expensive large scale computing needs that have the potential to make the solution expensive to use. Requirement RQ3 and RQ4 speak to the scaling characteristics of the solution. It should be able to handle high dimensional, large scale data. This makes the solution usable applied to larger datasets such as the INTERPOL database of Stolen Works of Art mentioned in Section 1.2.

There are a number of potential candidates for approximate search:

- KD-Tree based nearest neighbours
- BallTree based nearest neighbours
- Local Sensitivity Hashing (LSH)
- Hierarchical Navigable Small World (HNSW)

KD-Tree and BallTree based nearest neighbour search use efficient data structures to find the approximate neighbours. While BallTree is more robust against the problem than KD-Tree, both of them suffer from the curse of dimensionality (Peng et al., 2023) – as the

number of dimensions increase, their performance degrades. KD-Tree in particular degrades into exhaustive search under high dimensions (Tiwari, 2023). Additionally, they are not great at handling large datasets relative to LSH and HSNW. Requirements RQ3 and RQ4 are not met by KD-Tree and BallTree based nearest neighbours.

Hierarchical Navigable Small World (HSNW) is a popular algorithm, used in a number of vector databases. It has been shown to outperform other approximate search algorithms. However, it is memory intensive at scale (Malkov and Yashunin, 2016). As we wish to make the solution light on computational resources, as stipulated in Requirement R2, we exclude this from our candidates.

Locality Sensitive Hashing (LSH) meets the set of requirements. It is a well established algorithm and has seen usage by many large scale companies such as Google and Netflix (Chen et al., 2019). The algorithm is discussed further in Section 5.

4.3 Feature Extraction

As mentioned in Section 4.1, the solution proposed uses feature embeddings. The feature embeddings are compared against one another when searching for duplicate images. To generate these embeddings, we use convolutional neural networks (CNN). To perform well at computer vision tasks, such as image classification, a CNN must learn a representative set of features of each class of image. Due to their outstanding performance at these tasks, they have been proven to be good at feature extraction. They are used as such throughout the research landscape (Barbhuiya et al. (2021), Jogin et al. (2018), Liu et al. (2021a)).

We use the ResNet-50. It is fit for purpose having been used in a variety of research such as autonomous driving (Chen et al., 2021) and medical analysis (Sarvamangala and Kulkarni, 2022). Improving upon the model choice is left for future work. To use this model as a feature extractor, we remove the last fully-connected layer. In doing so, the model outputs the features used to classify the image. This feature embedding is then used downstream in our solution. Lastly, instead of training the model from scratch, we used the weights from a pre-trained model. This model was trained on ImageNet, a large image database that has served as an industry standard in computer vision research (Deng et al., 2009). Additional to the pre-trained approach, we also used a fine-tuned version which involved fine-tuning it to the artefacts at hand. This approach is further elaborated on in Section 8.

4.4 Similarity Measures

Similarity measures aid in determining how closely related images or embeddings are to each other. There are multiple similarity measures including Euclidean distance, cosine similarity and Hamming distance. For RBNN, Euclidean distance or cosine similarity can be used. Euclidean distance is a straight line distance between two embeddings in Euclidean space whilst cosine similarity is the cosine of the angle between two vectors which focuses on the orientation of the vectors more so than the magnitude of them. Cosine similarity is preferred in this instance over Euclidean distance because of its magnitude independence and the fact that it is better suited towards high-dimensional data (Qian et al., 2004).

In Figure 4.2, images close to each other have vectors in a similar direction.

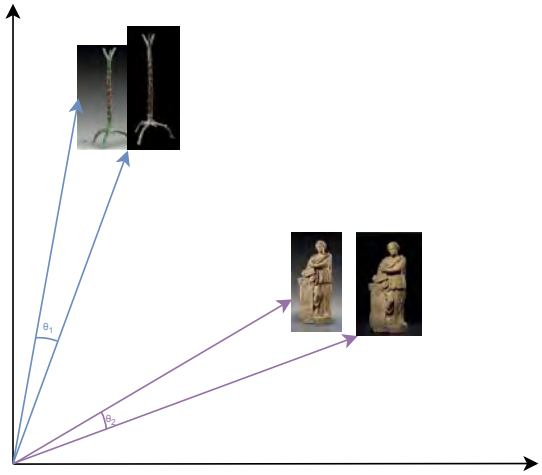


Fig. 4.2 Duplicate images will lie close to each other in embedding space whilst those that are different will lie further away from each other. The angles between the blue lines and between the purple lines are θ_1 and θ_2 respectively.

According to the cosine similarity, these will be similar to each other. This is the measure used in our evaluation. Formally, we define the cosine similarity between a feature vector z and a query vector q as follows:

$$\text{Cosine Similarity}(\mathbf{z}_i, \mathbf{q}_j) = \frac{\sum_{k=1}^{d} z_{ki} q_{kj}}{\sqrt{\sum_{k=1}^{d} z_{ki}^2} \cdot \sqrt{\sum_{k=1}^{d} q_{kj}^2}} \tag{4.1}$$

where

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{d1} & z_{d2} & \cdots & z_{dn} \end{bmatrix} \quad \text{and} \quad \mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ q_{d1} & q_{d2} & \cdots & q_{dm} \end{bmatrix} \tag{4.2}$$

Note that i indexes the feature vectors and j indexes the query vectors.

A cosine similarity between two vectors can range between -1 and 1. A cosine similarity of 1 would mean that the vectors are identical, a cosine similarity of 0 would mean that the vectors are orthogonal and there is no similarity between the vectors and as such the images. Finally, a cosine similarity of -1 would mean that the vectors are diametrically opposed. In our scenario, we want the cosine similarity for duplicate images to be close to 1.

As a complimentary measure, we can look at the cosine distance which is a linear transformation of the cosine similarity, defined as follows:

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity} \tag{4.3}$$

This in turn ranges from 0 to 2, where 0 is indicative of identical vectors (cosine similarity is 1), 1 is indicative of orthogonal vectors (cosine similarity is 0) and 2 is indicative of opposite or diametrically opposed vectors (cosine similarity of -1).

Chapter 5

Duplicate Identification System

In Chapter 4 we discussed the components of an image retrieval system. A number of design decisions were taken that shaped the solution. Specifically:

- A pre-trained ResNet-50 convolutional neural network is used as a feature encoder.
- For exhaustive search we use radius-based nearest neighbours (RBNN).
- For approximate search, we use locality sensitive hashing (LSH).
- The similarity measure of choice is the cosine similarity.

This chapter provides insight into the solution developed. Section 5.1 gives a high level overview of the architecture. Section 5.2 and 5.3 give a detailed account of the search algorithms, RBNN and LSH respectively.

5.1 System Overview

The high level architecture of the system developed for performing duplicate identification is pictured in Figure 5.1.

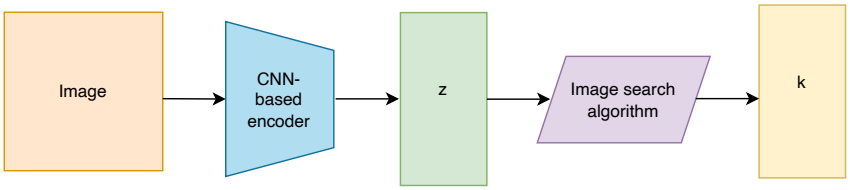


Fig. 5.1 Image Search Architecture. Note that this is the pipeline for one image. z represents the feature vector of the image, and k represents a vector of k nearest neighbours.

First, the image is passed through a CNN-based feature encoder. As mentioned, this is a pre-trained ResNet-50 (He et al., 2015). Once passed through the encoder, we obtain a set of feature vectors or embeddings denoted as z. These features are then fed through an image search algorithm which compares the embeddings against the embeddings database (derived from the original image database by passing them through the same CNN). To reiterate, for exhaustive search, the image search algorithm of choice is radius-based nearest neighbours search and for approximate search it is locality sensitive hashing. Finally, it results in k nearest neighbours which are considered to be duplicates – images containing the same object as in the query image.

5.2 Radius-based Nearest Neighbours

Radius-based nearest neighbours (RBNN) is a nearest neighbours algorithm that treats all points within a given radius as neighbours. It is exhaustive in nature and thus is exact – it is guaranteed to return the exact nearest neighbours, provided they are within the radius.

RBNN can be divided into three distinct steps.

1. Let the matrix $\mathbf{Z}$ depict the feature vectors where each row z_i is the feature vector of the i_{th} image.

2. Compute the distances based upon cosine distance as outlined in Equation 4.2 and 4.3.

3. We then compute a radius-based nearest neighbour search which means that given the query vector q_j and the radius r, all the points z_i in $\mathbf{Z}$ that satisfy the equation are returned:

$$\text{distance}(q_j, z_i) \leq r \tag{5.1}$$

The challenge in this method is choosing the correct radius. Recall our objective is to optimise for recall, since we care about identifying every instance of an object in the database. Trivially this can be achieved by making the radius arbitrarily large. However, this is unhelpful for the archaeologist as it would return far too many false positives. Instead, trading-off recall to the appropriate degree to improve the precision is desirable. The experiments done to find the optimal radius are discussed in Section 6.3.1.

A disadvantage of radius-based nearest neighbours is its time complexity. For a given input image, the time complexity is linear. As the dataset grows, the time to answer a given query grows accordingly as it must perform a comparison of the query against every object in the dataset. This is prohibitive at scale. An approximate search method such as locality sensitive hashing provides an answer to this problem.

5.3 Locality Sensitive Hashing

Locality sensitive hashing (LSH) uses hashing to improve performance. Hashing maps high-dimensional data to a low-dimensional representation through the use of a hash function. Of importance is that a hash function can map data to a fixed-sized set of values. For a trivial example, take the modulo function. The modulo function returns the remainder after a division. Assume we have a modulo function with a divisor of four. The output of the first nine numbers is as follows:

$$\begin{aligned} 0 &\mod 4 = 0 \\ 1 &\mod 4 = 1 \\ 2 &\mod 4 = 2 \\ 3 &\mod 4 = 3 \\ 4 &\mod 4 = 0 \\ 5 &\mod 4 = 1 \\ 6 &\mod 4 = 2 \\ 7 &\mod 4 = 3 \\ 8 &\mod 4 = 0 \end{aligned} \tag{5.2}$$

This is a simple hash function that maps every whole number to one of $[0,1,2,3]$. The size of the output range is fixed to four. Assume that each value output by the hash function is used to group values together and that values that are hashed to the same output are similar. To find similar values to any input value, we take the hash, and consider all the values with the same hash as similar. With the modulo 4 function above, the entire input space of whole numbers is mapped to just four hash values, effectively splitting it into four partitions. As the dataset size increases, the computational time for the algorithm does not increase linearly alongside it when searching for similar values. LSH leverages the same principle, applied to high-dimensional data. LSH partitions the dataset and each partition is assigned an integer value which in turn is used to retrieve the images in said partition. This makes it ideal for databases as it is possible to create an indexed table that maps the partition ID to the image ID. Locality sensitive hashing works with many different hash functions. The original algorithm proposed using random projections (Gionis et al.). First, a random hyperplane is generated. Then, it is determined whether a data point lies on the positive or negative side of the plane. This can be done by taking the dot product of the data point and the normal vector of the hyperplane. Data points in the same direction as the normal vector will be positive and vice versa. The positive side of the hyperplane is assigned 1 while the negative side is assigned

0. This is repeated for N hyperplanes. This generates a binary vector of length N. Through this method, a feature embedding is hashed into a binary vector. These effectively serve as a bucket, where each hash will have zero or more items in it. Figure 5.2 shows how hashes are determined by the location of the hyperplanes.

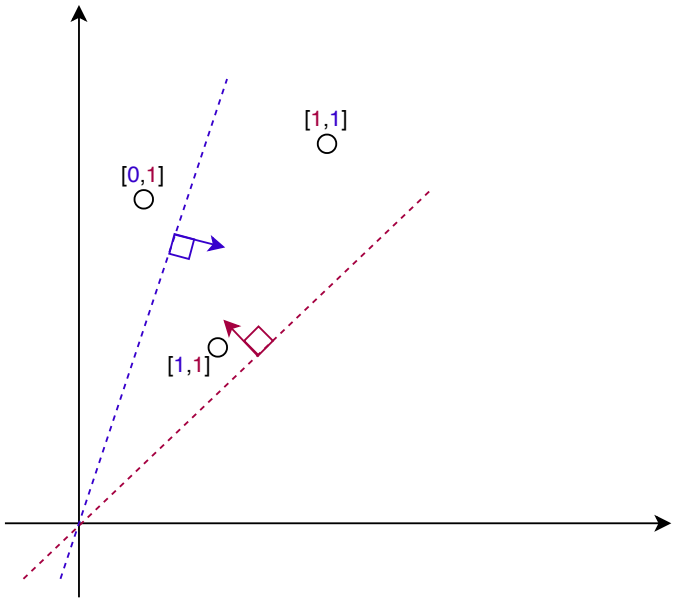


Fig. 5.2 A set of data points with their hashes determined by their location respective to the hyperplanes.

Finally, we use the Hamming distance to determine the neighbours for a given input. The Hamming distance is the bit difference between two binary vectors. All images within the specified Hamming distance are treated as neighbours. It follows that the smaller the Hamming distance, the more alike two images are. Figure 5.3 is a visual representation of calculating the Hamming distance. The emphasised blocks are locations where there are bit differences. In this case, the Hamming distance is 3 as there are three bit differences between the binary vectors. Once we have all the images within the specified Hamming distance, they can be ranked using the cosine similarity between the original embeddings.

For a formal treatment:

1. Create a hash function where similar binary vectors are mapped to the same hash bucket with a high probability. To do so we generate a random vector r from a Gaussian distribution. For a point z, we define the hash function as follows:

$$h(z) = \text{sign}(r \cdot z) \tag{5.3}$$

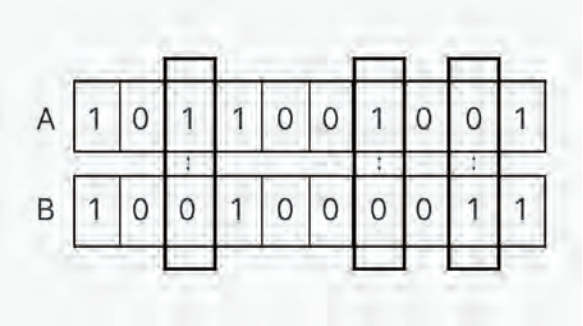


Fig. 5.3 A visual representation of calculating the Hamming distance between two binary vectors.

$r \cdot z$ is the dot product between the vectors which tells us the similarity between the vectors. The equation will output either +1 if the dot product is positive or -1 if the dot product is negative - this determines the side of the hyperplane the point z will lie on. Similar vectors are likely to lie on the same side of the hyperplane and therefore have the same hash value. In practice, we use multiple random hash functions to increase the chances of capturing similarity. To do so we generate c random vectors $r_1, r_2, ..., r_c$ and the hash function will generate a binary vector of length c. Note that identical or similar vectors will map to identical or similar binary vectors:

$$h(z) = \text{sign}(r_1 \cdot z)\text{sign}(r_2 \cdot z)...\text{sign}(r_c \cdot z) \tag{5.4}$$

2. Each vector v_i is hashed using the hash function $h(z)$ to give vh_i. Vectors that are hashed to the same value are grouped together under the hash.

3. Assume we have a query vector q. It is hashed using the same hash function $h(z)$ to give qh.

4. The Hamming distance is computed for the hash qh to every other hash, vh_i.

5. For a Hamming distance, d, values v_i are considered neighbours if they are grouped within a hash where:

$$\text{Hamming}(qh, vh_i) \leq d \tag{5.5}$$

Similar to radius-based nearest neighbours, the challenge posed is to find the optimal Hamming distance in which we consider all images as neighbours. Section 6.3.2 covers the methodology used to find the optimal Hamming distance.

Chapter 6

Experiments

This chapter lays out the experiments performed in order to answer the research questions stated in Section 1.1. The experiments cover parameter tuning of the various models and algorithms, determining their performance on test data and investigations into their scaling characteristics.

6.1 Data Preparation

This section outlines the dataset used throughout parameter tuning and evaluation.

6.1.1 Dataset

This dataset comprises of images containing a variety of artefacts. These include statues, vases and figurines, amongst others. Each image is labelled by the type of artefact pictured in it. Images with the same label are considered duplicates – the artefact in it is the same across the images. The dataset is divided into a train and test set. The train set is 80% of the dataset, with the test set making up the remaining 20%.

To ensure the dataset is of good quality, a number of transformations are applied to it before use: removal of images, background subtraction and resizing and normalisation. The sections following detail these transformations.

6.1.2 Removal of Images with Multiple Objects

There are some images that have multiple objects within them, as shown in Figure 2.4. These are ambiguous in nature. We manually identified images that included multiple objects and removed them from the dataset.

6.1.3 Background Subtraction

After a first pass at using the feature embeddings and their cosine similarity, we observed a failure mode in which matches were predominantly determined by their background colour. To solve this, we used background subtraction. We used manual masks as the background colours were standard across board: red, white and black. Specifically, the input image was converted to a hue, saturation and value format and colour ranges for red, white and black were defined as follows:

- (0, 50, 50) to (10, 225, 225) for one range of red.
- (170, 50, 50) to (180, 255, 255) for another range of red.
- (0, 0, 0) to (180, 255, 30) for black.
- (0, 0, 200) to (180, 20, 255) for white.

Masks were then generated from these ranges and combined to form a mask that covers all these colours. We inverted this mask so that it represented the areas of the image that are not these colours i.e. the foreground of the image, and extracted the foreground using this mask. We then converted the image back to RBG format for further processing.

An illustration of this is seen in Figure 6.1. On the left is the original image with a red background, and the resultant foreground image on the right, after the red background has been removed. Figure 6.2 and 6.3 display the resultant nearest neighbours prior to and post background subtraction, respectively. When cosine similarity between the images was computed, one notes the that images were matched based on the red background as seen in Figure 6.2. Once the background subtraction masks are applied to all the images, the resultant nearest neighbour changes as seen in Figure 6.3.

Fig. 6.1 Original image (left) and foreground image (right) after background subtraction applied.

Fig. 6.2 Nearest neighbour match prior to application of background subtraction.

Fig. 6.3 Query Image and nearest neighbour after background subtraction applied.

6.1.4 Image Resizing and Normalisation

Images are resized to 224 x 224 pixels so as to be compatible with the ResNet-50. As the pre-trained ResNet-50 was trained on ImageNet, we normalise images to the same specification. Images are normalised using the mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$, applied to each colour channel respectively.

6.2 Evaluation Metrics

There are three metrics used to evaluate the duplicate identification system: recall, precision and mean average precision (mAP).

Recall measures the ability of the system to find all duplicates within the dataset. The precision measures how correct the set of retrieved images are. These are both useful metrics in context of our problem. We want to recall all duplicate object images within our dataset, as we care about understanding the provenance of said objects. However, for the sake of the operator, we want those results to be as precise as possible – the retrieved images should only be duplicates.

Recall is defined as the fraction of relevant images that are retrieved. That is, from the total relevant images in the dataset, the number of them that were retrieved. Mathematically, it is denoted as follows:

$$\begin{aligned} \text{Recall} &= P(\text{retrieved} \mid \text{relevant}) && (6.1)\\ &= \frac{\#\,(\text{relevant images retrieved})}{\#\,(\text{relevant items})} && (6.2)\\ &= \frac{\text{TP}}{\text{TP}+\text{FN}} && (6.3) \end{aligned}$$

where TP is True Positives and FN is False Negatives. Precision on the other hand is defined as the fraction of relevant images among the retrieved images. That is, of the images retrieved, the number of them that are correct. Mathematically, it is denoted as follows:

$$\begin{aligned} \text{Precision} &= P(\text{relevant} \mid \text{retrieved}) && (6.4)\\ &= \frac{\#\,(\text{relevant images retrieved})}{\#\,(\text{retrieved items})} && (6.5)\\ &= \frac{\text{TP}}{\text{TP}+\text{FP}} && (6.6) \end{aligned}$$

where TP is True Positives and FP is False Positives.

The mean average precision (mAP) is used to measure the quality of retrieved images, according to their rank. Once we retrieve images, they would be shown to the operator in order of best to worst match. The idea being that even if precision is low overall, the duplicate images would be present in the first K images (where K is the number of duplicates of that object). If the mean average precision is high, it indicates the order of our results is good, whereas low means the duplicate images are scattered throughout the retrieved images.

mAP determines the average of precision scores across all recall levels for each query image and then averages these values across all queries. Mathematically, it is denoted as:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \tag{6.7}$$

where AP is the average precision of query i and is the area under the precision-recall curve. AP is defined as:

$$\text{AP} = \frac{1}{\text{RI}}\sum_{k=1}^{n} P(k)r(k) \tag{6.8}$$

where RI is the number of relevant images for query i, n is the total number of queries, $P(k)$ is the precision at k, $r(k)$ is the relevance of the k_{th} retrieved image. $r(k)$ is an indicator function that is 0 if the image retrieved is not relevant, and 1 if it is relevant.

These metrics are used in optimising the parameters of the search algorithms and for the evaluation of the duplicate identification system.

6.3 Parameter Optimisation Experiments

Both radius-based nearest neighbours (RBNN) and locality sensitive hashing (LSH) have parameters that require tuning to perform optimally. For RBNN, it is the radius parameter. For LSH it is the Hamming distance. The optimisation of these parameters is driven by a trade-off that exists between precision and recall. Take RBNN for example. A radius that is too small will miss the appropriate duplicates and one that is too large will include images that are not duplicates within the resultant set. While we have a tolerance for incorrect images being retrieved, it is still in the best interest to reduce the number of such cases.

6.3.1 Determining the radius for RBNN

The radius determines the points considered as neighbours for a given query. They are all the points that lie within the radius.

To determine the optimal radius in the RBNN method, we generated a distribution of distances to the query images on a set target recall of 90%. The process is as follows:

1. For a given query, the cosine distances are calculated to every other embedding in the database.
2. The cosine distances are sorted from closest to furthest.
3. We loop through each distance (which has a known label), calculating the recall. For context, we know the number of instances of a given query in the database. As we loop through, if a neighbour is a duplicate, it is tallied in our calculation. As more duplicates are identified, the recall increases. In Python, it is similar to the code shown in Listing 6.1.
4. When the target recall is achieved, we collect the distance.

With the set of distances that achieve the target, kernel density estimation is performed to generate the probability density function (PDF). The 75th percentile of this PDF is used as the optimal distance or radius within which to search for duplicate images.

Listing 6.1 Python code to calculate recall

```
total_relevant = 5
correct_label = 1
cosine_distances = [0.1, 0.2, 0.3]
labels = [1, 0, 1]
target_recall = 0.9
distribution = []

recall = 0
relevant = 0
for cd, l in zip(cosine_distances, labels):
    if l == correct_label:
        relevant += 1
        recall = relevant / total_relevant

    if recall >= target_recall:
        distribution.append(cd)
```

6.3.2 Determining the Hamming distance for LSH

The LSH hashes used in our system are fixed to 16-bit binary vectors. In order to choose the optimal Hamming distance:

1. Iterate through distances 0 to 15.
2. At each Hamming distance, we iterate through the entire dataset of image embeddings.
3. Each image embedding is used as a query, retrieving all the neighbours within the Hamming distance.
4. A running tally is kept of true positives, true negatives, false positives and false negatives on a per-class basis.
5. After iterating through all the embeddings, the precision and recall is calculated for each class. They are averaged across all classes.
6. The optimal distance is selected by hand. It is optimised for recall but balances the precision so it is still above 70%.

6.4 Query Time Experiments

This experiment is necessary to determine the computational efficiency of RBNN and LSH. We expect RBNN to take longer than the LSH as the dataset grows in size. RBNN grows linearly with increasing dataset size as it is exhaustive while LSH grows sublinearly. Therefore, LSH is expected to be faster with larger dataset sizes.

This experiment uses a sample of the main dataset with approximately 3500 images. This sample is divided into subsets so as to evaluate the performance at different dataset sizes. Specifically, it is divided from sizes 10 to 100 in steps of 10, size 100 to 1000 in steps of 100 and size 1000 to 3500 in steps of 500. The dataset is randomly sampled according to these dataset sizes, and the RBNN and LSH algorithms are fit on this random subset of features from the dataset. Next, one random query image is selected as a query image. Query time is recorded for each dataset size and averaged over 10 runs. Note that the optimal parameters are determined prior to this experiment and are used here.

Chapter 7

Results and Discussion

The results of the experiments laid out in Chapter 6 are presented in this chapter. The performance of both methods is determined by the precision, recall and mean average precision values achieved on a test set of the data. The retrieval capabilities of both approaches is depicted through example query images. Finally, a comparison on the query time between the RBNN and LSH methods is shown.

7.1 RBNN radius results

The objective of this experiment, as outlined in Section 6.3.1, is to determine the optimal radius for RBNN. Figure 7.1 displays the probability density function (PDF) of the distances when a target recall of 90% is set and Figure 7.2 displays the distribution of the distances in a boxplot. The mode of the distribution is 0.13 which indicates that majority of the images are able to reach 90% recall within a small radius of 0.13. To select an optimal radius, we chose a distance value that covers 75% of the distribution. This corresponds to a radius value of 0.35. This serves as a good candidate since it will capture the majority of the query images whilst avoiding outliers.

To evaluate the performance of the chosen optimal radius, the precision and recall is calculated on the test set. Figure 7.3 shows the per-class precision, where a class is a set of duplicate object images. The average precision, taken as the average of the precision across the classes, is 0.44. As evidenced in the results, the precision is highly dependent on the class. A number of classes have perfect precision, though it must be noted that a majority of them have only a single duplicate. Others perform poorly, indicating that further work needs to be done in order to handle those cases.

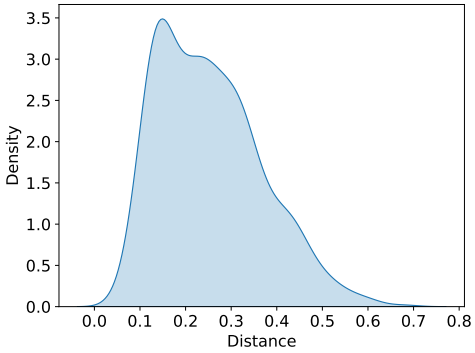


Fig. 7.1 KDE plot showing distribution of distances at recall value of 90% for RBNN with pre-trained ResNet-50.

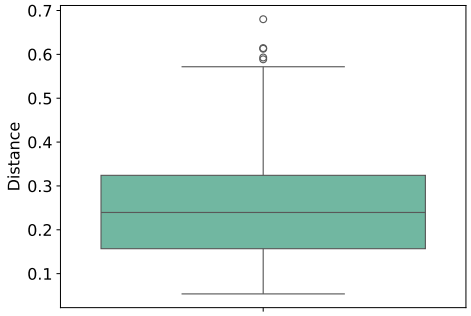


Fig. 7.2 Boxplot showing distribution of distances at recall value of 90% for RBNN with pre-trained ResNet-50..

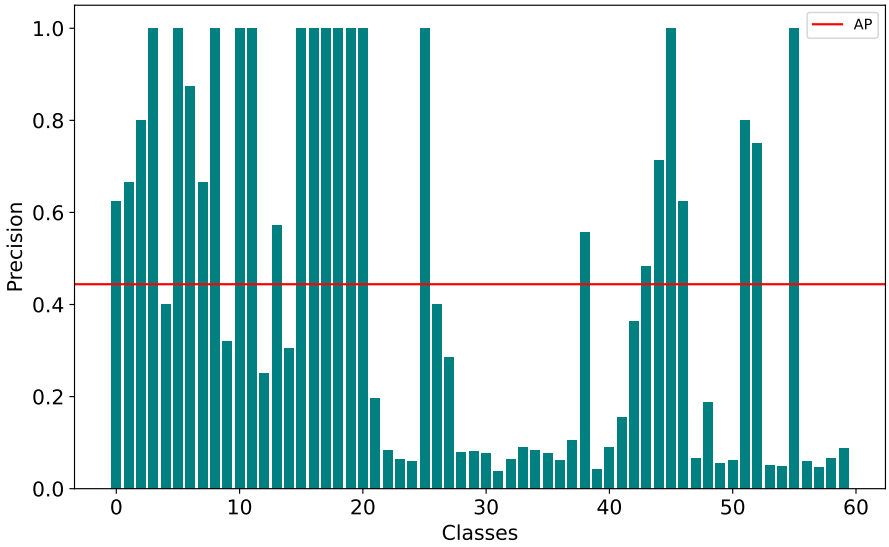


Fig. 7.3 Precision per class for RBNN with pre-trained ResNet-50. The red line is indicates the average precision across the classes.

Figure 7.4 shows the per-class recall. The average recall across classes is 0.84. Similarly, many classes have perfect recall but contain a single duplicate. The recall has variability across classes but is less so than precision. This is an appropriate result as we are biased towards stronger recall performance. The lowest recall of 0.3 indicates there are cases where the system could be improved.

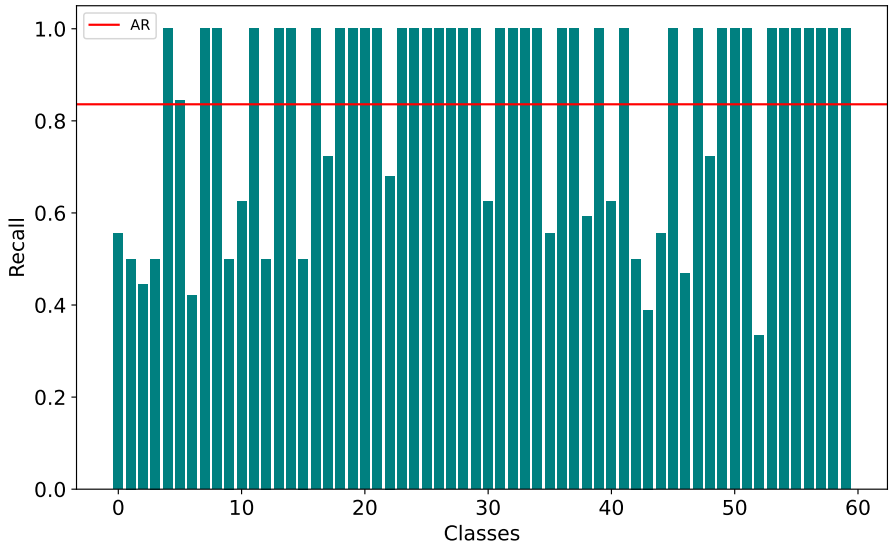


Fig. 7.4 Recall per class for RBNN with pre-trained ResNet-50. The red line is indicates the average recall across the classes.

The mAP over the dataset is 0.73. The system is able to retrieve duplicate images within the top-ranked positions – the RBNN ranks the duplicate items higher than non-duplicates 73% of the time on average.

The nature of the results are in line with expectations. According to Manning et al. (2009), there are easy information needs and difficult information needs. It is common place for a large variance in precision and recall when dealing with image retrieval tasks across a varying set of images.

Visually, we can see the discrepancies that exist within this approach. Figure 7.5 shows an instance of high recall and high precision, specifically a recall of 0.875 and precision of 1. Not all duplicates were retrieved, but all that were retrieved are indeed duplicates. Specifically two duplicates were missed. For one case, this is a radius issue as the missing image was a distance of 0.37 away from the query image. The other is an orientation issue. The image missed is the back of the statue which has a different shape profile to the front and side angles, pictured in Figure 7.6.

Fig. 7.5 Image example of the RBNN with pre-trained ResNet-50: Good recall (0.875), good precision(1)

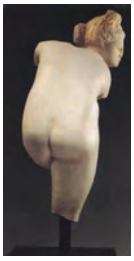


Fig. 7.6 Statue not recalled due to orientation.

Figure 7.7 shows a case where recall is low at 0.3 whilst precision is high at 1. All images retrieved were duplicates, but there are 7 instances that were not retrieved. Majority of the images not retrieved are full images of the statue and not a cropped version as pictured in Figure 7.8. Interestingly, one full image was able to be retrieved.

Fig. 7.7 Image example of the RBNN with pre-trained ResNet-50: Poor recall (0.3), good precision (1)

Fig. 7.8 Statues not recalled due to orientation.

Figure 7.9 shows the case where recall is high at 1 whilst precision is very low at 0.04. In this instance the only true duplicate is the first ranked image.

Figure 7.10 displays the case where recall is high at 1 whilst precision is moderate at 0.45. All duplicate images were retrieved, however there are others retrieved that are not duplicates. The first four neighbours are duplicates whilst the rest are not. This an ideal situation as the expert will be able to filter through these and select the true duplicates.

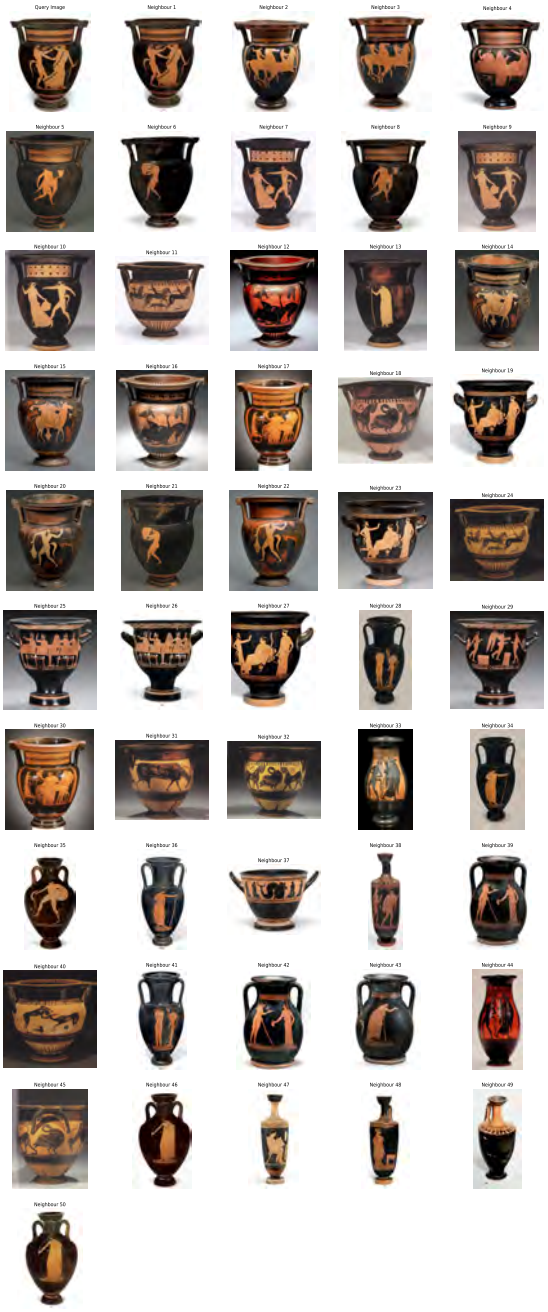


Fig. 7.9 Image example of the RBNN with pre-trained ResNet-50: Good recall (1), poor precision (0.04)

Fig. 7.10 Image example of the RBNN with pre-trained ResNet-50: Good recall (1), moderate precision (0.45)

7.2 LSH Hamming distance results

In this experiment, detailed in Section 6.3.2, the objective was to find the optimal Hamming distance. Figure 7.11 and 7.12 display the precision and recall achieved at different Hamming distances ranging from 0 to 15. Figure 7.13 displays the precision-recall curve showing the trade-off between the two. Notably, the recall plot displays the behaviour we would expect – it starts off low and increases as the Hamming distance increases. It reaches perfect recall at a Hamming distance of 12. Precision also follows an expected trend but is more drastic in its degradation. It starts off at 0.65, dropping below 0.1 at a Hamming distance of 2 and then plateauing at 0 from a distance of 8 onwards. The trade-off depicted in the precision-recall curve shows that for increasing recall, the precision decreases. The shape of this curve is however unexpected. As the recall increases, the precision drops off at a faster rate. For this reason, it is difficult to empirically choose an optimal Hamming distance and is indicative of the LSH not being an ideal approach for the duplicate identification task. For the sake of demonstration of this effect, we optimise for recall and choose a Hamming distance of 5. At this point, the recall is 0.85. Figure A.3 in Appendix A shows an example of this occurring and has been truncated as it returned far too many images as duplicates (specifically returning 129 images) being highly imprecise.

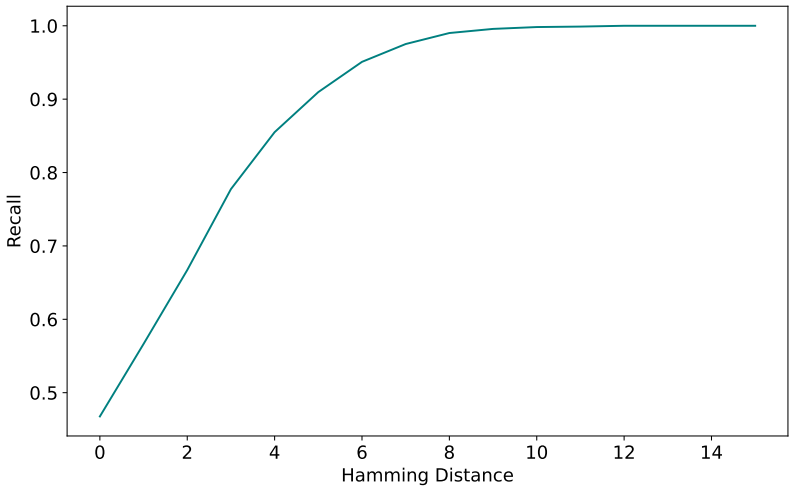


Fig. 7.11 Recall at Hamming Distance 0 to 15.

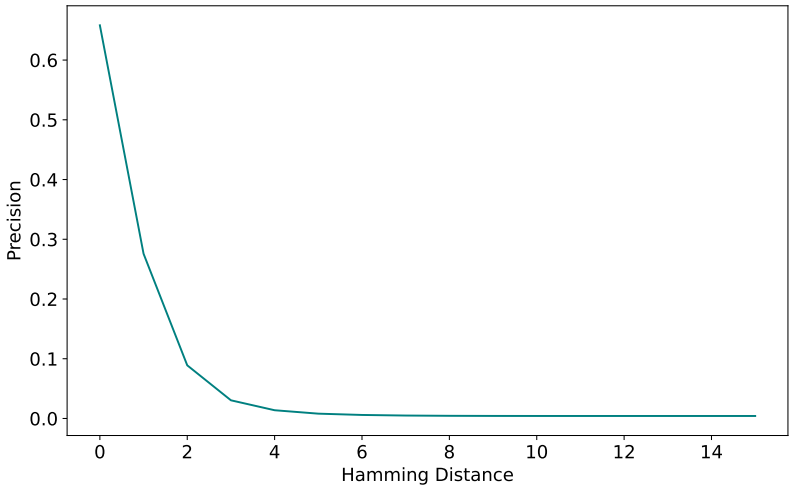


Fig. 7.12 Precision at Hamming Distance 0 to 15.

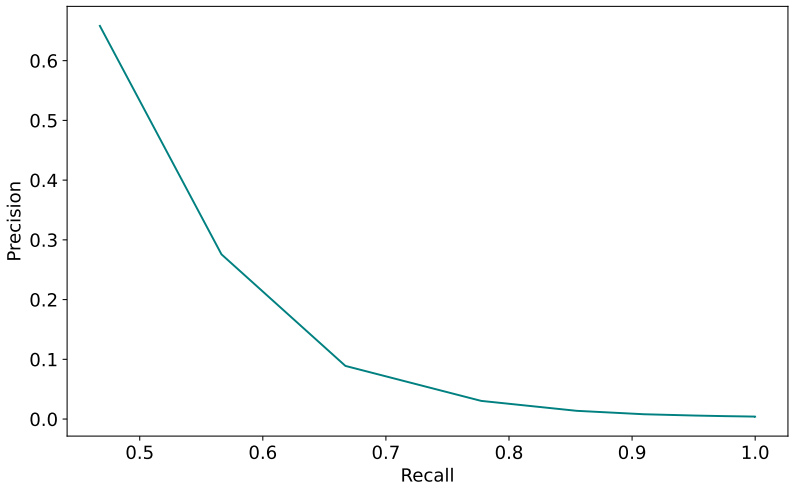


Fig. 7.13 Precision-Recall plot across Hamming distances.

The Hamming distance of 5 is used against the test set to evaluate the recall, precision and mAP. Figure 7.14 shows the per-class precision. As expected, it is low with the highest precision at approximately 0.13. The average precision is 0.04. LSH is highly imprecise meaning – many retrieved images are not duplicates of the query image. This indicates that the hash function used is suboptimal. With low precision, it means that multiple unrelated images are contained within a specific hash and hashes within a small Hamming distance. To improve the precision, the embeddings need to be hashed such that only closely related images are hashed to the same hash. A difficulty with that is that for perfect precision, you would need to have as least as many hashes as classes so that each hash contains only one class. This may be infeasible and is not the objective of this task but it highlights a challenge with LSH for this problem.

The recall is shown in Figure 7.15. The recall performs well with many classes having perfect recall. The average recall is 0.89. The lowest recall is approximately 0.5. There is room for improvement for these classes. For this system, we achieve a mAP of 0.76. Similar to the RBNN, the system is able to retrieve duplicates images within the top-ranked positions.

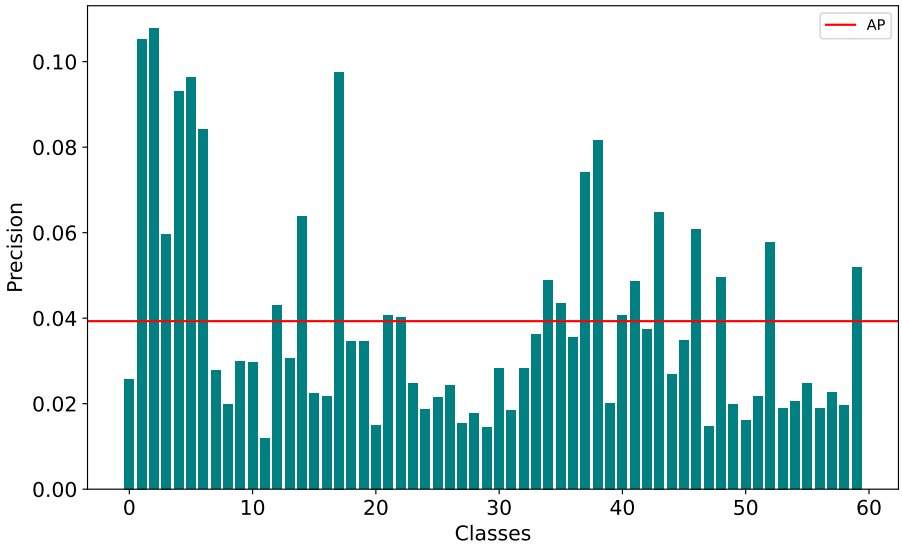


Fig. 7.14 Precision per class for LSH with pre-trained ResNet-50. The red line indicates the average precision over all the classes.

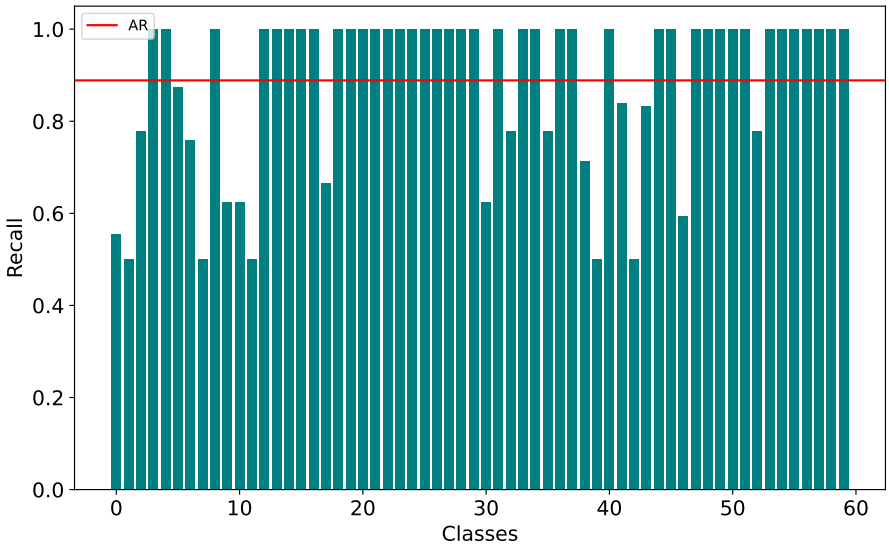


Fig. 7.15 Recall per class for LSH with pre-trained ResNet-50. The red line indicates the average recall over all the classes.

7.3 Query time comparison

In Section 6.4, the experiments for investigating the query time are presented. The objective is to investigate the speed of the algorithms as the dataset increases in size.

Figure 7.16 shows the query time against different dataset sizes for both the RBNN and LSH methods. For RBNN, as the dataset size increases, the query time increases linearly. This is due to the fact that the cosine distance is being calculated between each query vector and every image embedding within the database. It will be the case that at a certain dataset size, the query time is prohibitive. LSH scales sublinearly as the dataset size increases as shown in Figure 7.16. This is a result of embeddings being hashed into a finite number of values, much smaller than the size of the original dataset.

These results are in line with expectations. For large dataset sizes, the sublinear characteristic of LSH makes it preferable.

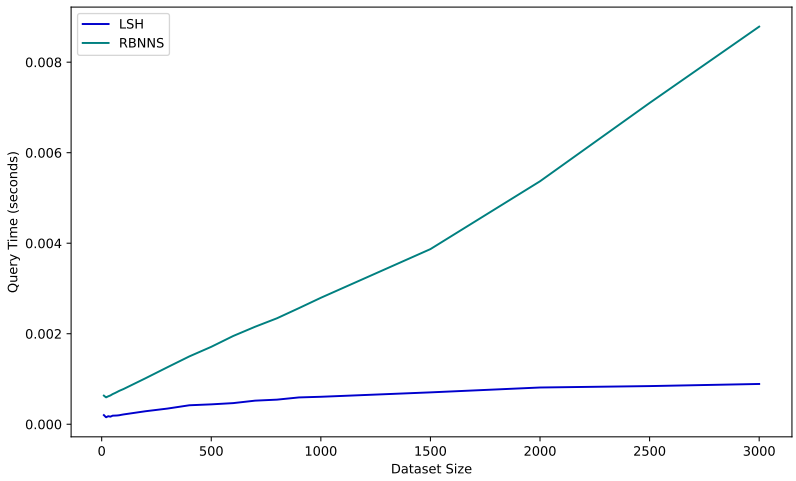


Fig. 7.16 Average query time for LSH and RBNN over varying dataset sizes. LSH scales sublinearly whilst RBNN scales linearly.

Chapter 8

Fine-tuning ResNet-50

It is hypothesised that fine-tuning the ResNet-50 will lead to an increase in precision for radius-based nearest neighbours (RBNN) and that locality sensitive hashing (LSH) will have increased recall. Fine-tuning it to the artefacts at hand should improve its performance due to the model learning the specific features, shapes, colours, contours and other such attributes of the artefacts. In doing so, the embeddings generated by the fine-tuned model will carry more distinguishing information, enabling the search algorithm to identify duplicates more accurately. As such, we fine-tune the ResNet-50 according to the different categories designed by the forensic archaeologist, with the expectation that it will improve the recall and precision of our duplication identification system.

8.1 Class Distributions

To begin we explore the high-level categories. Figure 8.1 displays the high-level categories on the x-axis and the number of images in each category on the y-axis. The distribution of images amongst the categories is highly unequal, causing a class imbalance problem. In total there are 63,500 images of 28 categories, making it a 28-class classification problem.

8.2 Fine-tuning Strategy

Figure 8.2 describes the strategy for fine-tuning the ResNet-50. We replace the final fully-connected layer, responsible for classifying an image, with our own custom classifier. This classifier comprises of a fully-connected linear layer, followed by a ReLU non-linearity, followed by a dropout layer with a dropout probability of 0.5 and ends with a final fully-connected linear layer. The first fully connected layer is used to reduce the dimensionality of

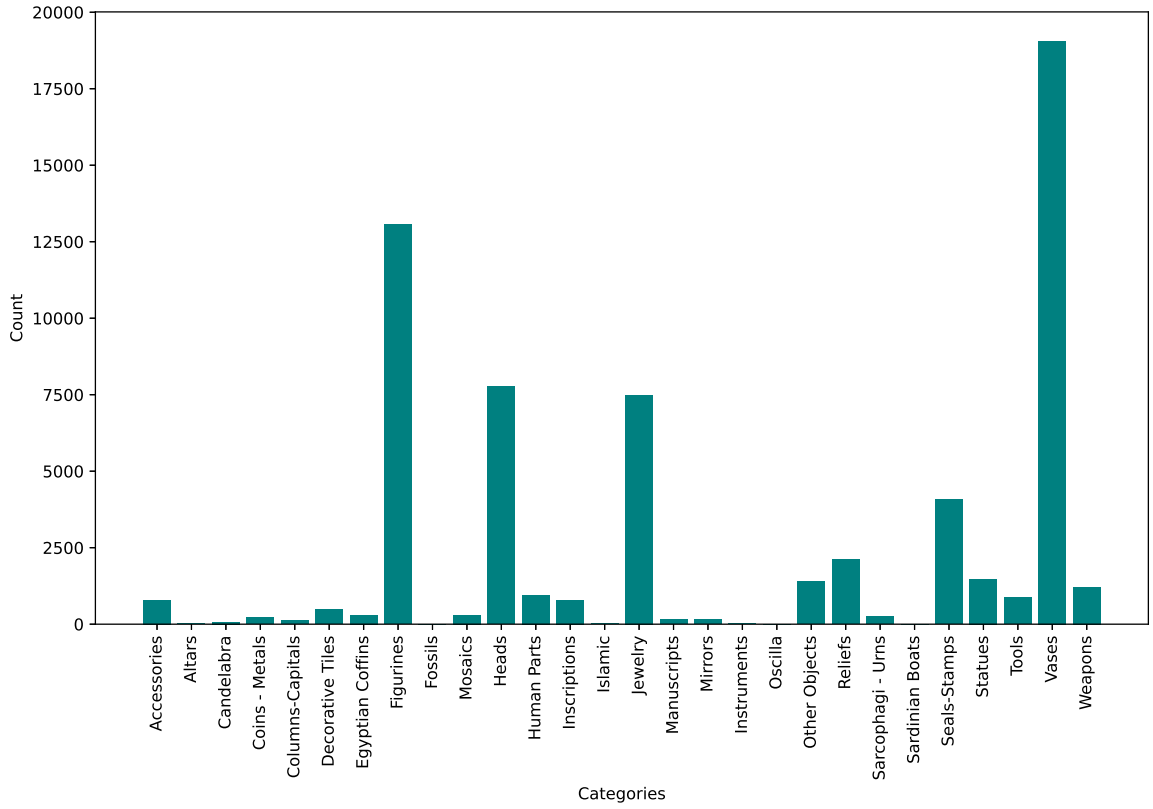


Fig. 8.1 Class distribution over 28 classes. Note the heavy class imbalance.

the ResNet-50 from 2048 to 256. The addition of the ReLU layer allows for the introduction of non-linearity into the final feature representation which in turn allows for more complex decision boundaries to be captured. The dropout layer is used as a regularisation technique in order to prevent overfitting. Using a dropout probability of 0.5 means that 50% of the neurons will be zeroed each time an image is passed through the layer during training. The final fully-connected linear layer is added to adapt the output to the specific number of classes for our task - in this case 28 classes.

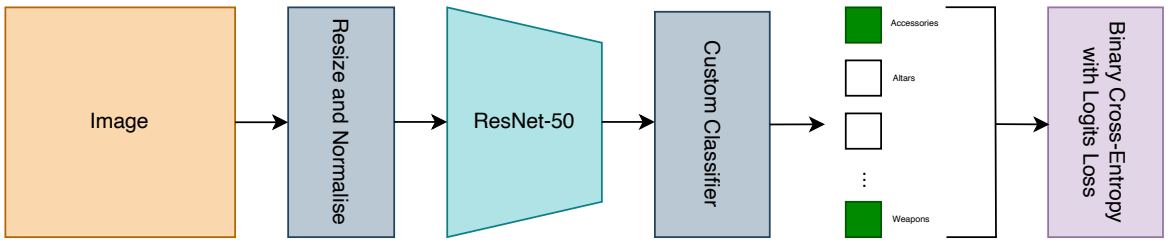


Fig. 8.2 This is an end-to-end multi-label classification training strategy adapted from (Winterbottom et al., 2022). The model has predicted the input is both a weapon and an accessory.

Recall in Chapter 2, we discussed the multi-label nature of the dataset. Objects may exist across multiple labels, depending on their hierarchy as well as being labeled under different hierarchies altogether. To handle multi-label classification, a loss function has to be used that takes it into consideration. Binary cross entropy is such a loss (Kobayashi, 2023) and is employed in our fine-tuning.

The weights are updated via backpropagation using the binary cross-entropy (BCE) loss. We know that our labels are tensors with binary values of 1s and 0s. Say for example there are 3 classes A, B and C. If an image belongs to class A and not to class B or C, the vector would be [1, 0, 0]. If it belongs to classes A, B and C, the vector would be [1, 1, 1]. In our case, each label tensor is a 28-dimensional vector. Prediction of each label occurs independently, meaning that each label is treated as an individual binary classification task. As such, using a BCE-type loss becomes a natural choice.

The specific loss we use combines binary cross-entropy loss with a sigmoid. It is defined as follows:

1. A sigmoid activation function takes the logits (raw output) and converts it into a probability between 0 and 1. We define the sigmoid activation function as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{8.1}$$

   These probabilities are then used in the BCE loss function.

2. The binary cross-entropy loss measures the difference between two probability distributions and it is calculated for each output label. The average across all the labels is then computed. For a single output, we define the loss as:

$$\text{BCE} = -[y\log(p) + (1-y)\log(1-p)] \tag{8.2}$$

   where y is the true label (0 or 1) and p is the predicted probability.

However, to combine these, we apply the BCE loss and sigmoid in tandem and as such the formula becomes:

$$\text{Binary cross-entropy} = -[y\log(\sigma(x)) + (1-y)\log(1-\sigma(x))] \tag{8.3}$$

where x is the raw output (logits), y is the true label and $\sigma(x)$ is the sigmoid of the logit. This formulation of BCE loss is more numerically stable than using a sigmoid followed by the BCE loss. This is due to it computing the loss in such a way that it avoids extremely small and large values (Mao et al., 2023).

As mentioned in section 8.1 above, class-imbalance exists. This means that the the classes or categories that are less represented may not be predicted at all, whilst the majority class could be predicted at all points and the still lead to good model performance overall. To address this, we weight the loss function for each class by the inverse of its relative occurrence such that objects with fewer images are weighted higher (Winterbottom et al., 2022). This ensures that the loss will account for the distribution of each class.

In terms of an optimisation algorithm, the standard optimisers used for image classification are stochastic gradient descent (SGD) and the Adaptive Moment Estimation (Adam) optimiser. A trade-off exists between these two, where Adam converges faster for image classification problems whilst SGD has better generalisation capabilities (Gupta et al., 2021). In this case, Adam was chosen as the optimiser. We also adjust the learning rate over time using a learning rate scheduler and start with a learning rate of 0.001.

8.3 Fine-tuned Model Evaluation

To evaluate the performance of the classification task, we look at the per-class precision and recall. We note that the dataset was divided into train, validation and test sets, with 70% of the data assigned to the training set, and 15% of the data assigned to validation and test sets respectively. We tracked the training and validation loss across 10 epochs. The results are shown in Figure 8.3 and 8.4. Here, we see varying precisions and recalls across classes, with seven classes being ill-represented. We hypothesise that the class-imbalance problem has not been successfully mitigated.

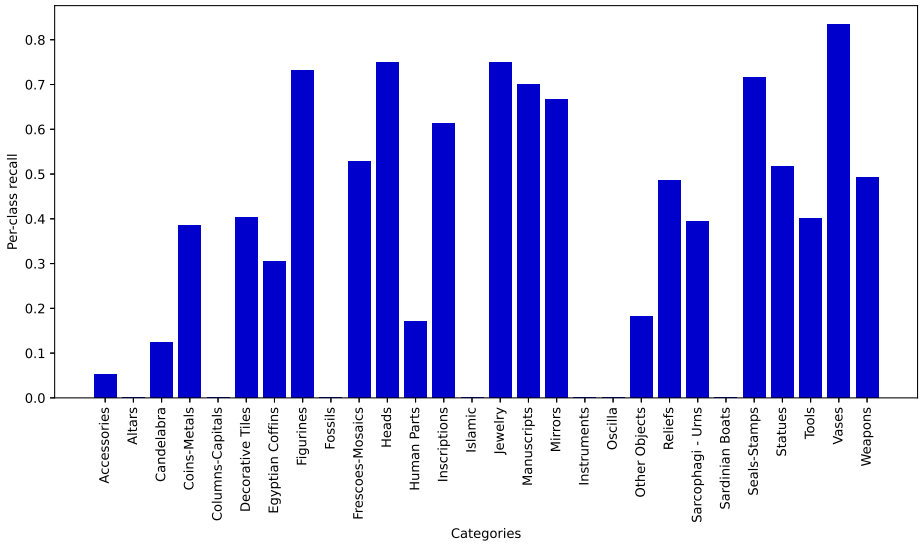


Fig. 8.3 Per-class recall for each category in the dataset provided by the expert.

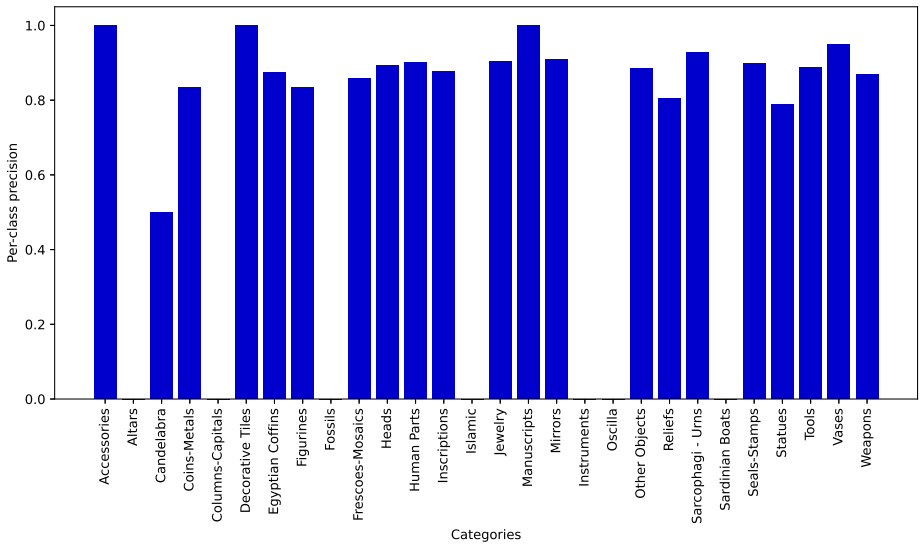


Fig. 8.4 Per-class precision for each category in the dataset provided by the expert.

The Table 8.1 shows the precision, recall and F1 score of the fine-tuned model over the test dataset. We calculated these on a per-sample basis and then averaged them across the samples. We see that on average, 69.73% of the labels that were predicted as positive are indeed correct and 70.01% of the true positive labels are identified by the model. F1 balances these two measures and is 69.82%. Since this is close to the precision and recall values, we note that the model has a good balance between making accurate positive predictions and capturing a large portion of the true positives.

Precision (%)	**Recall (%)**	**F1 Score (%)**
69.73	70.01	69.82

Table 8.1 Precision, Recall and F1 Score for the 28-class classification problem.

Chapter 9

Fine-tuned results

This chapter presents the results of the duplicate identification system using the fine-tuned model. It evaluates both the RBNN and LSH method on the same metrics of precision, recall and mAP.

9.1 RBNN radius results

Figure 9.1 displays the probability density function (PDF) of the radius when a target recall of 90% is set for the fine-tuned ResNet-50. Figure 9.2 provides further insight into the distribution of the dataset. As before, the optimal radius is chosen as a distance that covers 75% of the distribution. Therefore, the optimal radius is 0.008.

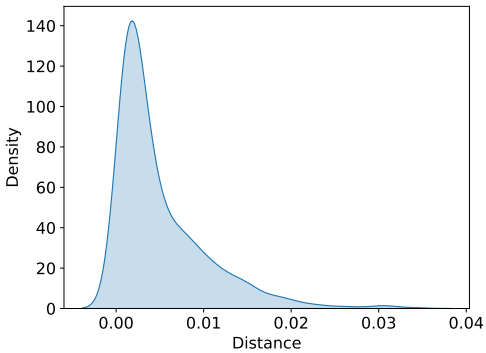


Fig. 9.1 KDE plot showing distribution of distances at recall value of 90% for RBNN using fine-tuned ResNet-50.

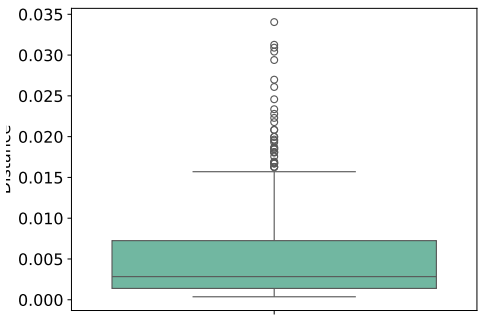


Fig. 9.2 Boxplot showing distribution of distances at recall value of 90% for RBNN using fine-tuned ResNet-50.

Feature extractor	Precision	Recall	mAP
Pre-trained ResNet-50	0.44	0.84	0.71
Fine-tuned ResNet-50	0.34	0.74	0.54

Table 9.1 A comparison of the results of the duplicate identification system using RBNN with a pre-trained model and with a fine-tuned model.

Figures 9.3 and 9.4 shows a side-by-side comparison of the PDFs of radius when a target of 90% is set when using the pre-trained ResNet-50 and the fine-tuned ResNet-50 respectively.

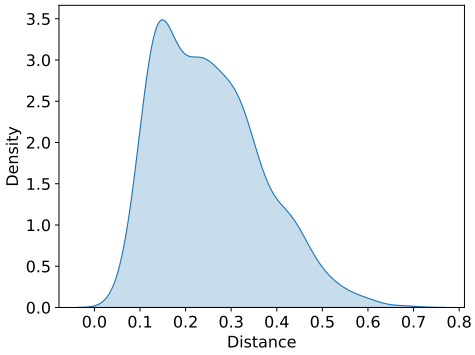


Fig. 9.3 PDF showing distribution of distances at recall value of 90% when using pre-trained ResNet-50.

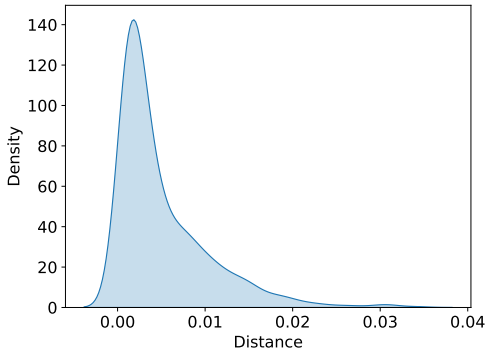


Fig. 9.4 PDF showing distribution of distances at recall value of 90% when using fine-tuned ResNet-50.

To evaluate the performance of this optimal radius, the precision and recall is calculated on the test set. As before, Figure 9.5 shows the per-class precision while Figure 9.6 shows the per-class recall. The average precision over the classes is 0.34 while the average recall is 0.74. The mAP is 0.54. In Table 9.1, we compare the results against the system using the pre-trained model.

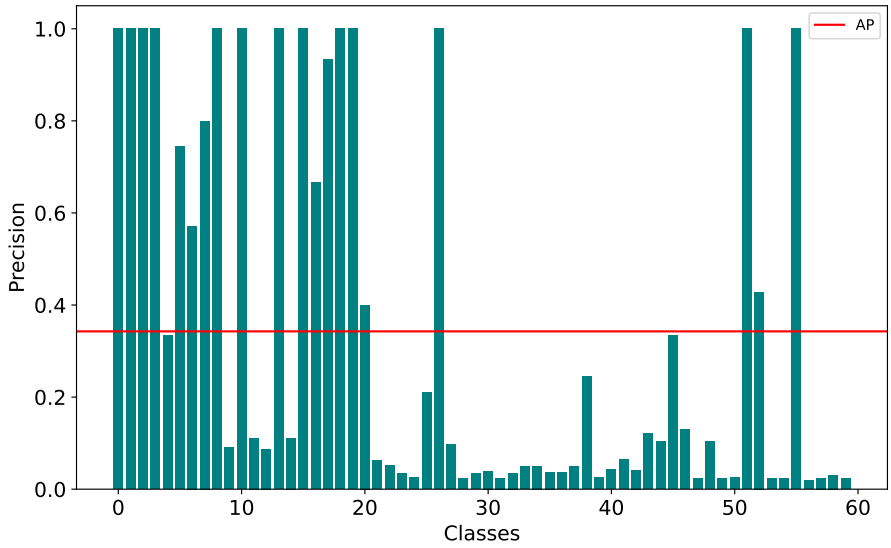


Fig. 9.5 Precision per class for RBNN with fine-trained ResNet-50. The red line indicates the average precision over all the classes.

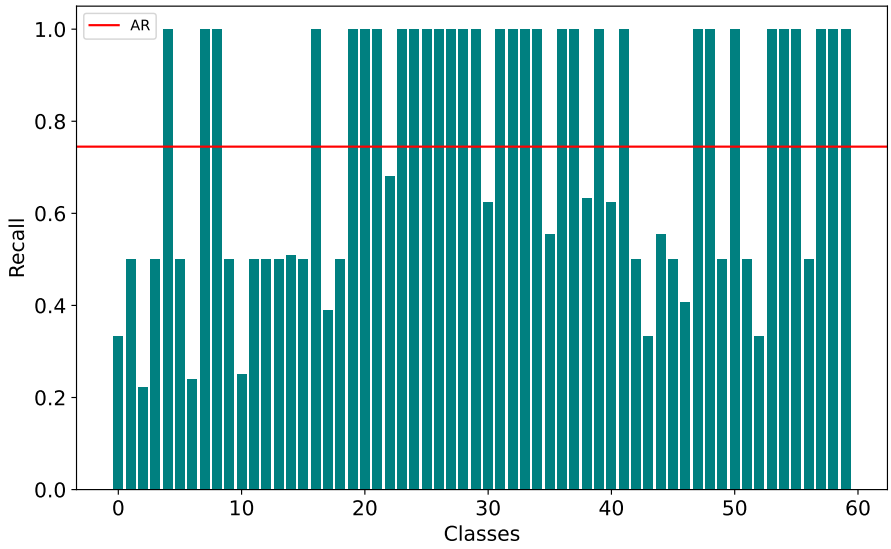


Fig. 9.6 Recall per class for RBNN with fine-trained ResNet-50. The red line indicates the average recall over all the classes.

The fine-tuned model as the feature extractor performs worse across all metrics. The leading hypothesis is that the model is overfitting to the dataset and forgetting information specific to the images. The model was not incentivised to retain information about the

differences between vases, for example, because there was not anything in the loss function that incentivised it. This is potentially detrimental for feature extraction use cases as the model is not forced to learn differences between similar objects but that are not the same. However, further investigation into the model performance is needed to verify this and other hypotheses. We consider them as future work.

Visually, the same query images are used as before to see the performance on an example set of images. Comparing Figure 7.5 from before with Figure 9.7, the fine-tuned version has poorer recall of 0.25, but high precision of 1. The vase examples from Figure 7.9 and 7.10 before have perfect recall in the fine-tuned case, but significantly more images are returned in the fine-tuned case, 40 and 30 more respectively leading to decreased precision in both cases. The full set of retrieved images are seen in Figure A.1 and A.2 in Appendix A.

Fig. 9.7 Image example of RBNN using fine-tuned ResNet-50: Poor recall (0.25), good precision (1).

9.2 LSH Hamming distance results

Figures 9.8 and 9.9 display the precision and recall achieved at different Hamming distances ranging from 0 to 15 respectively. Figure 9.10 displays the precision-recall curve showing the trade-off between the two. Notably, the recall plot displays the behaviour we would expect - it starts off low and increases as the Hamming distance increases. It reaches near-perfect recall within a Hamming distance of 5 and ultimately peaks at a Hamming distance of 8. Precision also follows an expected trend, although it starts off significantly lower than the

pre-trained version at 0.013 at a Hamming distance of 0. It then decreases rapidly between Hamming distance 0 and 6 and plateaus at 0 from a Hamming distance of 6 onwards. The trade-off depicted in the precision-recall curve shows that for increasing recall, the precision decreases linearly with increasing recall until a recall value of 0.8 is achieved. From this point onward, it follows the expected jaggered shape of a precision-recall curve. For this system, we achieve a mAP of 0.09. An optimal Hamming distance is difficult to choose in this setting too, and as such the fine-tuning has not assisted with the previous LSH issue. A Hamming distance of 3 was chosen as the optimal value.

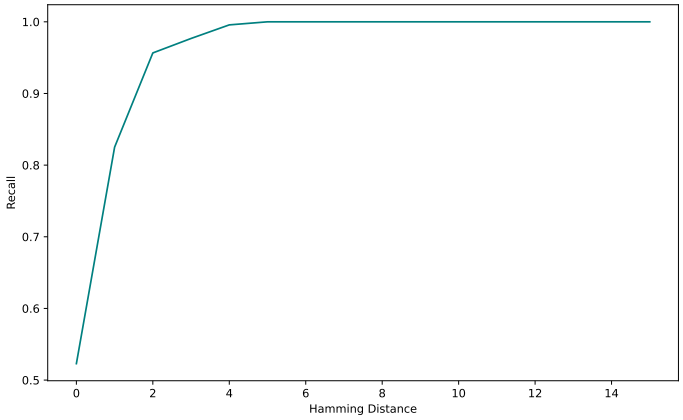


Fig. 9.8 Recall at Hamming Distance 0 to 15.

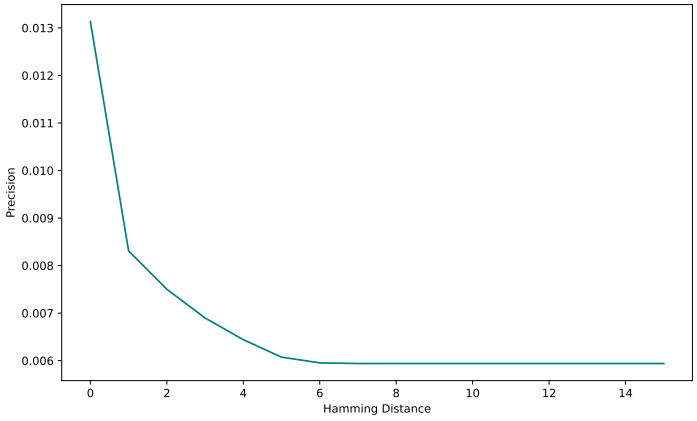


Fig. 9.9 Precision at Hamming Distance 0 to 15.

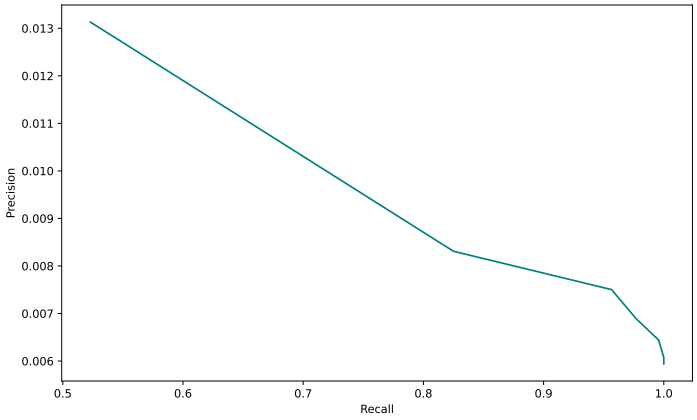


Fig. 9.10 Precision-Recall plot across Hamming distances.

Using the Hamming distance of 3, the system was evaluated on the test dataset. Figure 9.11 shows the per-class precision. It is extremely low with the maximum precision at approximately 0.13 with an average precision of 0.02. Similar to the LSH using the pre-trained ResNet-50, it is highly imprecise.

Figure 9.12 shows the per-class recall. The recall is high all round, with the majority of classes having perfect recall.

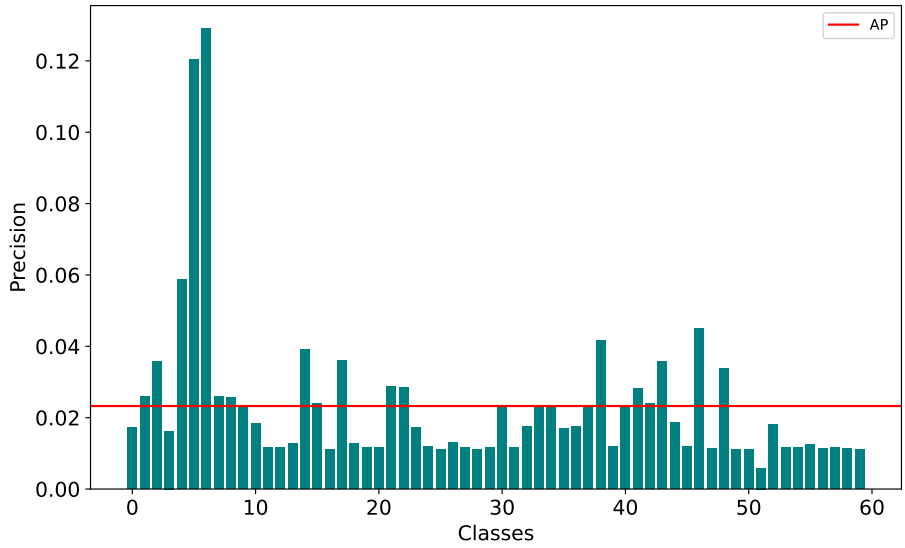


Fig. 9.11 Precision per class for LSH with fine-trained ResNet-50. The red line indicates the average precision over all the classes.

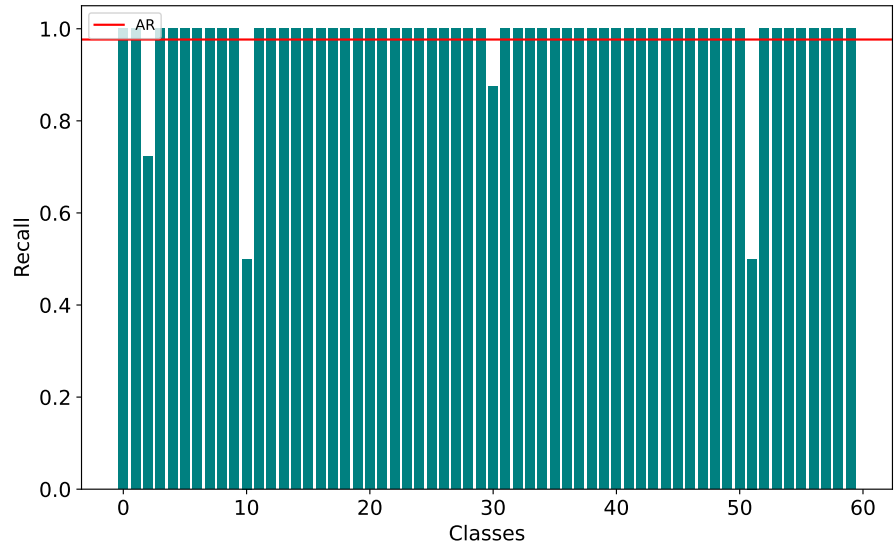


Fig. 9.12 Recall per class for LSH with fine-trained ResNet-50. The red line indicates the average recall over all the classes.

In Table 9.2, we compare the results against the system using the pre-trained model. The fine-tuned model has marginally lower precision. This is of little consequence as the precision is near 0 in both cases. The recall is higher. However, it is higher at a smaller Hamming distance of 3 whereas for the pre-trained feature extractor it was at 5. This indicates the feature extractor is not as powerful at distinguishing the relevant classes, hashing more images to a smaller set of hashes. Figure A.4 in Appendix A displays a truncated version of a test image when queried. The full version returns 169 images from the test set of 200 images. The mAP is lower as well, indicating a less precise model even after ranking. Fewer duplicate images are in the top-ranked retrieved images.

Feature extractor	Precision	Recall	mAP
Pre-trained ResNet-50	0.04	0.89	0.76
Fine-tuned ResNet-50	0.02	0.98	0.64

Table 9.2 A comparison of the results of the duplicate identification system using LSH with a pre-trained model and with a fine-tuned model.

Chapter 10

Conclusion

This work explored a duplicate identification system for identifying images picturing the same artefact. The system utilised convolutional neural networks for generating feature embeddings and two search algorithms, radius-based nearest neighbours (RBNN) and locality sensitive hashing (LSH) for finding duplicates. We evaluated their performance over a set of duplicate data. Our system has proven to be useful in solving the duplicate identification problem. However, there is much room for improvement. Both search algorithms had good recall ability but were relatively imprecise, LSH being the worst offender in this regard. To improve results, an attempt at fine-tuning the feature extractor was made. It did not work as expected. Further work will be necessary in order to identify the reasons behind its failure and subsequently to bring recommendations for improvements. This and other concerns are discussed as future work. In this section we review the research questions we attempted to answer, concluding our efforts on this work.

10.1 Research Question Review

Three research questions were posed at the start. We review the research questions in context of the work performed and determine whether we were successful in answering them.

10.1.1 Research Question 1

Research Question 1 was defined as: *Can we develop a system for the task of identifying potential duplicates in an artefact image database given a new artefact image query?*

In Chapter 5, we described our system for identifying duplicates given a query image. This system proved moderately successful at the task. The pre-trained ResNet-50 as a feature extractor paired with the RBNN proved to be capable as it achieved the highest average

per-class precision and recall as well as mAP out of the 4 models run (pre-trained RBNN, fine-tuned RBNN, pre-trained LSH and fine-tuned LSH). We conclude that yes, a system can be developed to solve this problem. However, there is room for improvement, which is highlighted in the Section 10.2.

10.1.2 Research Question 2

Research Question 2 was defined as: *Is it possible to determine duplicate images in a computationally efficient way?*

This question was motivated by the exhaustive nature of nearest neighbour search. To handle large dataset sizes, an algorithm with improved scaling characteristics would prove beneficial. We used LSH as a potential solution to the scaling problem. LSH scales sublinearly with dataset size, as shown in Figure 7.16. While it scales favourably, the precision was worse than the RBNN algorithm while only offering a modest improvement in recall. While we believe there are improvements that can be made to make LSH or other such approximate search algorithms work well for this task, in context of the work done, the answer is no. We were not able to determine duplicate images in a computationally efficient way while still offering decent results.

10.1.3 Research Question 3

Research Question 3 was defined as: *Can the performance of the system be improved through a more powerful feature extractor?*

Similarly, we still believe that an improved feature extractor will improve the precision and recall of the duplicate identification system. However, the attempt did not work successfully. There are several hypotheses for this which are highlighted under Section 10.2 which states the future work needed to improve upon this.

10.2 Future Work

This work can be improved through a number of avenues. These are discussed in this section.

10.2.1 Improve fine-tuned ResNet-50

The fine-tuned ResNet-50 did not perform well and was in fact worse at feature extraction than the pre-trained ResNet-50 without tuning. There are many hypotheses why this was the case:

- The model is overfitting to the dataset and forgetting information specific to the images.
- The loss function did not correctly account for the class imbalance.
- The loss function needs a term that penalises the model for drifting too far from the original embeddings.
- The model design may be flawed, e.g. using a linear layer to compress the ResNet vector.
- The use of cosine distance without a corresponding constraint that the feature vectors be embedded on the unit hypersphere.

We believe that work done to improve the fine-tuning of the ResNet-50 could improve the overall precision and recall of the system developed.

10.2.2 Using superior CNNs for feature extraction

As it was not in scope, we did not attempt to find the optimal CNN to use as a feature extractor. There have been many advancements in the field since the introduction of the ResNet, including DenseNet (Huang et al., 2018), vision transformers (Dosovitskiy et al., 2021) and SWIN transformers (Liu et al., 2021b). Exploring these models as the feature extraction component could yield improved results.

10.2.3 Locality Sensitive Hashing

The locality sensitive hashing algorithm has multiple avenues to explore. Two relevant avenues are the hashing algorithm and parameter optimisation of the hyperplanes.

Optimise hyperplanes

In our work, we did not optimise for the number of hyperplanes. The number of hyperplanes contributes to the effectiveness of the algorithm. It follows that optimising this parameter would lead to improved results. Specifically, in the context of this work, we have very high recall but poor precision. This means we could look into the effect of increasing the number of hyperplanes so as to further optimise for precision.

More advanced hashing algorithms

The hashing algorithm we used is fairly simplistic in nature. Investigating the hash function and its corresponding distance metric may prove worthwhile. Examples of further exploration include the MinHash algorithm (Broder, 1998) which uses Jaccard similarity as the distance metric, using Euclidean distance as a metric in p-stable LSH or simply implementing perceptual hashing.

References

Tanveer I. Bagban and Prakash J. Kulkarni. Template Based Clustering of Web Documents Using Locality Sensitive Hashing (LSH). In Brijesh Iyer, P. S. Deshpande, S. C. Sharma, and Ulhas Shiurkar, editors, *Computing in Engineering and Technology*, pages 567–584, Singapore, 2020. Springer. ISBN 978-981-329-515-5. doi: 10.1007/978-981-32-9515-5_54.

Abul Abbas Barbhuiya, Ram Kumar Karsh, and Rahul Jain. CNN based feature extraction and classification for sign language. *Multimedia Tools and Applications*, 80(2):3051–3069, January 2021. ISSN 1573-7721. doi: 10.1007/s11042-020-09829-y. URL https://doi.org/10.1007/s11042-020-09829-y.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars, April 2016. URL https://arxiv.org/abs/1604.07316v1.

Paul Boon, Laurens van Der Maaten, Hans Paijmans, Eric Postma, and Guus Lange. Digital Support for Archaeology. *Interdisciplinary Science Reviews*, 34(2-3):189–205, September 2009. ISSN 0308-0188. doi: 10.1179/174327909X441108. URL https://journals.sagepub.com/doi/abs/10.1179/174327909X441108. Publisher: SAGE Publications.

A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, Salerno, Italy, 1998. IEEE Comput. Soc. ISBN 978-0-8186-8132-5. doi: 10.1109/SEQUEN.1997.666900. URL http://ieeexplore.ieee.org/document/666900/.

Neil Brodie, Olga Batura, Gabriëlle op 't Hoog, Brigitte Slot, Niels van Wanrooij, and Donna Yates. *Illicit trade in cultural goods in Europe: characteristics, criminal justice responses and an analysis of the applicability of technologies in the combat against the trade : final report*. Publications Office of the European Union, 2019. ISBN 978-92-79-99359-6. URL https://data.europa.eu/doi/10.2766/183649.

Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya Razenshteyn, and M. Sadegh Riazi. SANNS: Scaling Up Secure Approximate k-Nearest Neighbors Search, March 2020. URL http://arxiv.org/abs/1904.02033. arXiv:1904.02033 [cs, stat].

Lin Chen, Hossein Esfandiari, Gang Fu, and Vahab Mirrokni. Locality-Sensitive Hashing for f-Divergences: Mutual Information Loss and Beyond. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/21b29648a47a45ad16bb0da0c004dfba-Abstract.html.

Long Chen, Shaobo Lin, Xiankai Lu, Dongpu Cao, Hangbin Wu, Chi Guo, Chun Liu, and Fei-Yue Wang. Deep Neural Network Based Vehicle and Pedestrian Detection for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 22(6): 3234–3246, June 2021. ISSN 1558-0016. doi: 10.1109/TITS.2020.2993926. URL https://ieeexplore.ieee.org/abstract/document/9440863. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

Yingjun Chen and Yongtao Hao. A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, 80:340–355, September 2017. ISSN 0957-4174. doi: 10.1016/j.eswa.2017.02.044. URL https://www.sciencedirect.com/science/article/pii/S0957417417301367.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848. URL https://ieeexplore.ieee.org/document/5206848. ISSN: 1063-6919.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs].

Tansel Ersavas, Martin A. Smith, and John S. Mattick. Novel applications of Convolutional Neural Networks in the age of Transformers. *Scientific Reports*, 14(1):10000, May 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-60709-z. URL https://www.nature.com/articles/s41598-024-60709-z. Publisher: Nature Publishing Group.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing.

Aman Gupta, Rohan Ramanath, Jun Shi, and S Sathiya Keerthi. Adam vs. SGD: Closing the generalization gap on image classification. 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv:1512.03385 [cs].

J. Hu, D. Gingrich, and A. Sentosa. A k-Nearest Neighbor Approach for User Authentication through Biometric Keystroke Dynamics. In *2008 IEEE International Conference on Communications*, pages 1556–1560, Beijing, China, 2008. IEEE. ISBN 978-1-4244-2075-9. doi: 10.1109/ICC.2008.301. URL http://ieeexplore.ieee.org/document/4533337/.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks, January 2018. URL http://arxiv.org/abs/1608.06993. arXiv:1608.06993 [cs].

Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. A Survey on Locality Sensitive Hashing Algorithms and their Applications, February 2021. URL http://arxiv.org/abs/2102.08942. arXiv:2102.08942 [cs].

Xudong Jiang. Feature extraction for image recognition and computer vision. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 1–15, August 2009. doi: 10.1109/ICCSIT.2009.5235014. URL https://ieeexplore.ieee.org/abstract/document/5235014.

Manjunath Jogin, Mohana ., M Madhulika, G Divya, R Meghana, and S Apoorva. *Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning*. May 2018. doi: 10.1109/RTEICT42901.2018.9012507. Pages: 2323.

Lavanya K, Stuti Tiwari, Rahul Anand, and Jude Hemanth. YOLO and LSH-based video stream analytics landscape for short-term traffic density surveillance at road networks. *Turkish Journal of Electrical Engineering and Computer Sciences*, 31(6):1099–1112, October 2023. ISSN 1300-0632. doi: 10.55730/1300-0632.4036. URL https://journals.tubitak.gov.tr/elektrik/vol31/iss6/12.

Takumi Kobayashi. Two-Way Multi-Label Loss. pages 7476–7485, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Kobayashi_Two-Way_Multi-Label_Loss_CVPR_2023_paper.html.

Nikolaos Kouiroukidis and Georgios Evangelidis. The Effects of Dimensionality Curse in High Dimensional kNN Search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45, September 2011. doi: 10.1109/PCI.2011.45. URL https://ieeexplore.ieee.org/document/6065061.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://dl.acm.org/doi/10.1145/3065386.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, December 2022. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3084827. URL https://ieeexplore.ieee.org/abstract/document/9451544. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Yao Liu, Hongbin Pu, and Da-Wen Sun. Efficient extraction of deep image features using convolutional neural network (CNN) for applications in detecting and analysing complex food matrices. *Trends in Food Science & Technology*, 113:193–204, July 2021a. ISSN 0924-2244. doi: 10.1016/j.tifs.2021.04.042. URL https://www.sciencedirect.com/science/article/pii/S0924224421003022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, August 2021b. URL http://arxiv.org/abs/2103.14030. arXiv:2103.14030 [cs].

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://link.springer.com/10.1023/B:VISI.0000029664.99615.94.

Laurens van der Maaten, Paul Boon, Guus Lange, Hans Paijmins, and Eric Postma. Computer Vision and Machine Learning for Archaeology | CAA Online Proceedings. 2006. URL https://proceedings.caaconference.org/paper/cd49_maaten_et_al_caa2006/.

Tanya Makkar, Yogesh Kumar, Ashwani Kr Dubey, Álvaro Rocha, and Ayush Goyal. Analogizing time complexity of KNN and CNN in recognizing handwritten digits. In *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pages 1–6, December 2017. doi: 10.1109/ICIIP.2017.8313707. URL https://ieeexplore.ieee.org/abstract/document/8313707.

Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs, March 2016. URL https://arxiv.org/abs/1603.09320v4.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. Introduction to Information Retrieval. 2009.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-Entropy Loss Functions: Theoretical Analysis and Applications, June 2023. URL http://arxiv.org/abs/2304.07288. arXiv:2304.07288 [cs, stat].

Pallabi Parveen and Bhavani Thuraisingham. Face Recognition Using Multiple Classifiers. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 179–186, Arlington, VA, USA, November 2006. IEEE. ISBN 978-0-7695-2728-4. doi: 10.1109/ICTAI.2006.59. URL http://ieeexplore.ieee.org/document/4031896/. ISSN: 1082-3409.

Petros Patias and Charalampos Georgiadis. Fighting Illicit Trafficking of Cultural Goods—The ENIGMA Project. *Remote Sensing*, 15(10):2579, January 2023. ISSN 2072-4292. doi: 10.3390/rs15102579. URL https://www.mdpi.com/2072-4292/15/10/2579. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

Dehua Peng, Zhipeng Gui, and Huayi Wu. Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect, December 2023. URL https://arxiv.org/abs/2401.00422v2.

Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 1232–1237, New York, NY, USA, March 2004. Association for Computing Machinery. ISBN 978-1-58113-812-2. doi: 10.1145/967900.968151. URL https://doi.org/10.1145/967900.968151.

Rubina Raja and Christos Tsirogiannis. *Shaping Archaeological Archives*, volume 4. Brepols Publisher, Turnhout, Belgium, 2023. ISBN 978-2-503-60564-7.

Luke Rottok, Wilson Cheruiyot, and Kennedy Ogada. *K-Nearest Neighbour in image retrieval based on color and texture*. August 2018. doi: 10.13140/RG.2.2.27360.53767.

D. R. Sarvamangala and Raghavendra V. Kulkarni. Convolutional neural networks in medical image understanding: a survey. *Evolutionary Intelligence*, 15(1):1–22, 2022. ISSN 1864-5909. doi: 10.1007/s12065-020-00540-3. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7778711/.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, March 2006. ISBN 978-0-262-25695-7. doi: 10.7551/mitpress/4908.001.0001. URL https://direct.mit.edu/books/book/2322/Nearest-Neighbor-Methods-in-Learning-and.

Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, May 2014. ISSN 10773142. doi: 10.1016/j.cviu.2013.12.005. URL https://linkinghub.elsevier.com/retrieve/pii/S1077314213002361.

Vijay Tiwari. Developments in KD Tree and KNN Searches. *International Journal of Computer Applications*, 185:17–23, June 2023. doi: 10.5120/ijca2023922879.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep Image Prior. *International Journal of Computer Vision*, 128(7):1867–1888, July 2020. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-020-01303-4. URL http://arxiv.org/abs/1711.10925. arXiv:1711.10925 [cs, stat].

Thomas Winterbottom, Anna Leone, and Noura Al Moubayed. A deep learning approach to fight illicit trafficking of antiquities using artefact instance classification. *Scientific Reports*, 12(1):13468, August 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-15965-2. URL https://www.nature.com/articles/s41598-022-15965-2. Publisher: Nature Publishing Group.

Mahmood Yousefi-Azar, Leonard G. C. Hamey, Vijay Varadharajan, and Shiping Chen. Malytics: A Malware Detection Scheme. *IEEE Access*, 6:49418–49431, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2864871. URL https://ieeexplore.ieee.org/abstract/document/8463441. Conference Name: IEEE Access.

Yi Yu, Michel Crucianu, Vincent Oria, and Lei Chen. Local summarization and multi-level LSH for retrieving multi-variant audio tracks. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 341–350, New York, NY, USA, October 2009. Association for Computing Machinery. ISBN 978-1-60558-608-3. doi: 10.1145/1631272.1631320. URL https://doi.org/10.1145/1631272.1631320.

Zoran Zivkovic. *Improved Adaptive Gaussian Mixture Model for Background Subtraction*, volume 2. September 2004. ISBN 978-0-7695-2128-2. doi: 10.1109/ICPR.2004.1333992. Journal Abbreviation: Proceedings - International Conference on Pattern Recognition Pages: 31 Vol.2 Publication Title: Proceedings - International Conference on Pattern Recognition.

Appendix A

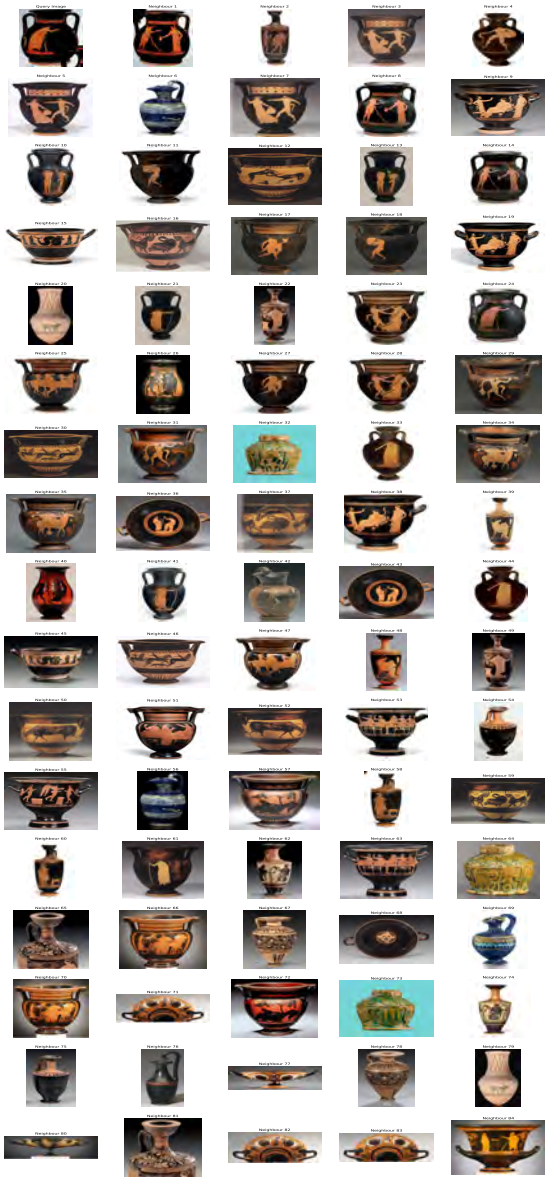


Fig. A.1 RBNN with pre-trained ResNet-50:Good recall (1), poor precision (0.05)

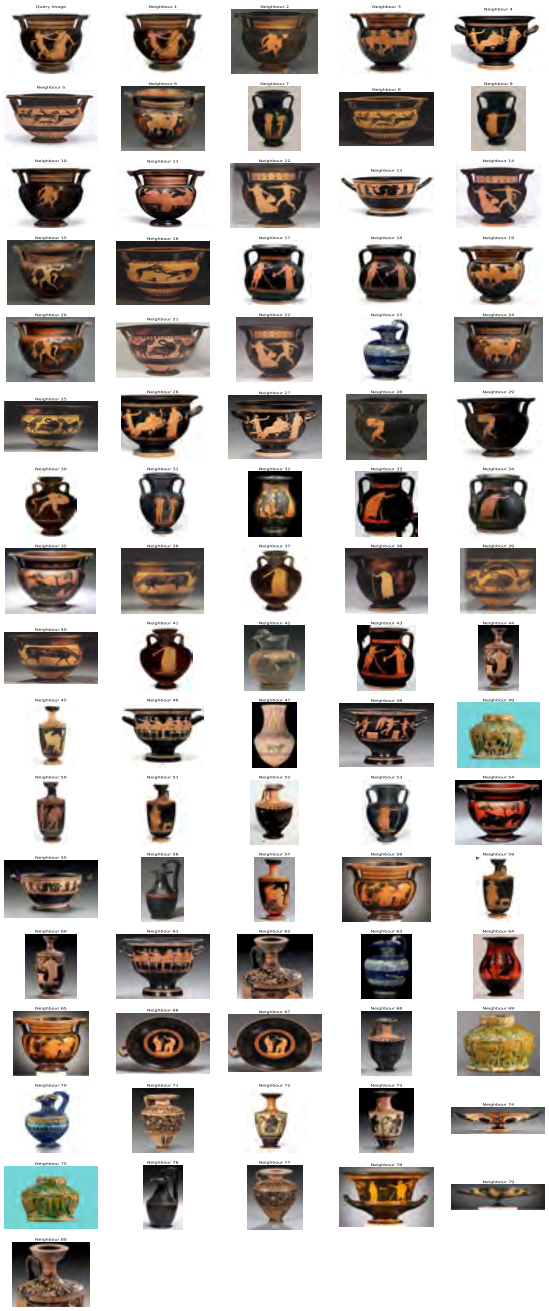


Fig. A.2 RBNN with pre-trained ResNet-50: Good recall (1), poor precision (0.02)

Fig. A.3 LSH with pre-trained ResNet-50 recalls 129 of the images in the test dataset. This is a truncated snapshot.

Fig. A.4 LSH with pre-trained ResNet-50 recalls 169 images out of 200 images in the test dataset. This is a truncated snapshot.