A NOVEL INDEXING METHOD USING HIERARCHICAL CLASSIFICATION FOR FACE-IMAGE RETRIEVAL

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisor

Prof. Russel Pears

January 2020

By

Vibhav Sunil Chitale

School of Engineering, Computer and Mathematical Sciences

Abstract

Finding matching images of a person from large-scale databases efficiently and quickly is still a difficult challenge in face-image retrieval. Most of the existing systems cannot scale with the ever-increasing size of image databases or suffer from a loss in accuracy introduced due to an approximate search. Therefore, this research study proposes a novel indexing method using a hierarchy of classifiers that predict attributes such as age, gender, and ethnicity on which the database is indexed through a hash table. Only a small subset of the database is selected from the indexed hash table for the matching process, thereby reducing retrieval time while simultaneously maintaining low computational complexity. The hierarchical classifiers are trained using transfer learning with a pre-trained Convolutional Neural Network. To minimize the classification error introduced by the classifiers, a novel Probabilistic Back-Tracking algorithm is proposed that rectifies miss-classifications using conditional probabilities. Another method, Dynamic Thresholding, is proposed that dynamically sets a threshold for matching computations based on the predicted attributes. Rigorous testing of classifiers was conducted to assess their performance, which shows comparable results with state-of-the-art methods. Experimental evaluation of the proposed indexing strategy under hard tests on a large-scale database demonstrates a significant reduction in retrieval time and a considerable increase in retrieval accuracy over existing methods for face-image retrieval. Finally, statistical tests prove the importance of the proposed probabilistic back-tracking algorithm and the dynamic thresholding technique.

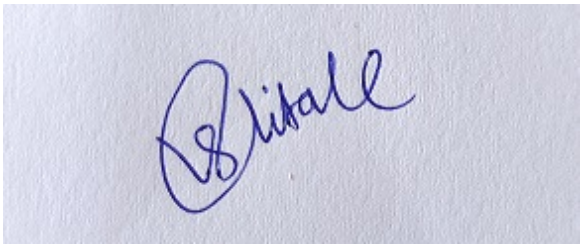
Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.



Signature of student

Acknowledgements

First and foremost I would like to thank my supervisor, Associate Prof. Russel Pears for his contributions and constant support throughout this research. I have learned a great deal about machine learning, data science, and research skills under his supervision. He has continuously inspired and pushed me to exceed my limits and steered me in the right direction.
I wish to express my sincere appreciation towards our student coordinator Sharda Mujoo who has been very welcoming, supportive and patient throughout the administrative process as well as during the Master's degree.
I want to thank all my friends who despite the distance always believe in me and are ready to help me. I am heartily thankful to Anuja Sahasrabuddhe for her continuous support and encouragement throughout the study.
Last but not least, my utmost thanks to my family members who have been the biggest support in all of my endeavours. This accomplishment would not have been possible without them. I cannot thank them enough.

Chapter 1

Introduction

1.1 Background and Motivation

With rapid advancements in face-recognition technologies, governments have recently started to incorporate facial-recognition systems by implementing them in various public e-services such as passport verification at airports, identity verification at offices, or even by local law enforcement agencies for criminal identification. These systems, as they automate most of the tasks, drastically reduce the time and manual labour which can be crucial, say for criminal identification.

The nucleus of these facial-recognition systems is the face-image data. Nowadays, face-images are publicly available with ease, thanks to the social media explosion. In 2013, Facebook published that its users had uploaded more than 250 billion photos while 350 million new photos were being uploaded each day (D. Wang, Otto & Jain, 2015).

The main motivation behind this research was its application and usefulness in identifying potential suspects as criminals. As law enforcement agencies generally tend to have billion scale datasets, efficient and accurate criminal identification is vital as the time taken to retrieve a potential match of a suspect's image could be one of the

deciding factors. A report published in 2014 showed the FBI's plans to include over 50 million new face-images into their dataset (Lynch, 2014). Their goal was to provide any investigative leads by searching their dataset with a suspect's image for potential matches.

Any false matches generated by the system would prove to be unnecessary public harassment of innocent individuals and can cause interference with their private lives. In fact, a couple of recent news articles published in 2019 (England, 2019) and (Dearden, 2019), showed a flawed implementation of UK police's use of facial-recognition system, which had astoundingly 81% and 96% misidentification rates respectively, wherein innocent individuals were falsely identified to be potential suspects as criminals.

Due to the million or even billion scaled nature of image databases, some of the issues regarding image retrieval that needs to be tackled are reduction in storage and computational costs, efficient image data manipulation, quick and accurate retrieval of images from the database.

1.2 Content-Based Image Retrieval

As large datasets normally require a tremendous amount of manual work and time for labelling, researchers have developed a unique way of image retrieval called content-based image retrieval (CBIR). CBIR systems work by retrieving most visually similar images to the given input query image, from a large dataset.
The steps involved in any image retrieval system usually include the following:

- Image representation
- Image indexing
- Image matching and
- Image re-ranking

In Image representation, features are extracted from images based on the exhibited shape or texture variations and are represented as n-dimensional vectors. These feature vectors should be compact and robust to transformations, scaling, or illuminations. In the past, feature extraction methods represented images using low-level descriptors such as Scale Invariant Feature Transform (SIFT) or Local Binary Patterns (LBP) or Wavelet patterns (Deselaers, Keysers & Ney, 2008). With recent advancements in deep learning, there has been a shift of focus towards feature learning, where a deep neural network automatically extracts features.

Image indexing allows for the efficient arrangement of the image representations (feature vectors) in a database for quick access. Some of the popular indexing techniques are based on hashing, inverted indexing, and clustering which are reviewed further in Chapter 2.

Image matching is responsible for computing similarity between the images and retrieving most visually similar images to the given query image. Some of the well-known similarity measures are cosine similarity, Euclidean distance, Manhattan distance, Minkowski distance, etc. (Cha, 2007).

Finally, the retrieved visually similar images are re-ranked removing any falsely matched images. CBIR systems are advantageous when searching a large scale dataset and are particularly useful in the retrieval of images from medical image datasets (CT Scans, MRI), or Scientific datasets (Jabeen et al., 2018) and (Lew, Sebe, Djeraba & Jain, 2006).

The following Figure 1.1 illustrates an overview of the working of CBIR systems.

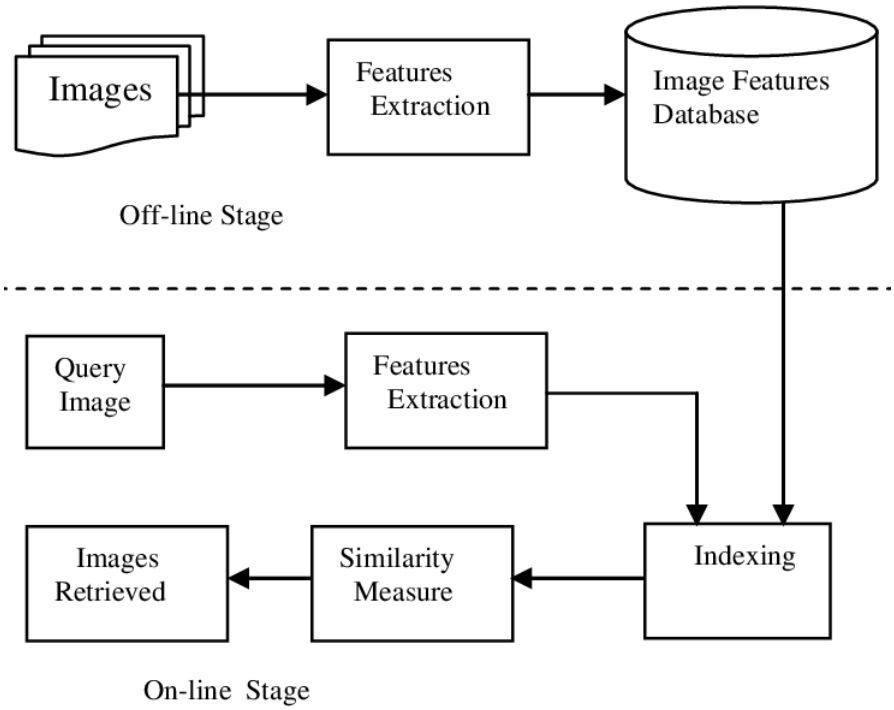


Figure 1.1: Content-based Image Retrieval (CBIR), retrieved from (Alkhawlani et al., 2015)

1.3 Face-Image Retrieval

Face-image retrieval is a kind of a specific application or use-case of content-based image retrieval. It works in a similar fashion as CBIR systems; however, some additional pre-processing steps such as face-detection, face-alignment, and necessary transformations must be incorporated before the usual steps of image representation, image indexing, matching and re-ranking as discussed earlier in the previous section. The key for quick and accurate retrieval of matching face-images, from a large dataset lies in three parts:

1. **Face-Image Representation** as to how face-images are represented and stored,
2. **The Indexing Strategy** used for storing these features and
3. **Face-Image Matching** and **Re-ranking** algorithms

For face-image representation, researchers previously used anthropometric features, which are geometrical ratios of various facial measurements (Kwon & da Vitoria Lobo, 1999) or used hand-engineered low-level features such as Local binary patterns (LBP) or Histograms of Oriented Gradients (HOG). Recently, deep-learning methods have been successfully used for automatic feature extraction from face-images. These features are then indexed using optimized hash tables (Tang, Li & Zhu, 2018), vector quantization methods (D. Wang et al., 2015) or inverted index methods (B.-C. Chen, Chen, Kuo & Hsu, 2013). Finally, the matching images are often retrieved quickly using variations of approximate nearest neighbour algorithms (ANN).
However, there are two major issues of concern associated with face-image retrieval that have not been adequately researched so far:

- Loss in accuracy due to quick search and,
- Increase in computational complexity with increased database size

Finding similar images from a very large database efficiently and quickly is still a difficult challenge in face-image retrieval. Most of the existing systems cannot scale with the ever-increasing database sizes or suffer from a loss in accuracy introduced due to an approximate quick search. The major disadvantage of these existing systems is having to operate and compute on the complete databases of a billion scale resulting in increased computational complexity. Thus, there is a need for a robust and scalable face-image retrieval architecture, which minimizes the loss in accuracy introduced due to quick search while simultaneously maintaining the feasibility of computational complexity when working with large databases.

1.4 Research Overview and Objectives

The main objective of this research was to build an indexing method that allowed for efficient matching and accurate retrieval face-images from large scale databases. This research study proposes a novel indexing strategy based on hierarchical classification. It was inspired by the success machine learning models in classifying high-level human attributes such as age, gender, or ethnicity which are reviewed thoroughly in Chapter 2.

The proposed methodology works using a divide-and-conquer strategy. The whole database is indexed hierarchically on the high-level human attributes such as the person's age, gender, and ethnicity. This essentially divides the database into smaller subsets. These subsets are mapped and stored in an index table. When an image is queried, the attribute values for age, gender, and ethnicity are predicted using trained classifier models. Using these predicted values, an index table is queried to retrieve a smaller subset of the database. The retrieved subset has the maximum likelihood of containing the matching images (different images of the same person) for a given query image. Since prediction time is almost negligible it does not add any significant overhead.

With the proposed indexing strategy, the database is filtered and reduced to a smaller subset. As only a smaller portion of the database is searched, the retrieval time is lessened, computational complexity is lowered, and the loss in accuracy is minimized due to the removal of any outliers or false matches.

The following Figure 1.2 provides an overview of the proposed methodology for face-image retrieval.

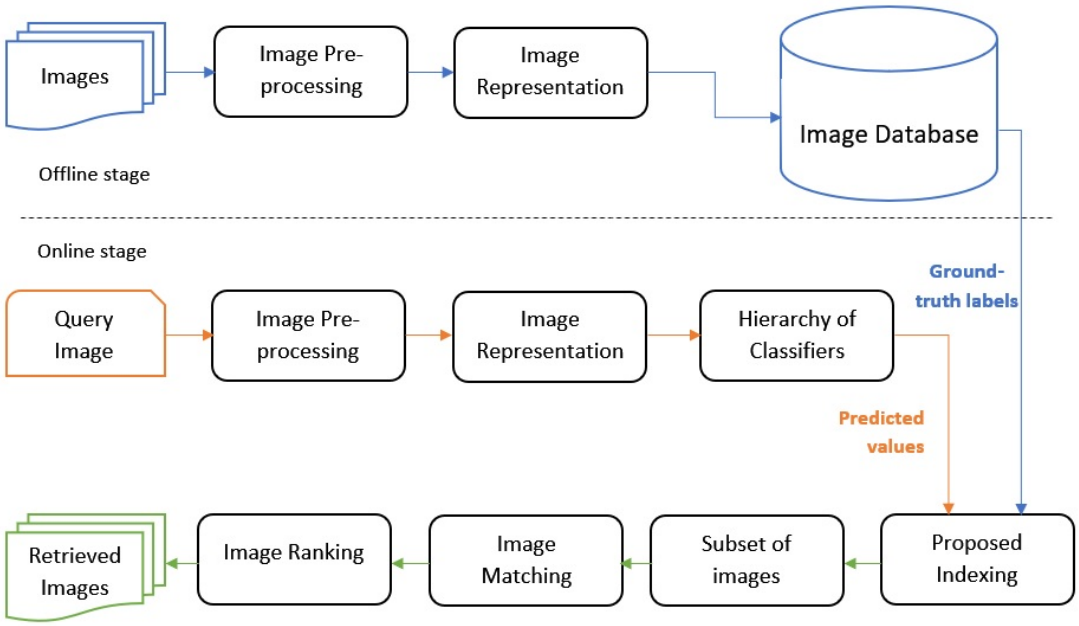


Figure 1.2: Overview of Proposed Face-Image Retrieval Architecture

The research questions tackled in this study are:

- Can the proposed indexing strategy based on hierarchical classification successfully filter the database to retrieve matching images quickly and accurately?
- How can the classification error be minimized which is naturally introduced due to the hierarchical classification?
- Can loss in accuracy with respect to quick search and retrieval be reduced with the proposed indexing method?
- How good is the quality of the features learned by a pre-trained neural network using transfer learning for classification tasks?

1.5 Contributions

This research study has produced the following contributions:

- A Novel database indexing method based on hierarchical classification for fast face-image retrieval from large scale databases.
- A Novel probabilistic back-tracking algorithm for minimizing the classification error.
- Dynamic thresholding of distance measures based on classification attributes such as ethnicity for improved matching performance and reduction in loss of accuracy as demonstrated through experiments.
- State-of-the-art age-group, gender and ethnicity classification results using pre-trained neural network and transfer learning.
- Performance comparison of different state-of-the-art face-image retrieval methods.

1.6 Thesis Structure

This thesis is structured as follows:

The second chapter presents a literature review that covers age, gender, and ethnicity classification, facial-recognition and face-image retrieval systems.

Chapter 3 provides background information on the existing technology used in this study, such as face-image pre-processing algorithms and the variants of pre-trained neural networks using various transfer learning strategies.

The proposed methodology is divided into two separate chapters. Chapter 4 details the hierarchical design, the classification attributes, and the underlying indexing structure. Chapter 5 focuses on explaining the proposed probabilistic back-tracking algorithm and the challenges faced during its implementation.

The experimental analysis and results of the proposed methodology are provided in Chapter 6 along with the statistical tests conducted for the proposed back-tracking algorithm and dynamic thresholding techniques.

Finally Chapter 7, concludes the study with a brief discussion on a few of the intriguing findings, achievements, limitations of the research and also presents possible future work directions.

Chapter 2

Literature Review

The proposed methodology draws on three strands of research, namely Classification tasks for Age, Gender and Ethnicity; Facial Recognition; and finally, Face-Image Retrieval owing to the interdisciplinary nature of this research study. Accordingly, each literature is reviewed and prior research is discussed within each of these themes.

2.1 Age, Gender, Ethnicity Classification

For the scope of this research, only three classification attributes were used: namely age, gender, and ethnicity.

The proposed systems' success was dependent on the performance of the classification models trained on these attributes. Hence it was necessary to achieve if not better state-of-the-art results in all of the above classification categories.

2.1.1 Traditional Approach

Earlier approaches for various face-classification tasks utilized the science of measuring facial dimensions and ratios based on facial geometry known as anthropometric models (Kwon & da Vitoria Lobo, 1999). (Farkas, 1994) provided a comprehensive overview

of face anthropometry. Their experimental data showed the successful application of face anthropometric measurements in characterizing facial growth based on age, gender, and ethnicity. (Alvi, Pears & Kasabov, 2018) have successfully used anthropometric models for age and gender classification. The face-images in their study were encoded as 7D features computed from distances and ratios between 68 facial landmarks. After applying a facial alignment algorithm, these features were fed into the NeuCube software based on a spiking neural network architecture proposed by (Kasabov, 2012). The experimental results achieved were 98% and 95% age-group classification accuracy on FGNET (Fu, Hospedales, Xiang, Gong & Yao, 2014) and MORPH (Ricanek & Tesafaye, 2006) datasets respectively. In terms of gender classification, 99% accuracy was obtained on the FGNET dataset.

With the advent of content-based image retrieval, numerous global and local image descriptors such as Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), Speed-up Robust Features (SURF) (Bay, Tuytelaars & Van Gool, 2006), Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG), Color Histograms surveyed by (Kaur & Rai, n.d.) were used for image representation in tandem with robust classification algorithms like K-Nearest Neighbours (K-NN) or Support Vector Machine (SVM) for age, gender, and other classification tasks. A successful classification of 73 different facial attributes such as age, gender, race, hairstyle, hair colour and more was presented by using the above-mentioned image descriptors with K-NN and SVM as the classification algorithms (Kumar, Berg, Belhumeur & Nayar, 2011).

(Chao, Liu & Ding, 2013) presented an age-oriented regression algorithm, which was a hybridization of K-NN and SVMs that captured the complex process of facial-ageing. Active Appearance Models (AAMs) (Cootes, Edwards & Taylor, 2001) were used as a face-encoding method that jointly extracted shape and texture variations from the face-images. As compared with other methods, AAMs achieved the lowest mean absolute error of around 4.4 for age estimation on the FGNET dataset.

(Eidinger, Enbar & Hassner, 2014) presented a unique approach for age and gender classification by using a modification of Support Vector Machines known as drop-out SVM. This method was inspired by the drop-out learning used in neural networks wherein some features for each sample are omitted by the network with a fixed probability. This helps the model to avoid overfitting the dataset (Choi, Lee, Lee, Park & Kim, 2011) and (Riesenhuber & Poggio, 1999). The face-images were encoded using an LBP and its variation called Four-patch LBP. The experimental results show about 79% and 76% accuracy was achieved for age and gender classification tasks respectively on an extremely hard unconstrained self-developed Adience dataset.

Although these methods achieved decent classification rates, most of the datasets used in these studies were of small scale and had controlled image conditions, or these datasets have now have been deprecated and supplanted by modern deep learning methods.

2.1.2 Modern Deep Learning Approach

Face-images can now be directly fed as an input to deep neural network architectures such as Convolution neural networks (CNN) for feature extraction as well as for classification. When used for feature extraction, deep learning models can learn and produce compact low dimensional features. Due to their low dimensionality, such features are computationally inexpensive to process as compared to traditional handcrafted features based on LBP or HOG. Such methods based on deep neural networks have been successfully used for gender and age classifications on large scale datasets.

The results presented in an earlier study by (Eidinger et al., 2014) were further improved significantly by using deep learning methods based on CNN. (Levi & Hassncer, 2015). An increase of 10% in prediction accuracy was seen for both age and gender classification on the same private Adience dataset of unconstrained images.

(Dhomne, Kumar & Bhan, 2018) have used a Deep-CNN based on VGGNet architecture developed by (Simonyan & Zisserman, 2014) and achieved 95% gender classification accuracy on Celebrity Face Dataset (Liu, Luo, Wang & Tang, 2015) with more than 100K images. Similar studies by (Jeong, Lee, Park & Park, 2018) and (Duan, Li, Yang & Li, 2018) have used deep learning architectures for automatic facial feature extraction or as a direct classifier for age estimation and gender recognition.

Based on the performance of deep learning models on age and gender classification, a similar trend was visible for ethnicity classification tasks. Deep learning methods were consistently outperforming traditional machine learning algorithms.

A comparison study by (H. Chen, Deng & Zhang, 2016) of a 3-class ethnicity classification problem showed that K-NN and SVM produced the worst results with a classification accuracy of only 60%, whereas a CNN achieved almost 90% accuracy on the same dataset. Other recent studies by (Vo, Nguyen & Le, 2018) and (Heng, Dipu & Yap, 2018) have produced comparable results of up to 90% on similar datasets for ethnicity classification using deep learning architectures.

It was evident from these studies that the current trend for achieving state-of-the-art results in age, gender, and ethnicity classification tasks was by utilizing the power of deep learning architectures either for feature extraction or classification. There was a significant performance gap in terms of feature extraction on large scale datasets between deep learning based methods and traditional methods such as anthropometric features or LBP or HOG. One of the most significant advantages of deep learning methods was that feature extraction could be automated, thus the tedious task of manual labelling (in case of anthropometric models) was avoided. It was also possible to prevent the high dimensionality of generated features associated with traditional feature extraction methods of LBP and HOG. Moreover, deep learning methods are resistant to limited data availability, meaning they tend to avoid overfitting on small datasets (Levi & Hassncer, 2015), which is a common problem associated with image classification tasks.

Therefore, deep learning models have gained popularity over traditional approaches in recent years due to their significant prowess in automatic feature extraction and high performance, particularly on large-scale datasets.

2.2 Facial Recognition Systems

2.2.1 State-of-the-art face-recognition

The standard facial recognition pipeline has been drastically changed over the past few years. A facial recognition pipeline usually included a face detection algorithm followed by a face alignment operation, which was followed by a feature extraction method that represented face-images as a vector to be used as an input for various machine learning algorithms.

Earlier studies by (Li, Wang & Zhang, 2007), (Geng & Jiang, 2009) and (D. Huang, Shan, Ardabilian, Wang & Chen, 2011) relied heavily on feature extraction methods such as Haar-like features, SIFT (Lowe, 1999), SURF (Bay et al., 2006) or LBP. These methods worked by using raw pixel values of input images and then transforming them into n-dimensional feature vectors, which can later be used as an input to train classification models. Such features, however, were required to be handcrafted and generally had high-dimensionality.

The performance of facial recognition systems using the above-mentioned feature extraction methods was limited to small scale datasets. These methods are completely dominated and outperformed by modern deep learning methods on large scale datasets. There has been a shift of focus towards feature learning instead of feature extraction. Feature learning involves the application of deep learning methods to automatically learn a feature extractor based on the statistical properties of the training data. The learned features could then be extracted from a single layer of the deep neural network

and can be used as an input to classical machine learning algorithms. This process is fully automatic, however it is a black-box method meaning that the extracted features are non-interpretable or extremely hard to understand on what basis the neural network computed these features. This is in contrast to anthropometric features that could be easily interpreted.

In 2014, a paper published by Facebook (Taigman, Yang, Ranzato & Wolf, 2014) called DeepFace, described the facial recognition system developed by Facebook's research group, which achieved 97.5% recognition accuracy on the benchmark dataset of Labelled Faces in the Wild (LFW) (G. B. Huang, Ramesh, Berg & Learned-Miller, 2007). The deep learning architecture was an end-to-end Siamese network with 9 layers and 120 million parameters which was trained on more than 4 million images. The deep network was used for feature learning which generated a facial representation based on a cross-entropy loss function. This loss was minimized by updating the parameters of the network using a stochastic gradient descent algorithm with standard error back-propagation. A 3D face frontalization method was used during pre-processing to create a 3D face model. A series of affine transformation operators frontalized or aligned the input image based on the generated 3D model. This step enabled for better alignment of unconstrained face-images leading to higher face-recognition accuracy. This paper represented a breakthrough in facial recognition systems as it showcased the practical application of deep learning in a real-world scenario with a high recognition accuracy of 97.5% on the benchmark LFW dataset.

In 2015, following Facebook's release of DeepFace (Taigman et al., 2014), Google released a paper called Face-Net (Schroff et al., 2015) which used a deep CNN architecture to learn the facial features which they called as face-embeddings. Experimental results showed that 99.63% accuracy was obtained for facial recognition on the benchmark LFW dataset (G. B. Huang et al., 2007), which bettered the results obtained by Facebook. The Face-Net outputs a 128D feature vector using a triplet loss function

based on a Large Margin Nearest Neighbour Distance Metric (Weinberger & Saul, 2009). The triplet loss function was designed and trained on 2 positive images (i.e., images of the same person) and 1 negative image (i.e., an image of a different person). The deep CNN was trained such that the intra-class distances within the positive pair were minimized, and the inter-class distances between the positive pair images and the negative image were maximized. The advantage of triplet loss function was that it reduced the problem of intra-class variation as it minimized the distance between the positive pair images which were of the same person with slight variations in pose or illuminations.

Further papers followed in the footsteps of Facebook and Google by using deep learning methods for facial recognition. A deep neural network architecture known as DeepID3 (Sun, Liang, Wang & Tang, 2015), achieved 99.53% accuracy on the LFW dataset (G. B. Huang et al., 2007). This network was trained on a private dataset which comprised of only 200,000 images when compared with Google and Facebook's extremely large datasets, and yet the results obtained were better than Facebook's (Taigman et al., 2014) and comparable to Google's (Schroff et al., 2015) results on the LFW dataset.

There is a general agreement on the fact that the performance of features learned using deep neural networks have now saturated the benchmark LFW dataset (Learned-Miller et al., 2016), (Masi, Wu, Hassner & Natarajan, 2018) and (Wen, Chen, Cai & He, 2018). The training data from these benchmark datasets of around a few thousand is insufficient for training deep neural networks. As shown from the below table 2.1, all the state-of-the-art methods trained the deep networks on their own private datasets having many millions of images. These deep learning methods, combined with the large training datasets have now been shown to surpass human capabilities on the LFW dataset (Schroff et al., 2015) and (Learned-Miller et al., 2016). Thus, new publicly available benchmark datasets consisting of millions of images must be created for

comparing the performance of facial recognition using deep learning methods under unconstrained conditions.

Table 2.1 shows the state-of-the-art face-recognition results and the feature extraction methods with the dataset size used for training the models.

Table 2.1: State-of-the-art Face-Recognition performance on LFW dataset
Source, Learned-Miller et al., 2016

Reference	**Algorithm Name**	**Feature Extraction**	**Train Dataset (Millions)**	**Accuracy**
(Berg & Belhumeur, 2013)	POOF-HOG	HOG	NA	0.9280
(D. Chen et al., 2013)	High-dim LBP	LBP	0.099	0.9517
(X. Cao et al., 2013)	TL Joint Bayesian	Joint Bayesian	0.099	0.9633
(Taigman et al., 2014)	Facebook's DeepFace	CNN	4.4	0.9735
(Sun et al., 2014)	DeepID2	CNN	0.16	0.9915
(D. E. King, 2009)	DLIB	Res-Net	3	0.9938
(Sun et al., 2015)	DeepID3	CNN	0.16	0.9953
(Schroff et al., 2015)	Google's FaceNet	CNN	200.0	0.9963

2.2.2 Application of State-of-the-art Facial recognition systems

(Y. Wang, Bao, Ding & Zhu, 2017) applied a face-recognition system on real-world surveillance videos using a deep learning architecture. Instead of building a deep neural network from scratch, a pre-trained model based on VGG architecture (Simonyan & Zisserman, 2014) was used for fine-tuning its parameters. The pre-trained model had 8 convolutional layers and was trained on 2.2 million images. Face-detection and tracking helped extract the face-images from a real-world captured surveillance video sequence, which generated a large dataset.

A unique graph-clustering method was used to prune this dataset by removing duplicate and erroneous images. The graph was constructed based on a set threshold value which identified a central node image for each identity. Similar images to this central node image were determined based on a cosine similarity metric. Edges were created by linking these similar images to their identified central node image. Many such central nodes and their corresponding similar images formed a group of distinct clusters. Each cluster consisted of a central node image, and similar images linked to it formed the graph edges. As a result of this graph clustering method, a robustly pruned dataset was generated, and a 92.1% face-recognition rate was achieved on real-world video surveillance sequences by fine-tuning the pre-trained VGG model.

A study by (Zeng, Zeng & Qiu, 2017) applied deep learning methods for analysing forensic face verification of suspect images from a video sequence. The deep learning architecture used for face-recognition and verification was based on Google's Face-Net architecture (Schroff et al., 2015). Using the proposed method, the suspect's images were successfully captured from a video sequence and verified. However, the proposed system came up short when the suspect's face-images had black spots or scars on their faces, suggesting a potential limitation of deep learning based methods for facial recognition or verification.

Another study by (Zhou, Wan, Kuang & Tong, 2018) proposed the use of a facial recognition system with deep learning for unmanned supermarkets. A deep convolutional neural network of 4 layers was used. Motion blur and a large number of customers shopping simultaneously were the two main problems of focus in this study. The study proposed the usage of data-augmentation techniques to introduce motion blur during the training phase, which resulted in increased recognition accuracy when the supermarket camera captured blurry images. To reduce the face-recognition error, they used a gender classifier trained on the same images which filtered potential mismatches. The deep CNN and the gender classifier used in this study were trained using tensorflow and python, which achieved more than 95% face-recognition rate and 96% gender classification rate respectively.

2.3 Face-Image Retrieval Indexing Techniques

Face-image retrieval techniques can be roughly categorized based on the different indexing strategies used for storage and retrieval of face-images from large scale datasets.

2.3.1 Hashing

Hashing has been widely utilized for Approximate Nearest Neighbor (ANN) search due to its fast retrieval speed and low storage cost. Compact binary hash codes can be used to build the database and to compute similarities between two images using simple logical operations like XOR.

A hashing method based on deep learning was introduced by (Jang, Jeong, Lee & Cho, 2018) called as deep-clustering and block hashing. The focus of this study was to alleviate the problem of intra-class variation and inter-class similarity between different face-images. The intra-class variation problem is due to illumination, lighting, or the

same people presenting different poses when their images were captured. Inter-class similarity groups people who look similar. A deep neural network based on VGG (Simonyan & Zisserman, 2014) was used along with a new block-hashing method. A centre clustering loss and a classification loss function were used to minimize the distances between face-images and the cluster centres belonging to the same class, thereby reducing both the intra-class variation and inter-class similarity problems. The deep neural network had 10 convolutional layers and a maxing pooling layer with a stride of 2 which reduced the image size by half in each pass.

The block hashing layer was used as the last layer in the network, which produced compact binary hash codes from the features extracted by the fully connected layer prior to the block hash layer. The k-dimensional features were divided into M different blocks. Since each block was generated from a face image descriptor of a separate image slice, it effectively contained or preserved some characteristics of facial representations. A softmax function was used to calculate the output from each of these M blocks, and the final output was concatenated to generate a k-bit binary hash code representation of the input image. K was a tuneable parameter and was obtained based on K number of nodes present in the block hashing layer.

The experiments were performed on the Youtube Face (Wolf, Hassner & Maoz, 2011) and FaceScrub datasets (Ng & Winkler, 2014) by comparing mean average precision (mAP) over the test images which happened to be around 99%. A result comparison with and without their proposed centre clustering method clearly highlighted its importance based on the high mAP values achieved with it.

A paper published in 2018, (Tang et al., 2018) proposed a novel hashing technique based on deep learning called deep supervised hashing. This approach was similar to feature learning, except in this case, a deep convolutional neural network was used to automatically learn the hashing function instead of a feature extractor. The deep neural network was used to generate face-encodings, and the last layer or the output layer

was used to generate a k-bit binary hash code by quantizing the previously generated face-encodings. Similar face-images generated similar binary hash codes and logical XOR operation was used for distance computation between face-images. The network had 4 convolutional layers with max-pooling which were used for feature extraction. The last layer in the network was designed to use a sigmoid activation function which mapped the input image data to the k-bit binary code.

This method was novel as it was based on minimizing both classification as well as quantization errors. The supervised deep hashing generated binary codes which preserved the similarity or dissimilarity between the images. Thus, these hash codes were also useful in the classification or prediction of image labels. The experiments were performed on 3 popular datasets namely, Youtube Face dataset (Wolf et al., 2011), FaceScrub dataset (Ng & Winkler, 2014) and CIFAR-10 dataset (Torralba, Fergus & Freeman, 2008). The proposed method using supervised deep hashing achieved the highest performance on all three datasets as compared with other hashing methods.

2.3.2 Product Quantization

Product Quantization is a technique where high dimensional feature vectors are decomposed as a Cartesian product of low dimensional subspaces and quantizing each subspace separately. The two fundamental parameters for product quantization are the number of subspaces and size of the codebook which are generally represented as **m** and **z** respectively.

A study known as Face-Search at Scale (D. Wang et al., 2015) used product quantization for performing an exhaustive search of face-images in large scale datasets. This paper proposed the use of a cascaded face retrieval system with 80 million images in a gallery under both closed-set (target query present in the database) and open-set protocols (presence of target query in a database is unknown).

The cascaded system had 3 phases:

1. **Template Generation:** A convolutional neural network based on VGG network (Simonyan & Zisserman, 2014) was used for generating a 320D face-image encoding. The model was regularized using a dropout learning method, where 60% of the features were retained while the remaining 40% were dropped using a softmax loss function and standard back-propagation method during the training of the network.
2. **Face Filtering:** Product Quantization was used to retrieve a list of top-k images which were most similar to the input query image. The parameters for product quantization were set empirically to the number of subspaces as 64 and the size of codebook as 256 which produced a 512 bits code. Cosine similarity and Euclidean distance were used as the similarity measures when computing the distances between the images for retrieving a list of top-k most similar images. Based on the results presented, it was found that cosine similarity performed better in accurately retrieving a list of top-k images as compared to Euclidean distance.
3. **Re-Ranking:** The last step was to re-rank the retrieved k-images using several face-matchers having different similarity measures. This study also mentioned the use of a commercial face-matcher for this re-ranking step.

The experimental results showed their proposed system even outperformed the commercial software used for face retrieval. The datasets used were LFW (G. B. Huang et al., 2007), IJB-A (Klare et al., 2015) and CASIA (Yi, Lei, Liao & Li, 2014). The average search times per input image were 0.9 seconds on a gallery size of 5 million images and about 6.7 seconds for a gallery size of 80 million images which appeared to be a very impressive performance on a huge dataset of 80 million images. The experiments were run on a PC with a single Intel(R) Xeon(R) CPU clocked at 3.10GHz. However,

the deep features were extracted using a very powerful Tesla K40 GPU. The study also demonstrated the search performance on a real-world case study involving the video frames of the Tsarnaev brothers of the 2013 Boston marathon bombing. In this case study, the proposed system was able to find one of the suspects' images at rank 1 in 1 second on a 5 million gallery size and at rank 8 in 7 seconds on an 80 million gallery size.

2.3.3 Inverted Indexing and Bag-of-Words

An inverted index is an indexing strategy that is widely used in all forms of image retrieval problems. It is a convenient data structure used for indexing and distance computation between image descriptors. Typically, K-means clustering is applied over the image descriptors to create a set of visual words. In image retrieval, this set is called as a visual vocabulary. Each descriptor is then quantized in its nearest visual word (or nearest cluster centroid) making the visual vocabulary. Inverted indexing technique or bag-of-words or visual vocabulary trees are known to provide excellent performance on million scale image datasets with standard consumer hardware (Uriza, Fernández & Rais, 2018). There are various implementations of inverted indexing or bag-of-words model; however using a hierarchical clustering method for creating a vocabulary tree based on an inverted index was shown to be more advantageous due to its scalability towards large datasets, its ability to traverse only smaller sections of the dataset and add images into the dataset with ease during run-time (Nister & Stewenius, 2006) and (Uriza et al., 2018).

(Wu, Ke, Sun & Shum, 2011) proposed an inverted indexing technique based on a combination of both local and global image descriptors. The local features used in this study were a histogram based T3hS2 descriptors (Winder & Brown, 2007), which were computed from 5 square face patches of two mouth corners, two eyes

and one nose tip. These descriptors were then quantized to form the inverted index table. This quantization process degrades the retrieval performance as two images of the same person may be quantized differently and clustered in a different group inside the inverted index table because of intra-class variations like pose, lighting conditions, or expressions. The solution to this problem proposed in this research was to use an identity-based quantization method where the descriptors with variations in pose and illumination had a strong probability of being quantized into the same cluster. This probability was calculated after comparison with multiple reference images.

The local features described above were used to traverse the index and retrieve a list of top candidate images matching with the query image. These candidate images were re-ranked using a small 40-byte Hamming signature based on a new multi-reference distance metric. The multi-reference distance metric computes the average distance of each candidate image to the multiple reference images of that candidate. Using this distance, the candidate images were re-ranked. Experiments were conducted on the LFW dataset (G. B. Huang et al., 2007) of varying sizes by selecting 220 query images and computing the mean average precision (mAP). The retrieval time taken per query image was around 0.5 seconds on a 1 million image dataset using a 2.6Ghz CPU and 16 GB of RAM.

(B.-C. Chen et al., 2013) further improved on the above study by (Wu et al., 2011) with the addition of human attributes such as ethnicity, gender, and hairstyle. A novel indexing technique known as attribute embedded inverted indexing was implemented for face-image retrieval. To overcome the intra-class problems of pose and illumination faced even with the traditional inverted indexing methods, they proposed the use of high-level human attributes such as gender, ethnicity, or hairstyle that could be easily captured from face-images in tandem with the inverted indexing technique. Local Binary Patterns (LBP) were used as the feature extraction method. This LBP was comprised of 59-dimensions for each patch and was computed from 5 different face

patches such as the nose, eyes, and lips. The two core components of this study were the attribute-enhanced sparse coding and the attribute embedded inverted indexing:

1. **Attribute-Enhanced Sparse Coding:** Sparse coding technique was used to solve a linear optimization problem of feature representations where each extracted feature vector from a single patch was prevented from becoming arbitrarily large. A dictionary or a visual vocabulary of features was created such that the actual features were a linear combination of columns in the dictionary. The high-level human attributes such as gender, ethnicity, and hairstyle were used to divide the large dictionary into separate sections based on their values. For gender, the dictionary was divided into two sections, one with all males and the other with all females and similar methods were followed for the rest of the attributes. Attribute detection was done automatically using support vector machines with radial basis function as the kernel based on the method proposed by (Kumar et al., 2011) for their study on 73 different face attributes classification. Finally, a sparse representation was generated for each face-image.
2. **Attribute-Embedded Inverted Indexing:** A binary signature was assigned to represent each human attribute. This signature along with a codeword (sparse representation of face-image from the previous step) was used to build the inverted indexing structure. Efficient retrieval was possible with a simple logical XOR operation which computed similarity scores based on Hamming distance.

The LFW (G. B. Huang et al., 2007) and PubFig datasets (Kumar, Berg, Belhumeur & Nayar, 2009) were used for experimental results comparison. More than 90% mean average precision (mAP) was achieved with around 120 images used as the test or query images. The average retrieval time using sparse coding with linear search was around 1.73 seconds, while sparse coding with inverted indexing boosted the retrieval times significantly by a factor of 57.7, which resulted to a retrieval time of only 0.03 seconds

on LFW dataset with 13,000 images. The experiments were ran using a single core Intel(R) Xeon(R) CPU clocked at 2.4GHz. Based on these experimental results, it was claimed (but not proved experimentally) that the retrieval of an image from 1 million image database would be around 0.2 seconds using the attribute embedded inverted index technique.

2.3.4 Human Attributes and Hierarchical Clustering

(Park & Jain, 2010) made use of demographic attributes such as gender and ethnicity along with their proposed facial mark detector for accurate face image retrieval. A blob detector based on the Laplacian of a Gaussian filter or Difference of Gaussian was applied for facial mark detection. Face similarities were computed based on the detected mark and the location of the mark. The experimental results revealed that 92% face-recognition accuracy was achieved on FERET database (Phillips, Moon, Rizvi & Rauss, 2000). The retrieval time for each image query was about 15 seconds.

(Kumar et al., 2011) used Support Vector Machines with a radial basis function as the kernel for successfully classifying 73 different attributes that were extracted from face-images from the LFW dataset (G. B. Huang et al., 2007) with high classification accuracy rates. Colour histograms based on RGB and HSV colour spaces were used for feature extraction along with the intensity and magnitude of edges detected from the image. The experimental results achieved 85.8% gender classification accuracy, while 96.5% accuracy was achieved for sunglass detection, whereas ethnicity classification was around 92%.

(Bhattarai, Sharma, Jurie & Pérez, 2014) proposed a semi-supervised hierarchical clustering algorithm based on human attributes such as gender, hairstyles, hair density. All the images were grouped into a single node at the initial step and a logistic loss function was used to minimize the errors. K-means was used to create clusters based on

attributes such as gender, and these clusters were used as two separate child nodes of the hierarchical tree. This process was continuously repeated for all the attributes to form a binary tree structure. The study claimed the same approach was easily adaptable for the use of k-ary trees instead of a binary tree. Local binary patterns (LBP) were used for feature extraction as they are highly resistant to illumination variations. Product Quantization was used for nearest neighbour retrieval based on Euclidean distance measure. Experimental results on the LFW dataset (G. B. Huang et al., 2007) show a speed boost of about N times with N being the number of leaf nodes or the depth of the tree. The empirical results obtained were a boost of about 2.8 times, 5.9 times, and 10.2 times with tree depth of 4, 8, and 16 nodes respectively on 1 million images with an Intel Xeon 2.8 GHz CPU running on Linux.

The process of creating a hierarchical tree based on clustering was automated, and hence the study observed the different types of clusters that were created by visually inspecting the individually clustered images. It was found that the Clusters 1–12 had predominantly male images whereas the clusters 13–16 had female images. Cluster 15 seemed to specialize females with bangs as their hairstyle and Cluster 14 with females having short hair and smiling faces. Cluster 2 seemed to have images of bald males with glasses.- A huge advantage of this process was that it was completely automated and thus tremendous amount of time spent on annotation and manual labelling was avoided. However, since it was automated there might be some images that were wrongly clustered. These miss-clustered images need to be manually indexed and grouped correctly, which can be quite a tedious task in a large scale dataset. A good solution would be to create a machine learning model that could automatically correct the incorrectly clustered images, thereby making this hierarchical model more robust and scalable.

In 2015, a paper known as “What Else Does Your Biometric Data Reveal?” presented a survey on how human attributes could be deduced and extracted from face-images and

their potential applications in image retrieval systems. Soft-biometric data could contain demographic attributes such as age, gender, ethnicity, eye colour, hair colour, etc. or anthropometric or geometric attributes such as facial geometry. These attributes are computationally inexpensive and have a semantic interpretation in a sense that humans can readily understand the description; for example, the description "Young, Asian ethnic, Female" is easily understood by humans, which could be a piece of additional information needed for forensic applications (Dantcheva, Elia & Ross, 2015).

A very recent paper published in September 2019 (Khan & Jalal, 2020) proposed a framework for automatic retrieval of images from a suspect's sketch by using linguistic information. The objective of this study was to retrieve a list of potential suspect's images based on linguistic information such as apparent age, gender, face shape, size, and colour provided by the onlookers or the witnesses. A linguistic module was used to aggregate the features from the provided face description ontology into a single feature vector. These linguistic attributes were categorized into three categories based on the priority information of the suspect's identification. The first category contained age and gender information; the second category contained colour information and the third category contained facial geometry description.

These three categories were used to divide the database images into four different clusters. The first cluster contained the images from all the three categories attributes that matched, the second cluster contained images that matched with the first and second categories while the third cluster contained images that matched with the first and third categories. This clustering step was crucial as the information provided by the witnesses is based on their memory and some of the information might be missed out. Thus, to retrieve the best possible matches with the limited information, the clusters were generated, and the suspect's images were retrieved by an ensemble classifier. Experiments performed on the FERET database (Phillips et al., 2000) showed recognition accuracy of 92% with the use of 18 different linguistic attributes.

2.4 Summary

After the literature review, it was evident that deep learning methods either used for feature learning and / or for classification achieved state-of-the-art results in age, gender, ethnicity classification as well as in facial recognition as compared to traditional methods. Most of the deep learning methods reviewed employed convolutional neural networks based on Google's Facenet (Schroff et al., 2015) or the VGG architecture (Simonyan & Zisserman, 2014). Each indexing technique was scalable to at the least a one million-scale dataset and the retrieval times recorded were somewhere between 0.2 to 0.5 seconds per image query on standard consumer hardware. Storage costs and computation costs for the generated face-encodings were reduced with the use of small Hamming signatures that converted the face-encodings into k-bit binary codes. From the review it could be concluded that the introduction of high-level human attributes such as gender, ethnicity, eye colour, etc. at some level of the indexing process had a positive impact by reducing retrieval times. The use of attributes also added an additional interpretable layer which could be beneficial for forensic or law enforcement applications especially if the whole process was automated.

Most of the reviewed indexing techniques followed the usual steps of image retrieval which included feature extraction, retrieval of top-k candidate images, and finally re-ranking the retrieved top-k images. These methods applied their proposed indexing techniques on the complete database and the matching images were retrieved using an approximate nearest neighbour (ANN) search. However, there is a further possibility of filtering the ANN search results using high-level human attributes such as age and gender. Based on these reviewed methods and their existing research gaps, there is still a lot of potential to improve the performance of state-of-the-art face-image retrieval systems further. Thus, this research study proposes a novel indexing method based on hierarchical classification for efficient and accurate face-image retrieval.

Chapter 3

Existing Technology

As discussed briefly in Chapter 1, the main objective of this research was to build an indexing method that allows for efficient matching and accurate retrieval of face-images from large-scale databases. Based on the findings from the literature review conducted in Chapter 2, this research study proposes a novel indexing method using hierarchically arranged classification models trained on high-level human attributes such as age, gender, and ethnicity.

The following Figure 3.1 gives an overview of the face-image retrieval system with the proposed indexing strategy based on hierarchical classification.

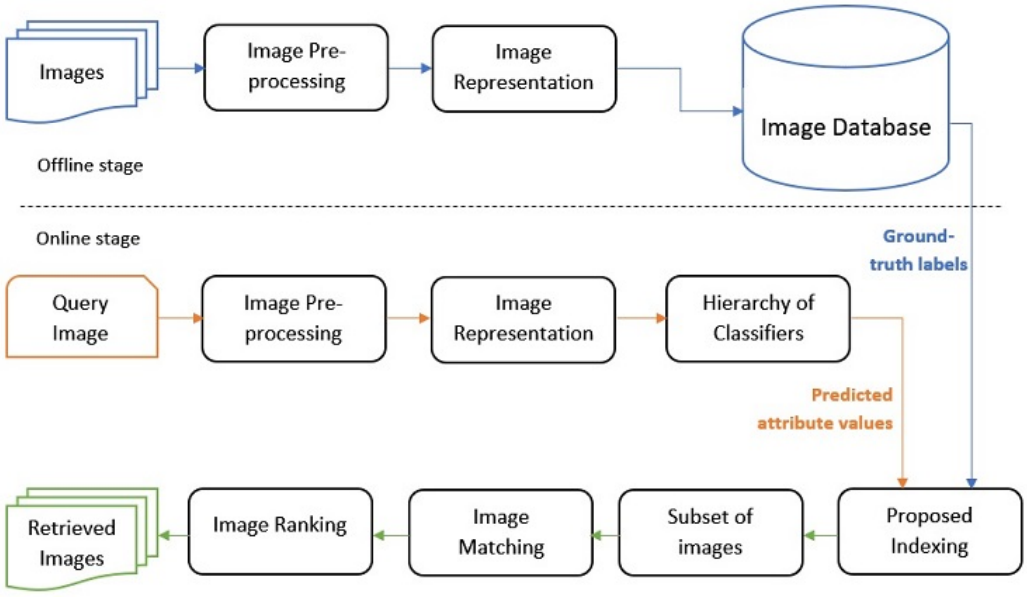


Figure 3.1: Overview of Proposed Face-Image Retrieval Architecture

Before proceeding with the proposed methodology, this chapter focuses on providing detailed background information on the already existing technology that was used as a part of this research study.

The next chapters 4 and 5 provide an in-depth walk-through and working of the proposed methodology and specifically highlight the contributions made by this research study.

3.1 Background Information

This section breakdowns the conceptual and practical components of the already existing technology that was used as a part of this research study. Each section explains the theoretical knowledge followed by the corresponding libraries used for its implementation.

The common steps associated with most face-image retrieval system are as follows:

- Image Pre-Processing
- Image Representation
- Image Indexing
- Image Matching
- Image Re-Ranking

3.2 Image Pre-Processing

Image pre-processing is an additional step required for any face-image retrieval system. As we are only concerned with face-images, pre-processing steps such as detecting a face, cropping, and aligning must be performed before proceeding further.

3.2.1 Face Detection

The initial step in pre-processing is to detect faces from the image. Once the face region is detected, the input image undergoes cropping for extracting only the area surrounding the detected face region. Face-detection became mainstream after the release of Viola and Jones' face detection algorithm presented in this paper (Viola & Jones, 2001), which was fast and computationally inexpensive to run even on mobile cameras. However, much more reliable and efficient solutions were developed later. The two face detection algorithms used in the study are based a on Histogram of Oriented Gradients (HOG) and a Convolutional neural network (CNN) methods.

Histogram of Oriented Gradients (HOG) Face Detector

Histogram of Oriented Gradients extracts HOG features by computing gradients based on the magnitude and direction of change in the pixel intensities of the image. These extracted HOG features are used to train a machine learning model to detect faces.

Although HOG was first proposed more than a decade ago, it is still a powerful and efficient face detection algorithm that is used even today.

HOG features are incredibly potent at identifying the structural shape and appearance of an object in an image, thereby making them an excellent choice for object classification. However, since HOG captures the local intensity of gradients and edge directions, they are also a good texture detector.

Advantages of HOG Face Detector:

1. High-speed computation using only CPU resources.
2. Good frontal and slightly non-frontal face detections.
3. Invariant to small occlusions.

Disadvantages of HOG Face Detector:

1. Does not work properly for side faced and extreme non-frontal faces, such as upwards or downwards looking face-images.
2. In rare cases, an inaccurate bounding box may be drawn for the detected face region which could result in false positive detection.

Library Used:

This research study uses a HOG + SVM implementation for face detection provided by an open-source dlib library [1]. The library model was built from 5 HOG filters, namely: front looking, left looking, right looking, front looking but rotated left, and front looking but rotated right. The dataset used for training the model consists of 2825 images which were obtained from the LFW dataset (G. B. Huang et al., 2007) and manually annotated by Davis King, author of the dlib library [1].

[1] `https://github.com/davisking/dlib`

Convolutional Neural Network (CNN) Face Detector

Following the trends in deep learning, CNN based face-detectors have gained popularity in recent years. Even though HOG based face detectors are extremely efficient, they lack the detection of non-frontal faces or of faces that exhibit complex pose variations. CNN based face-detectors can be used for detecting faces from any angle.

Advantages of CNN Face Detector:

1. Extremely accurate and robust to occlusions.
2. Can detect non-frontal faces and even faces with complex pose variations.

Disadvantages of CNN Face Detector:

1. Poor detection speed on CPU.
2. GPU requirement.

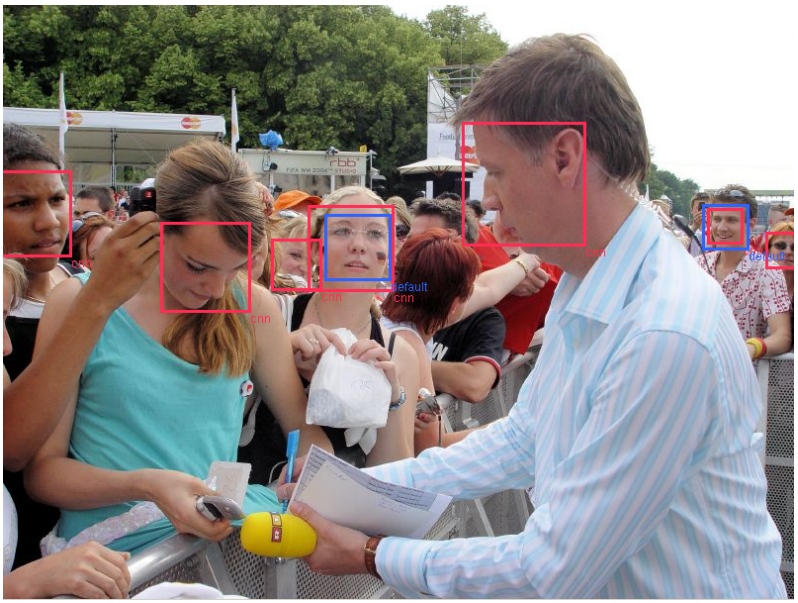
Library Used:

The CNN face detector used in this research study is based on a max-margin object detector (MMOD) (D. E. King, 2015) provided by same open-source dlib library[1]. MMOD improves face detection by applying a commonly used post-processing technique known as non-maximum suppression (Neubeck & Van Gool, 2006), (Malisiewicz, Gupta & Efros, 2011). A common problem faced in object detection or face detection is that of multiple detections of the same object or faces, leading to overlapping bounding boxes around a single detected object or a face. Non-maximum suppression reduces these overlapping detections and predicts a compact bounding box, thereby optimizing detections and reducing false positives. The training process for the CNN method is fairly straightforward and a custom face detector can be trained easily with limited amount of data.

[1] `https://github.com/davisking/dlib`

Comparison of HOG vs CNN Face Detections

Figure 3.2 illustrates example comparisons between a HOG based (blue coloured box) and a CNN based face detector (red coloured box).



(a)

(b)

Figure 3.2: HOG (blue) vs CNN (red) Face Detection, retrieved from (D. King, 2014)

It is clearly visible from these examples that the HOG based face detector struggled at detecting non-frontal images whereas a CNN face detector detected almost all faces in the images at any given angle even with occlusions. Although being a powerful face detector, a major drawback of the CNN face detector was the requirement to be run on a GPU for achieving comparable detection speed with respect to an HOG based face detector.

3.2.2 Face Alignment

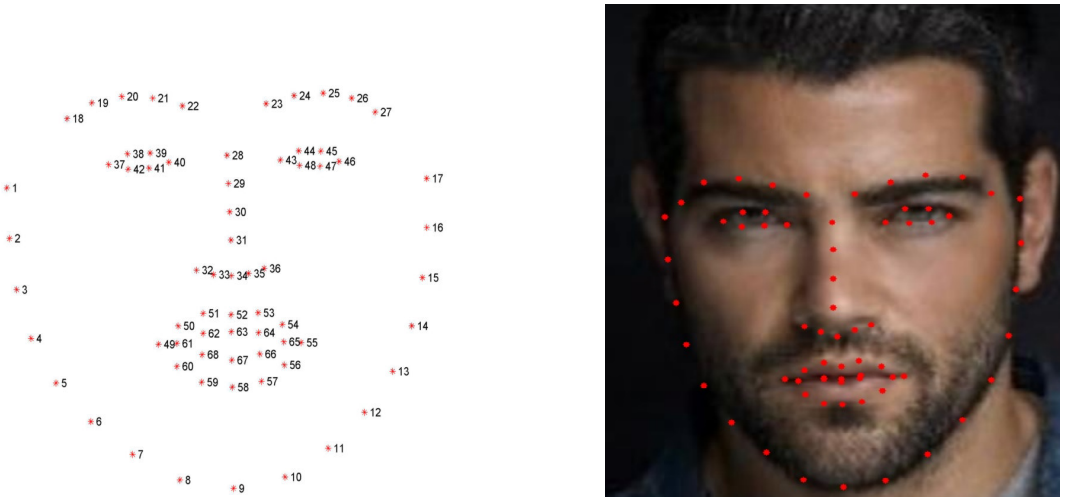
The next step in the pre-processing stage after detecting faces from an image is to align the detected faces to a front-facing position. As seen from the above examples, not all images are frontally aligned. Unaligned images may cause a common problem called as intra-class variation whereby images of the same person with varying poses and angles can be treated as two different entities.

For face-alignment, some methods try to impose a pre-defined 3D face model and then apply transformations such that the landmarks on the input face-image match with the landmarks present on the pre-defined 3D model. One such example of 3D alignment was used by Facebook presented in their DeepFace paper (Taigman et al., 2014). However, this research study uses face-alignment techniques based on simple affine transformations (co-linearity preserving) such as rotation and scaling, which frontally align the images with respect to a reference template.

Face alignment algorithm works as follows:

- **Step1 :** To generate the reference template, a face landmark estimation algorithm based on (Kazemi & Sullivan, 2014) was used to recognize common face landmarks such as the eyes, mouth or lip co-ordinates.
- **Step2 :** For each input image, 68 face landmark co-ordinates (x, y) were estimated that mapped to facial structures on the face. These co-ordinates are based on

common face locations such as the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. The 68 co-ordinates or annotations are part of a 68-point model created in the iBUG 300-W dataset (Sagonas, Antonakos, Tzimiropoulos, Zafeiriou & Pantic, 2016). Figure 3.3 demonstrates the 68-point landmark detection.



(a) (b)

Figure 3.3: Face landmark detection using 68-point reference template

- **Step3 :** From these detected landmark co-ordinates, we can extract the eyes, mouth, and lip co-ordinates of the face. Transformations are applied to the image such that the eyes and mouth are centered as best as possible. This process is also known as frontalization or centering of the face. Thus, to frontalize the face, eye centers as well as the angle between the eyes were computed. This angle was responsible for correcting the rotation and frontalizing the face.
- **Step4 :** The next step was to compute the scale of the image by taking the ratio of the distance between the eyes in the input image to the distance between eyes in the reference template.
- **Step5 :** The midpoint between the eye co-ordinates was computed as a reference

point around which the image was rotated. Simple scale and rotation transformation matrices were applied to align the image centered around the eyes with the given reference template.

- **Step5 :** After the alignment operation, the image size was kept fixed at 256 x 256 pixels. However, provisions were made to upscale or downscale the image as required.

Library used:

Both face-detection and face-alignment operations were implemented using open-source dlib [1] and opencv [2] libraries for python.

Figure 3.4 demonstrates the output result from face detection and face-alignment pre-processing operations.

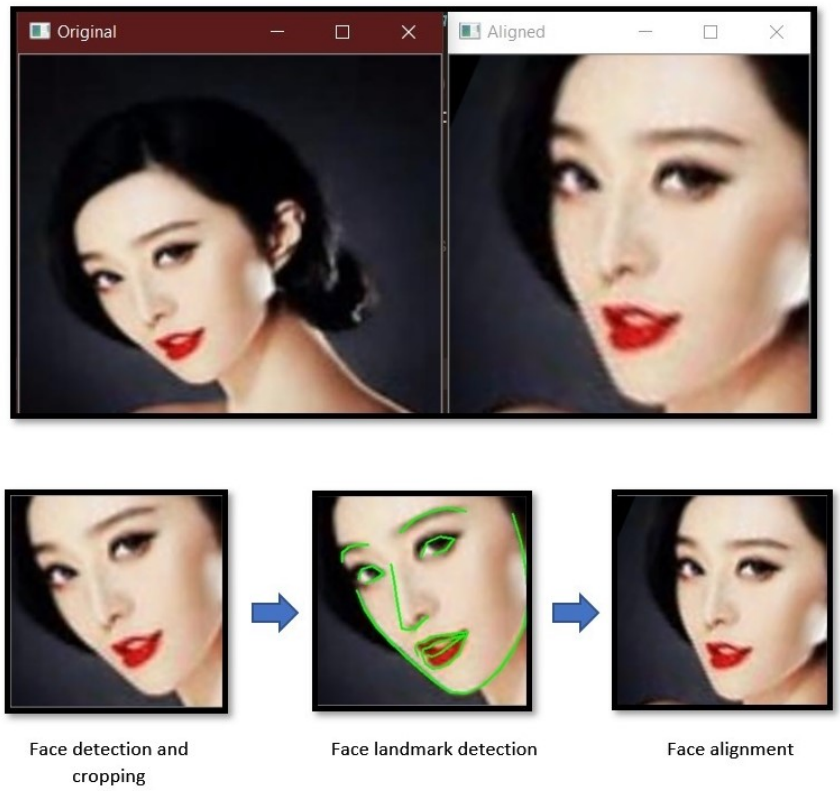


Figure 3.4: Image Pre-Processing Pipeline

[1] https://github.com/davisking/dlib
[2] https://github.com/opencv

3.3 Image Representation

After the pre-processing steps of face-detection, cropping, and alignment, the most critical component was of image representation. This is the most crucial step in any face-image retrieval system as all the steps that follow image representation are influenced drastically. The retrieval algorithm, the matching computation, the indexing technique, the storage costs all are affected by how the images are represented. Generally, most image representation techniques extract features such as SIFT, HOG, or LBP to encode images into feature vectors. If these feature vectors have high dimensionality, it takes a toll on all the above steps as every step introduces increased computational complexity.

According to the discussion in Chapter 2 of Literature review, the current trend has seen a shift towards feature learning instead of feature extraction due to the advances in deep learning. Feature learning is nothing but letting the machine (in this case a deep neural network) to do the work of learning a feature extractor that automatically extracts relevant features of the image based on the statistics of the training data. This means that the deep neural network will infer on its own what measurements or computations are of importance for the given task based on the provided data.

Applying this technique to face-image data allows the machine to infer the importance of specific measurements or ratios present in the facial geometry or capture patterns in image pixels for the given classification or recognition problem. These measurements might be complex or even inconceivable for humans to understand or visualize. That is the beauty and, at the same time, the problem in using deep learning. Although these computed measurements may be highly accurate but they are virtually non-interpretable. These generated measurements may be something as simple as the distance between the eyes, facial width or could be any complex computation or series of computations. This is in contrast to the anthropometric features reviewed earlier in Chapter 2 where the anthropometric features are deduced based on fixed geometrical measurements and

ratios present in the facial geometry which are easily understandable by humans.

For image representation, this research study focused on the use of feature learning implemented using Convolutional Neural Networks (CNNs) and Transfer Learning.

3.3.1 Convolutional Neural Network (CNN)

Convolutional neural networks are neural networks used primarily to classify images or cluster images by similarity and perform object recognition within scenes. For example, CNNs can be used to identify anything from faces, individuals, street signs, cars, and many other objects. The success of CNNs is the main reason for the rising popularity of deep learning methods. Input to a CNN is usually an image. Unlike a regular neural network, the layers of a CNN have neurons arranged in 3 dimensions, namely: width, height, and depth. The CNN architecture comprises of different layers that transform the original image layer by layer, from the pixel values to the final output scores (classification score). Each layer of the CNN accepts an input 3D volume (height, width, depth) and transforms it into an output 3D volume which is the predicted class score.

CNN architecture

A simple CNN architecture comprises of the following layers:

- **Input Layer:** This layer holds the raw pixel values of the image in 3D such as the image width, image height, and depth. Here, the depth corresponds to the image colour channels (R, G, B).
- **Convolution Layer:** This layer is the core building block of a CNN that is responsible for most of the heavy computations. The primary purpose of convolution operation in the case of CNNs is to extract features from the input image.

Convolution operation preserves the spatial relationship between pixels by learning image features using small squares of input data at each step. The following Figure 3.5 illustrates a simple convolution operation using a 3x3 filter over a 5x5 image matrix.

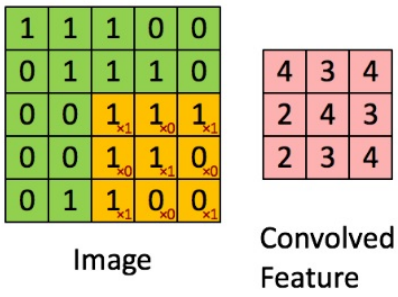


Figure 3.5: Convolution Operation, retrieved from (Johnson & Karpathy, n.d.)

In CNN terminology, the 3×3 matrix is called a 'filter' or 'kernel'. Another matrix is formed by sliding the filter over the image matrix and computing the dot product between them. This matrix is called the 'Convolved Feature' or 'Activation Map' or the 'Feature Map'. A CNN typically learns the values of these filters on its own during training. Increasing the number of filters increases the number of features that get extracted from the image, and thus the network becomes better at recognizing patterns in unseen images. The size of the feature map (convolved feature) is usually controlled by three parameters, namely depth, stride, and zero-padding which must be set before the convolution step. The depth here represents the number of filters to be used. Stride sets the number of pixels by which the filter matrix slides over the input image matrix. Zero-padding allows for the filtering of border elements as well as in controlling the size of a feature map.

- **ReLU layer:** ReLU stands for Rectified Linear Unit. ReLU is an element-wise operation applied on each pixel that replaces all negative pixel values in the

feature map with the zero value. The feature map produced after this layer is also referred to as a Rectified Feature Map. Other nonlinear functions such as $tanh$ or sigmoid functions can also be used instead of ReLU.

- **Pool Layer:** The pooling layer performs a spatial pooling also called subsampling or downsampling. This subsampling reduces the dimensionality of each feature map while retaining the most significant information. Spatial Pooling can be of different types such as Max-pooling or Average pooling, etc. Figure 3.6 shows an example working of max-pooling operation performed on a rectified feature map obtained from the ReLU layer.

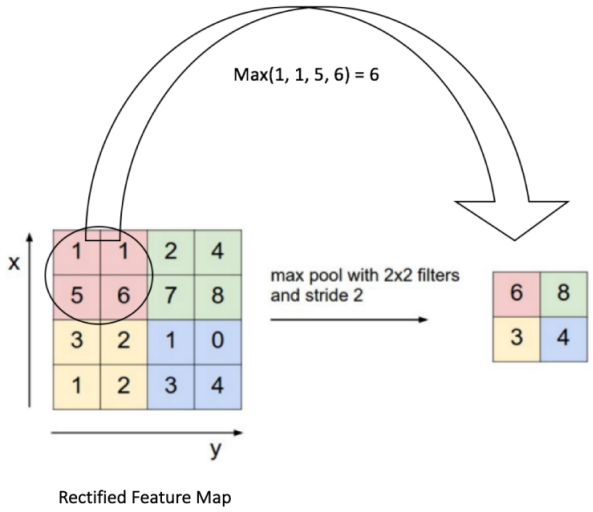


Figure 3.6: Max-pooling operation, retrieved from (Johnson & Karpathy, n.d.)

The function of Pooling is to reduce the spatial size of an input image progressively. After pooling, the network becomes invariant to small transformations, distortions, and translations in the input image as they do not affect the pooling operation (Johnson & Karpathy, n.d.).

- **Fully connected layer:** This layer is the same as that of a regular neural network that computes class scores producing an output volume dimensions as [1x1xN], where each of the N numbers corresponds to a class score for N number of classes.

Figure 3.7 represents a simple CNN architecture.

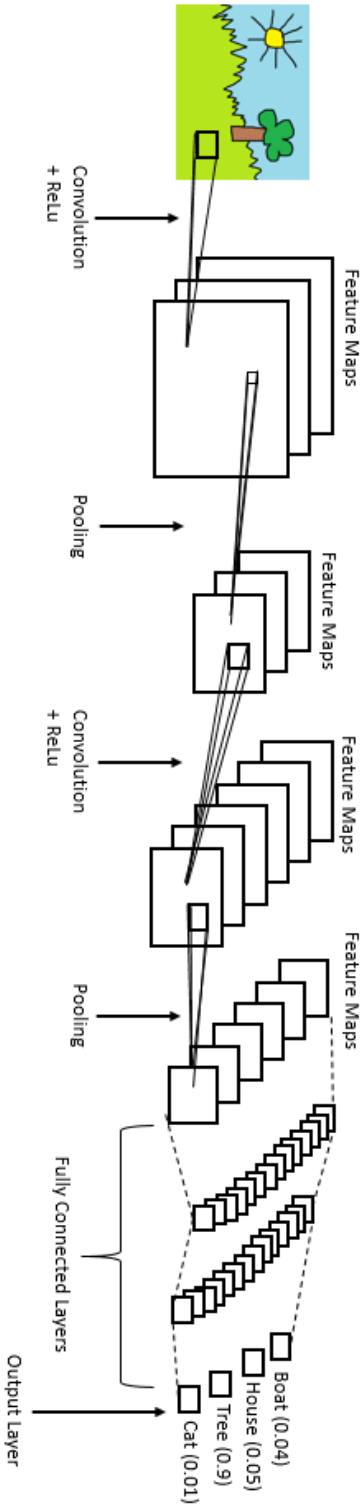


Figure 3.7: Simple CNN architecture, retrieved from (Prabhu, 2018)

Popular CNN architectures

There are several popular variants of CNN architectures as follows:

- **LeNet:** This was the first successful application of a CNN architecture developed in the late 1990s by Yann LeCun to read zip codes and digits (LeCun, Bottou, Bengio & Haffner, 1998).
- **AlexNet:** The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 (Russakovsky et al., 2015) and significantly outperformed the runner-up by more than 10% error difference (Krizhevsky, Sutskever & Hinton, 2012). It was composed of 5 convolutional layers followed by 3 fully connected layers with a filter size of 11x11 and a stride value of 4.
- **VGGNet:** This architecture was from the VGG group, Oxford. It made improvements over AlexNet by replacing large kernel-sized filters with multiple 3x3 kernel-sized filters one after another and had 16 convolutional and fully connected layers. The disadvantage of the VGGNet is that it is more expensive to evaluate as it uses more memory and has a large number of parameters (140 Million) (Simonyan & Zisserman, 2014).
- **ResNet:** Residual Network developed by (He, Zhang, Ren & Sun, 2016) was the winner of ILSVRC 2015 (Russakovsky et al., 2015). It features special skip connections and makes heavy use of batch normalization. The architecture is built without using the fully connected layers at the end of the network. ResNets and the models that followed thereafter such as **Inception** (Szegedy, Ioffe, Vanhoucke & Alemi, 2017) and **DenseNets** (G. Huang, Liu, Maaten & Weinberger, 2017) are currently the state-of-the-art Convolutional Neural Network models based on ImageNet ILSVRC challenge results (Russakovsky et al., 2015).

3.3.2 Transfer Learning

Training a CNN from scratch requires a huge amount of training data. This was evident from Table 2.1 where almost all the studies that used CNN had several hundred million images in their training dataset. Moreover, training these CNNs for achieving a decent accuracy score takes several weeks to months of training time. This was evident from Google's FaceNet paper (Schroff et al., 2015) where it was stated that training the FaceNet took around 1000 hours of CPU computation time. These CNN architectures also tend to have about million parameters (Google's FaceNet (Schroff et al., 2015) and Facebook's DeepFace (Taigman et al., 2014) had 140 million parameters) which affects the training time.

Instead, it is a common practice to use a pre-trained CNN model. This pre-trained CNN model is typically trained on a very large dataset such as ImageNet (Russakovsky et al., 2015), which contains millions of images with several thousand categories used for a classification task. This pre-trained model can be used as it is or can be customized using transfer learning to a given image library.

The Figure 3.8 shows the transfer learning process where a pre-trained source model on a large dataset of objects is used for new classification task on a relatively smaller dataset of bicycles and trucks.

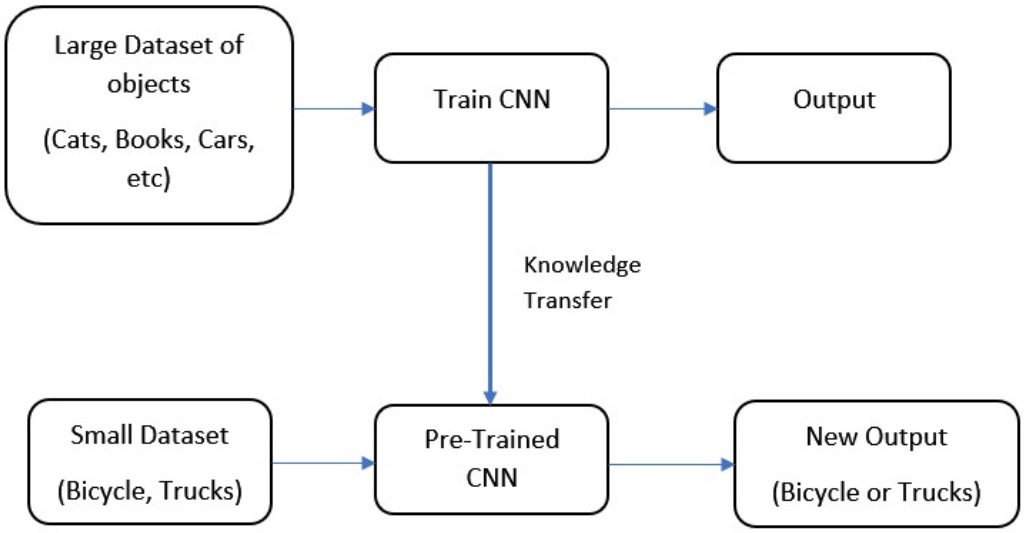


Figure 3.8: Transfer Learning

The intuition behind transfer learning is that if a model is trained on a large enough dataset, it can effectively be used as a generic model for any given image classification task. The already learned feature maps from the pre-trained CNN could be used without having to start training a model on a large scale dataset from scratch.

Transfer Learning Strategies

There are a few different transfer learning strategies out of which feature extraction and fine-tuning of weights are the widely used ones.

- **Feature Extraction:**

  A pre-trained CNN can be used to extract meaningful features from new training data based on the previously learned features. A new fully connected layer is simply added by removing and/or replacing the last or penultimate fully connected layers of the pre-trained model. This newly added fully connected layer re-purposes the previously learned feature maps to a specific classification task of our choice. This new layer is used to learn and extract features for the new

training data samples.

These extracted features are also known as embeddings and generally are of N-dimensions, where N depends on the number of nodes present in the last layer of the CNN model. Google's FaceNet (Schroff et al., 2015) model extracted a 128D face-embedding while the AlexNet (Krizhevsky et al., 2012) model extracted a 4096D embedding. These embeddings can be used to train a linear classifier such as Linear Support Vector Machine on a new dataset.

- **Fine-Tuning:**

  Another strategy of using transfer learning is to fine-tune the weights of a pre-trained network by continuing the back-propagation. It is possible to fine-tune all the layers of a pre-trained CNN model but usually, the earlier layers are kept fixed and the later layers are used for fine-tuning. This is motivated by the observation that the earlier features of a CNN model contain more generic features such as edge detections or colour detections that can be considered generically useful to many tasks. However, the later layers of the CNN model become progressively specific to a dataset. For example, in the case of ImageNet dataset (Russakovsky et al., 2015) which contains many dog breeds, a significant portion of the later layers may be devoted to learning the features that are specific to differentiating between dog breeds. Since training modern CNN takes about 2-3 weeks across multiple GPUs on the ImageNet dataset (Russakovsky et al., 2015), it is a common practice amongst researchers to release the final CNN model checkpoints that contain the network weights. The released weights can be used for fine-tuning instead of training a CNN from scratch.

Several studies have successfully used transfer learning for various tasks, including image classification and image retrieval challenges. A study by (Sharif Razavian, Azizpour, Sullivan & Carlsson, 2014) showed the use of transfer learning as a feature extractor for

object retrieval challenge. The experimental results indicated a Linear Support Vector Machine classifier trained on features extracted using transfer learning outperformed some of the scratch CNN models. A survey on applications of transfer learning (Pan & Yang, 2009) provided useful information and successful experimentation of transfer learning applied across various image classification and retrieval challenges.

Advantages of Transfer Learning:

1. Extremely fast training time of just seconds to minutes.
2. Can be trained on a single CPU core without any GPU requirement.
3. Training data can be of a small scale.

Disadvantages of Transfer Learning:

1. Accuracy of the developed model heavily depends on the performance of the pre-trained model.
2. No control over the pre-defined CNN architecture.
3. Can overfit if training data is small and similar to the original dataset.

3.3.3 Triplet Loss

An important aspect in terms of training a good CNN model is the loss function. In the FaceNet paper by Google (Taigman et al., 2014), a loss function called the Triplet Loss function was used to train the CNN. For a given input image sample, the triplet loss function works by minimizing the distance of an input image from the positive samples, while maximizing the distance from the negative samples. Here positive samples are those images which are the same as that of the input image, whereas negative samples are ones which are different from the input image.

The below equation 3.1 represents the triplet loss function to be minimized where x_a represents the anchor or input image and x_p represents positive samples which are

images of the same person as the input anchor image and x_n represents negative samples which are images of any other person.

$$\sum_{n=i}^{N}\left[\left\|f(x_a^i)-f(x_p^i)\right\|_2^2-\left\|f(x_a^i)-f(x_n^i)\right\|_2^2+\alpha\right]_+ \tag{3.1}$$

The main aim of employing the triplet loss function was to generate an embedding $f(x)$ from an anchor or input image x, such that the squared distance between all faces, independent of imaging conditions, of the same person was small, whereas the squared distance between a pair of face-images from different person was large as depicted in the below Figure 3.9.

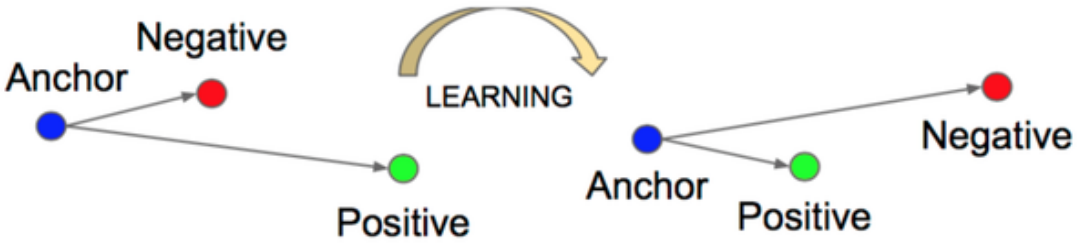


Figure 3.9: Triplet Loss minimizing and maximizing distances of anchor image from positive and negative samples respectively, retrieved from (Schroff et al., 2015)

The Figure 3.10 displays a sample training step using triplet loss function similar to the one used in the FaceNet paper (Schroff et al., 2015). The FaceNet architecture generated a 128D face embedding for each face-image. By repeating this step millions of times for millions of images of thousands of different people, the neural network learns to generate 128D embedding for each person reliably. Any different pictures of the same person should produce roughly the same 128D embeddings.

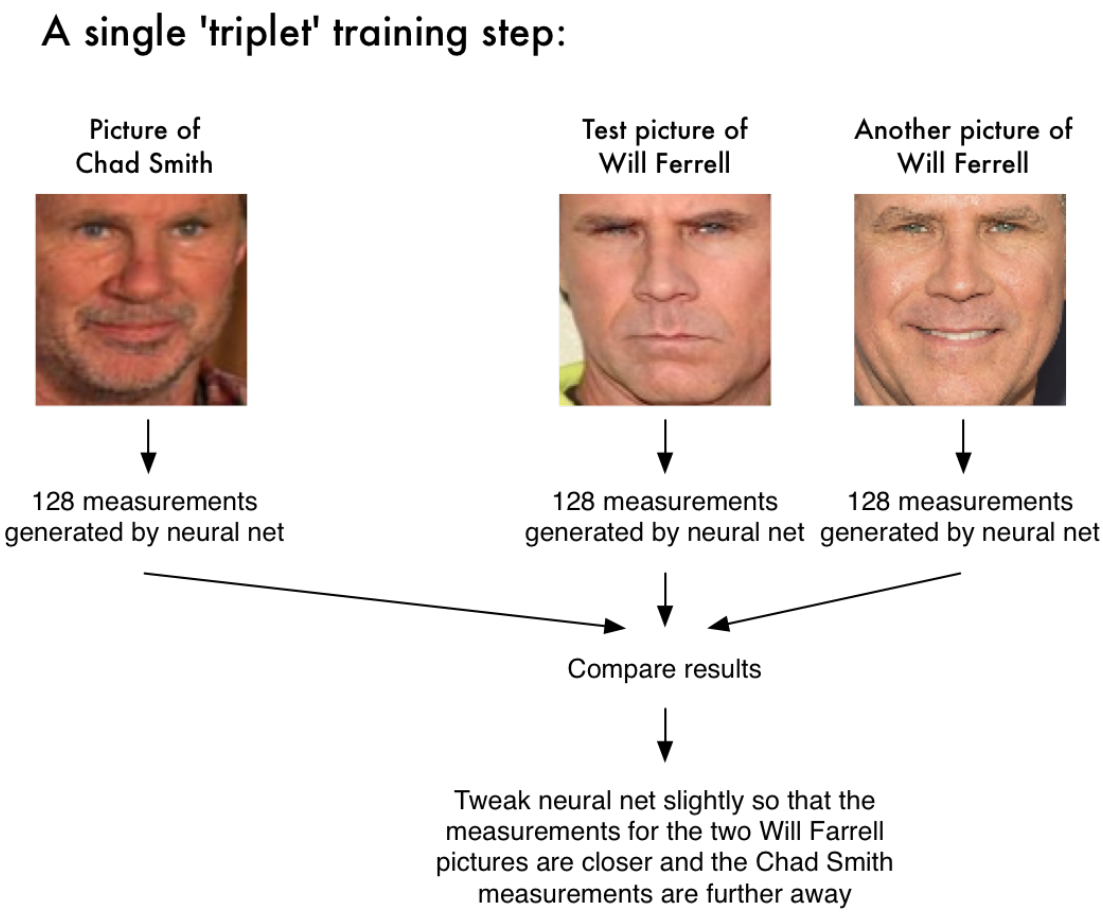


Figure 3.10: Triplet Loss training step, retrieved from (Geitgey, 2016)

3.3.4 Representing images as 128D feature vectors

Library used:

Due to powerful hardware requirements, the potential training time of some weeks, and the requirement of million images for training; the use of transfer learning was preferred over a CNN model trained from scratch for this research study. This research study uses both the above-discussed transfer learning strategies based on feature extraction and fine-tuning of weights.

For transfer learning as a feature extractor this research study uses the open-source

library dlib[1] by Davis King and an open-source implementation by David Sandberg[2] of Google's FaceNet (Schroff et al., 2015) architecture with different CNN variants based on Inception (Szegedy et al., 2017) and VGGNet (Simonyan & Zisserman, 2014) models, all of which were trained for face-recognition tasks. For transfer learning using fine-tuning of weights, Google's open-source library Tensorflow[3] is used to fine-tune the weights from checkpoints provided by a variety of pre-trained CNN models such as ResNets, Inception variants and MobileNets which are trained on the generic ImageNet dataset (Russakovsky et al., 2015) that are available to download from the Tensorflow's website[3].

Based on the experimental results of the above implementations discussed in Chapter 6, the open-source library dlib[1] performs the best and achieves state-of-the-art performance in age, gender and ethnicity classification tasks. The CNN architecture used by dlib library[1] is a Residual Net (ResNet) based on the architecture described by (He et al., 2016). The original architecture had 34 convolutional layers, whereas the dlib library[1] only uses 29 layers along with half the number of filters as compared to the original architecture. The network was trained from scratch on a dataset of about 3 million face-images with the total number of individual identities in the dataset being 7485. The network was initialized with random weights and employed a triplet loss function that generates a 128D feature vector for a given face-image. On the Labeled Faces in the Wild (LFW) dataset (G. B. Huang et al., 2007) the dlib network achieves a 99.38% face-recognition accuracy as noted from Table 2.1.

[1] `https://github.com/davisking/dlib`
[2] `https://github.com/davidsandberg/facenet`
[3] `https://www.tensorflow.org/`

The Figure 3.11 depicts the use of dlib's pre-trained model to generate a 128D feature vector as the face-image representation.

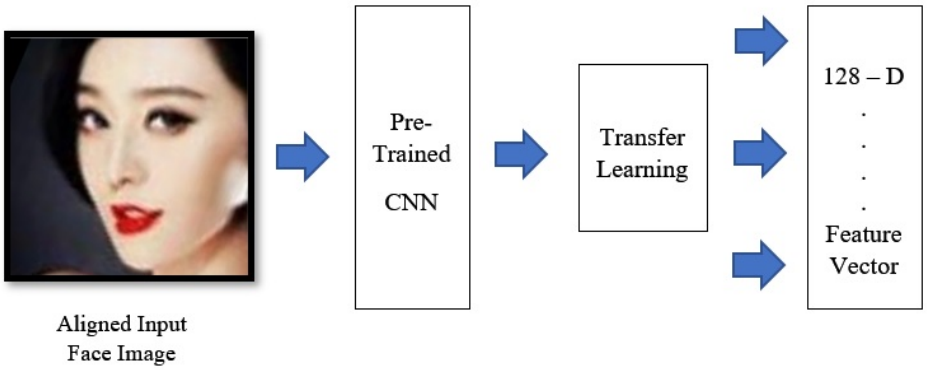


Figure 3.11: Transfer Learning to generate 128D feature vector

3.4 Image Indexing

After the Image Pre-Processing and Image Representation steps, the indexing techniques play a vital role in face-image retrieval systems. Image indexing techniques efficiently arrange, order, and store the generated feature vectors from the database. The main contributions of this research study are involved in the image indexing component. A novel image indexing strategy is proposed using hierarchical classification leading to quick and accurate face-image retrieval. This component is thoroughly described in the next Chapter 4.

3.5 Image Matching and Re-Ranking

Once the indexing techniques are applied, image matching algorithms are used to compute similarities between the given query image and the database images. These

similarities are computed based on well-known similarity or distance metrics such as Euclidean distance, Cosine similarity, Manhattan distance, etc. (Cha, 2007). Using the computed similarity score, a list of images is retrieved from the database, which contains the most similar images to the given query image. These most similar images are finally re-ranked removing any false matches thereby improving the accuracy of retrieval. Both these steps are a part of the proposed methodology and are explained comprehensively in the next couple of Chapters 4 and 5.

Chapter 4

Proposed Methodology: Design of Hierarchical Indexing Scheme

The proposed methodology will be covered in two chapters.

Chapter 4 provides a detailed description of the proposed database indexing strategy with an explanation of the hierarchical design and the underlying indexing structure. The chapter ends by presenting the indexing algorithms and the training procedure of attribute classification models.

The next Chapter 5 focuses on the novel probabilistic back-tracking algorithm proposed by this research study. The chapter explains the concept and the theoretical basis behind the back-tracking algorithm. Another important contribution of dynamic thresholding is presented and described in this chapter. The chapter ends with the discussion of the Image Matching and Ranking components which completes the overall processing pipeline of the face-image retrieval system.

4.1 Proposed Indexing Strategy

The proposed methodology is focused on the Image Indexing component as shown in Figure 1.2. This research study proposes a novel database indexing strategy based on hierarchical classification. By exploiting the predictive power of high-level human attribute classification models, a database can be reduced to a smaller subset by indexing it on high-level human attributes such as age, gender, ethnicity without significant time consumption and loss in precision.

Figure 4.1 demonstrates the proposed indexing strategy based on hierarchical classification. For a given query image, the hierarchy of classifiers predicts different attribute outcomes for age, gender, and ethnicity which are concatenated together to form a distinct key. Using this key, a hash index is computed from a deterministic hashing function. Each hash index value from the hash table points to a subset of the image database. An appropriate subset is retrieved by querying this hash table with the computed hash index. This subset of the image database is the one most likely to contain matching images for a given query image.

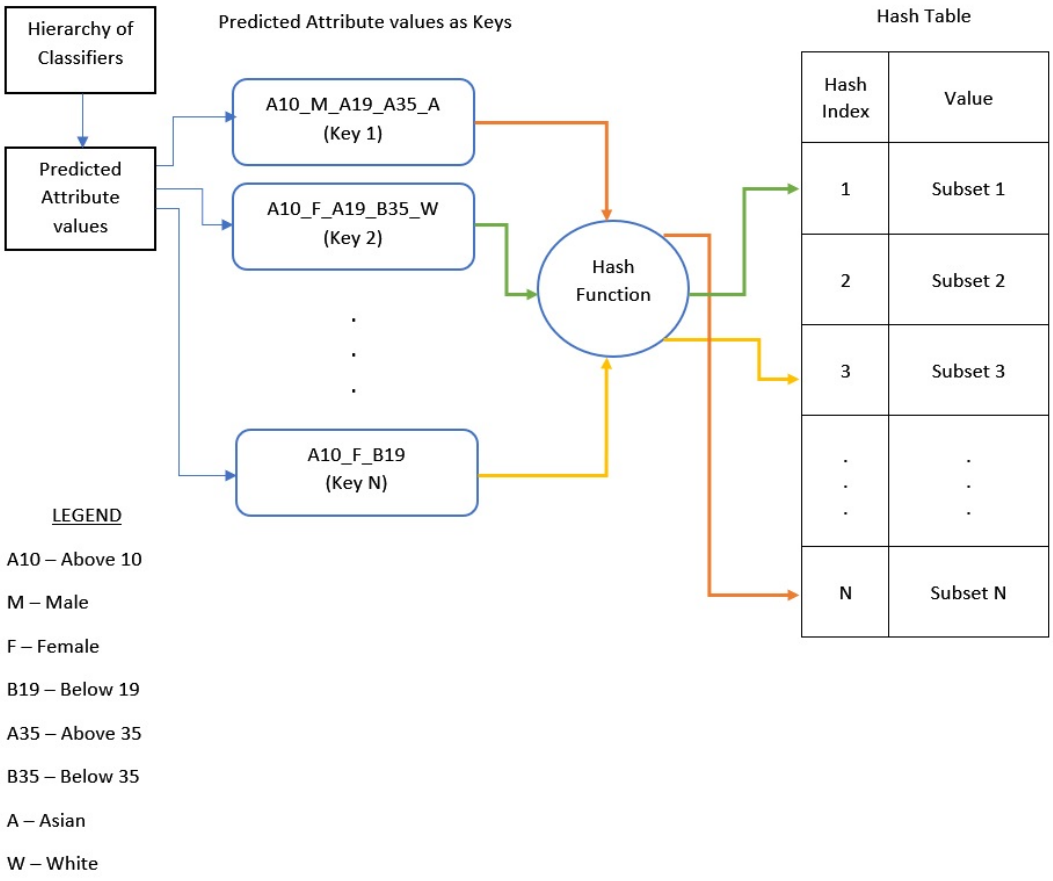


Figure 4.1: Proposed indexing strategy based on hierarchical classification

The hash table for a database is built using the ground-truth labels and is easily queried at run-time with the predicted labels obtained from the hierarchy of classifiers. A detailed description on the construction of the hash table and the underlying indexing structure is provided in the later sections of this chapter.

An efficient database indexing scheme can significantly accelerate the retrieval process and reduce memory usage substantially. Using the proposed indexing strategy, only a small subset of the database is selected for computing matches to a given query image, thereby drastically reducing retrieval time and computational complexity. The goal of the proposed indexing strategy is to efficiently and accurately obtain a subset of the database that is most likely to contain matching images for a given query image.

4.1.1 Hierarchical Classification vs Hierarchical Clustering

Similar to the proposed indexing strategy of hierarchical classification, there exist techniques based on hierarchical clustering. This section describes why hierarchical classification was preferred over hierarchical clustering by comparing their advantages and drawbacks.

Clustering is an unsupervised technique that groups similar objects together such that the objects in the same group are more similar to one another than the objects in the other groups. Hierarchical clustering is an unsupervised clustering process which involves creating clusters that have a predominant ordering from top to bottom in a hierarchical manner. Hierarchical clustering is divided into mainly two types:

- **Agglomerative Hierarchical Clustering:** The Agglomerative Hierarchical Clustering is the most common type of hierarchical clustering algorithm used to group objects in clusters based on their similarity. It is a bottom-up approach where initially each sample starts as a separate cluster and the closest samples get merged progressively as we move up the hierarchy.
- **Divisive Hierarchical Clustering:** Divisive hierarchical clustering algorithm works exactly opposite of Agglomerative algorithm. It follows a top-down approach where instead of starting with N number of clusters, a single cluster is formed initially which contains all the samples. At each iteration, the cluster is partitioned into two separate least similar clusters.

The main output of a Hierarchical Clustering is a dendrogram. A dendrogram is a type of tree diagram showing hierarchical relationships between different sets of data. The following Figure 4.2 depicts a sample dendrogram generated from a hierarchical clustering algorithm.

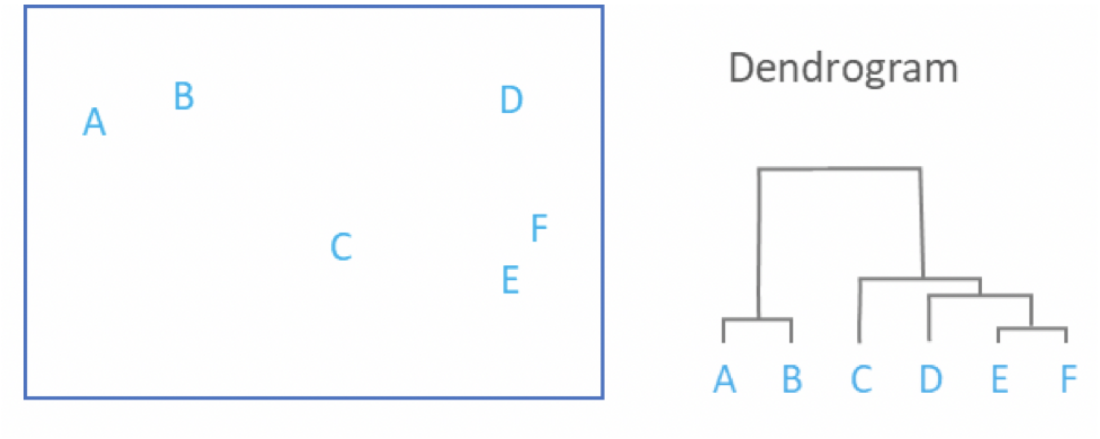


Figure 4.2: A Sample Dendrogram Output

Hierarchical clustering algorithms perform better than standard clustering algorithms such as K-means clustering by removing the need to pre-determine the number of clusters. However, they lack scalability and robustness shown by hierarchical classification models especially towards high dimensional data.

Below table 4.1 provides a comparison between hierarchical classification and hierarchical clustering methods. This comparison provides a rationale behind selecting hierarchical classification as the foundation structure for the proposed indexing strategy.

Table 4.1: Hierarchical Classification vs Hierarchical Clustering

Hierarchical Classification	**Hierarchical Clustering**
Robust to noisy data and outliers as incorrect classifications can be rectified	Sensitive to noisy data as incorrectly clustered samples cannot be rectified
Scalable to high-dimensional data	Difficulty in scaling to handle high-dimensional data
Split points of the hierarchy can be easily determined	Determining split points is challenging
Each split point from the hierarchy can employ its own strategies such as different feature selection methods, different distance metrics, etc.	All clusters need to use a single pre-defined strategy
Split points can provide additional information regarding a data point such as the age, gender, and ethnicity information extracted in this research study	Split points may or may not provide any additional information
Can be easily automated without losing interpretability of the formed hierarchy	Can be automated but interpretability of the formed hierarchical clusters could be lost for high-dimensional data
Can utilize back-tracking to revisit points higher up in the hierarchy	No possibility of back-tracking to higher levels in the hierarchy

4.1.2 Classification Attributes

This section provides a rationale behind the selection of different classification attributes used in the proposed indexing strategy.

The hierarchy of classifiers was constructed from high-level human attributes which can be classified with high accuracy. For this research study, the core attributes selected were age, gender, and ethnicity.

The rationale behind the selection of these three attributes is as follows:

- **Gender:** This attribute was chosen due to its simplistic classification with very high achievable accuracy rates of around 96% as reviewed in Chapter 2 of Literature Review. The gender attribute provides a high filtering factor on each gender category, as approximately half of the population lies within each division. Gender also provides vital information regarding a person in an easy-to-interpret manner. This information is extracted by default as a part of the proposed methodology, and thus no extra processing is required. It may prove useful in certain real-world applications such as those required by law enforcement or at airports for identifying fake individuals.
- **Age:** This attribute was treated as an age-group rather than a specific age number. As opposed to predicting specific age values, age-groups can be classified with high accuracy as noted from Chapter 2 Literature Review. Like gender, age-groups also provide vital linguistic information regarding an individual which may prove useful in real-world scenarios such as identification of missing individuals. Another benefit of age-groups lies in the identification of criminals who have naturally aged. If an image stored in the database of a suspect taken at an earlier age can be matched with the present-day image, it could provide valuable information of the suspect's details to the law enforcement agencies.
- **Ethnicity:** Ethnicity comprises of a large number of ethnic groups allowing for

the creation of plenty of subsets, one for each ethnicity. Since each subset would contain a fraction of samples, matching images can be found efficiently and accurately as compared to retrieving from the complete database of all images. This was the main reason for selecting ethnicity as one of the attributes due to its high cardinality. Similar to age-group and gender attributes, ethnicity also provides more detailed information regarding an individual as a part of the proposed methodology without any additional computation.

Thus, all three attributes provide valuable information that is extracted as part of the proposed methodology without any excess computation. To limit the error propagation, only the three above-discussed attributes were selected. However, more attributes such as hairstyles, eye-colour, etc. could be chosen if they can be successfully classified with higher accuracy scores. Increased number of attributes will increase granularity and increase the hierarchical depth leading to subsets with a fewer number of samples as compared with the whole database. However, classification errors are proportional to the number of attributes and the hierarchy depth. The greater the hierarchical depth and the number of attributes, the greater will be the classification error due to the propagation of errors from one level to the next.

4.2 Hierarchy of Classifiers

This section explains the structure of the hierarchical classification designed for implementing the proposed indexing strategy and the design decisions involved in its construction.

For an image, the hierarchy of classifiers predict different attribute values to form a key which is in turn used to query the hash table for retrieving an appropriate subset of the database. Thus, designing a good hierarchical structure of classifiers was crucial to the success of the proposed methodology.

4.2.1 Hierarchical Design

Seven classifier models were selected for designing the hierarchy based on the classification scores from amongst numerously trained classifiers. Multiple binary classifier models were preferred instead of multi-label classifier models due to their higher accuracy scores. Using multiple binary classifiers also improved the granularity of the developed hierarchy as compared with multi-label classifiers. The following table 4.2 shows the seven classifier models used in the hierarchy:

Table 4.2: Seven classification models

Classification Model	**Group**	**Type**	**Classification Categories**
Age-Group 10	Common	Binary	Above / Below 10 years
Gender	Common	Binary	Male / Female
Age-Group 19	Males	Binary	Above / Below 19 years
Age-Group 19	Females	Binary	Above / Below 19 years
Age-Group 35	Males	Binary	Above / Below 35 years
Age-Group 35	Females	Binary	Above / Below 35 years
Ethnicity	Common	Multi-Class	Asian / Indian / Black / White

Figure 4.3 depicts the hierarchy of classifiers. Each classifier from the hierarchy was built using one of the above-discussed attributes of age-group, gender, or ethnicity. The hierarchy was designed by following a top-down approach (depicted as left-to-right in Figure 4.3 for clarity) of decreasing classifier accuracy with the most accurate classifier

situated at the top level.

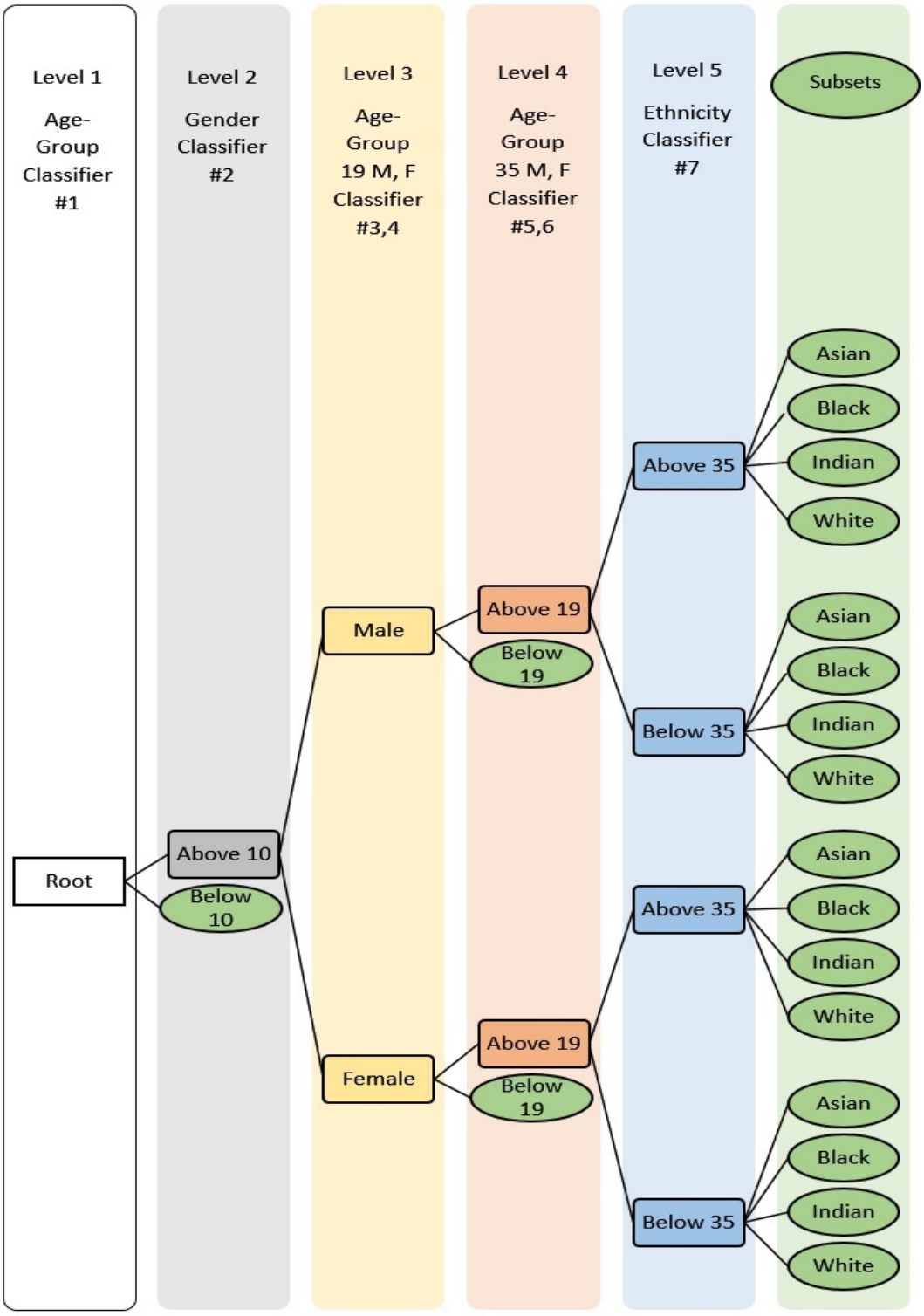


Figure 4.3: Hierarchy of Classifiers

Level 1 or Root

The classifier with the highest classification accuracy was selected at the root level to minimize error propagation due to miss-classifications. Based on the experimental results of each classifier model presented in Chapter 6, the Gender classifier was initially selected at the root level due to its highest classification accuracy.

However, according to the original paper of the pre-trained CNN model used from the dlib library (D. E. King, 2009), the pre-trained CNN does not work well with images of babies. It was empirically identified that the CNN model produced the worst results with images of kids who were ten years or younger.

Thus, due to this additional constraint, the root node was decided to be based on the binary age-group classification of above or below 10 years which also had a high classification accuracy of 98%. This classifier classified images into two categories of above or below 10 years. Images with age less than or equal to 10 years were stored as a separate subset and were not partitioned further. This ensured that the rest of the classifiers would be trained only on images with age values of greater than 10 years, overcoming the constraint of the pre-trained CNN model.

Level 2

Level 2 in the hierarchy is composed of a gender classifier trained on images belonging to above ten years coming from the previous level. The gender classifier classified images into a male or a female category. Each gender category formed its own sub-hierarchy, as depicted in the above Figure 4.3. Creating separate sub-hierarchies for male and female categories increased granularity, leading to finely tuned classifier models specifically trained for either male or female images only.

Level 3 and 4

Level 3 of the hierarchy has an age-group19 classifier that classifies images into two categories of above or below 19 years. Each sub-hierarchy belonging to the male or female category has its own separate age-group19 classifier trained on images specific to their respective sub-hierarchies. For both sub-hierarchies, the group of below 19 years is kept as an independent subset for which no further partitioning is involved.

Level 4 has a similar age-group35 classifier that classifies images into two categories of above or below 35 years. Like level 3, each sub-hierarchy has its own separate age-group35 classifier belonging to male or female images.

Level 3 and level 4 classifiers of age-groups 19 and 35 were selected based on the importance of the age-groups as well as their high classification accuracy. Age-group 19 filters the database as teenagers and adults while age-group 35 filters the database as young adults and older adults, both of which provide valuable linguistic knowledge regarding the age of a person.

Level 5

The final level 5 is composed of an ethnicity classifier that classifies into four ethnic groups of Indian, Asian, Black, and White. It is a single classifier common to both the sub-hierarchies. A common ethnicity classifier was preferred due to its higher classification accuracy as compared to more granular and separate ethnicity classifiers for male and female sub-hierarchies.

The ethnicity classifier was the least accurate amongst all the classifiers. Having the least accurate classifier at the last level 5 limited the error propagation as any miss-classifications that occurred would be present at the same level in one of the four ethnic groups.

Subset

The final subsets that would be retrieved are represented by green ovals in the above Figure 4.3. These subsets are retrieved during run-time based on the predicted attribute categories by each classifier at each level from the hierarchy.

Implementation

The hierarchy of classifiers was implemented using python's default dictionary data structure which internally acted as a tree hierarchy. A depth-first-search (DFS) approach was implemented using a last-in-first-out (LIFO) strategy for developing the hierarchy. However, different strategies and data structures can be used for further optimizations. A generic algorithm template for implementing the hierarchy of classifiers of any depth along with multi-label support is provided in the later section of this chapter.

4.2.2 Over-fitting

The hierarchical structure of classifiers reduces the amount of training data as we go down each level. At every level, classifiers become specific to a particular set of attributes. This results in an increased chance of over-fitting leading to poor generalization on unseen data samples. Over-fitting increases the probability of miss-classifications and the error rate grows further down the hierarchy. A high miss-classification rate can cause the proposed indexing strategy to fail as incorrect predictions would lead to an incorrect subset retrieval. If an incorrect subset is retrieved quite frequently, then the system would be unable to find matching images for a given query image despite being present in the database.

Thus, to minimize over-fitting, extreme care and rigorous testing on unseen datasets were conducted to assess the performance of trained classifiers on unseen data samples.

4.2.3 Error Propagation

Similar to over-fitting, another common problem associated with hierarchical classification is of error propagation. Classification error is associated with every classification model, as no model can have a 100% prediction rate. Due to the hierarchical structure, the classification error is propagated down the hierarchy and gets scaled at each level. This error propagation is minimized by ordering the hierarchy with decreasing classification accuracy from top-to-bottom. The most accurate classifier was at the top-most level that minimized error propagation while the least accurate classifier was at the bottom-most level.

To improve the performance of the hierarchical classification and significantly reduce the error propagation, this research study proposes a novel probabilistic back-tracking algorithm that rectifies miss-classifications based on conditional probabilities. This novel back-tracking algorithm is explained in detail in the next Chapter 5.

4.3 The Indexing Structure

The proposed indexing strategy utilizes a hash table that maps a unique hash index to a subset of the database. The hash indexes are uniquely computed using a deterministic hashing function applied on the keys. During run-time, this hash table is queried with the key generated from the predicted attribute values obtained from the hierarchy of classifiers, and an appropriate subset of the database is retrieved. An overview of this indexing structure was demonstrated previously in Figure 4.1.

4.3.1 Hash Table

Hash Table vs. B-Tree

For this research study, the hash table is built using python's default dictionary type. A quick research about the performance of python's dictionary vs commonly used database indexing structure of B-trees provided sufficient evidence that python's dictionary has an average lookup complexity of $O(1)$ as opposed to B-tree's average lookup complexity of $O(log\ n)$. Moreover, python dictionaries are extremely easy to implement with a built-in dictionary data structure support.

The main issue with hash tables is that the computed hash values (hash index) can collide i.e., two distinct keys can sometimes have the same hash value. These collisions are resolved using Open Addressing. In open addressing, all elements are entirely stored in the hash table. Thus, at any point, the size of the hash table must be greater than or equal to the total number of keys. For this research study, the total number of keys is finitely deterministic, which is equivalent to the total number of subsets and could be easily determined as 19 by merely counting from the hierarchical design depicted in Figure 4.3.

Keys

To build a unique hash index, distinct keys are required on which a hash function computes a unique hash value. In the proposed indexing strategy, a key is composed of a concatenated string of predicted attribute values. These values are depicted in the Figure 4.3.

An example of a key with predicted values of Above10, Male, Above19, Below35, Asian is as follows:

key = A10_M_A19_B35_A

A default hash function from the Python's library is used to compute a unique hash

index for a given key. This hashing function is deterministic i.e. the same key will always produce the same hash index.

The below equation represents a simple deterministic hash value computation:

$$index = Hash(key) \tag{4.1}$$

4.3.2 Mapping Database Subsets to Hash Table

Using the proposed indexing strategy, the hash table can be pre-computed for a given database of images and each hash index from the hash table will point to a particular subset of the database.

Database Subset Creation

For creating a total of 19 subsets, the original database was partitioned hierarchically using the ground-truth labels of all images mirroring the hierarchical design of the classifiers, as depicted in Figure 4.3. This partitioning was done at the initial offline stage using a simple folder-based storage similar to how files and folders are partitioned on Windows. The design and the partitioning of the database to create different subsets can be easily adapted and changed to any variant such as SQL tables or online cloud storage.

An example of database partitioning strategy using folder-based storage would produce hierarchical paths as shown below for all subsets:

- **Subset 1 :** Database/Above10/Female/Above19/Above35/Asian/
- **Subset 2 :** Database/Above10/Female/Above19/Above35/Indian/
- **Subset 3 :** Database/Above10/Male/Above19/Below35/White/ and so on.

The next step involved the extraction of 128D feature vectors using the pre-trained CNN model from dlib[1] library for all images from each of the 19 different subsets. These 128D feature vectors along with their local image paths were stored on the disk as 19 separate pickle files, one for each subset, and serialized using python's pickle[2] library. Each image record in a subset file had two entries of the 128D feature vector and the local image path.

The following Figure 4.4 depicts a sample subset file created with ground truth labels of Above10, Female, Above19, Above35, and Asian. These labels can also be noted from the image file paths. This subset file contains a list of images with their computed 128D feature vectors and their file paths.

A	B
Filepath	**128-D Vector**
/Database/Above10/Female/Above19/Above35/Asian\48_1_2_20170109002813437.jpg	[-0.09640963 0.1034953 0.04552925 -0.09670324 -0.12446711 -0.01329977
/Database/Above10/Female/Above19/Above35/Asian\39_1_2_20170116192459890.jpg	[-7.24911615e-02 1.42071649e-01 7.15052336e-02 -1.44412190e-01
/Database/Above10/Female/Above19/Above35/Asian\39_1_2_20170103183338179.jpg	[-9.06836390e-02 5.82381487e-02 8.86602998e-02 -1.17616683e-01
/Database/Above10/Female/Above19/Above35/Asian\42_1_2_20170104201224233.jpg	[-0.04198483 0.09569852 0.09201422 -0.09565828 -0.12268027 -0.04609513
/Database/Above10/Female/Above19/Above35/Asian\53_1_2_20170105174705949.jpg	[-0.06912729 0.07506439 0.08278649 -0.01212266 -0.06582824 0.03119363
/Database/Above10/Female/Above19/Above35/Asian\50_1_2_20170104023119069.jpg	[-0.06159835 0.03421573 0.19620125 -0.0812647 -0.06282505 -0.02015073
/Database/Above10/Female/Above19/Above35/Asian\74_1_2_20170105174904109.jpg	[-0.12736401 0.1127274 0.10874937 -0.01940372 -0.09800244 -0.03030964
/Database/Above10/Female/Above19/Above35/Asian\56_1_2_20170105000902100.jpg	[-0.06021681 0.09125583 0.22886924 -0.09512344 -0.02851994 -0.03563493
/Database/Above10/Female/Above19/Above35/Asian\45_1_2_20170109220409055.jpg	[-0.12762597 0.08305496 0.10259143 -0.062564 -0.14834639 -0.00194242
/Database/Above10/Female/Above19/Above35/Asian\50_1_2_20170104210635740.jpg	[-0.05914336 0.03474727 0.01643246 -0.00974853 -0.06732726 -0.0806943
/Database/Above10/Female/Above19/Above35/Asian\80_1_2_20170120223552933.jpg	[-0.07785255 0.15636827 0.03284593 -0.00367793 -0.1149388 0.0220891
/Database/Above10/Female/Above19/Above35/Asian\37_1_2_20170109013335990.jpg	[-5.50127961e-02 8.10918957e-02 7.06319958e-02 -6.56993911e-02
/Database/Above10/Female/Above19/Above35/Asian\40_1_2_20170116164250912.jpg	[-0.13638732 0.1288016 0.13475679 -0.15665804 -0.0949989 -0.01037885
/Database/Above10/Female/Above19/Above35/Asian\61_1_2_20170117123801917.jpg	[-1.73745066e-01 5.68968579e-02 1.44129604e-01 5.71257286e-02
/Database/Above10/Female/Above19/Above35/Asian\37_1_2_20170105172724340.jpg	[-0.05835754 0.02364703 0.08423079 -0.07452448 -0.10346545 0.00459544
/Database/Above10/Female/Above19/Above35/Asian\85_1_2_20170110183501116.jpg	[-0.09185798 0.07011067 0.01203827 -0.05387553 -0.11695434 -0.04767045
/Database/Above10/Female/Above19/Above35/Asian\39_1_2_20170105170215020.jpg	[-0.03165264 0.01862572 0.04659446 -0.05488689 -0.0746808 -0.06600889
/Database/Above10/Female/Above19/Above35/Asian\38_1_2_20170109140654630.jpg	[-0.05931004 0.05852328 0.13192284 -0.11346279 -0.04600082 -0.04005054

Figure 4.4: A Sample Subset File

Database Subset Mapping

Mapping the above created 19 subset files to the hash table was an easy task. A default hash function from python's library was used to compute a unique hash value (hash index) for each of the 19 keys corresponding to 19 subsets. A dictionary data structure was used to maintain the hash table where the dictionary's index was the computed hash value (hash index) that pointed to the path of the generated subset files. This dictionary

[1] https://github.com/davisking/dlib

[2] https://docs.python.org/3/library/pickle.html

stored 19 unique hash index values, each pointing to a different subset file.

The following Figure 4.5 illustrates the underlying subset mapping mechanism where the hash table is maintained logically as a dictionary data structure. The dictionary indexes represent 19 uniquely computed hashes for all the 19 keys pointing to one of the subset file paths.

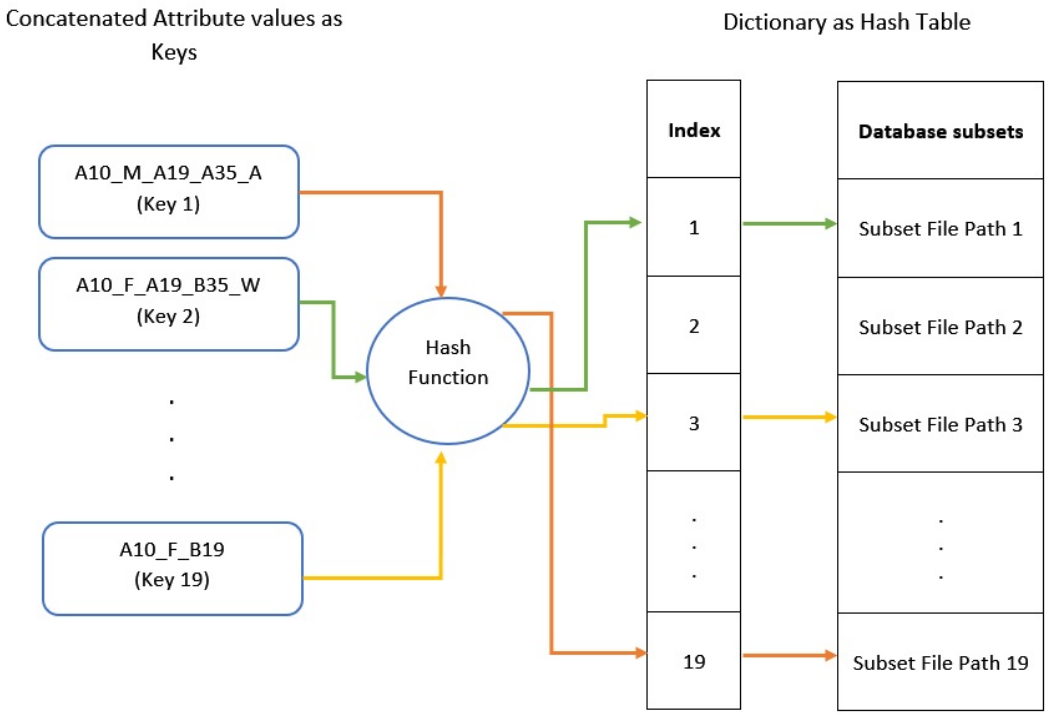


Figure 4.5: Mapping database subsets to the hash table dictionary

Run-time Subset Retrieval

For a given query image during run-time, the hierarchy of classifiers predict different attribute values which are concatenated to form a key. A hash value computed from this key is used to look-up the pre-computed hash table and retrieve an appropriate subset file containing the 128D feature vectors and the image paths. Top matching images are retrieved by computing similarity scores between the 128D feature vector of the given query image and all the 128D feature vectors contained inside the retrieved subset.

The following Figures 4.6 and 4.7 respectively depict the hierarchical and logical views for run-time subset retrieval using the proposed indexing strategy.

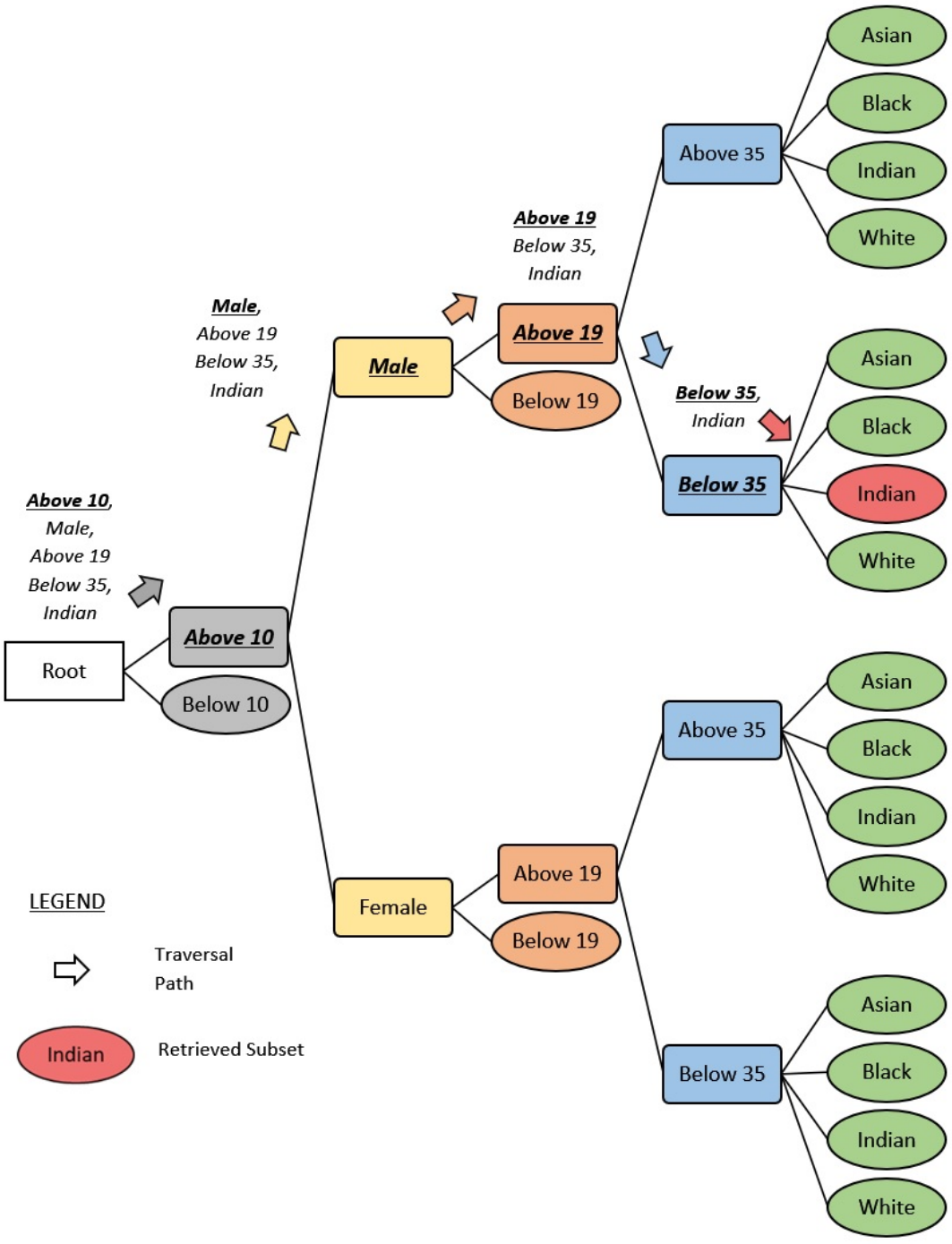


Figure 4.6: Hierarchical view of subset retrieval using proposed indexing strategy

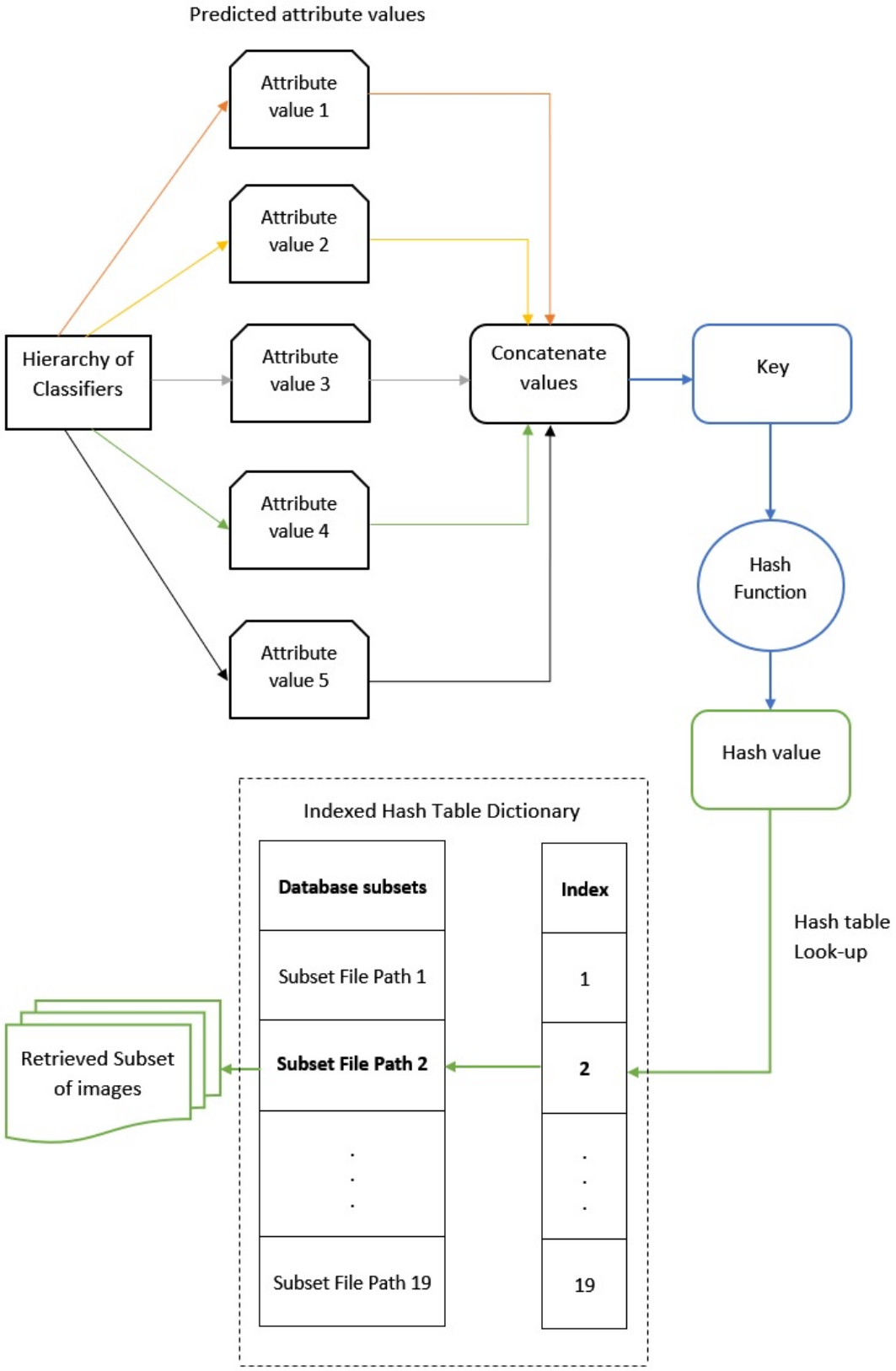


Figure 4.7: Logical view of subset retrieval using proposed indexing strategy

Figure 4.6 demonstrates the traversal path leading to a subset selection and retrieval based on the predicted attribute values obtained one by one from each classifier in the hierarchy.

Figure 4.7 shows a more logical view of subset selection and retrieval. The hierarchy of classifiers consists of seven classification models as noted earlier in table 4.2. These seven classification models are arranged hierarchically at a depth of 5 as depicted in Figure 4.3 such that they produce a prediction output of 5 different values one obtained from each depth. Only 5 predictions are obtained instead of 7 because the query image will either belong to a male or a female sub-hierarchy, thereby requiring only 5 classification models. These five attribute values are concatenated to form a key, generate a hash value and look-up the hash-table dictionary for retrieving an appropriate subset of the database.

Since the prediction time taken by classification models is negligible, the time required to retrieve the subset during run-time is also negligible. Thus, the whole database is reduced to a smaller fraction without any additional computation time. This drastically reduces memory usage and search time to find matching images. This is the main advantage of the proposed indexing strategy.

4.4 Indexing Algorithms

This section provides algorithms for subset generation, subset retrieval, hash table indexing and designing of hierarchical classification.

4.4.1 Hash Table Indexing Algorithm

Algorithm 1: Constructing indexed hash table

Result: Indexed hash table

1 Retrieve the list of **subset paths**

subset_paths = GetSubsetPaths()

2 Create an empty dictionary to maintain the **hash table**

hash_table = dictionary.empty()

3 **for** *Path p in subset_paths* **do**

4 Obtain the **key** from the subset path **p** by concatenating the first alphabet and age-group number (if present) from each value of the path (eg. $A10\_M\_A19\_A35\_I$)

key = ConvertPathToKey(p)

5 Compute the hash value (index) on the key

index = Hash(key)

6 Map the computed **index** of the dictionary to the subset's file path **p**

hash_table[index] = p

7 **end**

8 Serialize the hash table and save it to a file on disk using pickle library

pickle.dump(hash_table)

9 end of program

4.4.2 Subset Generation Algorithm

Algorithm 2: Subset Generation

Result: Generated database subsets of images

```
Initialize empty list of subset paths
  paths = list.empty()
for all images in database do
    Determine all label values for current image I using the ground truth labels
      Labels = Ground_Truth_Labels(I)
    for value v in Labels do
        Concatenate value v in each iteration to form a local Windows directory
          temp_p = string.concatenate(temp_p,v)
    end
    Create a local Windows directory at path p
      p = windows.mkdir(temp_p)
    if path p is not already added to paths list then
        Add the directory path p to the paths list
          paths.Add(p)
    end
    Move the current image I to the computed directory path p
      MoveImage(I, p)
end
for Path p in paths list do
    Initialize an empty subset data container
      subset = empty()
    Get list of images in the current directory at path p
      images = GetImagesInDir(p)
    Compute 128D vector for all images using pre-trained CNN model
      vectors = cnn.ComputeVectors(images)
    Add the generated 128D vectors into the subset data container
      subset.Add(vectors)
    Also add the image paths of all images in the current directory
      subset.Add(GetImagePaths(images))
    Serialize the subset and save it to a file at path p using pickle library
      pickle.dump(subset, p)
end
end of program
```

4.4.3 Subset Retrieval Algorithm

Algorithm 3: Subset Retrieval by Querying the Hash Table

Result: Retrieved subset of images

Retrieve the pre-computed **hash table**

hash_table = LoadHashTable()

Retrieve hierarchically arranged **classifiers**

hierarchical_classifiers = LoadClassifiers()

Fetch the input query image **Q**

Q = GetInputImage()

Initialize empty **key** string

key = string.empty()

for *classifier c in hierarchical_classifiers* **do**

  Predict attribute value **v** using the classifier **c**

  v = c.predict(Q);

  Concatenate first alphabet and the age-group digits (if present) for value **v** in each iteration to form the **key** (eg. $A10\_M\_A19\_A35\_I$)

  key = string.concatenate(key, v)

end

Compute the hash value (index) on the concatenated **key**

index = Hash(key)

Query the **hash table** using the computed **index** to retrieve an appropriate **subset**

subset = hash_table[index]

Return the retrieved subset of images

return **subset**

4.4.4 Algorithm for designing the hierarchy

Algorithm 4: Building hierarchy of classifiers using depth-first-search

Result: Constructed hierarchy of classifiers

1 Initialize empty tree data structure and retrieve list of trained classifiers

2 Select the next classifier **C_max** with maximum classification accuracy for a given classification category from the remaining classifiers in the list
C_max = Max(classifiers, category)

3 Add this classifier into the hierarchy (tree data structure)
tree.AddNode(C_max)

4 Fetch list of attribute categories **A** classified by **C_max** classifier
A = C_max.GetAttributeCategories()

5 Remove **C_max** model from the classifiers list
classifiers.Remove(C_max)

6 **for** *category c in Attribute Categories* **do**

7 | Create a new child node for current classification category **c**
| child_node = parent_node.CreateChildNode(c)

8 | Store this new **child node** in a list to visit later using Last-in-first-out (LIFO) strategy for depth-first-search
| child_nodes_list.Push() = child_node

9 **end**

10 **if** *child_nodes_list is not empty* **then**

11 | Remove next node from the child nodes list and assign it as the current parent node from which new child nodes would be created
| parent_node = child_nodes_list.Pop()

12 **else**

13 | goto Step 20

14 **end**

15 **if** *Current parent_node is not a subset* **then**

16 | Hierarchy is still incomplete thus, continue adding classifiers for current classification category
| category = parent_node.GetCategory()
| goto Step 2

17 **else**

18 | If this node is a subset, no further operations are required, so skip it and select the next child node to visit from the child nodes list
| goto Step 9

19 **end**

20 Return constructed hierarchy of classifiers
return tree

4.5 Training Classification Models

This section explains the training process and the steps applied to train all classifier models. Seven classifier models were trained as presented in Table 4.2.

Achieving high accuracy on the trained classification models was extremely crucial for the success of the proposed methodology. Having a low classification error was a necessity for the proposed methodology. An incorrect prediction at any level in the hierarchy would lead to error propagation and ultimately retrieval of an incorrect subset which may not include any matching images for the given query images. If this was to be a reoccurring theme of selecting an incorrect subset, then it would be catastrophic and could be considered as a system failure since matched images were not retrieved despite being present in the database.

Following Figure 4.8 depicts the training process for each classification model.

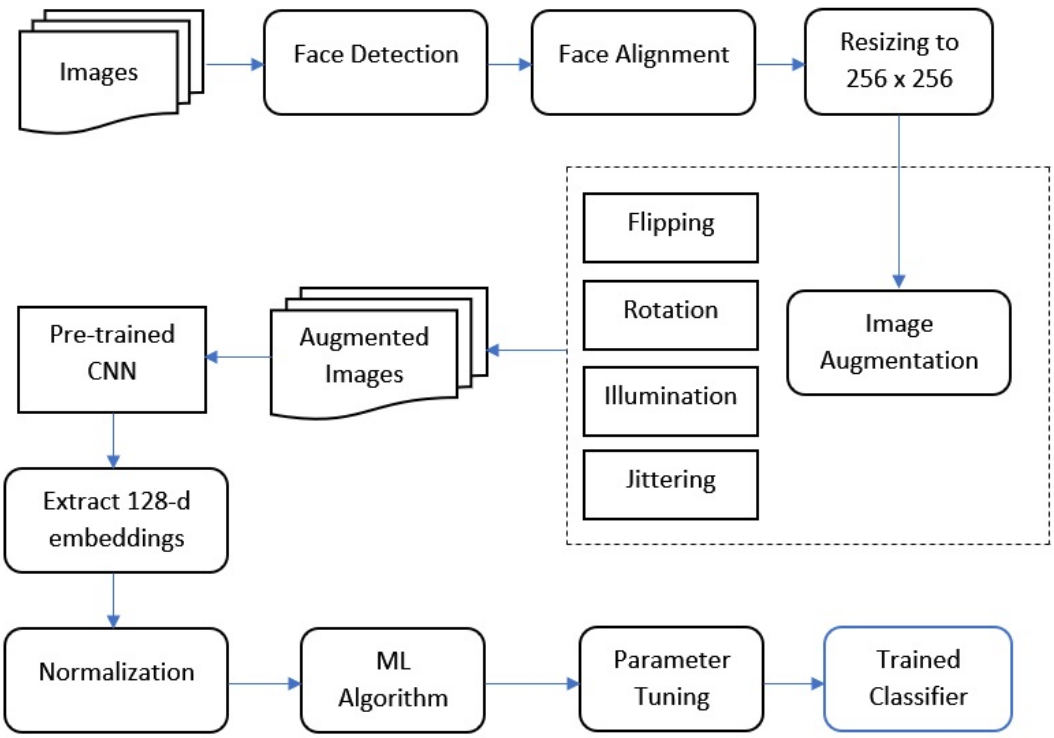


Figure 4.8: Training procedure for classification models

Training Process

- **Step1:** All database images underwent previously discussed pre-processing steps of face-detection, cropping and alignment and finally getting resized to a size of 256 x 256 pixels. Images where a face was not detected were simply discarded.
- **Step2:** Image data augmentation techniques were applied such as modifying the original image rotation, image flipping, image jittering, tweaking illumination conditions such as brightness and contrast. Ten new augmented images were created for each image using the above-mentioned augmentation techniques. The number 10 was chosen to limit the time requirement and limit the amount of training data due to additional hardware requirements and time constraints.
- **Step3:** Next step was to extract the 128D embeddings for all images using the pre-trained CNN model from the dlib library.
- **Step4:** Extracting 128D embeddings for all images in the database was a time-consuming process. Thus, to drastically speed up the extraction operation, multi-processing libraries provided with Python packages were utilized. This allowed for using all 8-CPU cores which reduced the time taken for feature extraction by a factor of approximately 8.
- **Step5:** Quick dimensionality reduction techniques were applied to decompose the 128D embeddings into a 2D vector space for visualization. This visualization provided evidence (based on discretely created cluster groups for each classification label) on whether the generated 128D embeddings could be used for age-group, gender and ethnicity classifications. Dimensionality reduction was performed using well-known Principal Component Analysis (PCA) (Jolliffe, 2011) and t-Distributed Stochastic Neighbour Embedding (t-SNE) (Maaten & Hinton, 2008) techniques.
- **Step6:** Visual inspection of the 128D embeddings showed the need to normalize

the embeddings before feeding them to train a classifier model for improving the performance.

- **Step7:** Normalized 128D embeddings were fed to different classifier models such as Support Vector Machine (SVM), Linear SVM, Multi-layer perceptron (MLP), K-nearest neighbour (K-NN) and variants of Naïve Bayes. Ensemble methods such as Gradient Boosting, Random Forest, and Voting classifiers were also used to determine if it improved the baseline performance.
- **Step8:** After determining the best classifier, parameter optimization was performed using random search and grid search techniques for further enhancing and maximizing the model performance.

Detailed training configurations and results for each classifier model are presented in Chapter 6 of Experimental Results.

Chapter 5

Proposed Methodology: Probabilistic Back-Tracking Algorithm

5.1 Proposed Probabilistic Back-Tracking Algorithm

This research study proposes a novel probabilistic back-tracking algorithm for improving the performance of the proposed indexing strategy based on hierarchical classification. Classification models introduce classification error because of miss-classifications as they cannot achieve 100% prediction accuracy on unseen data. For the proposed methodology, incorrect predictions will lead to incorrect subset retrieval, leading to false matches or no retrieved matches for a given query image.

Figure 5.1 illustrates a logical view of an incorrect subset retrieval. The hierarchy of classifiers consists of seven classification models, as described earlier in Figure 4.3. These seven classification models predict 5 different attribute values obtained hierarchically one by one from each classifier prediction. Only 5 predictions are obtained instead of 7 because the query image will either belong to a male or a female sub-hierarchy, thereby requiring only 5 classification models.

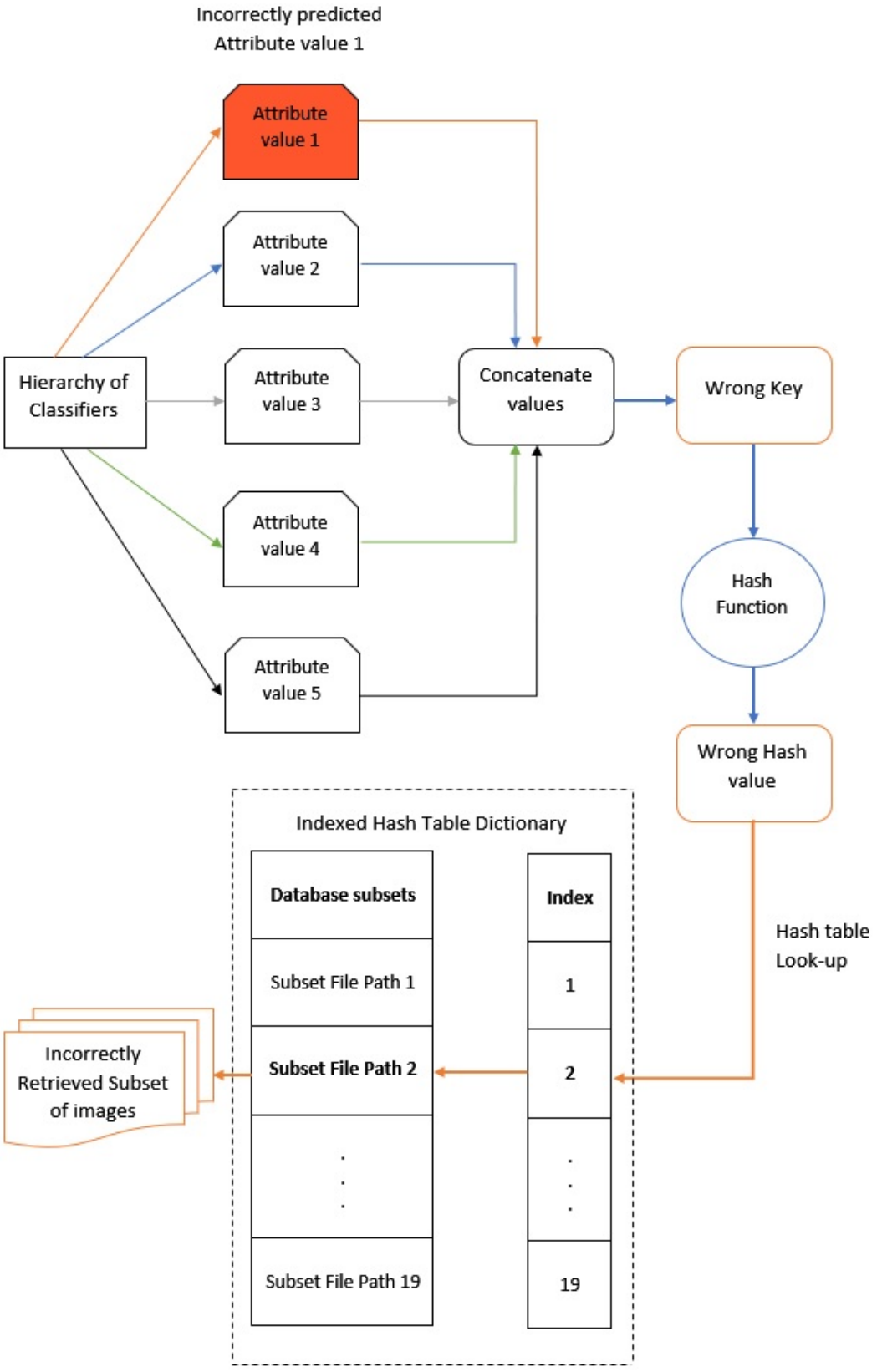


Figure 5.1: Incorrect subset retrieval

However, due to a miss-classified attribute value 1, a wrong key is generated after the concatenation of the five predicted attribute values, which in turn computes a wrong hash value. Querying this wrongly computed hash value with the hash table dictionary returns an incorrect subset. This subset is most likely to not contain matching images for a given query image due to a miss-classification on attribute value 1. Incorrect subset retrieval may produce false matches or cannot find matching images despite being present in the database.

To overcome this miss-classification problem, a novel probabilistic back-tracking algorithm is proposed in this research study. A back-tracking search is an exhaustive search algorithm that systematically assigns all possible combinations of values to variables and then checks if these assignments constitute a solution. This type of search is complete in the sense that a solution is guaranteed to be found. However, the worst-case running time complexity of a back-tracking search is exponential in nature (Egri & Shultz, 2015).

The back-tracking algorithm proposed in this research study is based on the use of conditional probabilities as formulated by the Bayes Theorem.

The Bayes theorem is stated mathematically as the following equation:

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)} \tag{5.1}$$

where,

- A, B are events and P(B) ≠ 0.
- $P(A \mid B)$ is the likelihood of event A occurring, given that B is true. It is also called as a Posterior probability.
- $P(B \mid A)$ is the likelihood of event B occurring, given that A is true. It is also called as likelihood.
- P(A), P(B) are probabilities of events A and B respectively happening independent

of each other.

- P(A) is referred to as the Prior probability of A occurring, while P(B) represents the evidence associated with event A.

5.1.1 Probabilistic Hypothesis Modelling

In equation 5.1 $P(B \mid A)$ and $P(A \mid B)$ are known as conditional probabilities. Conditional probability $P(A \mid B)$ can be defined as the probability of an event A occurring, given that another event B has already occurred.

Bayes theorem can also be used for hypothesis modelling. It can provide a probabilistic model that describes the relationship between a hypothesis and evidence (or observation) gathered in support of the event in question occurring.
Bayes theorem can be represented as follows in terms of hypothesis modelling:

$$P(H \mid O) = \frac{P(O \mid H) * P(H)}{P(O)} \tag{5.2}$$

where,

- H is the modelled hypothesis to be tested given the observation O.
- $P(H \mid O)$ is the posterior probability of the hypothesis holding true given the observation.
- $P(O \mid H)$ is the likelihood of observation given the hypothesis.
- P(H) is the prior probability of hypothesis before the observation.
- P(O) is the probability of observation regardless of the hypothesis.

Equation 5.2 of the formulation of Bayes theorem is utilized in this research study for computing the maximum likelihood of a correct classification value when the proposed system has miss-classified. Using this value, we can back-track the hierarchy to retrieve a proper subset.

Example usage of hypothesis modelling

Here, the hypothesis to be tested would be a classification value that is likely to be the actual value when a miss-classification from its classifier is observed.
As an example, consider that for a given query image the ethnicity classifier predicted an incorrect ethnicity value as Black, when the actual ethnicity value was Indian.

Hence the hypothesis to be tested would be the likelihood of having an actual ethnicity value of Indian given an observation of a miss-classified Black ethnicity value.

Consider predicted values of other attributes to be Above-10, Male, Above-19, and Above-35. Let us assume these to be correct predictions.
Substituting these values in equation 5.2, we get:

$$\mathbf{P}(\mathbf{H}_{\mathbf{indian}} \mid \mathbf{O}_{\mathbf{black\ mc}}) = \frac{\mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{indian}}) * \mathbf{P}(\mathbf{H}_{\mathbf{indian}})}{\mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}})} \tag{5.3}$$

where,

- $\mathbf{H}_{\mathbf{indian}}$ is the modelled hypothesis of correct classification value of ethnicity as Indian.
- Observation $\mathbf{O}_{\mathbf{black\ mc}}$ observed is a miss-classified ethnicity value of Black.
- $\mathbf{P}(\mathbf{H}_{\mathbf{indian}})$ is the prior probability of the modelled hypothesis of having a correct ethnicity value as Indian.
- $\mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}})$ is the probability of observation of a miss-classified ethnicity value as Black.
- $\mathbf{P}(\mathbf{H}_{\mathbf{indian}} \mid \mathbf{O}_{\mathbf{black\ mc}})$ is the posterior probability of the modelled hypothesis of having a correct ethnicity value as Indian when a miss-classification of Black ethnicity value has occurred.
- $\mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{indian}})$ is the likelihood of the occurrence of a miss-classified ethnicity value of Black given the hypothesis of ethnicity value to be Indian.

Equation 5.3 gives the probability of the hypothesis of taking a correct ethnicity value of Indian, given a miss-classified ethnicity value of Black. Using this probability computation, we can rectify the miss-classification of Black and back-track in the hierarchy to retrieve the proper subset belonging to the Indian ethnicity.

Figure 5.2 depicts the above example of observing an ethnicity miss-classification of Black. The algorithm performs back-tracking and retrieves the appropriate subset containing the correct ethnicity value of Indian.

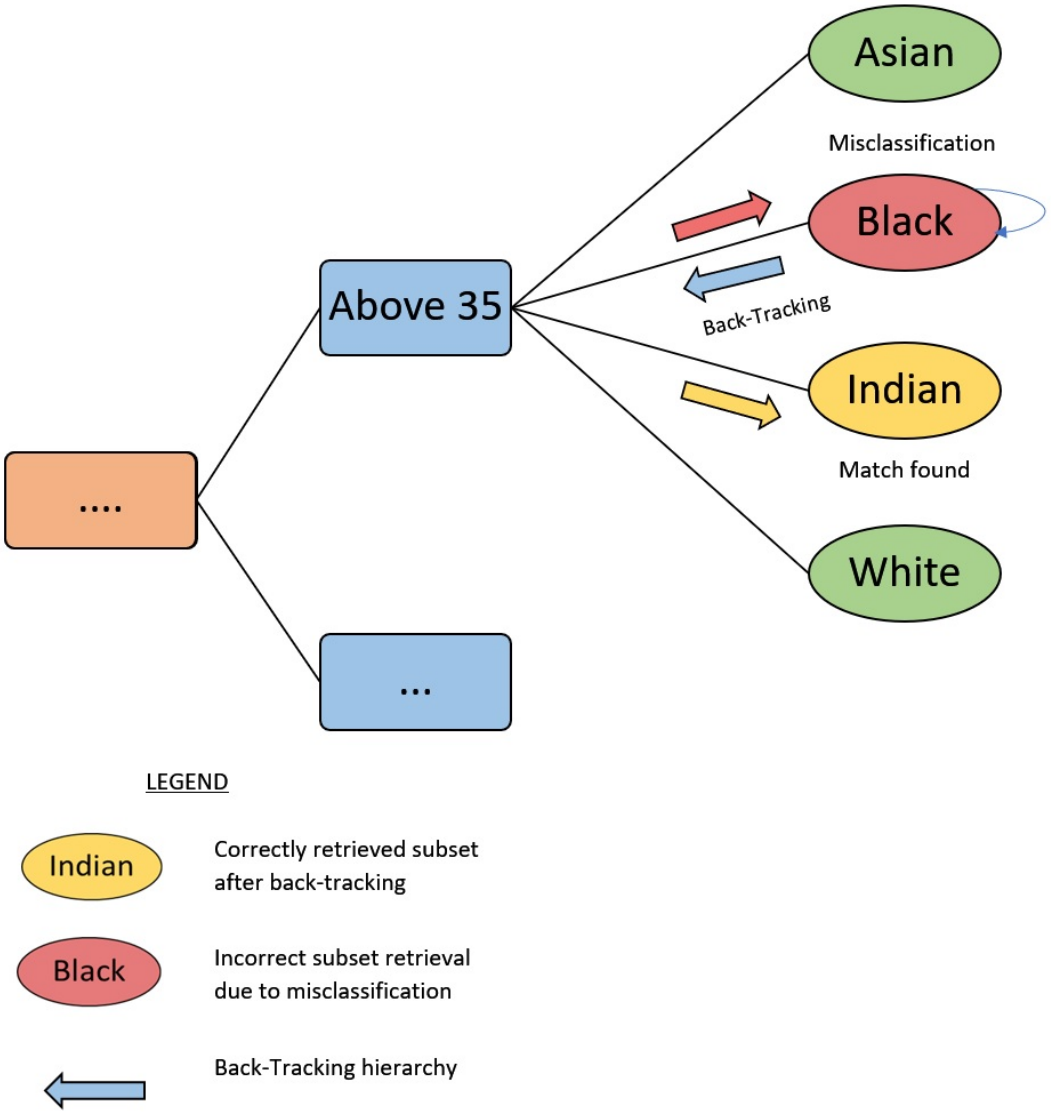


Figure 5.2: Back-tracking the hierarchy

5.1.2 Maximum-a-posteriori Hypothesis

In the above example, only a single hypothesis of correct ethnicity value to be Indian was modelled. However, in real-world scenarios, the actual values corresponding to miss-classifications are not known beforehand. Thus, multiple hypotheses (h1, h2, h3,...hn) need to be monitored simultaneously. Posterior probabilities of each modelled hypotheses (h1, h2, h3,...hn) are estimated with a test condition of each hypothesis holding to be true under the given observation corresponding to a miss-classification. Seeking a hypothesis with the maximum posterior probability is referred to as Maximum-a-posteriori, or MAP. The MAP hypothesis can be determined by using the above Bayes theorem 5.2, to calculate the posterior probability of each candidate hypotheses and selecting the one with maximum posterior probability.

Under this framework, the probability of the observation (O) is constant as it is used in the assessment of each hypothesis. Therefore, it can be removed from the calculation to give the simplified estimate function as follows:

$$\max \left[P(H_i \mid O) = P(O \mid H_i) * P(H_i) \right] \tag{5.4}$$

where,

- i is the i^{th} modelled hypothesis to be tested. (i ranges from 1 to n, n = total number hypotheses to be modelled)
- $\mathbf{P(H_i)}$ is prior probability of i^{th} modelled hypothesis.
- $\mathbf{P(H_i \mid O)}$ is the posterior probability of i^{th} modelled hypothesis holding true given an observation O.
- $\mathbf{P(O \mid H_i)}$ is the likelihood of observation given the i^{th} modelled hypothesis.

Example usage of MAP Hypothesis

Applying the above formula for this research study would correspond to modelling multiple hypotheses for different classification values when there is an observed miss-classification. As an example similar to the one discussed above of single hypothesis, if a miss-classification for the ethnicity value of Black is observed, then multiple hypotheses are modelled for each ethnicity category of Indian, Asian and White. An additional hypothesis for a classifier at one level up the current level is modelled for back-tracking to a different sub-hierarchy.

The MAP hypothesis is computed by finding the maximum posterior probability of each of the above candidate hypotheses. The MAP hypothesis represents the classification value having the maximum likelihood of being correct for a given miss-classification and is used to back-track and retrieve the appropriate subset. If the MAP hypothesis belongs to the one-level-up classifier, then the algorithm back-tracks to that level and proceeds thereon to a different sub-hierarchy.

Consider, for a miss-classification of ethnicity value of Black (under the assumption that other attributes such as Male, Above 35 are correctly predicted), the MAP hypothesis is estimated by computing individual posterior probabilities of each candidate hypotheses (h1,h2,h3, h4) and selecting the maximum as represented by the following equations:

$$\mathbf{h1} = \mathbf{P}(\mathbf{H}_{\mathbf{indian}} \mid \mathbf{O}_{\mathbf{black\ mc}}) = \mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{indian}}) * \mathbf{P}(\mathbf{H}_{\mathbf{indian}})$$

$$\mathbf{h2} = \mathbf{P}(\mathbf{H}_{\mathbf{asian}} \mid \mathbf{O}_{\mathbf{black\ mc}}) = \mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{asian}}) * \mathbf{P}(\mathbf{H}_{\mathbf{asian}})$$

$$\mathbf{h3} = \mathbf{P}(\mathbf{H}_{\mathbf{white}} \mid \mathbf{O}_{\mathbf{black\ mc}}) = \mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{white}}) * \mathbf{P}(\mathbf{H}_{\mathbf{white}})$$

$$\mathbf{h4} = \mathbf{P}(\mathbf{H}_{\mathbf{b35\ black}} \mid \mathbf{O}_{\mathbf{black\ mc}}) = \mathbf{P}(\mathbf{O}_{\mathbf{black\ mc}} \mid \mathbf{H}_{\mathbf{b35\ black}}) * \mathbf{P}(\mathbf{H}_{\mathbf{b35\ black}})$$

$$\mathbf{MAP\_H} = \max\left(\mathbf{h1}, \mathbf{h2}, \mathbf{h3}, \mathbf{h4}\right) \tag{5.5}$$

The goal of the proposed probabilistic back-tracking algorithm is to locate a hypothesis that best explains the observed data. In this case, based on probabilities, the MAP hypothesis would represent the most likely classification value for a given miss-classification. Using this MAP hypothesis, back-tracking can be performed to retrieve an appropriate subset.

5.2 Challenges in implementing the Back-Tracking Algorithm

5.2.1 Determining a misclassification

The main challenge in the proposed back-tracking algorithm was to determine when a miss-classification has occurred for applying the above-discussed formulae. As the actual values are not known beforehand, determining a miss-classification is difficult. One approach considered was to threshold the predicted class score to a low value such as 40%, below which would represent a miss-classification due to a low confidence of prediction. It would mean a faster rate of retrieval as the miss-classification check is performed before a wasteful retrieval of an incorrect subset. However, this solution does not cover all scenarios as there could be cases with high confidence values and yet actually be incorrect.

From a real-world application perspective, it was decided that a miss-classification occurs when no matching images are retrieved for a given query image. This guarantees that if no matches were found from the currently retrieved subset, the algorithm would check for matches at least once more using the next most likely destination subset based

on the computed posterior probabilities and the determined MAP hypothesis.

Therefore, the given observation condition is modified with an addition of “No Retrieved Matches” criteria because the posterior probabilities would be computed only when a “No Retrieved Matches” condition is satisfied deeming it to be a miss-classification. As a result, the above equation 5.3 slightly changes due to the addition of the "No Matches retrieved" criterion, as follows:

$$\mathbf{P}(\mathbf{H}_{\mathbf{indian}} \mid \mathbf{O}_{\mathbf{black\ \&\ nomatch}}) = \frac{\mathbf{P}(\mathbf{O}_{\mathbf{black\ \&\ nomatch}} \mid \mathbf{H}_{\mathbf{indian}}) * \mathbf{P}(\mathbf{H}_{\mathbf{indian}})}{\mathbf{P}(\mathbf{O}_{\mathbf{black\ \&\ nomatch}})} \quad (5.6)$$

where,

- $\mathbf{P}(\mathbf{H}_{\mathbf{indian}} \mid \mathbf{O}_{\mathbf{black\ \&\ nomatch}})$ is the posterior probability of the modelled hypothesis of having a correct ethnicity value as Indian given an ethnicity prediction as Black and no matches are retrieved.
- $\mathbf{P}(\mathbf{O}_{\mathbf{black\ \&\ nomatch}} \mid \mathbf{H}_{\mathbf{indian}})$ is the likelihood of the observation, of predicting ethnicity value as Black but no matches are retrieved given the hypothesis of ethnicity value to be actually Indian.
- $\mathbf{P}(\mathbf{H}_{\mathbf{indian}})$ remains the same prior probability of the modelled hypothesis of having a correct ethnicity value as Indian.
- $\mathbf{P}(\mathbf{O}_{\mathbf{black\ \&\ nomatch}})$ is the probability of observation of predicting ethnicity value as Black and no matches are retrieved.

All these probabilities can be pre-computed easily for all possible miss-classification combinations of different attributes of age-group, ethnicity, and gender. These probabilities were pre-computed based on simulations run using small samples selected by stratified random sampling technique to ensure that samples from all strata were equally selected, albeit randomly for each simulation.

5.2.2 Database Bias and dependency on prior probabilites

This algorithm so far did not consider the issue of bias. A database can be naturally biased towards one or more attribute values depending on the images. For example, a database may contain twice the number of images of the Indian ethnic group when compared to the White ethnic group, and so on. This results in an added bias due to the nature of the database which affects the probability computations. If prior probabilities were computed for the given database, these computations would be heavily biased representing the database state at the time of computation. In the real-world however, databases are not static and are updated regularly. Thus, prior probability computations would require regular updates making them database dependent.

If for example, assume that there are a total of 10,000 images present in the database of which 1000 belong to White ethnicity, 1500 belong to Black ethnicity, 2500 belong to Asians and the remaining 5000 belong to the Indian ethnic group. Thus, the static prior probability calculation $P(H_{indian})$, $P(H_{asian})$, $P(H_{black})$ and $P(H_{white})$ would be biased towards Indians as 50% of the database contains images of Indians. Thus, if the ethnicity classifier produced a miss-classification the proposed algorithm would always estimate the MAP hypothesis based on the above equation 5.6 as Indian due to the default database bias towards Indian ethnicity.

Below table 5.1 depicts the database bias and its effect on the prior probability computation.

Table 5.1: Database Bias and its effects on prior probabilities

Ethnicity	**Total Images**	**Prior Probability**
Asian	2500	2500 / 10,000 = 0.25
Black	1500	1500 / 10,000 = 0.15
White	1000	1000 / 10,000 = 0.10
Indian	**5000**	**5000 / 10,000 = 0.5**
Total	10,000	N/A

Dynamic prior probabilities

One solution for the above-discussed challenge of database bias and dependency would be to make all prior events equally likely so that the default bias of the database would be removed.

This would reduce the equation of modelling hypotheses:

$$\max\left[P(H_i \mid O) = P(O \mid H_i) * P(H_i)\right] \tag{5.4}$$

to the following equation:

$$\max \left[P(H_i \mid O) = P(O \mid H_i) \right] \tag{5.7}$$

where,

- i is the i^{th} modelled hypothesis to be tested. (i ranges from 1 to n, n = total number hypothesis to be modelled)
- $\mathbf{P(H_i \mid O)}$ is the posterior probability of i^{th} modelled hypothesis holding true given the observation O.
- $\mathbf{P(O \mid H_i)}$ is the likelihood of observation given the i^{th} modelled hypothesis.

However, this destroys the original nature of the database. While it removes the bias introduced in the prior probability computations, it is still dependent on the database due to the likelihood of observation which happens to be database dependent.

The solution to remove database dependency and default bias used in this research study was based on the extraction of dynamic probabilities. Instead of pre-computing the prior probabilities from the database simulations, they were extracted dynamically during run-time from the trained classifier model itself.

Most of the classification models have the capability to compute prediction probabilities for each of the classes. Exploiting this capability, we can easily extract the predicted probability for the rest of the classification categories and use this value as the dynamic prior probability. This prior probability can be considered as the current prior probability for the modelled hypothesis because it is computed prior to the observation of any miss-classification.

The final equation for estimating MAP Hypothesis after all modifications is as follows:

$$\mathbf{MAP\_H} = \max\left[\mathbf{P(H_i \mid O)} = \mathbf{P(O \mid H_i)} * \mathbf{P(H_{i\_classifier})}\right] \tag{5.8}$$

where,

- i is the i^{th} modelled hypothesis to be tested. (i ranges from 1 to n, n = total number hypothesis to be modelled)
- $\mathbf{P(H_i \mid O)}$ is the posterior probability of i^{th} modelled hypothesis holding true given the observation O.
- $\mathbf{P(O \mid H_i)}$ is the likelihood of observation given the i^{th} modelled hypothesis.
- $\mathbf{P(H_{i\_classifier})}$ is prior probability of i^{th} modelled hypothesis extracted from classifier during run-time.

An important point to be noted here is that, computing the dynamic prior probability during run-time by no means selects the second highest predicted class label as the MAP hypothesis. For example, for a miss-classified Black ethnicity, consider the predicted probabilities by the ethnicity classifier to be 5% for Asian, 20% for White, 60% for Black and 15% for Indian ethnic group. This does not result in guaranteed selection of White class label as the MAP hypothesis due to its second highest probability score; because the MAP hypothesis equation 5.8 also takes into consideration the likelihood of the observation under given modelled hypothesis ($\mathbf{P(O \mid H_i)}$).

5.2.3 Constrained Back-Tracking

Another challenge was in determining when to halt the back-tracking process. Back-tracking is a complete search meaning it will always find the correct solution if one exists. However, if back-tracking is not halted, the search time complexity could be

exponential and might be even worse than a brute-force method. Thus, determining how much to back-track and when to halt the back-tracking plays a vital role in quick and accurate retrieval of matching images.

The proposed indexing strategy has a hierarchical depth of five, with the last two levels having the lowest classifier accuracy scores. Thus, it was decided that back-tracking would be constrained to a maximum of one level above the current depth of miss-classification and limited for rectifying a maximum of two miss-classifications after which it would be halted. Back-tracking is only required in case of miss-classifications. Other classifier models at the upper levels of 1,2 and 3 in the hierarchical design have notably high accuracy scores leading to fewer miss-classifications and thus do not require to be back-tracked to these levels. This limited the back-tracking search to the lower depths of the hierarchy, where miss-classifications would be more likely when compared to the upper level of the hierarchy.

Constrained back-tracking also ensures that a true negative most likely remains as a true negative. A true negative in the scenario of face-image retrieval would mean the system returns no matches for the given query image when in fact, the given query image had no matching images in the database. Having a high true negative rate is also a vital performance metric for evaluating a face-image retrieval system as then fewer false matches would be retrieved due to a high true negative rate.

5.2.4 Back-Tracking Algorithm

This section presents the proposed probabilistic back-tracking algorithm that computes and retrieves an appropriate subset.

Algorithm 5: Back-Tracking

Result: New back-tracked subset

if *No Matches Retrieved and back-tracking is allowed* **then**
  Goto Step 6
else
  Goto Step 19
end

Model **n** number of candidate hypotheses for a given miss-classification

Estimate MAP Hypothesis by computing posterior probabilities of all candidate hypotheses using equation 5.8

$$\mathbf{MAP\_H} = \max\left[\mathbf{P(H_i \mid O)} = \mathbf{P(O \mid H_i)} * \mathbf{P(H_{i\_classifier})}\right]$$

if *MAP_H belongs to classifier one level above current level where miss-classification occurred* **then**
  goto Step 13
else
  goto Step 14
end

Discard the current subset and back-track to the classifier one level above the current level to obtain new predictions and
goto Step 15

Discard the current subset and back-track to the new subset which is present at the same depth to obtain a new key

Generate a new key

Generate a new hash value using the new key

Query the hash table using the new hash value and retrieve a new subset

Return newly selected subset

end of program

5.3 Image Matching and Ranking

After the indexing phase is complete and an appropriate subset of the database is identified, image matching and ranking operations are performed to retrieve top matching images for a given query image from the subset.

5.3.1 Approximate Nearest Neighbour Search

Nearest neighbour algorithms are famously used for image matching or similarity search. Unfortunately, they do not scale up and their computational speed decreases rapidly with large scale databases. Even though Nearest Neighbour algorithms are highly accurate, they tend to suffer from the curse of dimensionality as images are generally represented using high-dimensional feature vectors. A linear search using high-dimensional feature vectors on a large scale database is extremely expensive in terms of time and computational costs (Kumar, Zhang & Nayar, 2008).

To overcome this curse of dimensionality issue and to reduce time and computational costs, Approximate Nearest Neighbor (ANN) algorithms are preferred for similarity search or matching operations on large scale databases. ANN algorithms provide a healthy balance between performance and computational complexity.

ANN's are particularly useful as the returned approximate data samples can be as good as the exact ones. If the distance measure used to compute the distance between a query and its nearest points is able to estimate the relationship adequately, then small differences in the distance computations do not matter (Andoni & Indyk, 2006).

There are a variety of ANN algorithms such as Locality Sensitive Hashing (LSH) (Gionis, Indyk & Motwani, 1999), early terminating K-D trees (Ram & Sinha, 2019), Product Quantization, and other hashing or inverted index algorithms reviewed earlier in Chapter 2. For this research study, two open-source ANN implementations are used of

which one is based on product quantization[1] based on original paper by (Jegou, Douze & Schmid, 2010) and other is called ANNOY[2] based on random projections which is an open-source algorithm commercially used for music recommendations in Spotify[3]. These two algorithms were specifically chosen for the performance comparison of face-image retrieval with the proposed methodology. Product quantization[1] is one of the best algorithms from the reviewed existing state-of-the-art face-image retrieval systems and ANNOY[3] is a top-performing ANN algorithm as per the ANN benchmarks[4] also published in the paper (Aumüller, Bernhardsson & Faithfull, 2017).

5.3.2 Distance Metrics or Similarity Measures

A distance metric or a similarity measure quantifies the similarity between two objects. A good selection of distance metric or a similarity measure will help the ANN algorithm to return matching images for the given query image accurately. There are different kinds of distance metrics or similarity measures. This research study uses Euclidean distance and Cosine similarity (Cha, 2007), as the distance metrics in the computations to find matching images for the given query image.

- **Euclidean Distance:** The Euclidean distance between two points is the length of the path connecting them. The Pythagorean theorem gives this distance between two points. It is the simplest and most widely used distance metric.

$$D(x, y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2} \tag{5.9}$$

- **Cosine Similarity:** Cosine similarity metric finds the normalized dot product of the two vectors. It focuses on the angle of orientation rather than the magnitude

[1] https://github.com/matsui528/nanopq
[2] https://github.com/spotify/annoy
[3] https://www.spotify.com/nz/
[4] https://github.com/erikbern/ann-benchmarks

of the vectors.

$$D(x,y) = \frac{\sum_{i=0}^{n-1} x_i y_i}{\sum_{i=0}^{n-1}(x_i)^2 * \sum_{i=0}^{n-1}(y_i)^2} \tag{5.10}$$

5.3.3 Thresholding

A distance metric measures a similarity distance between all images and a given input query image. The computation results in a value usually ranging from (0,1) based on the chosen distance metric, where 0 signifies a perfect match, whereas 1 signifies two different images. Therefore, a thresholding value needs to be set for which values less than the threshold are treated as matches and values greater than the threshold are treated as non-matches for a given query image.

Thresholding allows for the calculation of false acceptance and false rejection rates which are essential performance evaluation metrics for a face-image retrieval system. From a real-world application perspective, thresholding provides user control as the threshold value controls the strictness of the face-image retrieval system. A higher threshold value will return a greater number of matched images but may sacrifice accuracy, whereas a lower threshold value will return lesser and accurate matches but may not be able to detect some of the actual matches.

In this research study, the threshold value was empirically set to 0.5 for Euclidean distance measure and 0.05 for Cosine Similarity.

5.3.4 Proposed Dynamic Thresholding

This research study also makes a novel contribution related to thresholding.

Exploiting the capabilities of hierarchical classification, different threshold values can be set for different database subsets. This research study proposes a dynamic thresholding value to be set differently based on predicted ethnicity attribute and age-group below 19 category at run-time. The dynamic threshold values for Euclidean

distance measure and Cosine Similarity are set empirically as shown in table 5.2.

Table 5.2: Dynamic Threshold values

Category	**Euclidean Distance**	**Cosine Similarity**
Asian	0.4	0.04
Black	0.42	0.05
Indian	0.46	0.052
White	0.48	0.055
Below 19	0.3	0.03

For example, if for the given query image, the ethnicity was predicted as Asian, then for distance computation the threshold value would be dynamically set to 0.4 or if the ethnicity was predicted as Black, then the threshold value is set to 0.45 at run-time rather than using a single threshold value of 0.5 for all, as in the case of the Euclidean distance metric.

The dynamic thresholding scheme does not require any additional computations as the predicted attributes are a part of the face-image retrieval process. This scheme can be further extended to other attributes in the hierarchy as well based on the gender and different age-group attributes.

Experimental results in the next Chapter 6 provide evidence for the success of dynamic thresholding whereby the performance of the retrieval system is improved with the use of dynamic thresholding rather than using a single thresholding value.

5.3.5 Image Ranking

Image Ranking is an optional post-processing step implemented to re-rank the top retrieved images from the image matching process. Re-ranking helps to remove any falsely matched images thereby improving the accuracy of face-image retrieval systems. There are several different methods developed for re-ranking ranging from the use of multiple distance metrics or commercial software to deep learning based re-ranking algorithms. However, re-ranking usually involves additional post-processing computations, thereby increasing the retrieval time. Hence, due to the additional increment in retrieval time and experimental success without application of re-ranking algorithms, image re-ranking step was not implemented in this research study.

5.3.6 Image Matching Process

The following Figure 5.3 depicts the image matching and ranking process. The already computed 128D feature vector for a given query image is used as an input to the Approximate Nearest Neighbour algorithms for finding top matched images. The top matching image list is retrieved based on the distance / similarity measure used in the computation for finding the similarity score between the 128D vector of the input query image and 128D vectors of all images present in the retrieved subset. A threshold value determines the number of images retrieved in the list. Finally, an optional re-ranking step re-orders the images inside the list that may remove any falsely matched images.

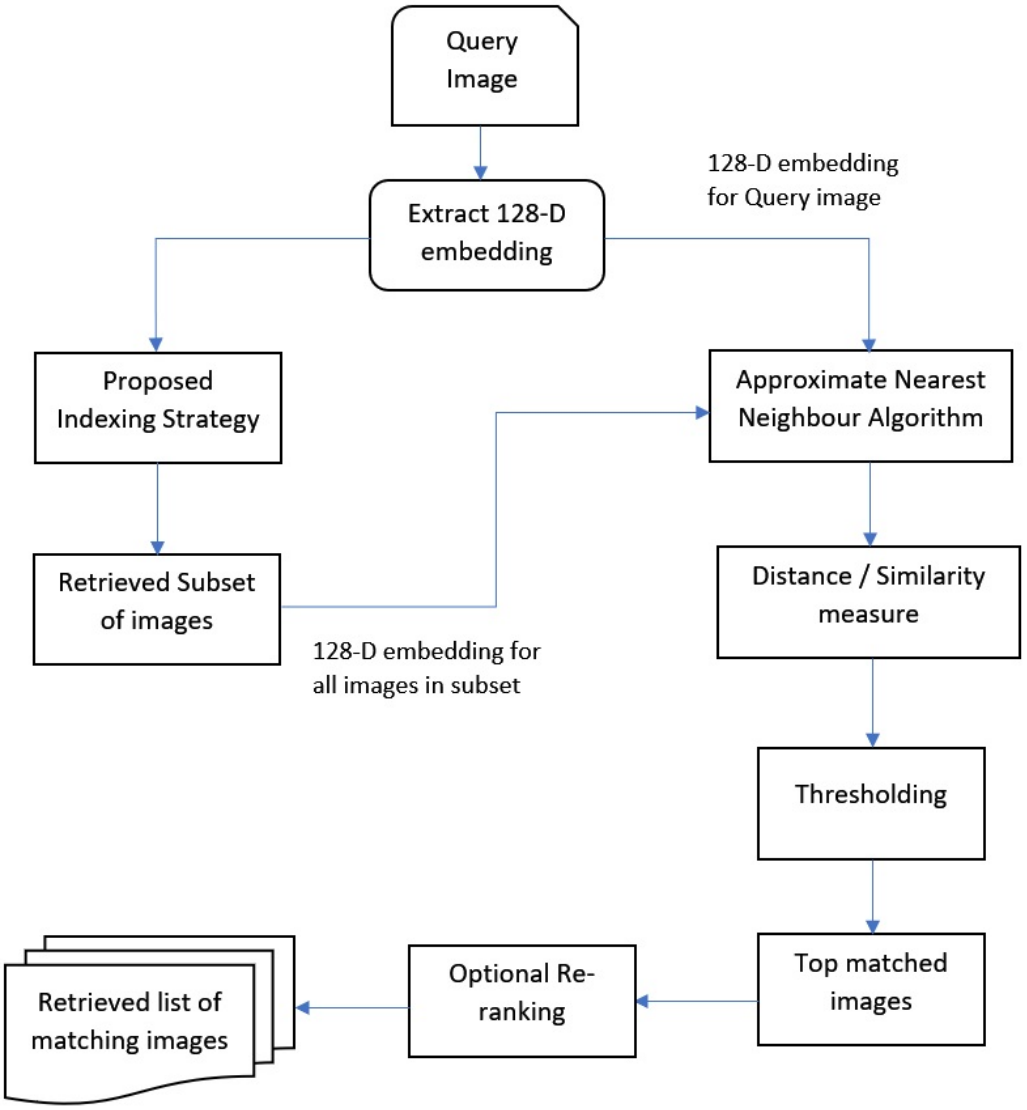


Figure 5.3: Image Matching Process

The distance comparison is made only on a small subset of the database, retrieved from the proposed indexing strategy thereby maintaining the computational complexity and reducing retrieval times significantly as compared to the entire image database.

Chapter 6

Experimental Results

This chapter details the experimental evaluation of the proposed methodology. Numerous experiments were conducted for comparing and evaluating the performance of the proposed indexing strategy, as follows:

- **Preliminary Test:** Preliminary tests were performed for selecting an appropriate pre-trained CNN model. The best transfer learning strategy of using the pre-trained CNN as a feature extractor or fine-tuning of weights was determined using the preliminary test results.
- **Evaluating Classification Models:** The trained classification models were evaluated for accuracy using 10-Fold cross-validation as well as hold-out strategies.
- **Closed-Set Evaluation:** The face-image retrieval system using the proposed methodology was evaluated under closed-set and open-set evaluation protocols. Closed-set meant the knowledge of whether the matching images are present in the database is known prior to the computation. For the closed-set protocol; accuracy, precision, recall, and other metrics were used to measure the performance.
- **Open-Set Evaluation:** In the open-set case, it is unknown beforehand whether

matching images are present in the database. Open-set reflects real-world scenarios and hence is an important evaluation protocol. For open-set protocol; mean average precision, precision@k, and retrieval time were used as performance measures.

- **Statistical testing:** The paired t-test was performed to analyze and statistically evaluate the significance of the performance of the proposed probabilistic backtracking and dynamic thresholding techniques.

6.1 Datasets

The main dataset used in this research study was the UTK Face dataset (Zhang, Song & Qi, 2017). This dataset contained over 20,000 images provided with ground truth labels for age, gender, and ethnicity attributes. Other datasets used either for testing, augmentation or database scaling were FGNET (Fu et al., 2014), IMDB-WIKI (Rothe, Timofte & Gool, 2016) and VGG Face2 (Q. Cao, Shen, Xie, Parkhi & Zisserman, 2018).

6.2 Hardware Configuration

Throughout the research study, the hardware used was an Intel i7 – 6th generation processor clocked at 2.6 GHz with 8 cores and NVIDIA 960M GPU with 4GB memory and 16 GB RAM. The code was written using Python 3.5.4 and PyCharm as the IDE with the required libraries and dependencies installed using python's package manager.

Multi-processing with all 8 cores was utilized only for speeding-up the feature extraction process. Performance comparison and test results were obtained using only a single processor core for a fair comparison.

6.3 Performance Metrics

6.3.1 Mean Average Precision

The goal of the proposed indexing strategy for face-image retrieval is to quickly and accurately retrieve matching images for a given query image. For systems that return a ranked sequence of images, it is desirable to consider the order in which the returned images are presented. Mean Average Precision (mAP) is a widely used metric that assesses the quality of retrieval results and thus is used as the primary performance measure for evaluating the performance of the face-image retrieval system using the proposed indexing strategy.

Average Precision (AP) is one of the most frequent methods used to evaluate the retrieval quality of a single query. AP takes into consideration both Precision and Recall scores.

Precision (P) is the fraction of retrieved images that are relevant, whereas Recall (R) is the fraction of relevant images that are retrieved.

$$P = \frac{TruePositive}{TruePositive + FalsePositive} \tag{6.1}$$

$$R = \frac{TruePositive}{TruePositive + FalseNegative} \tag{6.2}$$

Average Precision (AP) averages the precision values from the rank positions where relevant images are retrieved. AP is obtained by computing precision and recall at every position in the ranked sequence of images. The following equation 6.3 shows the computation of Average Precision for a single query image.

$$AP = \frac{\sum_{k=1}^{n} P(k) * rel(k)}{R} \tag{6.3}$$

where,

- $P(k)$ is the Precision at cut-off at k of top-k retrieved results
- $rel(k)$ is a binary indicator function equaling 1 if the k^{th} retrieved results are relevant to the current query image and 0 otherwise;
- R denote the number of relevant results for the current query image
- n is the total number of retrieved results

The mean average precision (mAP) is simply the mean of Average Precision (AP) scores over all queries. The following equation 6.4 is used to compute the mAP for all queries.

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{6.4}$$

Where, Q is the total number of queries and AP is the average precision for current query q.

6.3.2 Precision@K

The precision at particular rank-K accuracy is another important metric to evaluate face-image retrieval systems. P@K score refers to the average number of retrieved images within the top-K set of retrieved images. Precision at rank-K is used as a part of mAP computation. However, when used on its own it provides additional knowledge of the model performance.

$$\mathbf{P@k} = \frac{\mathbf{number\ of\ retrieved\ images@k\ that\ are\ \ relevant}}{\mathbf{number\ of\ images\ retrieved@k}} \tag{6.5}$$

6.3.3 Retrieval Time

The whole purpose of the proposed indexing strategy using hierarchical classification was to drastically reduce the time required to retrieve matching images from a large scale database.

Retrieval time was computed as the time taken to retrieve matching images for a given query image from the database and was used for comparing with the retrieval time of other indexing methods.

Other Metrics

Other additional metrics involved for measuring the performance of the face-image retrieval system using the proposed indexing method included the usual accuracy, precision, recall, and F-measure scores.

6.4 Preliminary Tests

Preliminary tests were conducted for selecting an appropriate pre-trained CNN model and a transfer learning strategy to use for the rest of the study.

6.4.1 Preliminary gender classification

To select the best performing pre-trained CNN model, quick preliminary tests were conducted for gender classification using the UTK Face Dataset (Zhang et al., 2017). The dataset was split into 70% for training and 30% for testing. Gender classification is the easiest and simplest classification and thus was used for conducting preliminary tests.

Based on the preliminary results shown in table 6.1, the pre-trained CNN model

of Dlib library achieved the highest classification accuracy of 96.6% for gender classification using transfer learning as a feature extractor. Hence, it was selected as a feature-extractor using transfer learning and generated 128D feature vectors for each face-image.

Table 6.1: Preliminary results of gender classification on UTK Face Dataset

Pre-trained CNN model	**Transfer Learning Strategy**	**Original Train Dataset**	**Output Feature Dimensions**	**Accuracy**
FaceNet-Inception ResNet[1]	Feature Extraction	CASIA-WebFace (Yi et al., 2014)	512D	83.1%
FaceNet-Inception ResNet[1]	Feature Extraction	VGGFace2 (Q. Cao et al., 2018)	512D	85.0%
Tensorflow-MobileNet v2[2]	Fine-Tuning	ImageNet (Russakovsky et al., 2015)	N/A	92.3%
Tensorflow-Inception v4[2]	Fine-Tuning	ImageNet (Russakovsky et al., 2015)	N/A	93.5%
Dlib-ResNet[3]	**Feature-Extraction**	**Private Face Dataset**	**128D**	**96.6%**

[1] https://github.com/davidsandberg/facenet

[2] https://www.tensorflow.org/lite/guide/hosted_models

[3] https://github.com/davisking/dlib

6.4.2 Visualizing 128D Feature Vectors

The selected pre-trained CNN model from the Dlib library uses transfer learning to generate a 128D feature vector for a given input image. These 128D feature vectors are used to train different classifiers such as the Support Vector Machine (SVM) and the Multi-layer Perceptron (MLP).

The generated 128D feature vector was projected onto 2D vectors using dimensionality reduction techniques such as PCA (Jolliffe, 2011) and t-SNE (Maaten & Hinton, 2008). The projection of the generated 128D feature vectors into a 2D vector space provided a quick visualization of the data. It also gave a preliminary indication of the potential of developing good classification models using the generated 128D feature vectors.

Figure 6.1 and Figure 6.2 provides a 2D representation of the 128D feature vectors using t-SNE method on all images of UTK Face Dataset used for gender and ethnicity classification tasks respectively. A good amount of separability between respective gender and ethnicity classes is clearly visible from both the figures, which gave an early indication that these 128D features can be used to train a robust classification model for gender and ethnicity classification tasks.

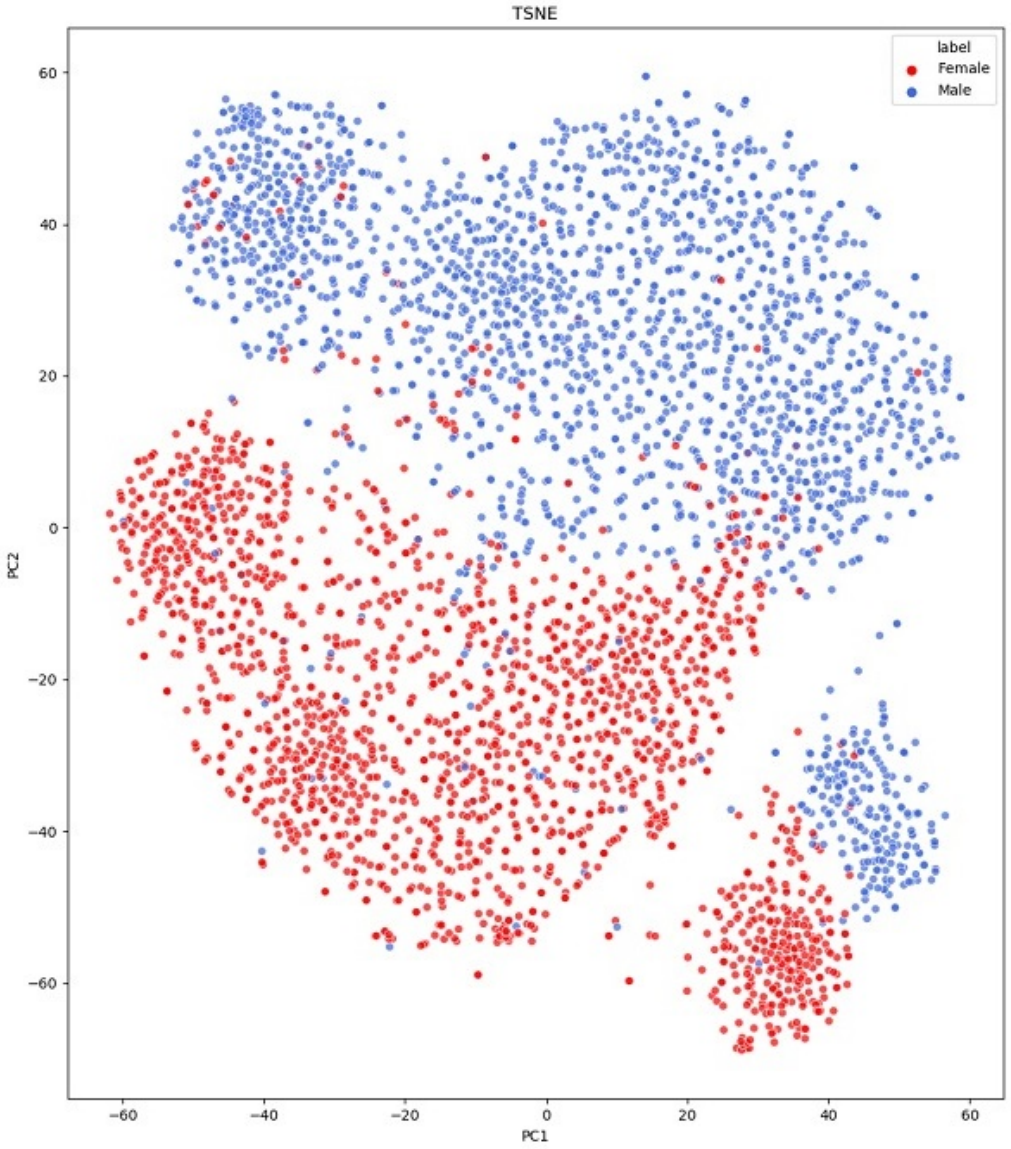


Figure 6.1: t-SNE visualization of 128D feature vectors for gender classification

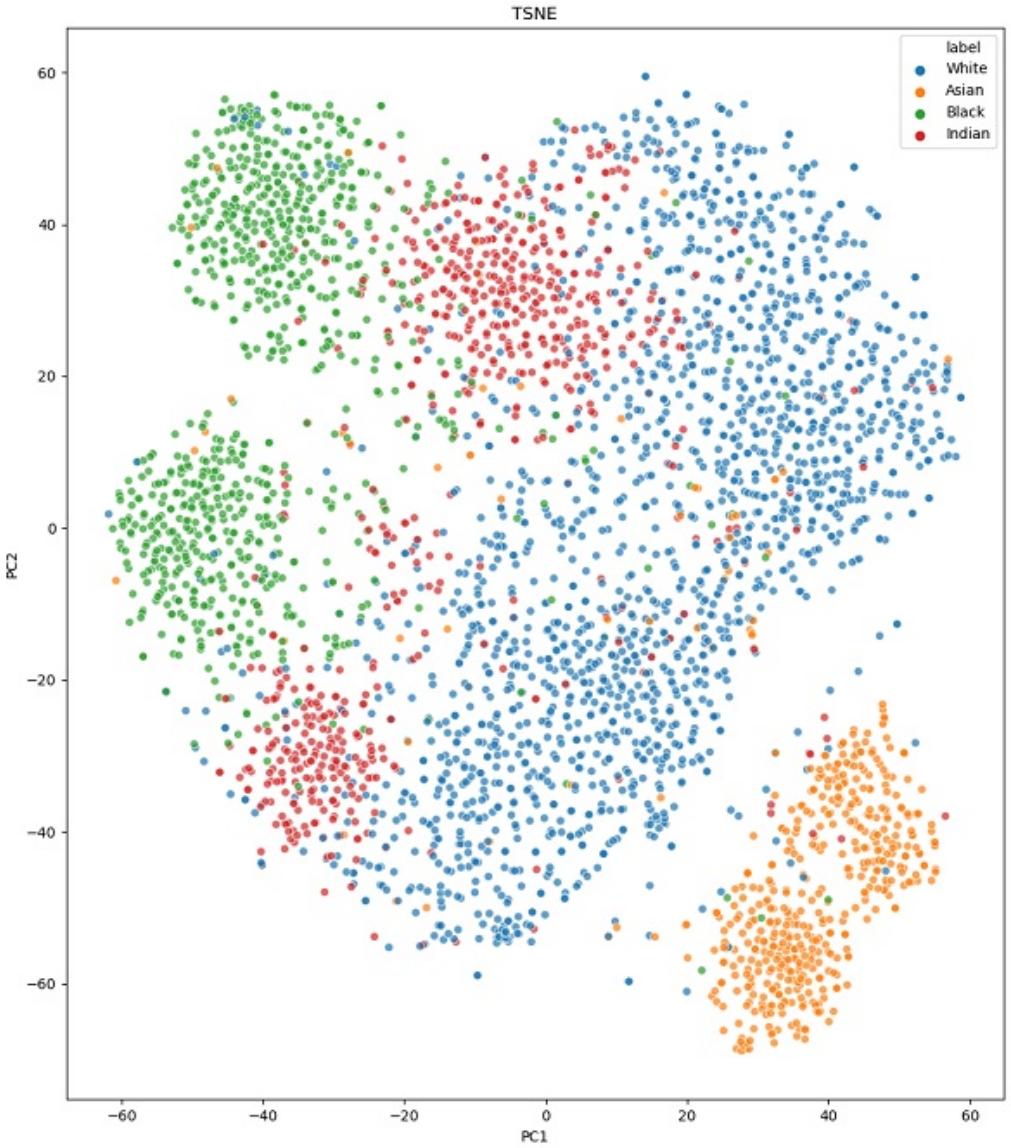


Figure 6.2: t-SNE visualization of 128D feature vectors for ethnicity classification

6.5 Evaluation of Classification Models

6.5.1 Classification Results

A total of seven different classification models presented in table 4.2 were required to be trained due to the proposed hierarchical design as depicted from Figure 4.3.

Each of these models were trained using only UTK Face Dataset (Zhang et al., 2017). The dataset was split into two parts with an 80-20 ratio. The 80% was used to train the classifier models with a 10-fold cross-validation strategy. The remaining 20% of the dataset was used for testing the model performance. Other datasets such as WIKI (subset of IMDB-WIKI) (Rothe et al., 2016) and FGNET (Fu et al., 2014) were also used as the test datasets to assess the generalizability of the classification models on entirely different and unseen data samples.

Age-Group 10 Classification Results

Table 6.2 presents the results obtained for age-group 10 classification of above or below 10 years on UTK Face Dataset. Results show that Linear SVM, RBF SVM, and MLP were the best performing classifiers with 98% 10-fold cross-validation accuracy and around 97.3% test set accuracy.

Table 6.2: Age-group 10 classification on UTK Face Dataset

Classification Model	**Training Accuracy (10-fold CV)**	**Test Accuracy**	**Time taken to train (secs)**
Linear SVM	**98.00%**	**97.41%**	**1.43**
RBF SVM	98.00%	97.36%	177.31
Decision Tree	94.3%	93.8%	9.7
MLP	98.00%	97.3%	9.45
K-NN	97.22%	96.54%	137.97
Random Forest	96.45%	96.4%	24.17
Gradient Boost	97.89%	97.2%	93.41

For testing the generalizability of transfer learning on age-group classification, the best age-group 10 classifier models were tested on other unseen datasets. A similar methodology was followed for all remaining classification models.

Table 6.3 shows that the MLP was the best performing classifier with consistent accuracy across other unseen datasets of WIKI and FGNET, whereas the other two SVM variants saw a performance dip as compared to MLP.

Table 6.3: Age-group 10 classification on other datasets

Classification Model	**UTK Train Accuracy**	**UTK Test Accuracy**	**WIKI Test Accuracy**	**FGNET Test Accuracy**
Linear SVM	98.00%	97.41%	95.23%	92.55%
RBF SVM	98.00%	97.36%	96.31%	93.00%
MLP	**98.00%**	**97.3%**	**96.45%**	**95.21%**

Gender Classification Results

Table 6.4 presents the results obtained for gender classification on the UTK Face Dataset. Only images of persons above 10 years were used for training this model.

Table 6.4: Gender classification on UTK Face Dataset

Classification Model	**Training Accuracy (10-fold CV)**	**Test Accuracy**	**Time taken (secs)**
Linear SVM	97.00%	96.27%	1.47
RBF SVM	96.4%	95.86%	118.75
Decision Tree	90.00%	88.54%	9.63
MLP	**97.00%**	**96.33%**	**4.02**
K-NN	96.5%	95.4%	47.28
Random Forest	95.8%	95.3%	47.28
Gradient Boost	96.00%	95.6%	59.8

Below table 6.5 shows that the Linear SVM classifier might have overfitted during training as it performs worse on other datasets, whereas MLP surprisingly shows an increase in test accuracy from the base score of 96.3% to 97.5% on the WIKI dataset. The accuracy slightly drops when tested on the FGNET dataset. This was expected as the FGNET dataset contains slightly different types of images as compared to UTK and WIKI datasets. The results show that MLP performed the best when tested with unseen data from the same UTK dataset and other datasets of WIKI and FGNET.

Table 6.5: Gender classification on other datasets

Classification Model	UTK Train Accuracy	UTK Test Accuracy	WIKI Test Accuracy	FGNET Test Accuracy
Linear SVM	97.00%	96.27%	92.56%	91.85%
MLP	**97.00%**	**96.33%**	**94.00%**	**95.5%**

Age-Group 19 and 35 Classification Results

A similar methodology was implemented for testing all remaining classification models and only the best results are provided from hereon for each classification model.

Table 6.6 and Table 6.7 provide the results obtained on all test datasets for age-group 19 classification of above or below 19 years for male and female sub-hierarchies respectively. The tables 6.8, 6.9 provide results for age-group 35 classification for male and female sub-hierarchies respectively.

Table 6.6: Age-group 19 males classification on all datasets

Classification Model	UTK Train Accuracy	UTK Test Accuracy	WIKI Test Accuracy	FGNET Test Accuracy
Linear SVM	97.03%	96.97%	95.43%	93.66%
MLP	**96.66%**	**97.10%**	**96.55%**	**96.38%**

Table 6.7: Age-group 19 females classification on all datasets

Classification Model	UTK Train Accuracy	UTK Test Accuracy	WIKI Test Accuracy	FGNET Test Accuracy
Linear SVM	95.33%	95.21%	95.1%	93.26%
MLP	**95.19%**	**95.25%**	**95.55%**	**93.49%**

Table 6.8: Age-group 35 males classification on all datasets

Classification Model	UTK Train Accuracy	UTK Test Accuracy	WIKI Test Accuracy	FGNET Test Accuracy
Linear SVM	91.73%	90.18%	89.88%	85.26%
RBF SVM	86.61%	85.31%	85.10%	85.26%
MLP	**92.6%**	**91.52%**	**90.55%**	**87.29%**

Table 6.9: Age-group 35 females classification on all datasets

Classification Model	UTK Train Accuracy	UTK Test Accuracy	WIKI Test Accuracy	FGNET Test Accuracy
Linear SVM	89.43%	89.83%	90.46%	86.31%
RBF SVM	89.00%	89.51%	88.70%	85.38%
MLP	**91.2%**	**91.05%**	**90.66%**	**86.34%**

Ethnicity Classification Results

Table 6.10 shows the results obtained for ethnicity classification of Asian, Black, Indian or White on UTK Face Dataset. This classifier model was common to both male and female sub-hierarchies and was trained using images of both males and females above 10 years of age. The ethnicity classifier could not be tested against other datasets of WIKI and FGNET due to the lack of ground-truth ethnicity labels.

Table 6.10: Ethnicity classification of the UTK Face Dataset

Classification Model	Training Accuracy (10-fold CV)	Test Accuracy	Time taken (secs)
Linear SVM	90.16%	89.52%	2.30
RBF SVM	90.07%	88.32%	135.77
Decision Tree	78.03%	78.67%	8.67
MLP	**90.5%**	**90.10%**	**10.83**
K-NN	89.61%	89.4%	35.96
Random Forest	88.87%	88.14%	15.41
Gradient Boost	90.48%	90.02%	171.90

6.5.2 Gender classification comparison

Table 6.11 presents a comparison of state-of-the-art results of gender classification on the UTK Face Dataset with the best gender classification MLP model built in this research study. The comparison clearly shows that the MLP model developed in this study outperforms state-of-the-art methods by a good margin of around 3%-4%.

Table 6.11: Comparison of proposed gender classifier vs state-of-the-art methods on the UTK Face Dataset, (Savchenko, 2019)

Method	Accuracy
Proposed Method's MLP	**97.00%**
MobileNet, fine-tuned	94.10%
MobileNet, pre-trained on VGGFace2	93.79%
MobileNet, fine-tuned from ImageNet	91.81%
DEX	91.05%
FaceNet	89.54%

The results from other methods such as with pre-trained MobileNets are in accordance with the results obtained from the preliminary tests performed in this study. These results were presented earlier in Table 6.1.

6.5.3 Summary

Table 6.12 summarizes the best performing models for all attribute classification on the UTK Face Dataset. From Table 6.12 it can be concluded that Multilayer Perceptron (MLP) was the best performing classifier on all attributes.

Parameter optimization was performed using Random Search cross-validation and Grid Search cross-validation techniques to obtain optimal parameter values and thereby improve the results of each classifier. The parameter optimization led to an average increase of approximately 2% in accuracy for all the classification models.

Table 6.12: Top 7 classification models trained on the UTK Face Dataset

Classification	**Classification Model**	**Training Accuracy (10-fold CV)**	**Test Accuracy**
Age-Group 10	MLP	98.00%	97.3%
Gender	MLP	97.00%	96.33%
Age-Group 19 (M)	MLP	96.66%	97.10%
Age-Group 19 (F)	MLP	95.19%	95.25%
Age-Group 35 (M)	MLP	92.60%	91.52%
Age-Group 35 (F)	MLP	91.20%	91.05%
Ethnicity	MLP	90.50%	90.10%

6.6 Experimental Configuration

This section explains the different query sets and the protocols used for the experimental evaluation of the proposed methodology.

6.6.1 Query Sets

Two different types of query sets were constructed for evaluation, namely: "easy query set" and "hard query set"; hereinafter referred to simply as easy and hard query sets respectively.

Easy Query Set

The easy query set consisted of images from the same database of the UTK Face dataset (Zhang et al., 2017) which were used for training classification models. However, the images in the query set were not a part of the training process but still belonged to the UTK Face dataset.

The easy set was constructed using a stratified random sampling technique which equally selected images belonging to each of the 19 different subsets of the database as depicted in the hierarchical design in Figure 4.3. This removed any bias towards one of the 19 subsets. Easy query set consisted of 180 images.

The following figure 6.3 shows some of the example images from the easy query set

Figure 6.3: Example images in easy query set

Hard Query Set

The hard query set consisted of images scraped from the web and social media webpages using simple selenium[1] based web-scraper which was scripted in Python. The images consisted of real-world examples with extreme illumination and lighting conditions or blurry images. Evaluation on the hard query set measured the performance of the proposed methodology on real-world like conditions. The hard query set consisted of a total of 579 images. Figure 6.4 shows examples of hard query set images.

[1] https://selenium-python.readthedocs.io/

Figure 6.4: Example images in hard query set

6.6.2 Protocols

A total of two evaluation protocols were used for evaluating the performance of the face-image retrieval system using the proposed indexing strategy. These protocols are Closed-set and Open-set evaluation protocols.

Closed-Set Evaluation Protocol

Closed-Set evaluation protocol meant it is known beforehand whether the matching images for the query set are present or absent in the database. This allows for evaluation of usual performance metrics of accuracy, precision, recall, and F-measures as we can plot a confusion matrix using prior knowledge of actual positive and negative matches.

Open-Set Evaluation Protocol

Open-set evaluation protocol reflected real-world scenarios where knowledge of matching images being present or absent in the database is not known beforehand. Thus, usual metrics deduced from a confusion matrix cannot be computed as total positive and negative matches are unknown. Thus, mean average precision (mAP) and precision at K-rank (P@k) are used to measure and evaluate the performance.

An open-set protocol reflects real-world scenario and hence, only the hard query set was used during the open-set evaluation protocol. For the closed-set protocol, both the hard and easy query sets were utilized.

6.6.3 Match Verification

While deciding on whether or not a match is obtained, it was noted that there could be a bias introduced depending upon the ethnicity of the tester. Thus, to remove this bias, 4 volunteers each belonging to one of the ethnic groups of Asian, Black, Indian or White were assigned as testers to verify for matches and assign ranks for the retrieved list of images using both query sets.

Match or no-match status was decided on the majority of votes from each volunteer. In case of a tie in votes, an additional tester was assigned to decide whether it is a match or a no-match for the given query image. This ensured that there was no self-test bias introduced while measuring the performance.

6.6.4 Experimental Setup

Due to the time-consuming match verification process, the two query sets had a relatively low number of images, whereby the easy set had only 180 total images whereas the hard set had 579 total images. More preference was given to the hard set as it depicted performance closer to real-world like conditions.

The dynamic threshold option was set throughout the closed-set and open-set evaluation protocols to achieve optimal performance. The threshold values are the same as presented in Table 5.2 in the previous Chapter 5. All processing and computations were handled using a single processor core clocked at 2.6 GHz for a fair comparison between different methods.

6.7 Experimental Evaluation of Proposed Methodology

6.7.1 Proposed Method vs Brute Force

The following tables 6.13 and 6.14 provide a comparison between the average retrieval times of proposed methodology and brute force on databases with sizes of 20k and 350k images respectively using euclidean and cosine distance measures. From both the tables, it is evident that the proposed methodology is several orders of magnitude faster than the brute force method.

Table 6.13: Proposed Method vs Brute Force average retrieval times on 20k Database

Method	**Distance Measure**	**Retrieval Time (secs)**
Brute Force	Euclidean	2.211
Proposed Method	Euclidean	0.1
Brute Force	Cosine Similarity	9.3
Proposed Method	Cosine Similarity	0.22

Table 6.14: Proposed Method vs Brute Force average retrieval time comparison on a 350K Database

Method	Distance Measure	Retrieval Time Per Query
Brute Force	Euclidean	1 minute
Proposed Method	Euclidean	0.1 seconds
Brute Force	Cosine Similarity	10 minutes +
Proposed Method	Cosine Similarity	2.2 seconds

6.7.2 Closed-Set Evaluation on Easy Query Set

This section presents results obtained on the easy query set under the closed-set evaluation protocol. Using prior knowledge a confusion matrix was computed from which accuracy, precision, recall, true negative rate (TNR), F1-score (F1), false positive rate (FPR) and false negative rate (FNR) were deduced. All results are obtained using both euclidean and cosine distance measures and are presented in the tables below.

Due to the added time-constraints for the manual testing process done by the volunteers, the database size was fixed at 20k images to keep retrieval times as minimum as possible.

Euclidean Distance

Table 6.15: Confusion matrix with euclidean distance on easy query set

(a) Without back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 122	FN = 14
	Negative	FP = 9	TN = 35

(b) With back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 132	FN = 3
	Negative	FP = 9	TN = 36

Cosine Similarity

Table 6.16: Confusion matrix with cosine similarity on easy query set

(a) Without back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 125	FN = 14
	Negative	FP = 2	TN = 39

(b) With back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 136	FN = 3
	Negative	FP = 2	TN = 39

Table 6.17: Closed-set evaluation results on easy query set

Method	**Accuracy**	**Precision**	**Recall**	**TNR**	**F1**	**FPR**	**FNR**
PM+EU [1]	87.22 %	93.13 %	89.71 %	79.55 %	91.39 %	20.46 %	10.3 %
PM+BT+EU [2]	93 %	93.62 %	97.78 %	80 %	95.65 %	20 %	2.2 %
PM+CS [3]	91.1 %	98.43%	89.93 %	95.1 %	93.98 %	4.8 %	10 %
PM+BT+CS [4]	97.22 %	98.55 %	97.84 %	95.1 %	98.2 %	4.8 %	2.1 %

[1] Proposed method without back-tracking + euclidean distance

[2] Proposed method with back-tracking + euclidean distance

[3] Proposed method without back-tracking + cosine similarity

[4] Proposed method with back-tracking + cosine similarity

From the above table 6.17, it can be noted that the proposed methodology with the back-tracking option and cosine similarity as the distance measure performed the best with an accuracy of around 97% with similar precision, recall and F1 scores. It can also be noted that there was an average increase in accuracy of about 4% with cosine similarity than with Euclidean distance as the similarity measure. It may be due to the fact that cosine similarity takes into consideration the angle between two face-images which may be an important factor resulting in improved performance with cosine similarity.

6.7.3 Closed-Set Evaluation on Hard Query Set

Similar to the previous section, this section presents results obtained on the hard query set under the closed-set evaluation protocol. Cosine similarity achieves better results as compared with Euclidean distance as noted from the previous section and thus only results with cosine similarity as the distance measure are provided here.

Cosine Similarity

Table 6.18: Confusion matrix with cosine similarity on hard query set

(a) Without back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 327	FN = 110
	Negative	FP = 30	TN = 112

(b) With back-tracking

		Predicted	
		Positive	Negative
Actual	Positive	TP = 400	FN = 44
	Negative	FP = 31	TN = 104

Table 6.19: Closed-set evaluation results on hard query set

Method	**Accuracy**	**Precision**	**Recall**	**TNR**	**F1**	**FPR**	**FNR**
PM+CS [1]	76.08 %	92.11 %	74.83 %	80 %	82.58 %	20 %	25.17 %
PM+BT+CS [2]	87 %	92.81 %	90.01 %	77.04 %	91.43 %	22.9 %	9.9 %

[1] Proposed method without back-tracking + cosine similarity

[2] Proposed method with back-tracking + cosine similarity

Table 6.20 provides a comparison of results under the closed-set protocol between easy and hard query sets. The performance on an easy set was as expected with almost all performance metrics measuring around 97%. However, this does not reflect the real-world performance of the system as all images were from the same database. Thus,

results obtained on the hard query set provide a closer view as to how the system will perform in real-world scenarios.

From the table 6.20, it is evident that the accuracy significantly dropped from 97% on the easy set to 87% on the hard set, which is still impressive considering the conditions of images in the hard set. The face-image retrieval system using the proposed indexing strategy managed to maintain high precision and recall scores of around 92% even on the hard query set.

Table 6.20: Comparison of Closed-set evaluation results between easy and hard query sets

Method	**Accuracy**	**Precision**	**Recall**	**TNR**	**F1**	**FPR**	**FNR**
PM+BT+ES [1]	97.22 %	98.55 %	97.84 %	95.1 %	98.2 %	4.8 %	2.1 %
PM+BT+HS [2]	87 %	92.81 %	90.01 %	77.04 %	91.43 %	22.9 %	9.9 %

[1] Proposed method with back-tracking + cosine similarity on the easy query set

[2] Proposed method with back-tracking + cosine similarity on the hard query set

6.7.4 Open-Set Evaluation on Hard Query Set

From the previous section, it was clear that the system performed very well under the closed-set evaluation protocol on both query sets. Even though the hard query set provides a close-to-real-world performance measure, it still does not entirely reflect real-world scenarios. In most real-world use-cases, the prior knowledge of matches present or absent in the databases is not known as generally the databases contain billions of images. Thus, an open-set evaluation protocol provides a better and close-to-real-world performance measure.

Rank-10

A large database of more than 350K images was formed by augmenting images from different datasets such as the VGG Face dataset (Q. Cao et al., 2018) and the IMDB-WIKI dataset (Rothe et al., 2016).

A list of top-10 matching images was retrieved for a given query image. Thus, precision at rank-10 (P@10) and the mean average precision at rank-10 (mAP@10) could be computed.

An example output of the retrieved top-10 matching image list is shown in Figure 6.5. This list is ordered and ranked on a scale of 1 to 10 with 1 being the most likely match while 10 represents the least likely match. A coloured overlay also adds another interpretable layer where the green coloured numbers show the images which are to be the most likely matches, whereas the orange numbers indicate it may be a match while the red numbers mean that it is least likely to be a match.

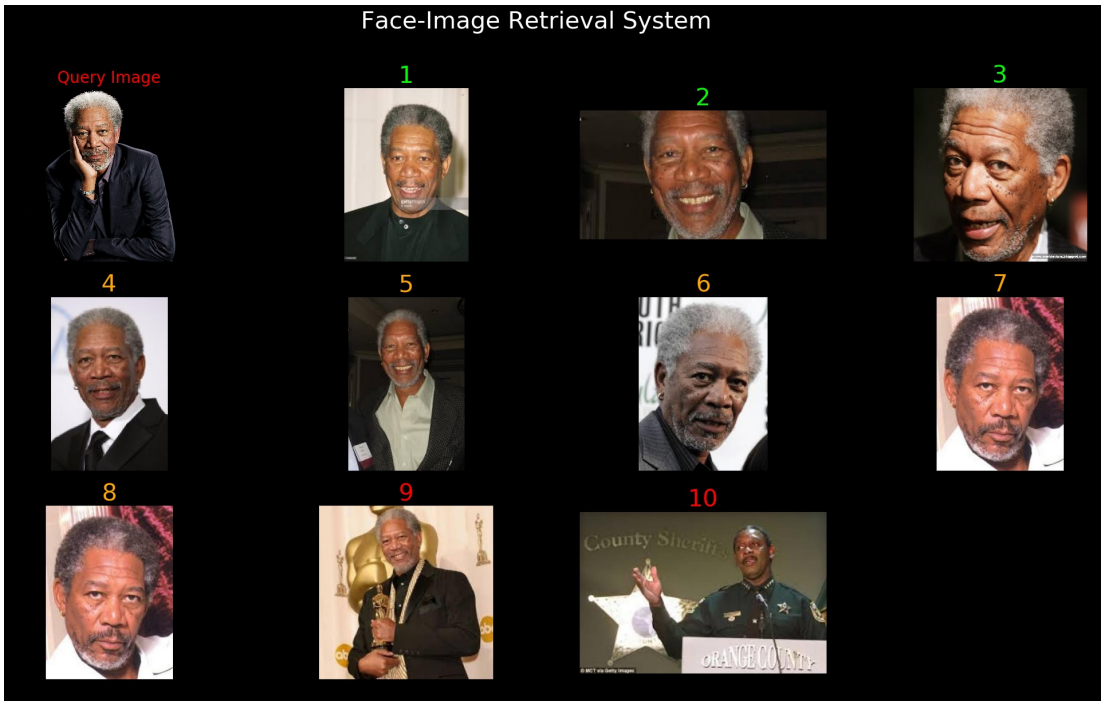


Figure 6.5: Sample output of face-image retrieval system

The open-set protocol was evaluated using mean average precision (mAP) and precision at rank-10 (P@10) where a list of top-10 retrieved images was ranked manually by the volunteers to compute the mAP and precision values at rank-10.

Table 6.21 presents the results obtained with the proposed methodology under open-set evaluation protocol with a hard query set on a 350K database with Euclidean and Cosine similarity as the distance measures. It is evident that the proposed methodology with cosine similarity produces the best results; however it is also 3 times slower than the Euclidean distance.

Table 6.21: Open-set evaluation protocol with hard query set on a 350K Database

Method	**mAP@10**	**Precision@10**	**Retrieval Time (seconds)**
PM+BT+EU [1]	80.15 %	79.72 %	1.1
PM+BT+CS [2]	82.61 %	80.11 %	3.43

[1] Proposed method with back-tracking and the Euclidean distance measure

[2] Proposed method with back-tracking and Cosine Similarity as the distance measure

6.7.5 Open-Set Performance Comparison on Hard Query Set

The proposed methodology is compared against two methods, namely: Product Quantization and ANNOY. Product Quantization is a state-of-the-art indexing method as reviewed earlier in Chapter 2 Literature review. ANNOY is a state-of-the-art top performing approximate nearest neighbour algorithm as per the ANN benchmarks published in the paper (Aumüller et al., 2017).

A hard set was queried against the 350k image database under the open-set protocol. Euclidean distance was used as the distance metric to provide a fairer comparison as other methods relied on Euclidean distance as opposed to cosine similarity.

The proposed indexing strategy acts as a filtering method which reduces a large

database to a smaller subset of images without significant time consumption. Thus, the proposed indexing strategy can be combined with state-of-the-art approximate nearest neighbour algorithms or other retrieval algorithms to boost the performance significantly. The proposed indexing method was combined with Product Quantization as well as with ANNOY and evaluated under the open-set protocol.

Optimal parameters were identified and set empirically for ANNOY and Product Quantization for best performance comparison.

For **Annoy:** Length of vector = 128, number of trees = 20, retrieved_results = 10, search_k = database_size / 2.

For **Product Quantization:** subspace_dimension = 32, subspace_size = 256, train_size = 20,000.

A comparison of mAP@10 and P@10 between different methods under an open-set evaluation is presented in Table 6.22.

Table 6.22: Performance comparison of open-set evaluation protocol with the hard query set on a 350K Database

Method	mAP@10	Precision@10	Retrieval Time (seconds)
PM+BT+DT [1]	80.15 %	79.72 %	1.1
ANNOY [2]	78.61 %	41 %	0.053
PQ [3]	81.47 %	79.63 %	0.83
PM+BT+DT+PQ [4]	**85.23 %**	**82.2 %**	**0.02**
PM+BT+DT+ANNOY [5]	**84.6 %**	**81 %**	**0.005**

[1] Proposed method with back-tracking + dynamic threshold

[2] ANNOY, `https://github.com/spotify/annoy`

[3] Product Quantization, `https://github.com/matsui528/nanopq`

[4] Proposed method with back-tracking + dynamic threshold + Product Quantization

[5] Proposed method with back-tracking + dynamic threshold + ANNOY

Several interesting trends are evident from Table 6.22. Firstly, we note that the proposed methodology of PM+BT+DT provided superior mAP@10 and Precision@10 scores than ANNOY. The product quantization method (PQ) had a similar level of performance to the proposed methodology with respect to the precision scores. Although PQ outperformed ANNOY in terms of precision, its retrieval time is far greater than that of the latter, and thus it is of interest to investigate whether a combination of methods would improve search time. As the proposed methodology operates on a hierarchical classification scheme which is different in nature to ANNOY and PQ it would be of interest to see whether the combination of methods would yield better precision and search performance. Table 6.22 shows that this is the case as the combination of the proposed method with either PQ or ANNOY significantly improved not just the search performance as expected but also the precision.

6.7.6 Comparison of Retrieval Time

In view of the fact that hybrid methods outperformed all others with respect to search performance, it would be of interest to consider scaling up of the methods with respect to the size of the database that is being searched. Considering a linear relationship of increase in retrieval time with increase in database size, a sample retrieval time comparison is depicted in the following Figures 6.6 and 6.7:

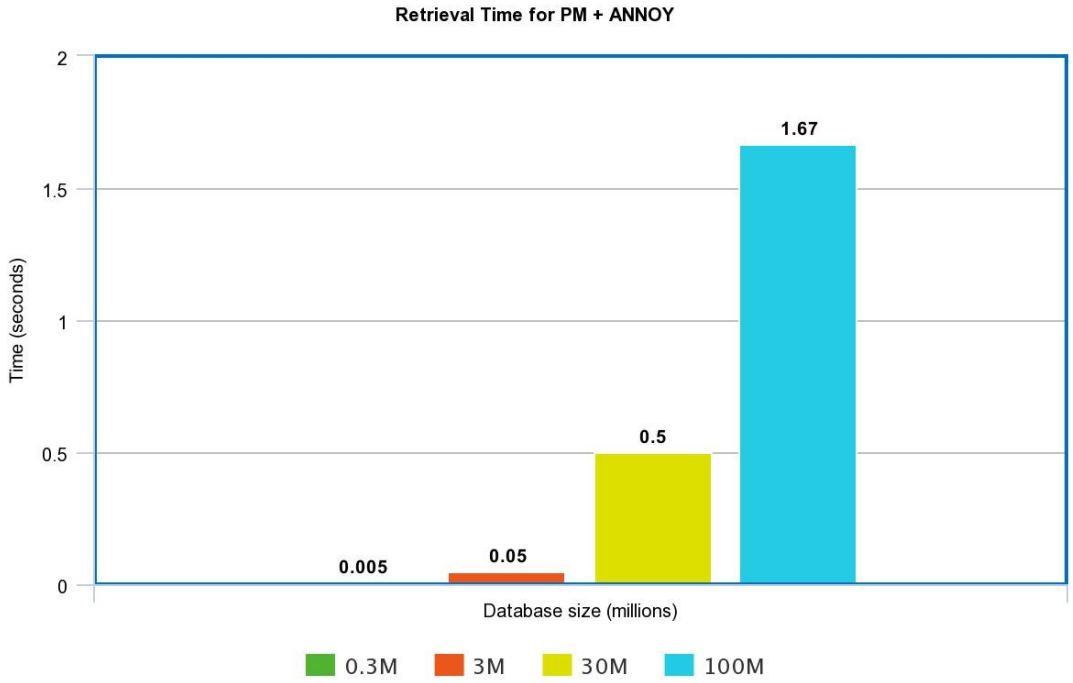


Figure 6.6: Retrieval times for Proposed Methodology with backtracking, dynamic thresholding + ANNOY on different database sizes

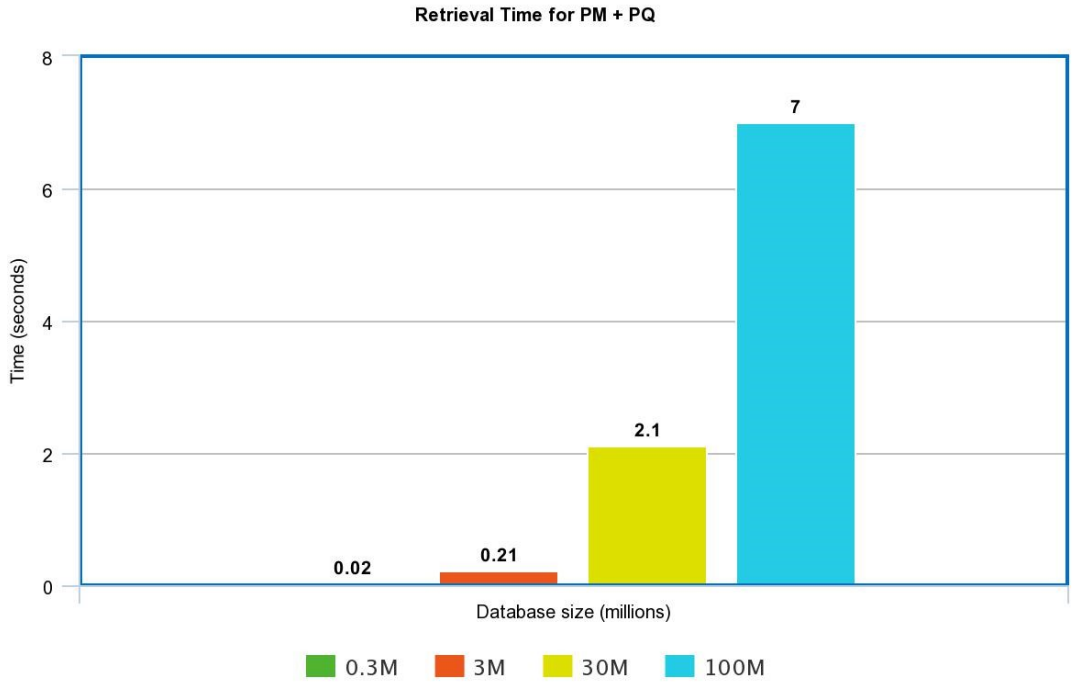


Figure 6.7: Retrieval times for Proposed Methodology with backtracking, dynamic thresholding + Product Quantization on different database sizes

The combination of the proposed methodology and ANNOY achieves a retrieval time of only 1.67 seconds for a 100 million database size under the assumption of a linear time-database size relationship. Under the same assumption however, the combination of the proposed methodology and Product Quantization achieves a drastic increase in retrieval time of up to 7 seconds.

Below table 6.23 presents the retrieval time comparison of the proposed method with various indexing methods used for face-image retrieval with at least one million database size.

(Wu et al., 2011) obtained 0.5 seconds retrieval time by using an inverted indexing method on a 1 million database while (B.-C. Chen et al., 2013) managed to improve these results by reducing the retrieval time to about 0.2 seconds on a 1 million database size. The combination of the proposed method and ANNOY achieves a retrieval time of only 1.67 seconds on a 100 million database.

The retrieval time achieved by (D. Wang et al., 2015) using Product Quantization for an 80 million database size was 6.7 seconds. These are comparable to the results obtained by the combination of the proposed method and product quantization of around 7 seconds. However, as stated earlier, the combination of the proposed method with ANNOY achieves a significantly low retrieval time of only 1.67 seconds per query image from a database of 100 million images, thus making it faster on a larger database (100 million as opposed to 80 million).

Table 6.23: Retrieval time comparison

Reference	**Indexing Method**	**Database size**	**Retrieval Time**	**CPU**
(Wu et al., 2011)	Inverted Index	1 million	0.5 sec	2.6GHz, 16 GB RAM
(B.-C. Chen et al., 2013)	Inverted Index	1 million	0.2 sec	Intel Xeon 2.4GHz
(D. Wang et al., 2015)	Product Quantization	5 million	0.9 sec	Intel Xeon 3.10 GHz
(D. Wang et al., 2015)	PQ	80 million	6.7 sec	Intel Xeon 3.10 GHz
Proposed Method	Hierarchical + PQ	30 million	2.1 sec	Intel 2.6GHz, 16 GB RAM
Proposed Method	Hierarchical + PQ	100 million	7 sec	Intel 2.6GHz, 16 GB RAM
Proposed Method	Hierarchical + ANNOY	30 million	0.5 sec	Intel 2.6GHz, 16 GB RAM
Proposed Method	Hierarchical + ANNOY	100 million	1.67 sec	Intel 2.6GHz, 16 GB RAM

6.8 Statistical Tests

Statistical tests such as the paired t-test are used to compare the means between two related groups of samples. The purpose of the paired t-test is to determine whether there is statistical evidence that the mean difference between the paired observations on a particular outcome is significantly different from zero.

Based on the experimental results presented in the earlier sections, the proposed methodology achieved superior performance over other methods. Thus, statistical tests

can be used for identifying the factors that were responsible for contributing to the superior performance of the proposed method.

Thus, the proposed probabilistic back-tracking algorithm and the dynamic thresholding technique were statistically analyzed for identifying the effects each of these techniques had in enhancing the performance of the proposed methodology.

6.8.1 Paired T-test for Proposed Probabilistic Back-tracking Algorithm

A total of 16 groups were formed using stratified random sampling such that all 16 groups included an equal number of images (50 per group) belonging to each of the 19 database subsets as per the hierarchical design described in Figure 4.3.

The evaluation was done under a closed-set protocol to calculate the accuracy of the system with the application of the proposed back-tracking algorithm and without the back-tracking algorithm. The accuracy scores were noted for each of the 16 groups with and without the back-tracking algorithm.

The following table 6.24 provides the obtained accuracy results for each of the 16 groups.

Table 6.24: Paired T-test evaluation for back-tracking

Groups	Accuracy without back-tracking	Accuracy with back-tracking	Difference in accuracy
1	88.89	91.67	+ 2.78
2	72.22	83.33	+ 11.1
3	83.33	88.89	+ 5.56
4	72.22	77.78	+ 5.56
5	75	77.78	+ 2.78
6	80.56	86.11	+ 5.55
7	77.78	86.11	+ 8.33
8	75	86.11	+ 11.11
9	80.56	91.67	+ 11.11
10	80.56	88.89	+ 8.35
11	72.22	88.89	+ 16.67
12	66.67	86.11	+ 19.44
13	83.33	83.33	0
14	86.11	97.22	+ 11.11
15	80.56	88.89	+ 8.33
16	77.78	91.67	+ 13.89

Hypothesis

The paired t-test was conducted with a one-tailed hypothesis test to measure the effectiveness of the probabilistic back-tracking algorithm.

The following hypotheses were framed for the one-tailed test:

Null Hypothesis (H_0) = Accuracy with back-tracking is the same or lower than accuracy without back-tracking.

Alternate Hypothesis (H_a) = Accuracy with back-tracking is significantly higher than accuracy without back-tracking.

Calculating the sample mean and sample standard deviation of the accuracy differences gives:

$mean(\hat{d})$ = 8.854 and s_d = 5.19, standard error (SE) = 1.3, t-statistic (t) = 6.81.

Looking this up in the tables with 0.05 cut-off value (for 99% confidence) gives a p-value of p = 0.000002 << 0.1. The p-value provides a probability of the null hypothesis (H_0) being true, and hence an extremely low p-value indicates that the null hypothesis is unlikely to be true, giving strong statistical evidence to reject the null hypothesis (H_0).

Thus, there is a strong support for the alternate hypothesis (H_a) that the application of the proposed back-tracking algorithm results in significantly increased accuracy values.

Confidence Interval of 99%

It would be useful to calculate a confidence interval for the mean difference to tell us within what limits the true difference is likely to lie. A 99% confidence interval for the true mean difference is given by the below equation 6.6.

$$\hat{\mathbf{d}} \pm \mathbf{t}^* \frac{\mathbf{s_d}}{\sqrt{\mathbf{n}}} \tag{6.6}$$

Where, t^* is the 0.5% point of the t-distribution under n-1 degrees of freedom.

Substituting the mean $(\hat{d}$ = 8.854) and the standard deviation $(s_d$ = 5.19), values in equation 6.6 we get,

$$\mathbf{8.854} \pm \mathbf{2.576} * \frac{\mathbf{5.19}}{\sqrt{\mathbf{16}}} = (\mathbf{5.51}\ ,\ \mathbf{12.196}) \tag{6.7}$$

Equation 6.7 states that the true mean accuracy increase after the use of the proposed back-tracking algorithm is likely to be between 5.5% to 12.2% with a confidence level of 99%.

6.8.2 Paired T-test for Proposed Dynamic Thresholding

The same 16 groups as mentioned in the previous section were used under the closed-set evaluation protocol to compute the accuracy with and without the proposed dynamic thresholding scheme.

Table 6.25 provides the obtained accuracy results for each of the 16 groups.

Table 6.25: Paired T-test evaluation for dynamic thresholding

Groups	**Accuracy without dynamic threshold**	**Accuracy with dynamic threshold**	**Difference in accuracy**
1	88.89	91.67	+ 2.78
2	80.56	83.33	+ 2.78
3	83.33	88.89	+ 5.56
4	80.56	77.78	- 2.78
5	77.78	77.78	+ 0
6	86.11	86.11	+ 0
7	75	86.11	+ 11.11
8	75	86.11	+ 11.11
9	91.67	91.67	+ 0
10	91.67	88.89	- 2.78
11	77.78	88.89	+ 11.11
12	88.89	86.11	- 2.78
13	83.33	83.33	0
14	97.22	97.22	+ 0
15	86.11	88.89	+ 2.78
16	77.78	91.67	+ 13.89

Hypothesis

This paired t-test was also conducted with a one-tailed hypothesis test to assess the overall impact of the proposed dynamic thresholding technique.
The following hypotheses were framed for the one-tailed test:

Null Hypothesis (H_0) = Accuracy with dynamic thresholding is equal or lower than the accuracy without dynamic thresholding.

Alternate Hypothesis (H_a) = Accuracy with dynamic thresholding is higher than the accuracy without dynamic thresholding.

Similar to previous calculations for sample mean and standard deviation, we get: $mean(\hat{d})$ = 3.3 and s_d = 5.58, standard error (SE) = 1.394, t-statistic (t) = 2.367. This results in a low p-value of p = 0.01594 providing statistical evidence to reject the null hypothesis (H_0) at the 95% confidence level.

Confidence Interval

Similar to previous calculations, computing a confidence interval of 95% gives,

$$\mathbf{3.3} \pm \mathbf{1.96} * \frac{\mathbf{5.58}}{\sqrt{\mathbf{16}}} = (\mathbf{0.57}\ ,\ \mathbf{6.02}) \tag{6.8}$$

Computing a confidence interval of 99% gives,

$$\mathbf{3.3} \pm \mathbf{2.576} * \frac{\mathbf{5.58}}{\sqrt{\mathbf{16}}} = (\mathbf{-0.29}\ ,\ \mathbf{6.89}) \tag{6.9}$$

Equations 6.8 and 6.9 respectively state that the true mean accuracy increase after the use of proposed dynamic thresholding is likely to be between 0.57% to 6% with a confidence level of 95% and between - 0.29% to 6.89% with a confidence level of 99%.

6.9 Summary

The experimental results obtained highlight the benefits and performance of the proposed methodology. The use of a pre-trained CNN model as a feature extractor proved to be a successful strategy for the age-groups, gender, and ethnicity classification tasks having matched or even bettered state-of-the-art results.

The proposed methodology, when evaluated under a closed-set protocol, was able to achieve up to 97% and 87% accuracy scores while only having 0.04 and 0.2 as false positive rates for the easy and hard query sets respectively. The closed-set evaluation also showed that the proposed methodology obtained better results when used with Cosine similarity over Euclidean distance as the distance measure.

The open-set evaluation provided a realistic performance of the proposed methodology. The proposed indexing strategy, when combined with the state-of-the-art approximate nearest neighbour (ANN) algorithms, tremendously boosted the face-image retrieval performance. The combination of the proposed indexing strategy and ANNOY achieved a retrieval time of only 0.005 seconds per query image for a database of size 350,000 images. Assuming a linear relationship between retrieval time and database size, the retrieval time per query image in a database of 100 million images is expected to take just 1.67 seconds. This represents a significant reduction by a factor of 10 when compared with the retrieval time obtained using state-of-the-art method of ANNOY. We also observe that improvements are not restricted to just retrieval time but also include accuracy. The mean average precision at rank-10 (mAP@10) was also significantly increased from 78% to 85%, which is an increase of approximately 9%.

Finally, the statistical tests indicate the use of the probabilistic back-tracking algorithm increases the mean retrieval accuracy by 5% - 12% with a confidence level of 99% while the dynamic thresholding technique increases the mean retrieval accuracy by 0.5% - 6% with a confidence value of 95%.

Chapter 7

Conclusions and Future Work

The goal of this research study was to build an indexing method for efficient and accurate retrieval of face-images from large-scale databases. A novel indexing strategy based on hierarchical classification was proposed.

With the proposed indexing strategy, the database is filtered and reduced to a smaller subset. As only a smaller portion of the database is searched, the retrieval time was reduced significantly, as demonstrated in Chapter 6. At the same time, a high rate of accuracy was obtained due to the removal of any false matches. The proposed methodology based on a hybrid scheme of hierarchical indexing and ANNOY was able to achieve an average query image retrieval time of only 0.005 seconds for a database of size 350,000 images. This retrieval performance is several orders of magnitude faster when compared with other indexing methods for face-image retrieval. The fundamental reason for this speed-up was due to the synergy between classification based indexing and other schemes based on the approximate nearest neighbour principle such as Product Quantization and ANNOY. In such methods, an image is represented using a vector of features that do not explicitly account for attributes such as gender, age, and ethnicity. This means that an ANN search will return results that have a scope for further filtering using these attributes. This represents the synergy between the hierarchical classification

approach and the ANN approaches, thus accounting for the speedups obtained with the hybrid approach.

The following sections conclude this research study with a brief discussion on the findings, achievements, future work opportunities and limitations of this research study.

7.1 Findings

This section briefly highlights the interesting findings observed throughout the research study.

7.1.1 Labelling Errors

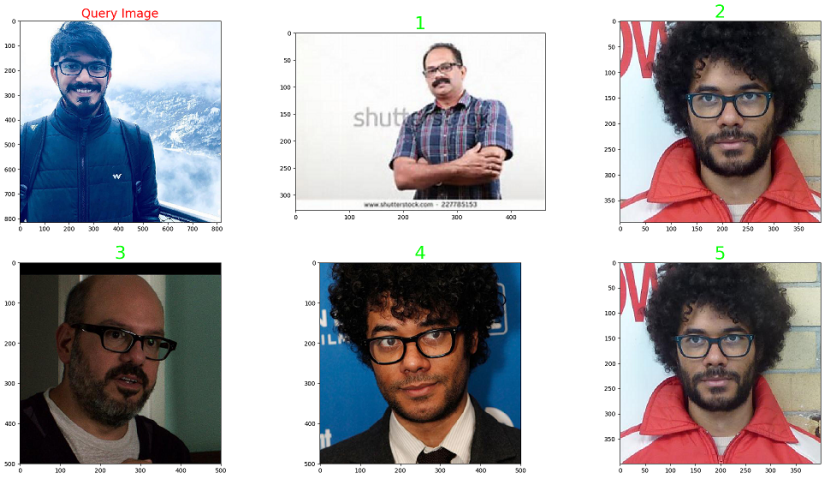
The proposed methodology was not only able to achieve significant speed-ups in retrieval time and precision but interestingly was able to uncover errors in labelling in the benchmark dataset used for experimentation. With the help of the classification models used in the hierarchical design, the system was able to identify labelling errors in the UTK Face dataset (Zhang et al., 2017). The original labelling was done using a DEX algorithm published by (Rothe, Timofte & Gool, 2015).

Figure 7.1 depicts some of the images associated with labelling errors identified for gender and ethnicity attributes.

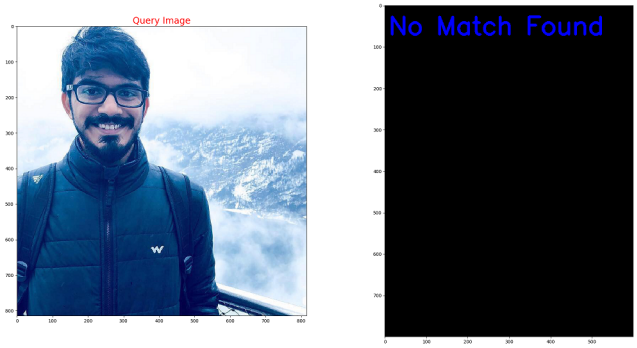
Figure 7.1: Images associated with gender and ethnicity labelling errors

7.1.2 Helpful Miss-classifications

One of the drawbacks of the proposed indexing strategy was the error introduced due to miss-classifications. However, for a given query image with no matching images present in the database, a miss-classification actually helped to correctly register a True Negative using the proposed method, whereas the other methods identified false matches for the same query image.



(a)



(b)

Figure 7.2: False matches with other methods (a) vs True Negatives obtained from proposed method (b)

Part **a** of Figure 7.2 shows false matches obtained with other indexing methods while part **b** depicts a true negative obtained with the proposed indexing method. Here the original ethnicity value was Indian but was misclassified to be White thereby removing likely false matches and maintaining a true negative.

7.1.3 Intra-class variation

According to the literature review conducted in Chapter 2, one of the main problems in face-recognition and retrieval systems was that of Intra-class Variation (ICV). Due to variance in the pose, face angle or facial accessories, the same person is mistaken as a different person.

The proposed system was tested with such images exhibiting high ICV in the hard query set, as mentioned in Chapter 6. The face-image retrieval system using the proposed methodology performed surprisingly well as most of the ICV conditions did not affect the retrieval performance.

Figure 7.3 shows example images that exhibit high ICV such as complex pose or face accessories like hats, masks, spectacles, or different hairstyles.

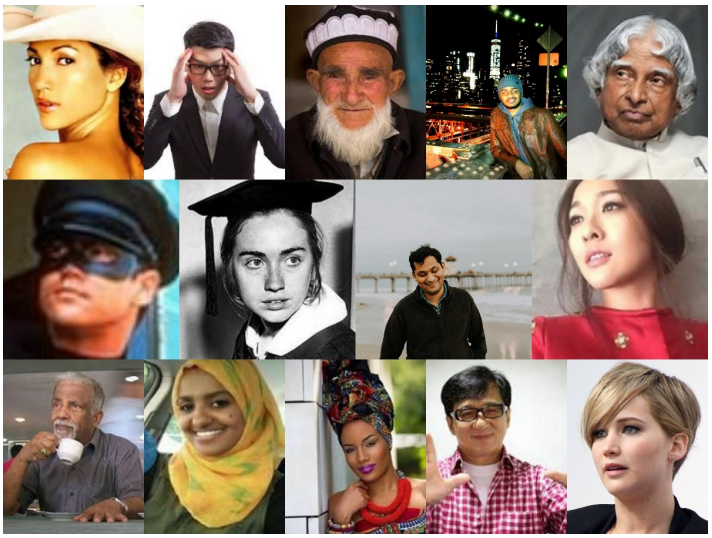


Figure 7.3: Images exhibiting Intra-class Variance

Figure 7.4 illustrates an example output of the successfully retrieved matches at rank-1 for the query image exhibiting high ICV using the proposed methodology.

Figure 7.4: Example output of a query image exhibiting ICV using proposed methodology

The below Figure 7.5 depicts an example output of falsely matching images retrieved except for a correct match at rank-7 for the same query image from the above Figure 7.4. The increased false matches are due to the effect of high ICV on other indexing methods exhibited by the query image.

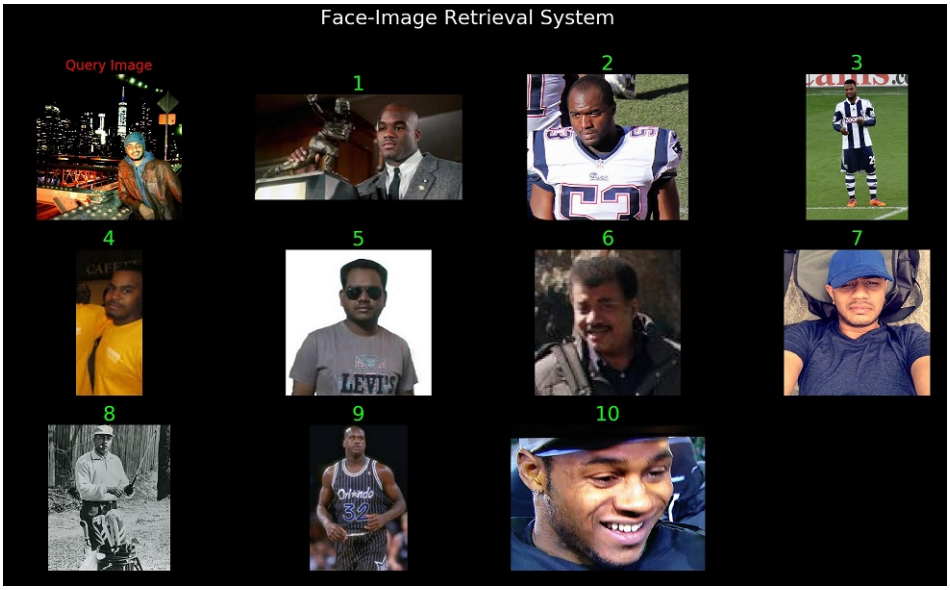


Figure 7.5: Example output of a query image exhibiting ICV using other indexing methods

7.1.4 Storage Costs

Another interesting finding was related to the storage cost. It was observed that the pre-computed index table for all 350,000 images required only about 200 MB when stored on disk with an unoptimized implementation. Other indexing methods such as ANNOY, required about 250MB when stored on a disk.

The storage costs can be reduced using various compression techniques on both the computed index table dictionary and the generated 128D feature vectors. These feature vectors can further be compressed by converting them into k-bit binary hash codes.

7.2 Achievements

This section highlights the key achievements and contributions made by this research study.

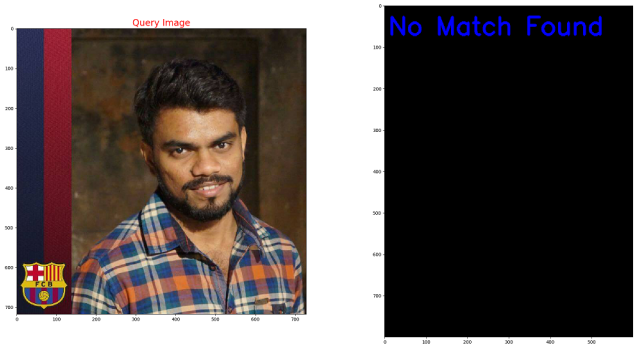
This research study has made two notable contributions. Firstly, a novel indexing method using hierarchical classification, and secondly a novel probabilistic backtracking algorithm.

Experimental results from Chapter 6 show that the combination proposed indexing strategy with ANNOY resulted in a reduction of retrieval time by a factor of 10 which is around 90% decrease when compared with state-of-the-art indexing method of ANNOY. It also resulted in an increased mAP@10 value by 9%.

The proposed methodology also obtains a higher number of relevant matching images which was evident from the high Precision at rank-10 value of around 80% compared to only 40% achieved by ANNOY. This means that irrelevant images are removed by the proposed methodology, thereby reducing human effort and inspection time associated with unnecessarily retrieved images.

7.2.1 Probabilistic Back-tracking Algorithm

The second notable contribution of this research study was a novel probabilistic back-tracking algorithm. The algorithm was responsible for identifying and correcting miss-classifications by performing a back-tracking operation which boosted the face-image retrieval performance significantly. Figure 7.6 provides an example comparison of the output obtained without and with using the back-tracking algorithm.



(a)

(b)

Figure 7.6: Results without back-tracking (a) vs with back-tracking algorithm (b)

It is clear that without the back-tracking algorithm (a), the system fails to find matching images indicating a miss-classification. When the back-tracking algorithm is used (b), a miss-classification of ethnicity value of Black is corrected to be Indian and matching images are found by the system.

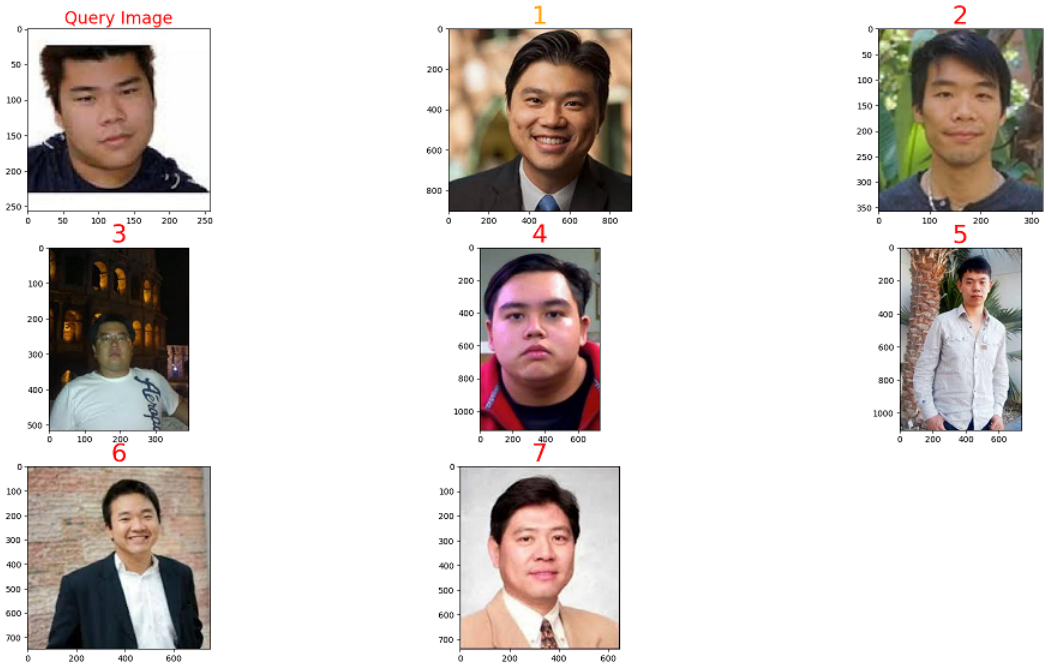
Statistical tests performed in Chapter 6 indicate with a confidence level of 99% that the use of this algorithm increased the mean accuracy by 5% - 12%.

7.2.2 Dynamic Thresholding

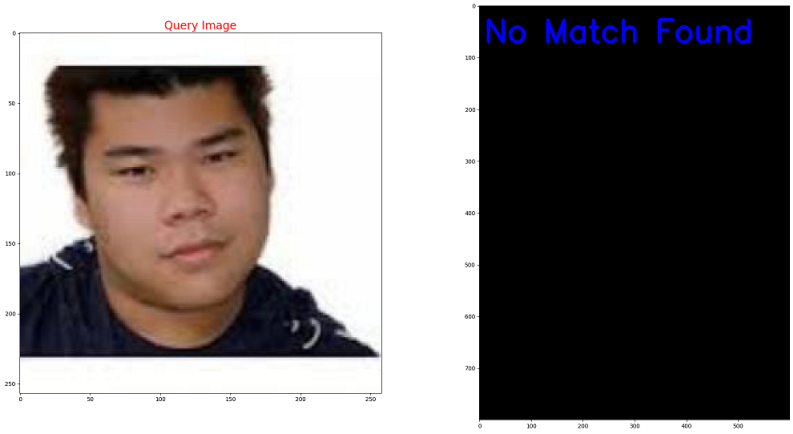
Another contribution worth mentioning is that of dynamic thresholding. Usually, most methods only apply a single, fixed threshold value to find matching images. However, it was observed that setting different threshold values to images with different attributes such as ethnicity or age improved the system performance by reducing the number of false matches.

This was particularly evident with images of Asian ethnicity, as shown in Figure 7.7 where the panel a shows the output with a fixed threshold of 0.5 containing many false positives. On the other hand, panel b shows that false matches are completely removed when a dynamic threshold of 0.4 is set based on the ethnicity value of Asian. The dynamic threshold values for Euclidean distance measure and Cosine similarity were presented in Chapter 5 in Table 5.2.

Statistical tests performed in Chapter 6 indicate with a confidence level of 95% that the use of dynamic thresholding will increase the mean accuracy by 0.5% - 6%.



(a)



(b)

Figure 7.7: Results without dynamic thresholding (a) vs with dynamic thresholding (b)

7.2.3 State-of-the-art Classification Models

Some of the trained classification models such as that of the gender classification model achieved better than state-of-the-art results on the UTK Face dataset for gender

classification which were presented in Table 6.11. These models were trained with a pre-trained CNN using transfer learning as a feature extractor and outperformed or matched some of the CNN models trained from scratch on classification tasks such as for the ethnicity classification as well.

7.3 Limitations

There are a number of limitations of the proposed methodology that need to be recognised. The following sections briefly highlight some of the limitations of the research study.

7.3.1 Privacy Issues

The main drawback or limitation of the proposed methodology is that of privacy concerns. The proposed indexing strategy extracts additional information such as age, gender, and ethnicity attribute values as a part of the procedure.

This additional information must be extracted only with the consent of the user. As part of a broader application used by law enforcement agencies or security surveillance, the public must be notified of the additional information that will be extracted and utilized such as their age-group, gender, and ethnic groups before automating the whole process.

Recently New Zealand police announced the use of a new state-of-the-art facial recognition system designed by an American company. It has sparked up privacy concerns and were raised recently by a law professor at the University of Auckland and a chairperson of Privacy Foundation NZ (Block, 2019).

In another article, the London police were informed that the use of any kind of facial recognition system must not introduce gender or racial bias (Perraudin, 2019). The public and the privacy officials have raised similar concerns in major countries like

China (C. Chen & Qu, 2019), India (Choudhury, 2019) and USA (Alba, 2019) due to the privacy threats associated with individual privacy.

Thus, if the privacy policy legislates against the use of additional information such as age, gender, or ethnicity, the proposed methodology will have to be augmented with suitable data perturbation methods.

7.3.2 Classification Error

Another limitation of the proposed methodology lies in the use of classification models. Each classification model will introduce some classification error right from level 1 of the hierarchy and thus matching images may not be found in some cases even if they do exist.

Training these classification models for a real-world application will require a large amount of training data that should include a sufficient number of images from all of the 19 different subsets as per the hierarchical design depicted in Figure 4.3.

Although the classification error can be minimized using techniques such as the proposed back-tracking algorithm, it can never be completely eliminated from the system.

7.3.3 Rigid Design

The hierarchical design depicted in Figure 4.3 is a rigid or static design. A major drawback is that it cannot be altered dynamically as the arrangement of classifier models is affected. Thus, finding an optimal hierarchical design can be challenging as well as a time-consuming task.

The database requires an initial labelling either done by a human or automated using a machine learning model. If incorrectly labelled, there is a possibility that the proposed methodology will never locate those matching images.

7.4 Future Work

This research study also presents opportunities for future research.

The CNN model used in this study was a pre-trained model. A better approach would be to train a CNN model from scratch according to the requirements. This would lead to increased classification accuracy, particularly for ethnicity classification which had the least accuracy of around 90% amongst the set of attributes used. This research study only used three attributes of age, gender and ethnicity. More attributes such as hairstyles, eye colours, facial hair, etc. can be used to reduce the database to an even smaller subset.

The index hierarchy used in this research study was designed according to the decreasing order of classification accuracy. A possible future work could be to optimize the hierarchical design. One approach could be to train a decision tree model or similar rule-based model to order the attributes according to their importance values and design the hierarchy based on the generated attribute importance scores.

Similar to the proposed dynamic thresholding scheme, retrieval algorithms can also be tuned dynamically based on one of the attributes. For example, for an ethnic group of White which has a significantly large number of images, a retrieval algorithm can be used which is faster with a comparable trade-off in the retrieval accuracy. Thus, exploiting the capabilities of hierarchical classification, different retrieval algorithms can be combined and used for optimizing the performance.

A unique future possibility is to explore the work done by Google on learned indexed structures published in (Kraska, Beutel, Chi, Dean & Polyzotis, 2018) and to identify whether it could be integrated with the proposed hierarchical classification models to enhance the performance by learning the index. The key idea of the original paper was that a machine learning model could be used to learn the sort order or the structure of lookup keys and use it effectively to predict the indexed position.

Another interesting future research could be to explore the possibility of developing a machine learning model that could predict beforehand which subset is most likely to be retrieved and hence only that subset can be cached in the memory for extremely faster retrieval operation. This process would be similar to the caching of user-specified data tables on Apache spark clusters[1] for faster computations. In this context, the selection of data table (subset) for caching will be predicted using machine learning models.

An optimized implementation of the proposed methodology would further improve the results. A proper indexing structure should be maintained that can be memory-mapped (mmapped) for optimal retrieval performance on large scale databases. The face-image pre-processing and feature extraction operations could be executed within a fraction of seconds using a GPU. Multi-processing can enable the proposed methodology to find matching images simultaneously for multiple queries which will be a significant boost to retrieval performance.

The proposed indexing method is not only restricted to face-image retrieval, but could also be easily extended to support different types of image-retrieval tasks from large databases. Instead of indexing on attributes such as age or gender, image categories such as types of vehicles or animals could be used for indexing depending upon the retrieval problem, if they can be successfully classified with high accuracy scores.

[1] `https://spark.apache.org/`

References

Alba, D. (2019, Jul). *Privacy and civil rights groups ask the us government to end its use of facial recognition tech on the public.* Retrieved from https://www.buzzfeednews.com/article/daveyalba/campaign-against-facial-recognition-house-homeland-security

Alkhawlani, M., Elmogy, M. & El-Bakry, H. (2015, 01). Text-based, content-based, and semantic-based image retrievals: A survey. *International Journal of Computer and Information Technology*, *4*, 58-66.

Alvi, F. B., Pears, R. & Kasabov, N. (2018). An evolving spatio-temporal approach for gender and age group classification with spiking neural networks. *Evolving Systems*, *9*(2), 145–156.

Andoni, A. & Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS06)*.

Aumüller, M., Bernhardsson, E. & Faithfull, A. (2017). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International conference on similarity search and applications* (pp. 34–49).

Bay, H., Tuytelaars, T. & Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404–417).

Berg, T. & Belhumeur, P. N. (2013). Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. *2013 IEEE Conference on Computer Vision and Pattern Recognition*.

Bhattarai, B., Sharma, G., Jurie, F. & Pérez, P. (2014). Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval. In *European conference on computer vision* (pp. 160–172).

Block, G. (2019, Dec). *Privacy concerns over police's new 'state of the art' facial recognition system.* Retrieved from https://www.stuff.co.nz/national/117957684/privacy-concerns-over-polices-new-state-of-the-art-facial-recognition-system

Cao, Q., Shen, L., Xie, W., Parkhi, O. M. & Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. In *International conference on automatic face and gesture recognition.*

Cao, X., Wipf, D., Wen, F., Duan, G. & Sun, J. (2013). A practical transfer learning algorithm for face verification. *2013 IEEE International Conference on Computer*

Vision.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, *1*(2), 1.

Chao, W.-L., Liu, J.-Z. & Ding, J.-J. (2013). Facial age estimation based on label-sensitive learning and age-oriented regression. *Pattern Recognition*, *46*(3), 628–641.

Chen, B.-C., Chen, Y.-Y., Kuo, Y.-H. & Hsu, W. H. (2013). Scalable face image retrieval using attribute-enhanced sparse codewords. *IEEE Transactions on Multimedia*, *15*(5), 1163–1173.

Chen, C. & Qu, T. (2019, Nov). *Data privacy woes grow as china's facial recognition tech advances.* Retrieved from `https://www.techinasia.com/data-privacy-woes-grow-china-facial-recognition-advances`

Chen, D., Cao, X., Wen, F. & Sun, J. (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3025–3032).

Chen, H., Deng, Y. & Zhang, S. (2016). Where am i from?–east asian ethnicity classification from facial recognition. *Project study in Stanford University*.

Choi, S. E., Lee, Y. J., Lee, S. J., Park, K. R. & Kim, J. (2011). Age estimation using a hierarchical classifier based on global and local facial features. *Pattern Recognition*, *44*(6), 1262–1281.

Choudhury, S. R. (2019, Oct). *Facial recognition technology needs controls on its use, world economic forum says.* Retrieved from `https://www.cnbc.com/2019/10/04/governments-must-ensure-fair-transparent-use-of-facial-recognition-wef.html`

Cootes, T. F., Edwards, G. J. & Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*(6), 681–685.

Dantcheva, A., Elia, P. & Ross, A. (2015). What else does your biometric data reveal? a survey on soft biometrics. *IEEE Transactions on Information Forensics and Security*, *11*(3), 441–467.

Dearden, L. (2019). *Facial recognition wrongly identifies public as potential criminals 96% of time.* Retrieved from `https://www.independent.co.uk/news/uk/home-news/facial-recognition-london-inaccurate-met-police-trials-a8898946.html?amp`

Deselaers, T., Keysers, D. & Ney, H. (2008). Features for image retrieval: an experimental comparison. *Information retrieval*, *11*(2), 77–107.

Dhomne, A., Kumar, R. & Bhan, V. (2018). Gender recognition through face using deep learning. *Procedia computer science*, *132*, 2–10.

Duan, M., Li, K., Yang, C. & Li, K. (2018). A hybrid deep learning cnn–elm for age and gender classification. *Neurocomputing*, *275*, 448–461.

Egri, L. & Shultz, T. R. (2015). Constraint-satisfaction models. *International Encyclopedia of the Social & Behavioral Sciences*.

Eidinger, E., Enbar, R. & Hassner, T. (2014). Age and gender estimation of unfiltered

faces. *IEEE Transactions on Information Forensics and Security*, *9*(12), 2170–2179.

England, R. (2019). Uk police's facial recognition system has an 81% percent error rate. *engadget*. Retrieved from `https://www.engadget.com/amp/2019/07/04/uk-met-facial-recognition-failure-rate/?guccounter=1`

Farkas, L. G. (1994). *Anthropometry of the head and face*. Raven Pr.

Fu, Y., Hospedales, T. M., Xiang, T., Gong, S. & Yao, Y. (2014). Interestingness prediction by robust learning to rank. In *European conference on computer vision* (pp. 488–503).

Geitgey, A. (2016, Jul). *Machine learning is fun!* Retrieved from `https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78`

Geng, C. & Jiang, X. (2009). Face recognition using sift features. In *2009 16th ieee international conference on image processing (icip)* (pp. 3313–3316).

Gionis, A., Indyk, P. & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Vldb* (Vol. 99, pp. 518–529).

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heng, Z., Dipu, M. & Yap, K.-H. (2018). Hybrid supervised deep learning for ethnicity classification using face images. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*.

Huang, D., Shan, C., Ardabilian, M., Wang, Y. & Chen, L. (2011). Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(6), 765–781.

Huang, G., Liu, Z., Maaten, L. V. D. & Weinberger, K. Q. (2017). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, G. B., Ramesh, M., Berg, T. & Learned-Miller, E. (2007, October). *Labeled faces in the wild: A database for studying face recognition in unconstrained environments* (Tech. Rep. No. 07-49). Massachusetts, Amherst: University of Massachusetts, Amherst.

Jabeen, S., Mehmood, Z., Mahmood, T., Saba, T., Rehman, A. & Mahmood, M. T. (2018). An effective content-based image retrieval technique for image visuals representation based on the bag-of-visual-words model. *PloS one*, *13*(4), e0194526.

Jang, Y. K., Jeong, D.-j., Lee, S. H. & Cho, N. I. (2018). Deep clustering and block hashing network for face image retrieval. In *Asian conference on computer vision* (pp. 325–339).

Jegou, H., Douze, M. & Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, *33*(1), 117–128.

Jeong, Y., Lee, S., Park, D. & Park, K. (2018). Accurate age estimation using multi-task siamese network-based deep metric learning for frontal face images. *Symmetry*, *10*(9), 385.

Johnson, J. & Karpathy, A. (n.d.). *Cs231n convolutional neural networks for visual recognition.* Retrieved from `http://cs231n.github.io/convolutional-networks/`

Jolliffe, I. (2011). *Principal component analysis*. Springer.

Kasabov, N. (2012). Neucube evospike architecture for spatio-temporal modelling and pattern recognition of brain signals. *Artificial Neural Networks in Pattern Recognition Lecture Notes in Computer Science*, 225–243.

Kaur, K. D. & Rai, P. (n.d.). An analysis on gender classification and age estimation approaches. *International Journal of Computer Applications*, *975*, 8887.

Kazemi, V. & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Khan, M. A. & Jalal, A. S. (2020). A framework for suspect face retrieval using linguistic descriptions. *Expert Systems with Applications*, *141*, 112925.

King, D. (2014, Feb). *Make your own object detector.* Retrieved from `http://blog.dlib.net/2016/10/easily-create-high-quality-object.html`

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*, 1755-1758.

King, D. E. (2015). Max-margin object detection. *arXiv preprint arXiv:1502.00046*.

Klare, B. F., Klein, B., Taborsky, E., Blanton, A., Cheney, J., Allen, K., ... et al. (2015). Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kraska, T., Beutel, A., Chi, E. H., Dean, J. & Polyzotis, N. (2018). The case for learned index structures. In *Proceedings of the 2018 international conference on management of data* (pp. 489–504).

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Kumar, N., Berg, A., Belhumeur, P. N. & Nayar, S. (2011). Describable visual attributes for face verification and image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(10), 1962–1977.

Kumar, N., Berg, A. C., Belhumeur, P. N. & Nayar, S. K. (2009). Attribute and simile classifiers for face verification. *2009 IEEE 12th International Conference on Computer Vision*.

Kumar, N., Zhang, L. & Nayar, S. (2008). What is a good nearest neighbors algorithm for finding similar patches in images? In *European conference on computer vision* (pp. 364–378).

Kwon, Y. H. & da Vitoria Lobo, N. (1999). Age classification from facial images. *Computer vision and image understanding*, *74*(1), 1–21.

Learned-Miller, E., Huang, G. B., RoyChowdhury, A., Li, H. & Hua, G. (2016). Labeled faces in the wild: A survey. In *Advances in face detection and facial image analysis* (pp. 189–248). Springer.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Levi, G. & Hassncer, T. (2015). Age and gender classification using convolutional neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Lew, M. S., Sebe, N., Djeraba, C. & Jain, R. (2006). Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *2*(1), 1–19.

Li, J., Wang, T. & Zhang, Y. (2007). Face recognition using feature of integral gabor-haar transformation. *2007 IEEE International Conference on Image Processing*.

Liu, Z., Luo, P., Wang, X. & Tang, X. (2015). Deep learning face attributes in the wild. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*.

Lynch, J. (2014). *Fbi plans to have 52 million photos in its ngi face recognition database by next year.* Retrieved from `https://www.eff.org/deeplinks/2014/04/fbi-plans-have-52-million-photos-its-ngi-face-recognition-database-next-year`

Maaten, L. v. d. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Malisiewicz, T., Gupta, A. & Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the 2011 international conference on computer vision* (pp. 89–96).

Masi, I., Wu, Y., Hassner, T. & Natarajan, P. (2018). Deep face recognition: A survey. In *2018 31st sibgrapi conference on graphics, patterns and images (sibgrapi)* (pp. 471–478).

Neubeck, A. & Van Gool, L. (2006). Efficient non-maximum suppression. In *18th international conference on pattern recognition (icpr'06)* (Vol. 3, pp. 850–855).

Ng, H.-W. & Winkler, S. (2014). A data-driven approach to cleaning large face datasets. In *2014 ieee international conference on image processing (icip)* (pp. 343–347).

Nister, D. & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06)* (Vol. 2, pp. 2161–2168).

Pan, S. J. & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Park, U. & Jain, A. K. (2010). Face matching and retrieval using soft biometrics. *IEEE Transactions on Information Forensics and Security*, *5*(3), 406–415.

Perraudin, F. (2019, May). *Facial recognition must not introduce gender or racial bias.* Retrieved from `https://www.theguardian.com/technology/`

2019/may/29/facial-recognition-must-not-introduce-gender-or-racial-bias-police-told

Phillips, P. J., Moon, H., Rizvi, S. A. & Rauss, P. J. (2000). The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(10), 1090–1104.

Prabhu, R. (2018, Mar). *Understanding of convolutional neural network (cnn) — deep learning.* Retrieved from https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Ram, P. & Sinha, K. (2019). Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1378–1388).

Ricanek, K. & Tesafaye, T. (2006). Morph: A longitudinal image database of normal adult age-progression. In *7th international conference on automatic face and gesture recognition (fgr06)* (pp. 341–345).

Riesenhuber, M. & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, *2*(11), 1019.

Rothe, R., Timofte, R. & Gool, L. V. (2015). Dex: Deep expectation of apparent age from a single image. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*.

Rothe, R., Timofte, R. & Gool, L. V. (2016, July). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211-252. doi: 10.1007/s11263-015-0816-y

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S. & Pantic, M. (2016). 300 faces in-the-wild challenge: Database and results. *Image and vision computing*, *47*, 3–18.

Savchenko, A. V. (2019). Efficient facial representations for age, gender and identity recognition in organizing photo albums using multi-output convnet. *PeerJ Computer Science*, *5*, e197.

Schroff, F., Kalenichenko, D. & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sharif Razavian, A. S., Azizpour, H., Sullivan, J. & Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, Y., Chen, Y., Wang, X. & Tang, X. (2014). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems* (pp. 1988–1996).

Sun, Y., Liang, D., Wang, X. & Tang, X. (2015). Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*.

Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first aaai conference on artificial intelligence.*

Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Tang, J., Li, Z. & Zhu, X. (2018). Supervised deep hashing for scalable face image retrieval. *Pattern Recognition*, *75*, 25–32.

Torralba, A., Fergus, R. & Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, *30*(11), 1958–1970.

Uriza, E., Fernández, F. G. & Rais, M. (2018). Efficient large-scale image search with a vocabulary tree. *Image Processing On Line*, *8*, 71–98.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, *1*(511-518), 3.

Vo, T., Nguyen, T. & Le, C. (2018). Race recognition using deep convolutional neural networks. *Symmetry*, *10*(11), 564.

Wang, D., Otto, C. & Jain, A. K. (2015). Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*.

Wang, Y., Bao, T., Ding, C. & Zhu, M. (2017). Face recognition in real-world surveillance videos with deep learning method. In *2017 2nd international conference on image, vision and computing (icivc)* (pp. 239–243).

Weinberger, K. Q. & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*(Feb), 207–244.

Wen, G., Chen, H., Cai, D. & He, X. (2018). Improving face recognition with domain adaptation. *Neurocomputing*, *287*, 45–51.

Winder, S. A. & Brown, M. (2007). Learning local image descriptors. In (pp. 1–8). IEEE.

Wolf, L., Hassner, T. & Maoz, I. (2011). *Face recognition in unconstrained videos with matched background similarity*. IEEE.

Wu, Z., Ke, Q., Sun, J. & Shum, H.-Y. (2011). Scalable face image retrieval with identity-based quantization and multireference reranking. *IEEE transactions on pattern analysis and machine intelligence*, *33*(10), 1991–2001.

Yi, D., Lei, Z., Liao, S. & Li, S. Z. (2014). Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*.

Zeng, J., Zeng, J. & Qiu, X. (2017). Deep learning based forensic face verification in videos. In *2017 international conference on progress in informatics and computing (pic)* (pp. 77–80).

Zhang, Z., Song, Y. & Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, F. Z., Wan, G. C., Kuang, Y. K. & Tong, M. S. (2018). An efficient face recognition algorithm based on deep learning for unmanned supermarket. In *2018 progress in electromagnetics research symposium (piers-toyama)* (pp. 715–718).