Scraping a shoeprint database and matching an unknown shoe print

Darren Gichuru Gitagama

03/05/2023

Abstract

In an industry saturated with a never ending stream of new shoe models and patterns, the task of maintaining effective oversight becomes increasingly unmanageable. Current leading systems update their shoe sole image repositories periodically, often struggling to maintain pace with the manufacturer releases. Thus an opportunity arises to develop a system capable of accurately classifying unknown shoeprints, particularly within forensic applications where new insights can be found. To address this, we propose a pipeline designed to automatically source data, extract key insights, and construct classifiers, all with the overarching objective of fully automating these processes to enable the reintegration of new insights. Our approach involves a comparative analysis of the most promising classification techniques, ranging from deterministic models such as LBP and SIFT to more complex, black-box models like CNNs. Additionally, we explore the potential enhancements in prediction through the merging of methods, with the promise of embedding local, global, and spatial information within categorisation decisions. We aim to provide a benchmark for performance, and build the foundation for a future fully automated system

I certify that all material in this dissertation which is not my own work has been identified.

Signed: Darren Gitagama

Contents

1 Introduction

The aim of this project is to create a shoe sole image classification system designed to accurately categorise unknown shoe prints images, particularly useful given the lack of regulation regarding the release of new shoe models and novel patterns to governing bodies. To achieve this, we have created a web scraper to automate the data collection of shoe sole images (i.e, figure 1) from online sources. From this, we propose a system that will extract key pattern and keypoint information to inform subsequent classifications. The primary emphasis of the report will be on leveraging the methodologies outlined in the Literature Review to obtain good quality class assignments while building a repository of features for aiding in future matching processes. A variety of approaches will be explored, ranging from deterministic models like Scale-Invariant Feature Transform (SIFT)[1], to the use of unexplainable and nondeterministic deep learning techniques. Through experimentation, efforts will be made to construct descriptors capable of encompassing local, global, spatial, and hierarchical information through feature-level fusion [2]. In our case, the term "descriptor" refers to the numerical representation of the pattern information. Ultimately, due to the novelty of this particular problem case, we aim to determine the most effective approach for shoe sole classification, providing a well defined benchmark of performance, and for future work.

Figure 1: Example of web-scraped shoe: Jordan 13 Outsole

Current leading American systems such as SoleMate and Treadmark [3], require constant maintenance in attempts to keep them up-to-date. This requires a hands-on effort to source new information, perform analysis and update these systems (often occurring periodically). But given the sheer volume of shoe releases, from an equally large number of manufacturers, such an approach quickly becomes difficult to sustain. Thus a research opportunity arises, centering on an approach to automate unknown shoeprint matching. Such research is important, having significant implications in forensics where valuable insights into a crime scene can be attained. Through the examination of patterns present in shoe soles images, investigators can learn new information regarding a shoe's make, model, and size, along with any unique branding that may help to establish links within the scene. Using a database, we can make such investigative procedures more efficient through automation, though you may be surprised to know the UK currently lacks such a capability and is in need of a comprehensive shoe outsole database service [4]. A key distinction to note is that our research focuses on classification rather than retrieval, where the former focus on query categorisation, and the latter centres on returning the top matches to a query image. Furthermore, we must clarify that we are not aiming to perform classifications on shoeprint impressions, but rather the outer sole of the shoe itself, as demonstrated by figure 1.

To make this project viable given the time limitations, we must constrain the scope of this problem. We therefore use a dataset containing only sneakers, more specifically from the Nike Jordan franchise, due to the following properties: (i) Their distinctive features and branding allowing for clear categorisation (ii) Variations in designs allowing for a diverse pool of training data. This data pool encompasses shoes with subtle pattern differences as well as highly unique models, which will be beneficial in fully assessing the models classifications abilities (iii) Their global use and recognition due to commonality. Such shoe soles are more likely to be found in places of interest due to wide availability, therefore producing more immediate value. We also limit the sourcing of data only from Google Images, thereby only requiring the use of a single script to handle the web scraping process, and removing the need to handle variations in different website structure (further explored in section 2.1). We must be wary of memory and speed constraints when using certain classification models, hence techniques such as dimensionality/feature selection will play an important part in ensuring reasonable speeds.

1.1 Requirements

- **Data Collection Requirements**: The dataset should exclusively consist of images of sufficient quality, ensuring the acceptance of only unobstructed images without excessive noise or severe

warping. Additionally, we need to provide means for the expansion of this dataset to accommodate a broader range of shoe types, aligning with our objective of accommodating a large variety of new shoes. This requirement emphasises the need for the system to be robust, where since information is solely retrieved from unvetted sources, mechanisms must be in place to enforce data quality standards and consistency.

- **Feature Extractors and Classifiers**: The success of the system will depend on the extent that distinctive pattern information is captured. As we are motivated to find the best suited approach for this, we will explore a variety of algorithms to produce a comparative study, aiming to explore their respective strengths, weaknesses, and performance characteristics. The best approaches will be distinguished by its capabilities of capturing subtle variations in shoeprints while being resilient to noise, distortions, and other sources of variability. Hence this research necessitates the implementation of multiple algorithms. While experimenting with ways we can perform feature fusion level, where we must ensure that prior to integration, features to be fused are indeed compatible, and there is potential value to be derived from the fusion.
- **Non-Functional requirements**: Due to the fact that we are working with a variety of algorithms and approaches, it is essential that components are designed to be modular, enabling us to combine, change and reuse the different aspects with ease. This will allow us to design common approaches for module interactions (ie: feature extractors), testing and evaluation. Similarly, the database system chosen must be able to handle semi-structured data, whilst having the needed scaling capabilities. This will allow us to store data of different types in a manner that is easily interpretable and flexible enough to accommodate any future changes.

1.2 Literature Review

We centre our research on the use of the three popular approaches for feature extraction and classification: Local Binary Pattern (LBP) [5], Scale-Invariant Feature Transform (SIFT) [1], and Deep Learning (CNNs) [6], each offering distinct advantages and considerations for success. Local Binary Pattern (LBP), is a texture analysis technique that offers the upsides of a simple and computationally light-weight image processing approach. It does this by encoding local regions into binary patterns through a process of thresholding, in which the intensity of each pixel is compared against its neighbours, mapping them to a concise set of 2 values. LBP has undergone extensive research, resulting in a multitude of variations aimed to address its underlying weaknesses For example multi-block LBP(MB-LBP) [7] works by applying the LBP operator onto every segment of an equally partitioned image, thereby producing a resulting descriptor with greater robustness to variations in regional intensities and noise. Concatenating the LBP codes into a feature histogram creates a description of the texture profile, including pattern distribution within the image (amongst other texture properties), enabling the differentiation of textures, using distance metrics/classifiers. This process is very computationally efficient due to the reduced dimensionality in data and simple calculations performed.

SIFT offers advantages over LBP by producing a descriptor that is robust to geometric transformations, scale and illumination. This is accredited to its four stage process consisting of: space-scale extrema detection, keypoint localization, orientation assignment, and the formation of the keypoint descriptor. This is done by creating a scale space of an image through convolutions with a Gaussian kernel at different scales, thereby creating a hierarchy of blurred images. Within each image octave in the hierarchy, a blur is applied before calculating the Difference of Gaussians (DoG). We can then determine local extrema after comparing pixels with their neighbours at different scales, enabling us to identify key points within an image. Keypoint localisation refers to the refining of the identified points through the elimination of points exhibiting low contrast or lying along edges. After establishing stability and scale invariance in keypoints, orientations are assigned by constructing orientation histograms through the analysis of gradient information in surrounding regions. Rotation independence is achieved by subtracting the orientation of each keypoint from the orientations in its neighbourhood, and in doing so each gradient rotation becomes relative to the keypoints rotation. However, SIFT's performance diminishes in complex backgrounds or poor-quality images, attributed

to reliance on gradient information for keypoint detection. Therefore if an image abstracts a lot of this information, or warps it, it can lead to difficulties in subsequent steps. ORB [8] positions itself as a faster approximation to SIFT, achieving heightened speeds using FAST algorithm (Features from Accelerated Segment Test) [9] for keypoint detection, and using BRIEF (Binary Robust Independent Elementary Features) [10] to formulate robust descriptors based on pixel intensity comparisons.

Feature-level fusion aims to combine features from a diverse range of sources and modalities using different techniques to produce a merged representation of salient information with greater discriminatory capabilities when compared to the respective feature vectors [2]. The primary benefits in this approach lies in its abilities to detect correlated features within respective algorithms and eliminate redundancy. By doing this we ensure we capture condensed sets of prominent features information thereby compacting the resulting vectors. We attempt this using the following techniques: (i) Principal Component Analysis (PCA)[11] - a technique that uses orthogonal operations to convert sets of correlated observations into linearly uncorrelated variables. It aims to maximise variance whilst eliminating existing correlations amongst variables thereby producing more compact representations of pattern information. (ii) concatenation to simply combine feature vectors (iii) K Means [12] - a clustering algorithm used to separate data into defined groups based on similarities shared. Data is iteratively assigned to its nearest cluster based on some distance measure whilst also updating centroid positions until the system convergens. This forms the basis of our Bag of Features pipeline (BoF)[13], which is instrumental in grouping related features in a form that can be processed by subsequent classifiers. Support Vector Machines (SVM)[14] is well suited for classifying such grouped data given that they prove to be very effective when handling high dimensionality, small datasets, as well as modelling complex relationships. It works by finding the best hyperplane that separates classes in a feature space, and given the choice of kernel we can capture both linear and non-linear relationships within the data.

SIFT and LBP are deterministic and focus on the representation of local information, however deep learning approaches are often preferable due to heightened abilities to capture global and nuanced information - an important factor in the comparison of similar shoe prints. Deep learning is particularly suited for image recognition tasks, demonstrating great capabilities for discovering intricate structures in high-dimensional data through the use of representation learning. To capture hierarchical and abstract features, non-linear modules are combined to transform representations at a given level into one higher [6]. Given enough module combinations, highly complex functions can be represented, being particularly desirable for classification tasks where higher layers amplify data vital for discrimination. While effective, deep learning encounters limitations in our use case due to data sparsity, thereby making a trained model more susceptible to overfitting and as a result incapable of making generalisations.

2 Design

Producing a functioning shoe sole matching system broadly requires the implementation of 4 key parts: data collection, preprocessing, feature extraction, and Classification. This section provides detailed justifications into the various design elements and choices made at each stage in the pipeline.

2.1 Scraping Data

A web scraper is the first stage within the system pipeline, playing a pivotal role in addressing the absence of a comprehensive dataset. To automate the collection of the dataset items, we need to interface with an image data service with the following properties and capabilities: (i) has a broad collection of image content that covers a wide variety of themes and topics, and is accessible within a centralised location. Such centralisation is beneficial for ensuring consistency (i.e., in image formats, sizes, etc), efficiency, and scalability as we pursue to accommodate the ever-increasing number of shoe patterns released. (ii) Advanced filtering parameters: essential for narrowing the search and finding images of relevance and value. (iii) Image metadata (brand, size, colour, etc.), providing necessary information for labeling data, (iv) has an API that is less susceptible to frequent changes, meaning that

our script functionality has more longevity and requires low maintenance. Google Images proves to be a good solution, meeting the stated criteria, as well as being an established avenue for web-scraping within the computer vision community. However, Google's lax approach to content verification in a publicly accessible service allows a lot of noise (irrelevant images) to be collected in the dataset during web-scarping, requiring an additional filtering process. This should not only ensure images are of the correct subject matter but enforce a consistent standard in which shoeprint images are of sufficient resolution and without obstruction or excessive noise. Whilst acknowledging existing solutions using automated filtering systems [15](using deep learning models), I intend to perform this stage manually for the following reasons: first to prevent scope creep due to the breadth of this task, thus we want to retain research focus on shoeprint matching rather than data collection. Secondly, proposed filtering solutions like [15] are not perfect, having levels of uncertainty that still include noise in the dataset, possibly having trickle-down effects on ML model performance or feature extraction.

Our overarching objective for this phase is to be extendable for future purposes as we want to have the functionality to continuously incorporate new shoe labels and data. We do this using arguments controlling the number of images to be scrapped per page, and an updateable list containing all web pages to be visited. The scraping is performed using the selenium library which provides ample tooling for locating, inspecting web page elements, and applying automation.

2.2 Data processing

In this stage, raw input data must undergo several transformations to produce an output void of noise (random variation), and inconsistencies, and in a correct format that retains key information necessary for analysis by subsequent algorithms. Determining which transformations are necessary depends on the algorithm as they each exhibit different constraints. For example, LBP calculations are dependent on the local spatial ordering of pixels, therefore we must maintain consistent image aspect ratios to preserve these spatial relations. This is also true for extreme variation in rotations, where additional re-orientation processes are required to enforce consistent direction in the dataset. Contrastingly, SIFT is not subject to these constraints, proving to be more robust to scale and rotation changes and alleviating the need for an equally rigorous preprocessing pipeline. Despite this, every image in the dataset undergoes the same preprocessing stages for the sake of consistency, allowing us to make comparisons without bias.

The following preprocessing stages are applied: First, the images are converted to greyscale. This forms the basis from which the algorithms are able to perform keypoint detection and thresholding operations. We then remove the background and noise from the image using techniques like thresholding and Gaussian blurs respectively, limiting interference from random variations present in the image. We then rescale the image to dimensions 150 * 400, a number chosen as a trade-off between resolution and eventual time costs as size directly impacts the number of key points detected within an image, resulting in a more time-consuming matching process. As we acknowledge the web scraped dataset is not representative of the final dataset due to the inclusion of redundancy and irrelevant images (section 2.1), a big risk going forward is overfitting due to small datasets. Possible solutions may involve scraping considerably more shoeprint images for an increased number of pages, but this approach ultimately becomes problematic due to the proportionally longer times required for subsequent manual filtering. Alternatively, augmentation aims to mitigate overfitting by enlarging the dataset through the application of geometric transformations, in a manner that reflects real-world variations. Such transformations include shear rotations, rotations, random cropping, zooming, and random noise.

2.3 Feature Extraction & Fusion

The goal in the feature extraction stage is to capture salient pattern information in a form that is representative of the underlying structures present within the image. This information serves as the input to classifying models, therefore its success has a downstream effect on classification outcomes. When choosing appropriate feature extractors, we consider the following properties; (i) ability to

capture spatial, global, and or local information in images. Such information should be sufficient enough for effective discrimination and generalisation (ii) They describe pattern data in a manner invariant to scale, rotation, and illumination, and perform in an efficient and scalable manner. Hence we centre our approach on LBP and SIFT. Whilst I acknowledge that LBP is comparatively limited in its capabilities, its extensive study and simplicity provide a good benchmark for consequent classifier performances.

We also investigate with an enhanced LBP-SIFT descriptor, anticipating greater discriminatory capabilities. Such a descriptor is assumed to leverage LBP's strengths as a fast, local regional descriptor and SIFT's robust properties. SIFT output is typically formed from a list of detected key points in the coordinate form (x,y) and a list of fixed-sized vectors representing associated descriptors. These descriptors provide a numerical encoding of the information contained within a patch around key points. Similarly, LBP descriptors are formed as a vector representation of the LBP histograms, where each bin within the histogram reflects a feature (or LBP code). Thus when referring to fusion at this level, we are finding the best way to combine the respective vectors whilst retaining their key information. However, fusion at the feature level poses several challenges: (i) feature vectors are required to be compatible for concatenation to occur, ensuring features are representative, relevant to the task, and on the same scale. Furthermore, as SIFT is dependent on gradient information to determine key points, the number of detected per image can be vastly different resulting in a dataset with large variations in vector size (ii) fusion may produce feature vectors with excessively large dimensionality, possibly introducing a lot of redundancy if data is related. (iii) success is dependent on the effectiveness of individual descriptors. Should LBP or SIFT fail to capture the necessary details, we are only building a shaky foundation, where individual shortcomings are likely to be propagated through any final model.

Several approaches are used to achieve successful feature-level fusion, the first of which being concatenation. However this is likely to create some issues or provide limited insights due to inconsistencies in the manner in which the contribute to the final vector. For instance, LBP vector size may be much larger than SIFT vectors, inadvertently giving itself a larger weighting due to larger contributions. Though SIFT may compact greater insights, therefore requiring larger weighting. (2) Bag Of Features (BoF) [13], uses K Means clustering to build a vocabulary of features from which subsequent inputs can be analysed against. Building the dictionary involves clustering related features together to form codewords, offering further advantages of compacting the vector sizes. Though this introduces an additional parameter that may need to be optimised - the number of clusters.

2.4 Classifiers

By this stage several descriptors have been obtained: LBP (and MB-LBP), SIFT (and ORB), LBP-SIFT, alongside corresponding raw filtered image data, which serve as input to our classifier models. Our goal while using these models is to learn the underlying function and relationships that map specific representations to a label. Once trained, the classifier can then be used to categorise the class labels of new, unseen instances based on their feature descriptors, which when quantified forms the basis of our evaluation procedure. The first classifier we intend to use is Support Vector Machines, being well suited for our dataset consisting of high dimensional data and complex relationships. SVM's can capture such relations through the use of the appropriate kernel functions. In our case, a polynomial kernel may best be suited to ensure maximum separation and minimise error. CNNs are also used to classify descriptors, as well as performing image classification themselves on the raw image data. We expect to uncover additional hierarchical representations contained in the image which may help to better inform predictions . CNNs are particularly well suited to solving problems regarding complex, structured data, thus it lends itself well to our problem, especially in cases where shoes are only distinguishable by more nuanced factors. To mitigate overfitting, we place a preference on simpler architectures that have shallower arrangements of convolutional layers to reduce complexity.

2.5 Expeprimental Design

We will quantify the success of the system using an array of statistical techniques to ensure we have a holistic understanding of the system's effectiveness. The following metrics are used: accuracy for determining overall correctness, precision for determining how often the models give correct positive predictions, and recall for evaluating how often a model identifies instances of a given class. These metrics will help enable us to assess the model's overall outcomes, but similarly, we want to dissect how well the models learned from or fit our data. This will allow us to increase the interpretability of our models and gain a better understanding of the decision-making process, also proving vital in evaluation. The way this is done is algorithm dependent, but this can range from visualisation techniques to training/validation loss data, alongside additional statistical measurements.

Through the experimentation process, we hope to uncover the extent to which fusion improves recognition accuracy, if it all, by comparing the performance of the combined approaches with their individual performances. While there aren't benchmarking results directly related to our context (shoe-outsole matching), we can identify similar problems using similar technologies, (ie: shoe print impression matching), loosely using their performance results as a reference point of what we can achieve. Moreover, the individual algorithms used are benchmarked on unrelated datasets to ensure they are performing as they should, therefore should any results be unexpected, we can be sure it is not in relation to how it is coded. The final dataset used contained 1860 images belonging to 13 classes, though there is an imbalance in the number of images obtained per class, therefore we reduced the usable dataset size to 1430 during CNN training. OpenCV 4.9.0, Python 11.0, and tensorflow 2.16.1 are used to conduct experiments, using a computer system with the following specifications: a 13th Gen Intel(R) Core(TM) i5-13450HX CPU @ 2.40 GHz with 16.0 GB RAM. Should current hardware prove to be insufficient, it may be more viable to use cloud-based services like Google Colab which offer additional TPU & GPU capabilities.

3 Development

Pertaining to the objectives and design discussed in section 1.1, the following section is purposed to outline the development process, accounting for the implementation process, all challenges encountered, resolutions, unintended changes in initial design alongside justified reasoning. We chose to develop the system modules using python due to its ease of use, extensive libraries and active communities, all of which help in providing an efficient development process.

3.1 Data Preprocessing

Preprocessing begins by converting the images into grayscale, reducing the complexity of the image's representation where each pixel value is representative of only intensity, thereby simplifying subsequent calculations. Next, to remove the background, a Gaussian blur is applied to the grayscale image to reduce details which is then thresholded to obtain the shoe's binary silhouette. A bounding box can now be formed using the 4 corners-most coordinates (the cornermost locations containing white pixels) from the silhouette, allowing us to remove extra background noise. However this approach is flawed for several reasons: (i) the sole does not occupy the whole box; some of the background noise persists and the quality of the box defined is highly sensitive to noise (ii) scraped images exhibit different characteristics like contrast and intensity distributions, therefore the optimal threshold value requires tuning for every image. This is initially addressed using a custom interface which lets me adjust the threshold value and visualise the results in real time (as shown in figure 2) . This includes a button to invert pixel values, aiming to address contrast discrepancies between background and foreground. We can then accept the value that gives the optimal result, though this system is still constrained to manual processes, therefore being time-consuming and tedious.

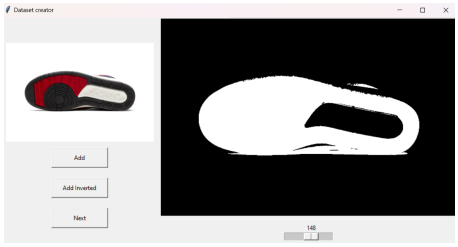

Figure 2: Pre-proccesing UI

In search of a more efficient procedure, we experiment with the following alternative approaches: (i) canny edge detection [16], and (ii) Otsu thresholding [17]. The former is a technique used in computer vision to find the edges present in an image, referring to regions in an image containing significant intensity changes. This approach is renowned for its ability to accurately detect edges in an image, despite noise or (low or high) contrast factors. This is an effective approach to demarcating the boundary between the foreground and background producing a well-tailored silhouette that addresses concerns of a poorly fitting bounding box. However it faces similar challanges as our custom interface, where performance is dependent on tuning the Gaussian filter parameter σ, which balances edge preservation and noise reduction. This is different for every image and particularly unfavorable given the nature of our dataset where there is a lack of standardisation, therefore we must work on the assumption that all images have different properties. Instead, Otsu thresholding presents an automatic thresholding algorithm that is suited to finding optimal thresholding values. It works on the notion that an image contains two classes of pixels: foreground and background. It thereby aims to find the threshold value that minimises the intra-class variance or maximises the inter-class variance between these two classes. We incorporate this approach into our interface, using it to generate the default threshold values for images within our interface, allowing us to further adjust outcomes if need be. After this, the images must be resized to a consistent size, though doing so may distort the image itself due to differing-sized images within the webscraped dataset. To maintain the image's aspect ratios, all images are resized by increasing width based on the calculated height and width ratio. Once we have the desired width we may notice the value for height may now fall short of the desired size, therefore additional padding can be added. If the height is in excess the image is cropped, though this is not an ideal approach due to loss of pattern information or key features. However, the dataset contains multiple instances of each given shoe model, therefore a cropped shoe should only have the same impact as an equivalent image augmentation, still serving as valid information. We finally apply a rotation, and erosion function to the preprocessed image, to increase our dataset size and incorporate variation that allows for more robust models. To rotate the images about a specified angle, we apply a rotation matrix about the centre point of the image. Erosion is emulated by adding grey (0-255) pixels to the image with a uniform distribution. We then threshold the grey pixels, where the threshold value, determines the intensity of erosion.

3.2 Feature Extraction

3.2.1 LBP

The LBP algorithm used is taken from the skimage library, requiring the optimisation of radius, number of points (P), and method parameters. Radius defines the size of the neighbourhood used around each pixel in LBP calculations, where a larger sized radius takes more points into consideration, thus placing emphasis on capturing global information. This comes at the expense of computational efficiency as we need to perform calculations on an exponentially increasing number of points as radius increases. Conversely, a smaller radius focuses on capturing intricacies in patterns, often overlooking the larger structural elements present. A number of points determine the number of sampling points considered along the radius-determined circumference. As we increase P, we can have a more detailed description of a local region at the expense of computation time. Finally, the method parameter determines the scheme used to encode patterns, we are interested in the uniform method which constrains patterns to have at most two bitwise transitions. After various iterations of tweaking these parameters, we settled on the following configurations: a radius of 1, 8 points, and a uniform method. We found negligible additions to performance when increasing values after this threshold, therefore considering any more points would only incur unnecessary computational costs.

As LBP calculations are dependent on pixel intensity, the performance of the operator can significantly degrade if there is not enough variation in intensity as we are unable to derive any meaningful information from the image. These conditions occur in low-contrast images, as mentioned in 3.1, where differences in pixel intensity are subtle, therefore producing the same LBP codes repeatedly across the image rendering the operator incapable of finding distinctive features. To resolve this, we use the image's extracted mask produced during the preprocessing thresholding stage. We first attempt to

perform LBP calculations on the binary mask and original grayscale image respectively, concatenating each feature vector to form an enhanced vector. Whilst LBP is not designed to work on black and white images, we can still perform calculations based on transitions between 0 and 1 pixels, providing an approach for detecting the edges of the image, and capturing additional structural elements. However, I ultimately settled on aggregating the binary mask onto the original image to produce a final image with more defined boundaries, allowing us to retain a more compact final vector. These new boundaries can now be amplified using openCVs dilate function which applies convulsions to the resulting image with a kernel to accentuate the gaps present.

3.2.2 SIFT

The sift Keypoints and descriptors are computed using OpenCV's SIFT feature detector, limiting the number of key points to 1000, in a tradeoff that captures the most insightful key points whilst aiming to minimise memory use and downstream time costs for matching. We apply the SIFT algorithm to all augmentations of the image, noticing that images that undergo the erosion operation have vastly more number of key points detected. This is as expected as the addition of grey pixels (during erosion transformation) adds significantly more changes in intensity and thus gradient information resulting in more key points detected. To determine the effectiveness of the SIFT descriptors we measure the quality of matches produced using a brute force matcher, an algorithm that exhaustively matches points within two separate descriptors based on a distance metric. A subsequent thresholding operation is used on retrieved matches, enhancing the selection criteria of what is deemed a valid match. We found that a value of 0.9 produced the best performance after several iterations of adjustments, where optimal accuracy was achieved at said value. The same approach was used to obtain the ORB descriptor as used in SIFT, resulting in a 32-bit floating point descriptor in contrast to SIFT's 128-bit. The results of their comparisons can be found in section 4.2, where we ultimately favour the use of SIFT despite the inferior computation times as our research focuses more on classification accuracy rather than efficiency. The differences in performance can be accredited to the amount of detail captured in the descriptor, where SIFT descriptors are based on gradient histograms in local neighbourhoods, capturing more detailed information about the image structure. On the other hand, ORB descriptors are generated based on intensity comparisons in a patch around each key point using BRIEF.

3.3 Feature Fusion

We first fuse LBP and SIFT features using concatenation. To do this, we must ensure both SIFT and LBP have the same dimensionality to make them compatible by flattening and padding the SIFT and LBP vectors to the same length. Normalisation serves to ensure equal vector contribution to the concatenated vector by mapping values to a standardised range. A challenge in this approach involves the high dimensionality of the resulting vector which adds a significant amount of complexity to calculations. Therefore we use PCA as a way to reduce dimensionality whilst maintaining the detail necessary for discrimination. This, however, is with limited success as PCA proves most effective when working on data with linear relationships. This type of connection is not shared between LBP and SIFT, as while they are both used to capture local information, they do by focusing on different aspects; LBP centres on local texture whereas SIFT is based on key points. In addition, the resulting non-sparse components produced often contain noise or lack visually interpretable patterns [18].

We overcome this using Bag Of Features (BoF), an approach to generate a vocabulary of features that are reflective of natural groupings of information in training data (section 1.3). Contrary to PCA, this approach allows us to aggregate features into common bags that enable subsequent algorithms to capture the underlying structures and relationships that aren't linear, thereby being better suited for our use. This is achieved using K Means clustering where we divide the feature space according to distance from a centroid, which forms the definition of a word. The goal is to have a dictionary of words that is comprehensive enough that it encompasses the most meaningful detail and representative enough so as to enable accurate class distinction. Using this vocabulary we can generate a dataset of feature histograms based on assignment to the nearest centroids. This provides us with a means to map

arbitrarily long vectors to a fixed-length vector output, addressing the issues of high dimensionality. The one apparent drawback is we must tune parameter K, which determines the number of centroids used and thus number of code words in the dictionary. We settled on 200 centroids, after several iterations of attempting to balance a good amount of captured detail whilst preventing overfitting.

As we now have the means to obtain a standardised dataset, we can now feed the normalised data to our classifiers, the first of which being SVC due to simplicity of implementation allowing us to quickly assess how viable this approach is. The choice of kernel is important as it decides the way we map data into high dimensional space which provides better separation in data. Given the complexity of our data, we use a polynomial kernel of degree 3 and a regularisation strength of 1. This configuration was found using grid search [19] (discussed more in section section 3.4), in a tradeoff that aims to promote simplicity yet being representative of the nonlinear relationships in data. As the success of the classifiers is dependent on the representations in data, we must ensure the groupings are meaningful. We do this using the following approaches; (i) visualisation using diagrams to enable the visual analysis of the groupings. It is important to note that as we are plotting high dimensional data on two axes so the output may not seem as expected, with clear groupings neccesitating additional dimensionality reduction. Though when using a large number of clusters, visualising cluster assignments may be difficult (ii) Using silhouette coefficient metric [20] we can measure the similarity within clusters compared to other clusters where a score of -1 indicates incorrect cluster assignment, and 1 is reflective of good separation.

3.4 CNN

To use deep learning the original web scraped dataset does not need to undergo the same preprocessing stages listed in section 3.1, as a CNN is well suited to working on coloured images, making accurate classifications irrespective of orientations, lighting conditions and noise. CNN performance is reliant on architecture, with structure design choices often being problem-dependent and requiring the consideration of a number of factors: the complexity of the task, size of the dataset, computational resources, and desired level of accuracy. Given that our dataset is moderately small, having around 1860 samples, we start off by assuming simpler models will be sufficient enough to capture patterns in data and to mitigate overfitting. Furthermore, we currently lack computational resources required to train complex models with deep arrangements of layers. We iteratively experiment with several CNN designs to determine one that boasts the best classification performance. The principles guiding the design of said architectures are twofold: (i) keeping the feature space wide and shallow in early development stages: capturing low-level features to collect as much local information as possible (ii) making it narrower and deeper towards the end: referring to the use of incrementally larger filter sizes to capture increasingly abstract and hierarchical features features. Until overfitting occurs, we continually add more layers to the model, at which point we can rectify the overfitting by using regularisation components. For instance, dropout layers do this by randomly dropping out neurons in training, forcing the network to learn the same encoding of information using a different ordering of neurons.

Hyperparameter-tuning is equally as important as defining the model's structure, providing performance improvements through a process of parameter tuning and optimisation. Some common hyperparameters include learning rate, optional padding, stride, batch size, optimisers and many others. To find the best hyperparameters for each model grid search is used. We do this by creating a grid containing a list of hyperparameters and their corresponding values. We then train a model for each possible combination of parameters and evaluate its performance on validation data. This method exhaustively tests every combination of hyperparameter values. The significant drawback in this approach is that it can be very inefficient in terms of training time and other costs, especially as the number of parameters to test grows, due the scope of this search. Instead of performing an exhaustive search, we use random grid search [21] to validate performance of random hyperparameter combinations from the search space (the grid). This prioritises computational efficiency and exploration through a selection of a diverse range of combinations (that may improve performance), without requiring the evaluation of every possible combination. Though as it is not comprehensive, it cannot

guarantee the optimal results, but instead provides good approximations. Our grid defines ranges for epochs, batch size, optimiser choice and activation function.

Following this, we start with a simple architecture defined as follows: Input → Conv2D(32 filters, 3x3) → MaxPooling2D(2x2) → Conv2D(64 filters, 3x3) → MaxPooling2D(2x2) → Flatten → Dense(128 units, ReLU) → Dense(13 units, softmax), and iteratively increase convolution, pooling layers, and filter sizes till an optimal configuration is found. We find that the initial CNN performs adequately well, successfully identifying key features necessary for class discrimination and mitigating overfitting through a lower number of parameters with respect to dataset size. However, due to the limited depth of this network, it may fail to capture more abstract features leading to a lower detection rate particularly for shoes with very similar in design. We therefore conclude on the model: Input → Conv2D(32 filters, 3x3) → MaxPooling2D(2x2) → Conv2D(64 filters, 3x3) → MaxPooling2D(2x2) → Conv2D(96 filters, 3x3) → Conv2D(128 filters, 3x3) → MaxPooling2D(2x2) → Flatten → Dense(128 units, ReLU) → Dense(13 units, softmax).

Pre-trained CNN models are especially useful in our use as they greatly reduce the training time and computational resources required to achieve a desirable outcome. This is done by capitalising on the low-level features and pattern information learned in the early layers of deep neural networks (i.e., edges, corner) which are applicable for use across different datasets. Instead of starting from scratch,we can leverage the transfer learning capabilities of pretrained networks, enabling us to benefit from the computational resources already invested in training the model. The exposure to extensive and diverse datasets during pre-training enables these models to gain rich, general-purpose representations that capture meaningful patterns and relationships within the data. The dataset we use for pretraining is ImageNet, being widely reputed as a comprehensive and diverse source. It contains over 14 million images distributed across 20,000+ categories with reliable labelling, therefore making it an ideal dataset for referencing generalisable features.

To investigate the impact of this approach we select two models renowned for their top-performance in image classification - VGG16 and ResNet50. Each architecture is distinguished by its unique design and innovations that contributed to its respective strengths and leading performances. VGG16 consists of 16 convolutional and fully connected layers. It is characterised by its relatively simple implementation, with a deep and uniform arrangement that makes it the reliable choice for many classification tasks. VGG16 produced notable performances when benchmarked on the ImageNet dataset, achieving 92.7% top-5 accuracy [22]. ResNet50, on the other hand, goes much deeper with an arrangement of 50 layers, though with such depth, innovations are required to resolve the degradation and vanishing gradient problem. This is resolved using skip connections which enables effective propagation of gradients during training, thus achieving a top-1 accuracy of 75% (on ImageNet)[23]. We can access these models from the Keras applications module, which provides an extensive repository of CNN models alongside pretrained weights. To use these models we must freeze the top layers of trainable parameters to prevent retraining, thus maintaining the information learnt during the initial training phase. We use these models as a convolution base, from which we can remove the top classification layers, replacing them with our own defined layers. We added the following additional layers: 256 fully connected neurons, dropout layer for regularisation, and 13 output neurons using softmax activation for classification.

3.4.1 Combining SIFT, LBP and CNN

We form a vocabulary based on our SIFT and LBP descriptors (following procedure outlined in section 3.3), from which we can generate the training data (feature histograms) by calculating word distances and assignments. As we've now acquired the training data, established the training-test split of 80/20 and encoded labels using one-hot encoding, we must now identify a CNN network best suited to learning representations in our new datasets. We have the choice of using 1D or 2D Convolutional Neural Networks (coined as 1D-CNN, 2D CNN). Operations in a 1D-CNN are fundamentally different from their 2D counterparts, as they focus on capturing sequential patterns along a single dimension through convolutions with filters of height 1. 2D ConvNets, however, specialise in extracting spatial

features across two dimensions (width and height), within input images using filters of varying sizes. The characteristics of our data lend itself nicely to the use of a 1D convolutional network as after the BoF clustering pipeline, each feature histogram is 1 dimensional. This is beneficial as we do not need to modify our data further to be compatible with the 1D-CNN network, as it can learn directly from the pattern relationships encompassed in feature histogram vectors. This is contrary to the needs of the 2D-CNN, where vectors will need to capture spatial information in an additional dimension for effective use. To represent the descriptors in such a way, we will form a 2D matrix where the X axis represents the words in the vocabulary, and the Y axis is a reference point for the largest codeword frequency in the dataset. By this, we mean that we iterate over all samples in the training data and retrieve the frequency for the most popular feature. This will be the upper bound limit on the Y axis. Each datapoint in the matrix (x,y), represents the frequency of a particular feature within that image. This way we can convert our dataset into 2D data, though the one downside in this approach is that the resulting matrix may be increasingly sparse depending on the Y value determined. As data is now in the correct format, we can now investigate performance using our defined CNN structures (section 3.4), as well as determining if we can also benefit from pretrained weights and models for added classification performance.

4 Testing, Results

As the success of a good classification system hinges on its ability to accurately categorise a query image despite variations in scale, translations, and rotations; we ought to measure how our system performs in relation to these factors using evaluation metrics as discussed in 1.2. Accuracy is a key indicator to determine prediction correctness. This is measured using top-n prediction scores (where $N \leq 5$), allowing us to account for the uncertainty amongst highly-likely outcomes. Though it's important to note that as n increases, the less informative the results are as we incorporate more randomness into predictions. To measure the resilience of our classifiers, for each query image we test our system against three of its variations including the original, rotated, eroded and rescaled versions of the query. To do this, each query image undergoes the same preprocessing pipeline as was used to get the original dataset, before we can retrieve the top-n prediction scores. Additionally we account for the speed of predictions, being the aggregate time for all processes until output. This is especially important with regards to the scalability of our system, where our motivations for building a comprehensive database rely on its ability to effectively handle increasingly larger datasets.

Method	**N = 1**	**N = 3**	**N = 5**
LBP	44.5%	63.6%	72%
SIFT	66.5%	68.4%	71%
LBP-CNN	28.6%	48%	67%
SIFT-Conv1D	65.3%	88.7%	92.3%
SIFT-Conv2D	53.2%	76.2%	87%
CNN	68%	85.5%	92%
Pretrained-CNN	**95%**	**96.5%**	**98%**

Table 1: Top-N accuracy for the different classification methods. As we increase N, we increase the set from which we consider whether the correct classification has been considered

4.1 LBP

To ensure the descriptor is working as it should, we benchmark performance on the Broadtz image dataset [12], of which research [11] shows accuracies of 91.2%. We achieved 75% using our implementation and parameter configurations. To get these results we perform distance calculations between the query image histogram and the dataset. Several measures of distance were used to test performance - Euclidian, Chi, and Kullback Leibler (KL)[24] distances. Euclidean distance often exhibited worse performance having a top-3 accuracy of 58%, whereas Kullback Leibler averages 64%. Here KL divergence is shown to be better at comparing probability distributions, capturing information regarding

the shape and the magnitude in which they differ. Euclidean distance on the other hand solely focuses on distances between points, potentially overlooking differences in distribution shapes. We see this in circumstances where two shoe soles have very similar shapes and distribution but may differ in scale where one image potentially was more effective at capturing the features. Though they are the same shoe, they are interpreted as different as Euclidian only considered straight-line distances. Contrary to expectation, LBP descriptors exhibited relatively poor classification performance on the dataset, having a top-1 accuracy of 44.46% (table 1) on normal shoe sole images and its augmented variations. This is likely attributed to its specialisations in capturing the finer texture elements within an image, as opposed to larger structural elements. I initially perceived its ability to capture intricacies (like edge & corner information) to be a strength, especially in circumstances where more detailed information is required to differentiate similar images.

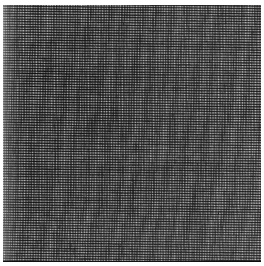

Figure 3: Image from Broadtz texture dataset

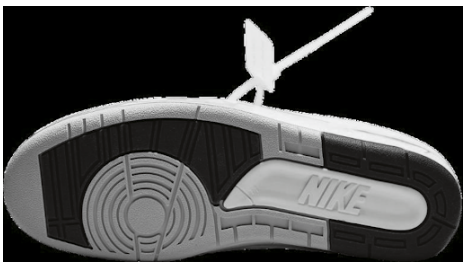

Figure 4: Image from training data

However, I failed to fully consider the characteristics of the images stored in my dataset, where the majority of shoes frequently feature large homogeneous regions. For instance, consider figure 3 (a data item from the Broadtz dataset), and figure 4, an image from my shoe print dataset. When analysing the differences, we notice figure 3 has far more granular detailing, containing far more changes and variations in intensity, all factors contributing to good LBP performance. On the other hand our image is composed of much larger shapes and patterns, with changes in intensity being much more subtle (especially prominent within the darker regions). We notice that dark regions in the shoe are large and continuous, directly impacting the LBP calculations performed, where there is a reliance on variations in local pixel intensities. To mitigate these effects, we tried different parameter configurations, where as discussed in section 3.2.1, increasing the size of radius and points arguments allows us to consider relatively larger regions within the image, though this had negligible impact on the outcome as it still only considers a relatively small area. The top-3 and top-5 accuracy of 63% and 72% respectively however imply that whilst the model needs to improve its precision, it still considers the correct classes amongst the top predictions.

MB-LBP was implemented with the intention of overcoming LBP's pitfalls as shown in section 1.3, where the summation of independently encoded regions should allow us to capture larger structural detail. Nevertheless, the performance surprisingly lagged behind that of LBP itself, achieving top-3 accuracy of 54.23% compared to LBP's 63.55%. This can be due to several factors: (i) using MB-LBP introduces a new parameter, the number of blocks, where a larger number of blocks divides the image into smaller segments, thus capturing finer details (and vice versa). We use 9 blocks which places greater focus on the larger structures in the image. (ii) Dividing the image removes the element of spatial continuity in an image, creating artificial boundaries that may introduce false features. It is difficult to determine if this is the root of the poor performance but it is still an important factor to consider. A solution to this issue is proposed in the paper [25], using a modified mb-lbp (mmb-lbp) that uses an additional grid shifted marginally to the right to incorporate the details lost in boundaries. However, I could not justify adopting an approach as the gains in performance appear to be negligible, and from our experimentation we have deduced that LBP is not suited in the first place.

4.2 SIFT

Compared to LBP, the SIFT descriptor averaged a much better accuracy of 66.5% on the normal dataset and its augmentations, as shown in Table 1. As discussed, such accuracy is expected as SIFT exhibits far greater capabilities in capturing key point information in a manner that is robust to transformations. Such stability is shown when breaking down accuracy for each augmented version of the image. For instance, normal, rotation, and eroded images averaged 68.46, 67.3%, and 41.5% respectively, showing a remarkable level of consistency. We however note that eroded images have a much lower detection rate than their counterparts, but this is expected with its cause linking back to large losses of information. Looking back, we discuss how this erosion augmentation was implemented by distributing grey pixels across the image with a normal distribution. This directly impacts the way the algorithm selects key points with SIFT relying on gradient information to select notable points. By introducing grey pixels, we are creating new local gradient information points where it never existed, and likely destroying them in their original places. This is also apparent when comparing the number of key points detected per image, where rotated and normal images have a relatively similar number of key points, and the eroded image has notably more. Using SIFT is not without cost, boasting much higher computational costs to calculate and especially match.

This reflects our biggest challenge when testing SIFT being the excessively long matching times that grow polynomially in relation to dataset size. We observe while testing where running the matching algorithm on a dataset of 260 samples takes 24.6 seconds whereas a dataset of 520 samples takes 106.9s seconds, approximately 4.5 times longer whilst only doubling the dataset size . This is due to the nature of the K-NN brute-force matcher, being an exhaustive algorithm that matches all points in one descriptor to all points in another to find the best matches. The matcher has a time complexity of O(mnd) * O(mn log n), where O(mnd) represents complexity for matching descriptors m, n with dimensionality d, and O(mn log n) represents sorting of nearest neighbours. Therefore it is impractical to execute this on the entirety of our dataset of 1860 images, and instead, we use a subset of 30 images for every class and transformation. This allows reasonable computation times, leading to more efficient testing.

Whilst ORB descriptors were not studied further, we had initially experimented with its use due to its promise to be a faster approximation of SIFT. To do so we used a separate dataset (generated using the same pipeline), though instead using ORB as the feature detector. In hindsight, this is not the ideal approach, as better quality and more fair comparison can be made if both descriptors are formed from the same dataset. However, during the formation of the original dataset, we did not include the original image information within each database data entry due to space limitations, thus finding the exact same images would prove difficult. The auxiliary dataset used had 732 samples though unevenly distributed amongst the 13 classes. The performance of ORB was quite shocking, achieving 82% accuracy on all augmentations, a sharp increase of about 15% from SIFT 67%. Given that there are class imbalances in the dataset, such results can be misleading as there is bias to classify a query under one of the more dominant classes, and achieve a higher accuracy rate because of this. Therefore we take these results with a pinch of salt as they are not formed from the same dataset, but still acknowledge its good performance despite being initially perceived as the lesser detailed descriptor. Had I tested this sooner I would have based downstream algorithms on ORB rather than SIFT, but refactoring all the code now would take a big effort so I remain with SIFT, though future work can further explore this.

4.2.1 SIFT-SVM

Figure 5depicts the SVM learning curve, illustrating how performances change given larger dataset sizes. In this graph we see that training score remains high irrespective of dataset size, whilst cross validation converges towards training score. As the cross validation score has not yet plateaued, it indicates that addition of more data can further improve performance. The SIFT-SVM achieved an encouraging accuracy of 0.78 using the following optimal configuration (obtained through grid search): a poly kernel with degree 3 and a regularisation coefficient of 10. With an average F1, precision, and recall score of

0.79, 0.80, and 0.79 respectively, we can reasonably deduce that the SVM has successfully identified the boundaries that maximise the separation between classes leading to good discrimination capabilities. To ensure these results are not due to overfitting, we use cross validation with 5 folds, in which we obtained consistent results across each fold (0.79 ± 0.01). Thus we can first, be sure that the outcome has not been affected by the train-test split and secondly, be confident in the models ability to generalise as it demonstrates good performances across five different unseen datasets.

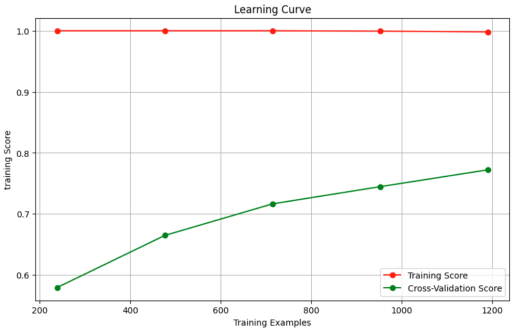

Figure 5: SIFT-SVM learning curve

4.3 CNN

In section 3.5, we discuss the use of simpler CNN architectures given the constraints in our dataset. Though a model that would be too simple would struggle to learn the underlying relationships, a factor becoming more evident when analysing accuracy by class. We notice a pattern emerging where shoes with visually similar shapes and patterns, such as Jordan 1s, 2s, 3s, and 4s exhibit lower accuracy scores of 11.76%, 58.82%, 46.15%, and 56.25% respectively. In contrast, later models like Jordan 10s, 11s, 12s, and 13s which feature more unique designs are detected at a higher rate of around 80.00%. 77.77%, 82.35%, and 83.33%. This is a particularly interesting trend as it further reiterates the influence of model complexity on performance, suggesting simple models may struggle to identify the more nuanced differences in images which prove vital for classification. Hence, we continue to add larger filters, convolutional layers and max pooling layers, monitoring performance on the testing and validation and testing sets to determine model limits.

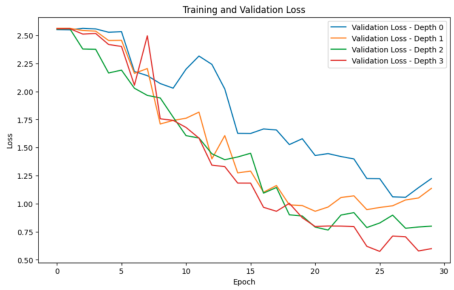

Figure 6: visualising Validation loss at different network depths

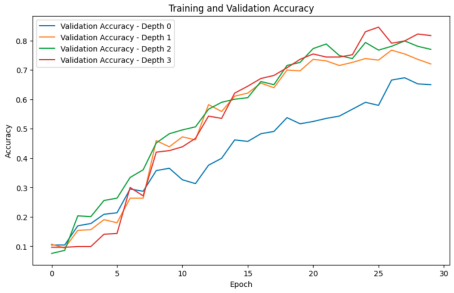

Figure 7: visualising Validation accuracy at different network depths

figure 6 and 7 describe a similar pattern, illustrating how validation accuracy and loss changes at different depths, with respect to the number of epochs. Lower validation loss scores suggest a better understanding of underlying structures due to better generalisations on the data, making less error. Similarly, high validation accuracy indicates good abilities in detecting true positives and true negatives, translating to better quality predictions. From these figures we notice that the best validation losses and accuracies are attained by more complex structures with higher depth, becoming more evident when breakdown accuracy per class, where Jordan 1s, 2s, 3s, and 4s now average 85% up from the average of 44.5%. When comparing our custom CNN performance against the individual algorithms (SIFT, LBP), CNN achieved significantly better results than LBP, and was marginally better compared to SIFT at an accuracy of 67.5%. Such results are exciting, showing that CNN is very effective in capturing all the spatial and hierarchical information necessary for discrimination, all the whilst doing it on a relatively small dataset. Thus if we extrapolate this performance, we can reliably assume that we can achieve further improvements using transfer learning, allowing us to benefit from models trained on much larger datasets.

4.3.1 Pretrained CNN

Before using the pretrained models on our dataset, we first benchmarked their performance on the ImageNet dataset to ensure the weights produce desirable classifications. Using a subset of 50 images, we achieved an accuracy of 94.4%, demonstrating its viability. As speculated, all the pretrained models used showed notable improvement in performance. Amongst those tested, VGG-16 proved to be best suited, likely relating back to the issue of model complexity, achieving an accuracy of 95% and top-5 accuracy of 98%. VGG16 is much simpler than ResNet50, using an arrangement of 16 layers as opposed to ResNets's deeper arrangement of 50. This makes ResNet way more computationally demanding, requiring larger datasets to achieve desirable results. Resnet uses a combination of techniques to mitigate overfitting for this purpose, ie, through the use of residual connections, and while it still achieved a good performance of 83%, it was suboptimal for our use case. This success is attributed , we are able to focus the entirety of our dataset into fine-tuning the model for our context, rather than learning more general pattern information. We also note that the classification performance has now exceeded all other classifiers, proving to be robust to various transformations and capable in distinguishing slight differences in shoe models. We interestingly observed the pretrained network achieved this result in a vastly fewer number of epochs of around 5, being a testament of the diversity captured within the pretrained weights, though performance after this point stagnates, likely due to overfitting and model complexity (especially noticeable in ResNet). However, it appears that 5 epochs was enough to capture all necessary representations as it attained the best performances when tested on the unseen dataset.

4.4 SIFT-CNN

The SIFT-CNN (1D) performed much better than LBP-CNN, achieving a performance that rivals SIFT at about 65.3%. We gain more interesting insights when comparing the top-3 and top-5 accuracy scores, where SIFT plateaus around 71% showing limited improvement, whereas the SIFT-CNN jumps to 92%. This is opposite to what we'd expect from SIFT, where considering a larger number of n should allow us to have a higher likelihood of selecting the correct label. This suggests a bottleneck in SIFTs ability to capture information, though we can immediately discredit this as we were able to achieve gains when using SVM. Therefore we can conclude the major pain points was the manner in which SIFT was classified or matched (using an exhaustive brute force-matcher). This is likely the reason as to why the SIFT-CNN worked quite well as rather than performing simple Euclidean distance calculations on a dataset harbouring complex relationships, we in fact learn more complex representations that contribute to more effective class discrimination. The 1D CNN structure is designed to learn from sequential information which lends itself very well to our original dataset, resulting in adequate performance without the need of introducing any redundancy in the form of padding.

Next we test the SIFT 2D-CNN using our newly formed descriptor (matrices) composed of the feature vocabulary on the X axis and max frequency count on the Y, allowing us to represent some form of spatial information. The ultimate goal with this approach is to see if we can exploit the benefits obtained from using pretrained weights, especially promising after observing the results on the image data (section 4.3). Similar to the testing process of the CNNs, we first investigated using our own simple architectures, before using pretrained VGG-16. Using our own CNN design, we achieved a top-1 and top-5 accuracy of 53.2% and 87% respectively, which is surprisingly worse than the 1D network. We first attribute this loss in performance due to the sparse representations of image data from adding a new dimension to the data. Within the 2D matrix representation for any given SIFT descriptor, there is only one data point per column, representing the frequency of a specific codeword (feature) within the original image. Therefore (for example) assuming maximum codeword frequency in dataset is 100 and we have 200 words in the vocabulary, we form a grid of dimensions 200*100, where only 200 locations (1%) contain meaningful data. This puts into perspective the number of zeros within each descriptor, reflecting a lack of sufficient information encoding where there exists large regions void of any relevant pattern information. This then leads to difficulties in learning meaningful representations, if any at all.

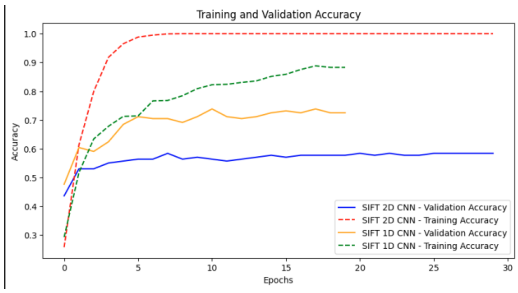

Figure 8: Training and Validation accuracy for SIFT CNNs (1D is red and blue and 2D is green and yellow lines)

Figure 9: Training and Validation loss for SIFT CNNs (1D is red and blue and 2D is green and yellow lines)

This is highlighted when plotting validation loss and accuracy for both 1D and 2D SIFT-CNNs in figures 8,9, where we observe the training loss being greatly lower than validation loss (for 2D SIFT-CNN). This is a clear indicator for overfitting as it suggests the model has fit too well to the training set but does not generalise in the slightest to the validation set. This is reflected in its training accuracy of 100% after just 5 epochs, whereas validation accuracy stagnates at 0.5, with no improvement in performance despite perfect predictions amongst training data. Finally, we adapt our pretrained VGG16 to classify our SIFT descriptors having expectations of even better results, inline with the results seen from the performance jump from normal CNN to VGG16 (as shown in table 1). Using this approach we achieved an accuracy of 24%, largely attributed to the amplified adverse effects of network and dataset refactoring discussed earlier. We must also be aware of the impact the pretrained datasets have on the network. Up until this point we have largely relied on ImageNet, capitalising on its diversity and ability to generalise. But it's worth noting that ImageNet does not capture all the categories or information of relevance to our specific task. Tasks involving objects and scenery have been extensively represented in ImageNet hence we saw good performance in our CNN models. The SIFT-CNN dataset, on the other hand, is a collection of clustered keypoint information, thereby falling outside of ImageNet's domain. To remedy this, I initially hoped to use different pretrained weights to monitor how it would impact (i.e, MNIST [26], COCO[27]), but the large majority of recognised pretrained datasets are based on visual information, objects or scenery.

4.5 LBP-CNN

The performance of LBP-CNN is a testament of how poor data representations affect consequent classifications. Given that we have determined LBP to be ill-suited for our dataset, evident in its relatively poor accuracy scores, we automatically have the assumption that the LBP-CNN will boast similarly poor performances. Despite the dataset and network being formed in the same manner to the SIFT-CNN (1D), we see significantly worse accuracy scores of 28.6%, down by 16% from LBP, suggesting either the dataset does not capture information relevant to the task, or incompatibility to our models (or both). This is because the representation built on top of poorly clustered features only compounds on its inability to generalise as the CNN struggles to learn meaningful patterns. This first becomes observable when visualising the clustering in a scatter plot where there appears to be an almost random relationship between centroids and descriptors. This is emphasised by a silhouette score of -0.053 being indicative of significant overlap between clusters and poor fit. To ensure this was not a matter pertaining to model complexity we made sure that overly simple models where used with 1 or 2 convolutional layers, though performance remained lacking and stagnant.

5 Critical Assessment

Reflecting back on our initial goals, we had set out to identify the effective classification techniques for shoe sole matching with the overarching objective of fully automating the data collection, feature extraction and classification pipeline.To assess the degree in which we fulfilled this, we will evaluate

adherence to the initial requirements as stated in section 1.2:

(i) is the final web-scraped dataset composed of high-quality, unobstructed images, being expandable to cover diverse shoe types with necessary mechanisms to enforce consistency?

While our current system demonstrates partial success in regards to the formation of the dataset, we acknowledge its limitations specifically regarding time concerns and class imbalances within formed datasets. We have been shown to successfully obtain a wide variety of class information during scraping, though as a precursor to using this data, we engage in a tedious manual filtering process in which we remove data not relating to our subject area. We currently lack the capabilities to automatically identify noise (irrelevant images) prior to its adoption within the dataset, therefore we are left with a cyclical approach to acquiring data as we can never really predict the final size of our dataset based on the original dataset size. In addition we have a very limited approach for scraping image metadata, where images are labelled based on information related to the query instead of the actual image properties. This constrains the amount of additional information we can capture. We have however seen some success in the preprocessing stages, where we have devised an efficient semi-automated pipeline for cropping, resizing, normalisation, labelling, and dataset updates. These transformations are successfully applied whilst preserving the underlying spatial relationships and information in the images. The datasets are also extendable as the scraper has been designed to scrape from a list of defined URLs, so expanding our search requires only updating this list to new sources.

(ii) To what extent were we able to form resilient and informative descriptors, perform fusion and investigate classification approaches?

To validate this requirement, we have rigorously tested the performance of feature and pattern analysis techniques used, namely SIFT, LBPs and CNNs. Each algorithm differs in process, complexity, and application and each generates a descriptor encapsulating varying levels of detail. By engaging in a comparative analysis, we journey from a more simplistic and intuitive approach (LBPs) to the black-box CNN model with greater capacities for capturing subtle and abstract information. According to our initial definition of success, we saw very limited success in the application of LBP where we found it to be unsuited for our dataset, based on poor benchmark results and sub-par accuracy results. It places greater focus on granular detail, often losing sight of global structures and more prominent features that in our case distinguish an image with greater certainty. We found that shoe sole patterns are typically composed of larger prominent detailing, meaning that the LBP is unable to take advantage of local variations in intensity, thus performance was not as we hoped. We saw dramatic improvements when using SIFT where we were able to successfully describe image patterns using keypoint information in a robust manner. This was demonstrated with accuracies staying consistent despite the query image undergoing rotation and erosion transformation. One particular area in which we did fail was in feature-level fusion, where we aimed to combine feature information of both LBP and SIFT to create a more descriptive feature vector. Our intended method used to accomplish this was concatenation and principal component analysis, but ultimately this failed for the following reasons: (i) In light of LBPs unsuitability to our dataset, it was first important to determine if there is any information can actually be gained from fusion as we risked degrading the current performance attained by SIFT. We did this by breaking down predictions made by class for each algorithm, observing the correct and incorrect predictions made to any find points of crossover. SIFT was correct in all the places that LBP was and more, essentially telling us that LBP is likely not adding any new perspective to the data but instead could be adding noise and uncertainty to predictions. This became more apparent when there was no performance improvement observed when we applied brute force matching the concatenated features (ii) secondly we rendered PCA ineffective with the knowledge that LBP features and SIFT features do not share linear relationships, thus undermining the assumptions made about underlying data relationships.

While feature-level fusion was not successful, we observed promising outcomes by using the extracted features as input for CNNs. This partially validated our assumption that learning additional representations from these informative features could provide improved performance. This was based on

the idea that our extracted features already encapsulate the most salient pattern information in a compact representation allowing for more efficient analysis. This was shown through the successful use of clustering to create meaningful feature groupings, allowing for standardised features sizing and enhancing the CNN decision-making process. A big challenge involved the search for a suitable CNN architecture and hyperparameters that could capture patterns in the data, whilst balancing time and resource costs. While initial findings show promising improvements with SIFT 1D-CNN proven by comparable accuracy metrics with traditional SIFT approach, transitioning to SIFT 2D-CNN introduced notable challenges, specifically in relation to representing the data in informative manners. Sparse data representation within the 2D matrix leads to ineffective pattern encoding as structures are largely unpopulated. This goes against what we hoped, as the 2D-CNN was shown to struggle grappling with these representations.

Reflecting back on the development process, we attribute a large amount of time cost to the structural design of the CNNs, requiring an iterative approach to determine optimal hyperparameters and structure. As previously highlighted, systems like grid search were employed to find reasonable configurations that would boast the best results. Though given the fact that we had to adapt the networks for different types of datasets, constant changes had to be made, and given the scope of the search space, grid search often proved to be an expensive solution. Hence a large part of the CNN development required randomness to ensure good amounts of exploration in a more efficient way.

(iii) non functional requirements

One of the primary objectives of my non-functional requirements was to address the challenges with scalability, emphasising the need for a modular design so that components can be independently developed and tested. We would be able to produce common interfaces that would streamline these processes, thereby preventing the need to constantly reinvent the wheel. This was of particular concern to me as I endeavoured to create a comparative study using a wide variety of algorithms, therefore I needed a process to minimise boiler plate code. This did not particularly translate for CNN designs as discussed previously, needs modifications based on dataset characteristics therefore requiring the use of several designs. Secondly, we emphasised the need for a database that could accommodate larger supplies of shoes, in a semi-structured manner. We made a large oversight with this respect, where we chose MongoDB to store our data. Whilst having exceptional capabilities with regards to flexibility and read/write speeds, we did not consider the pricing factor in early development. This became a noticeable problem when we overextended our storage limits, necessitating payment for subsequent use. Our fix to this was a temporary one, essentially recreating the dataset though using lossless compression on all descriptors. This will need to be properly addressed in future work.

5.1 Future Contributions

As reiterated throughout the course of the research, the overarching goal of this project has always been the full automation of the classification pipeline. Only by doing this can we accommodate the growing supply of shoes and provide meaningful predictions. Doing this requires automated filtering and dataset management processes, first to handle challenges discussed in section 2.1, pertaining to (i) irrelevant and noisy data, (ii) redundancy and near-duplicate images, and the proper labeling and categorisation of data. Our current implementation does this manually, using a scraping script that skips URLs already visited, though this does not account for actual image content. Research conducted in [15] presents an intuitive approach for real-time noise purification in social media (for natural disaster detection purposes), aiming to boost the robustness and quality of subsequent machine learning models. Deep learning has been shown to be successful in detecting irrelevant images by training images on data belonging to relevant labels and non-label labels - referring to images of partial or no importance. Deduplication is performed using perceptual hashing at key locations, using hamming distances to calculate similarity. These techniques may be replicated in our pipeline, to automate the sourcing of data whilst minimising the amount of noise, though we should be wary in our perception of a non-duplicate image, as by definition shoes of the same model will share large similarities, if not being identical. Hence this remains a large factor for future consideration.

While we have discussed future approaches for sourcing good quality data, we have yet to consider how new insights can be incorporated within existing models and, better yet our strategy for performing updates. Concept drift becomes an increasingly difficult challenge to handle as our dataset evolves, where the underlying relations between variables change over time in unforeseen ways and invalidate previously established models. Neural networks offer the benefit of being adaptable as their weights can continuously be updated to respond to changes in the dataset's underlying structure or the introduction of new class labels. First, we must determine when we want an update to be performed, whether that be periodically assuming we expect frequent changes, or based on some trigger. An example of such a trigger would be a drop in monitored performance using metrics like validation loss, etc. Next we decide our update strategy. One option is to only train the model on new data, placing greater emphasis on the learning from new insights. Alternatively, we can retrain the model on both old and new data, enabling us to retain previously learned patterns. Taking inspiration from transfer learning, we may choose to freeze the layers of our current models and extend them by adding new layers, allowing us to keep the generalisations and pattern information learned from early layers. Finally, we may choose to train a new model in an ensemble approach where classifications are made using a form of voting. Through future contributions we hope to decipher the best way to do this, thereby minimising human input and error.

Alongside the focus on automation, we must resolve the issues concerning the database used. While MongoDB is suited to our needs, it is limited to a capacity of 512MB (the basic subscription) and exists behind a paywall. Its further use will require funding, thus it might be best to consider other equally as capable models. MySQL is such an example, being a popular open-source relational database management system. It is widely known for its ease of use, scalability, and extensive active community support. Its primary benefits lies in its free use under the GNU General Public License (GPL), though commercial editions with additional features are also available. When migrating from a NoSQL database to MySQL, refactoring may be necessary due to differences in data structures and storage mechanisms. NoSQL databases often have flexible schemas or no schemas at all, while MySQL relies on a predefined schema for data storage. Therefore, adjustment will be needed to fit MySQL's relational model, potentially necessitating the refactoring of the database system.

5.2 5.2 Conclusions

In conclusion, this project has journeyed in the development of a robust shoe sole image classification system, addressing to the need for accurate categorisation of unknown shoe sole images in the absence of any oversights. Our approach closely resembles the exploration-exploitation model, where exploration involves delving into diverse ranges of encoding methods and information types (i.e, spatial, global, and local representations). The latter, exploitation, involves refining algorithms that exhibit promise, optimising their use in classification tasks. An example of this is reflected in the SIFT and CNN, where we saw promise in initial results of the respective algorithms and as such we investigated into merging the processes to further enhance recognition capabilities (ie, SIFT-CNN). Our methodologies had varying levels of successes, with the pretrained CNNs attaining the best perfomance at 95%. In contrast, LBPs, LBP-CNN, SIFT-CNN (2D), and pretrained SIFT-CNN (2D) exhibited poor performance. A common underlying factor observed in our less successful models was the manner in which we represented our data and its impact on subsequent predictions. Frequently, models that performed poorly were either ill-suited to our dataset or struggled to learn meaningful representations helpful to generalization. For instance, we found LBPs to be better suited for texture analysis contexts, in which our dataset featured limited texture characteristics. Also, pretrained weights used in the SIFT-CNN (2D) are intended for object recognition instead of descriptor classification. Nevertheless, our work demonstrated when the data is in the correct form and used with compatible models, we can gain further insights that heightened predictions. Going forward, future work should aim to fully automate our pipeline, with processes to filter image data and reintegrate new insights into our established models using possible stratergies outlined in 5.1.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[2] A. Ross, *Fusion, Feature-Level*. Boston, MA: Springer US, 2009, pp. 597–602. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5$_1$57

[3] R. Bowen and J. Schneider, "Forensic databases: Paint, shoe prints, and beyond," *NIJ Journal*, vol. 258, no. 2, p. 40, 2007.

[4] J. Christmas, "Private conversation," 2024, conversation held on March 18, 2024.

[5] A. Satpathy, X. Jiang, and H.-L. Eng, "Lbp-based edge-texture features for object recognition," *IEEE Transactions on image Processing*, vol. 23, no. 5, pp. 1953–1964, 2014.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li, "Face detection based on multi-block lbp representation," in *Advances in Biometrics: International Conference, ICB 2007, Seoul, Korea, August 27-29, 2007. Proceedings*. Springer, 2007, pp. 11–18.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[9] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 105–119, 2008.

[10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[11] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[12] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[13] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9*. Springer, 2006, pp. 490–503.

[14] V. Cherkassky and Y. Ma, "Practical selection of svm parameters and noise estimation for svm regression," *Neural networks*, vol. 17, no. 1, pp. 113–126, 2004.

[15] D. T. Nguyen, F. Alam, F. Ofli, and M. Imran, "Automatic image filtering on social networks using deep learning and perceptual hashing during crises," *arXiv preprint arXiv:1704.02602*, 2017.

[16] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved canny edge detection algorithm," in *2014 IEEE international conference on mechatronics and automation*. IEEE, 2014, pp. 577–582.

[17] X. Xu, S. Xu, L. Jin, and E. Song, "Characteristic analysis of otsu threshold and its applications," *Pattern recognition letters*, vol. 32, no. 7, pp. 956–961, 2011.

[18] A. De Pierrefeu, T. Löfstedt, F. Hadj-Selem, M. Dubois, R. Jardri, T. Fovet, P. Ciuciu, V. Frouin, and E. Duchesnay, "Structured sparse principal components analysis with the tv-elastic net penalty," *IEEE transactions on medical imaging*, vol. 37, no. 2, pp. 396–407, 2017.

[19] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 4, pp. 1502–1509, 2016.

[20] K. R. Shahapure and C. Nicholas, "Cluster quality analysis using silhouette score," in *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*. IEEE, 2020, pp. 747–748.

[21] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] M. Yamazaki, A. Kasagi, A. Tabuchi, T. Honda, M. Miwa, N. Fukumoto, T. Tabaru, A. Ike, and K. Nakashima, "Yet another accelerated sgd: Resnet-50 training on imagenet in 74.7 seconds," *arXiv preprint arXiv:1903.12650*, 2019.

[24] A. Vupputuri and S. Meher, "Facial expression recognition using local binary patterns and kullback leibler divergence," in *2015 International Conference on Communications and Signal Processing (ICCSP)*. IEEE, 2015, pp. 0349–0353.

[25] S. Alizadeh, H. B. Jond, V. V. Nabiyev, and C. Kose, "Automatic retrieval of shoeprints using modified multi-block local binary pattern," *Symmetry*, vol. 13, no. 2, p. 296, 2021.

[26] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.