baocao.docx

bởi An Hồ

Ngày Nộp: 20-thg 2-2025 10:54CH (UTC+0700)
ID Bài Nộp: 2593680135
Tên Tập tin: baocao.docx (4.74M)
Đếm từ: 12345
Đếm ký tự: 77703

VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

DO MINH QUAN – 521H0290
HO HUU AN – 521H0489

ENHANCED FACE CLASSIFICATION USING CENTROID-BASED REPRESENTATION

INFORMATION TECHNOLOGY PROJECT

COMPUTER SCIENCE

HO CHI MINH CITY, YEAR 2025

VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

DO MINH QUAN – 521H0290
HO HUU AN – 521H0489

ENHANCED FACE CLASSIFICATION USING CENTROID-BASED REPRESENTATION

INFORMATION TECHNOLOGY PROJECT

COMPUTER SCIENCE

Advised by

MSc. Nguyen Thanh An

HO CHI MINH CITY, YEAR 2025

ACKNOWLEDGMENT

We would like to express our sincere gratitude to MSc. Nguyen Thanh An from Ton Duc Thang University for his invaluable support and guidance throughout our project. His expertise in the field has been instrumental in shaping the direction of our work.

MSc. Nguyen Thanh An's commitment to excellence and his willingness to share his knowledge have significantly contributed to the success of this endeavor. His mentorship has not only enhanced our understanding of front-end and back-end data frameworks but has also inspired us to delve deeper into the intricacies of algorithm models designed to handle vast amounts of data.

We extend our heartfelt thanks to Mr. Nguyen Thanh An for his continuous encouragement and support, which have played a pivotal role in the completion of this project. His dedication to fostering a learning environment has been a source of inspiration, and we are grateful for the opportunity to work under his guidance.

Ho Chi Minh City, 20th January 2025

Authors

Do Minh Quan *Ho Huu An*

DECLARATION OF AUTHORSHIP

I hereby declare that this thesis was carried out by myself under the guidance and supervision of M.Sc. NGUYEN THANH AN; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

I will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

Ho Chi Minh City, 20th January 2025

Authors

Do Minh Quan *Ho Huu An*

ENHANCED FACE CLASSIFICATION USING CENTROID-BASED REPRESENTATION

ABSTRACT

Deep learning-based face recognition models achieve high accuracy but often lack interpretability, raising concerns about trust and bias. Limited training data can lead to reduced generalizability, affecting model robustness across different demographics.This study enhances both accuracy and interpretability in face classification using centroid-based representations in the latent space. Feature maps from ResNet-50, MobileNetV2, and EfficientNetB0 are clustered to derive centroids, and distance vectors are computed to capture spatial relationships. These representations are then classified using Support Vector Machines (SVM) and Random Forests, bridging deep learning with interpretable machine learning.By integrating clustering insights with conventional classifiers, this approach improves performance while providing a transparent framework for understanding facial feature discrimination. Additionally, it addresses challenges of imbalanced data by capturing essential feature variations. The findings contribute to Explainable AI (XAI), shedding light on clustering dynamics and decision-making pathways in face classification.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial intelligence
API	Application Programming Interface
BE	Back-end
CAGR	Compound annual growth rate
CBA	Component-Based Architecture
CNNs	Convolutional Neural Network
CORS	Cross-Origin Resource Sharing
CV	Computer Vision
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DTs	Decision Trees
FCN	Fully Convolutional Networks
FE	Front-end
GANs	Generative Adversarial Networks
GMM	Gaussian Mixture Models
GMM	Gaussian Mixture Models
HOG	Histogram of Oriented Gradients
JSON	JavaScript Object Notation

KNN	K-nearest neighbors
LLMs	Large language models
ML	Machine learning
MLE	Maximum Likelihood Estimation
MLP	Multi-layer Perceptron
MVC	Model-View-Controller
OpenCV	Open Source Computer Vision Library
PCA	Principal Component Analysis
PWAs	Progressive Web Apps
ResNet	Residual Networks
RPN	Region Proposal Network
SEO	Search Engine Optimization
SIFT	Scale-Invariant Feature Transform
SSG	Static Site Generation
SSR	Server-Side Rendering
UI	User Interface
UX	User Experiences
ViTs	Vision Transformers
VUIs	Voice User Interfaces
XAI	Explainable AI

XML	Extensible Markup Language
YOLO	You Only Look Once

CHAPTER 1. OVERVIEW OF THE TOPIC

1.1 Overview of the current situation

1.1.1 Artificial intelligence

Artificial intelligence (AI) is shifting from rapid development to more stable and scalable integration, emphasizing its practical application across diverse data types and real-world workflows. By 2025, AI is expected to be a core component in many sectors, significantly boosting productivity, customer experience, and service. Major changes include:

Multichannel AI: AI is expanding its interface beyond text to include voice, images, video, and direct on-screen actions, making it a versatile tool for various operations.

Agentic AI: This emerging technology enables AI systems to make independent decisions and adapt their actions based on their environment, using machine learning to learn and adjust with minimal human input.

AI in Software Development: Increased use of AI in software development enhances employee productivity and boosts sustainability efforts, streamlining algorithm creation.

AI in Cybersecurity and Defense: AI applications are becoming more common in cybersecurity and defense systems.

AI Legislation: Governments worldwide are focusing on the ethical aspects of AI and enacting legislation to promote its responsible usage.

Beyond Chatbots: AI developers and corporate clients are exploring applications that leverage large language models (LLMs) on the back end to summarize or interpret unstructured data, allowing generative AI to scale more easily.

Commoditization of Generative AI Models: The competitive advantage is shifting from having the best model to excelling at fine-tuning pre-trained models or building specialized tools.

Domain-Specific AI: There is a growing demand for limited, highly specialized models for enterprises, with businesses examining how AI technology will be utilized, who will use it, and in what sector.

1.1.2 Computer Vision

Computer Vision (CV) is quickly growing, fueled by AI developments, and is poised to change industries by allowing robots to understand and analyze visual data. CV is predicted to increase productivity, scalability, and creativity in a variety of industries by 2025.

Edge AI: The combination of edge computing with AI will transform real-time decision-making in computer vision applications by processing visual input closer to its source. The edge AI market is expected to reach $8.06 billion by 2025, at a CAGR of 20.3%. This is especially beneficial in cases that need immediate reactions, such as driverless cars and real-time monitoring.

Generative AI transforms computer vision by creating synthetic pictures and movies. These breakthroughs are being driven by technologies such as generative adversarial networks (GANs) and diffusion models.

3D Modeling: As 3D modeling technologies advance, more accurate representations of real-world things emerge, which is particularly essential in architecture, gaming, and virtual reality.

AI and robots Integration: More collaboration between AI and robots is predicted, with CV playing a key role in driving productivity increases across sectors.

Vision Transformers (ViTs) are neural network designs that analyze pictures utilizing self-attention processes, finding detailed correlations between pixels and improving image categorization and object recognition accuracy. ViTs have greater scalability and versatility than CNNs and are ideal for sophisticated, high-precision computer vision applications such as medical imaging, autonomous cars, and industrial automation.

Ethical Considerations and Regulation: As AI technologies grow more prevalent, ethical concerns and regulatory frameworks will gain relevance. Stricter

rules and industry norms for the development and deployment of CV technologies are expected, demanding experts with a solid grasp of AI ethics, data protection, and regulatory compliance.

Multimodal AI Integration: Combining computer vision with other AI technologies, such as natural language processing and speech recognition, broadens the scope of multimodal AI applications, allowing for more sophisticated use cases like augmented reality and advanced customer service automation.

Advancements in Object recognition and Tracking: Further development of object recognition and tracking algorithms will result in considerable gains in accuracy, speed, and resilience. These developments are critical for self-driving cars, surveillance systems, and sophisticated robotic applications.

1.1.3 Latest development status of Front-end and Back-end

Front-end development

Major Progress: In 2025, front-end development will focus on efficiency, elegance, and innovation1. No-code tools, AI-driven interfaces, and an emphasis on sustainability are among the current trends.

No-Code/Low-Code Platforms: These platforms allow non-developers to create working apps by designing and creating them with drag-and-drop interfaces.

Component-Based UI Libraries: Libraries like as React, Vue, and shadcn UI encourage modular design approaches, with shadcn UI enabling developers to copy individual components into their projects for lightweight code and customisation.

AI and Machine Learning: Developers are upgrading user interfaces with frameworks such as TensorFlow.js and ml5.js.

Progressive Web Apps (PWAs): PWAs combine the stability of a website with the functionality of a mobile app, resulting in quicker load times and offline capabilities.

Voice User Interfaces (VUIs): As speech-activated devices gain popularity, developers must grasp voice recognition APIs.

Top Frameworks/Technologies:

- React: A popular JavaScript library for building user interfaces.
- Vue.js: Noted for its gentle learning curve, making it ideal for beginners.
- Angular: Suited for large-scale applications due to its comprehensive framework.
- Svelte: Known for high performance, as it shifts much of the work to compile time.

Back-end development

Major Changes: Cloud technology and AI are reshaping backend development, with cloud services making application construction and scaling easier.

Cloud Services: Cloud platforms such as AWS, Azure, and Google Cloud are critical, with AWS dominating the market share.

AI and ML Integration: AI automates code reviews and testing, which improves development operations. Hyperautomation connects automated functions across companies.

Modern DevOps approaches, including as containerization (Docker), GitOps, and CI/CD pipelines, are critical for effective development and deployment.

Cloud-Based AI: Cloud-based AI is transforming backend development by making machine learning operations more accessible.

Frameworks: ASP.NET, Ruby on Rails, CakePHP, Django, Laravel, Flask.

Containerization: Docker is the go-to platform for containerization.

CI/CD Pipelines: Tools like Jenkins, GitHub Actions, and GitLab CI are improving bug detection and update deployment.

Platform Engineering: Platform engineering provides developers with all necessary resources in one place.

1.2 Meaning of Research

This study on improved face categorization using centroid-based representation is significant from both theoretical and practical viewpoints:

1.2.1 Theoretical Significance

This study contributes to the theoretical understanding of face recognition by rigorously testing the performance of a novel classification model. By employing a centroid-based representation of facial features extracted from deep neural networks, we aim to demonstrate improvements in accuracy, robustness, and explainability compared to traditional methods.

Evaluate the effectiveness of centroid-based feature representation: We assess how well this representation captures essential discriminatory information from facial images.

Compare performance against benchmark methods: The performance of the proposed model will be compared against established face recognition techniques, such as those employing global average pooling or other feature extraction methods, to quantify any gains achieved.

Analyze the impact of key parameters: We investigate how parameters such as the number of clusters (k) in the k-means algorithm and the dimensionality reduction technique (PCA) affect the model's performance.

Assess generalization ability: The model's ability to generalize to unseen data will be evaluated using appropriate cross-validation techniques and independent test datasets.

Provide insights into model behavior: By visualizing learned centroids and feature maps, we aim to gain a deeper understanding of how the model makes its decisions, contributing to the interpretability of deep learning models.

The theoretical contribution lies in providing a comprehensive evaluation of a novel approach to face classification, advancing the state-of-the-art in computer vision and machine learning.

1.2.2 Social Applications

This research advances both theoretical understanding and practical applications, particularly in large-scale national face databases. A reliable face

classification system can serve as a critical decision-making tool across societal domains, provided ethical challenges are addressed. Key applications include law enforcement, where it could aid in suspect identification, locating missing persons, or recognizing crime victims, enhancing public safety. In national security, it could strengthen border control, identity verification, and threat detection. Citizen services could also benefit, streamlining processes like passport issuance, voter registration, and social welfare administration. Additionally, in disaster relief, the system could help identify victims, enabling swift emergency responses.

However, deploying this technology requires addressing ethical concerns. Privacy protection is paramount to safeguard personal data and prevent unauthorized access. Bias mitigation is essential to avoid discriminatory outcomes, while transparency ensures accountability through public scrutiny. System security must also be prioritized to prevent misuse or malicious attacks. Balancing these ethical considerations with the technology's benefits is crucial for its responsible and effective implementation.

1.3 Research Objectives

The "Enhanced Face Classification Using Centroid-Based Representation" approach can be further refined through several avenues. Feature extraction could benefit from alternative architectures like EfficientNet or vision transformers, potentially enhanced with attention mechanisms to focus on critical facial regions. Dimensionality reduction might be improved using t-SNE or UMAP instead of PCA, while hybrid representations combining centroid-based distances with descriptors like Local Binary Patterns could enrich feature quality.

Clustering techniques could move beyond k-means to explore Gaussian Mixture Models, hierarchical clustering, or adaptive k-selection using metrics like silhouette score. Fuzzy clustering could capture nuanced feature relationships, and visualizing centroids in the image space might reveal discriminative facial attributes.

For classification, ensemble methods, advanced algorithms like XGBoost, and regularization techniques could improve robustness and reduce overfitting. Data

augmentation and diverse evaluation metrics would enhance performance assessment. Adversarial testing could further validate robustness.

Explainability tools like saliency maps and interactive visualizations could clarify model decisions, while comparative analyses with traditional methods would highlight the centroid-based approach's advantages. For real-world applications, optimizing computational efficiency and ensuring generalization across datasets and conditions are critical. Privacy concerns could be addressed through security measures and anonymization.

The development plan includes a deep learning model with advanced clustering and representation strategies, a mobile frontend for real-time detection, and a backend for performance evaluation through detailed metrics like confusion matrices.

3 main objectives of research:

- Elucidate the foundational principles underlying the decision-making processes of a classification model by analyzing the spatial distribution of learned features within the latent space, thereby providing a comprehensive understanding of how feature representations influence model predictions.
- Introduce a novel embedding methodology predicated on the metric distances to distinct feature clusters within the latent space, and subsequently conduct a rigorous empirical evaluation to assess its efficacy in enhancing model performance and interpretability.
- Design and implement a robust software architecture encompassing both front-end and back-end components, aimed at demonstrating the recognition capabilities of the model and providing an interpretative framework based on the retrieval and visualization of semantically similar images, thereby facilitating a deeper insight into the model's decision-making rationale.

1.4 Report structure

This report is structured to provide a comprehensive overview of the enhanced face classification system, starting with foundational knowledge and progressing through the methodology, implementation, and evaluation. Chapter 2, "General Understanding," lays the groundwork by exploring computer science problems relevant to face classification, reviewing existing algorithm models for face recognition, examining the current technology landscape in front-end and back-end development, and detailing back-end architecture technologies and cloud services. Chapter 3, "Methodology," defines the specific problem type addressed, details the model's structure, explains image pre-processing, describes the data used, and presents the experimental setup and results. Chapters 4 and 5 focus on the practical implementation, with Chapter 4 detailing the front-end application's technologies, functionality, and main functions, and Chapter 5 describing the back-end application's architecture, request flow, and performance metrics. Finally, Chapter 6, "Conclusion," summarizes the project's achievements, highlights advantages, discusses challenges, and outlines future development goals. A list of cited sources and related works is included in the "References" section

CHAPTER 2. BACKGROUND KNOWLEDGE

2.1 Knowledge of computer science problems

In the field of computer science, particularly in artificial intelligence and computer vision (CV), various problems arise that require sophisticated algorithms and computational models to solve. These problems can be broadly categorized into several types, including object detection, segmentation, classification, and regression, each of which plays a significant role in image analysis and pattern recognition.

Object detection is a fundamental problem in computer vision that includes recognizing and finding items inside an image or video frame. Unlike classification, which just identifies whether an object exists, object detection offers bounding box coordinates, allowing for exact localization. This job is commonly employed in facial recognition, autonomous driving, and security surveillance systems, where identifying many objects in real time is critical.

Segmentation is the process of dividing a picture into relevant areas to facilitate analysis. It may be separated into two types: semantic segmentation, which labels all pixels in a certain class, and instance segmentation, which distinguishes between unique objects within the same class. Face classification can employ segmentation techniques to identify facial characteristics, eliminate background noise, and improve classification accuracy.

Classification is another key issue in computer vision that entails assigning predetermined labels to pictures based on their content. In the context of face classification, this entails sorting facial photographs into separate categories such as identities, moods, or age groupings. Deep learning techniques, such as convolutional neural networks (CNNs), have dramatically increased classification accuracy by extracting high-level information from pictures. The centroid-based representation strategy utilized in this work intends to improve the performance of classic classification models by exploiting centroid-based metrics.

Instead of categorical classification, regression is used in computer vision to predict continuous values. This approach is often used in age estimation, where a model predicts a numerical age using face traits. Regression models may also assess facial features such as posture estimation, skin tone fluctuations, and facial landmark identification, which are critical for boosting face classification robustness.

One major challenge is the extraction of significant information from raw photos. Feature extraction has progressed from simple handmade methods to sophisticated deep learning systems that automatically build hierarchical representations of complicated patterns in facial data. This automated extraction is crucial since the quality of the features influences future tasks like clustering and classification. Clustering is critical; it entails grouping comparable face embeddings into natural clusters, with centroids serving as representative characteristics for each class. This notion is closely connected to unsupervised learning approaches, which leverage the data's intrinsic structure to enhance classification judgments rather than depending exclusively on labeled samples.

Another significant issue is object tracking, particularly in situations where face categorization must be performed to video feeds or dynamic settings. Tracking guarantees that a face, once spotted, is consistently observed in subsequent frames. This continuity is critical for applications like surveillance or interactive systems, where identity preservation is as vital as initial detection.

Transfer learning and domain adaptability broaden the scope of these difficulties. Transfer learning uses pre-trained models from broad, varied datasets to enhance performance on specialized tasks like face classification by adapting learnt characteristics to new settings. Domain adaptation is especially important when there is a mismatch between the settings of the training data and the operational environment, ensuring that models stay resilient even when confronted with changing illumination, occlusions, or position variations.

Furthermore, anomaly detection has developed as a vital component, particularly in applications that require identifying outliers or unexpected inputs. In

the context of face categorization, finding anomalies can assist indicate occurrences that depart considerably from established patterns, increasing the system's overall dependability. Ensemble approaches add to this ecosystem by combining the capabilities of different models to get more consistent and accurate results.

2.2 Computer Vision Algorithms

Within the domain of computer vision, a plethora of sophisticated algorithms have been meticulously engineered to address a diverse array of challenges, including but not limited to object detection, classification, regression, feature extraction, clustering, object tracking, transfer learning, and anomaly detection. These algorithms are intricately tailored to enhance efficacy and precision, often being optimized for singular tasks or a constellation of interrelated functions. Their development underscores the relentless pursuit of computational efficiency and accuracy in interpreting visual data.

2.2.1 Object Detection Algorithms

Object detection necessitates the capability of models to discern both the spatial coordinates and categorical labels of objects within an image or video frame. Region-based Convolutional Neural Networks (R-CNN) employ a two-stage methodology, initially generating candidate object regions through Selective Search, followed by the classification of each region utilizing a convolutional neural network (CNN). To expedite this process, Faster R-CNN introduces a Region Proposal Network (RPN), which supplants the Selective Search mechanism, thereby significantly augmenting computational efficiency. In contrast, the You Only Look Once (YOLO) framework adopts a singular, holistic approach by processing the entire image in one forward propagation, segmenting it into a grid system, and concurrently predicting bounding boxes and class probabilities for each grid cell. Analogously, the Single Shot MultiBox Detector (SSD) architecture capitalizes on multi-scale feature maps to facilitate the detection of objects across varying dimensions, thereby optimizing detection accuracy and robustness. These

methodologies collectively underscore the advancements in algorithmic design aimed at achieving real-time, high-precision object detection.

2.2.2 Image Classification Algorithms

Image classification involves the assignment of predefined labels to images based on their visual content. Convolutional Neural Networks (CNNs) have fundamentally transformed this task by hierarchically extracting high-level features through successive convolutional and pooling layers. Architectures such as VGG-16, ResNet, and DenseNet have set benchmarks in classification performance, with ResNet introducing residual connections to mitigate vanishing gradients and DenseNet fostering feature reuse through dense inter-layer connectivity. More recently, Vision Transformers (ViT) have emerged as a paradigm shift, eschewing convolutional operations in favor of self-attention mechanisms to capture long-range dependencies and contextual relationships across image regions, thereby offering a novel approach to image understanding.

2.2.3 Feature Extraction Methods

The extraction of salient features from raw image data has transitioned from traditional, hand-engineered techniques to sophisticated deep learning-based methodologies. The Scale-Invariant Feature Transform (SIFT) algorithm identifies and delineates local image features that remain invariant to variations in scale and rotation, thereby ensuring robustness in diverse visual conditions. Similarly, the Histogram of Oriented Gradients (HOG) method captures shape-related attributes by analyzing the distribution of gradient orientations within localized regions of an image. In contrast, deep learning models leverage convolutional layers to autonomously derive hierarchical feature representations. Rather than employing the entire network for classification, the penultimate convolutional layers are often repurposed as feature extractors, providing rich, transferable representations for downstream tasks such as object detection, segmentation, or clustering.

2.2.4 Clustering Algorithms

Clustering algorithms play a pivotal role in unsupervised learning by grouping analogous image representations into coherent clusters, thereby uncovering inherent structures within the data. K-Means clustering operates by iteratively partitioning the dataset into a predefined number of clusters, minimizing the intra-cluster variance by optimizing the distance between data points and their corresponding cluster centroids. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) adopts a density-centric approach, identifying clusters as regions of high data density separated by areas of sparsity, which renders it particularly effective for detecting irregularly shaped clusters and isolating noise. Gaussian Mixture Models (GMM) further enhance clustering flexibility by assuming that the data distribution is composed of multiple Gaussian components, allowing for probabilistic assignment of data points to clusters and accommodating more complex data structures compared to K-Means. These algorithms collectively facilitate the exploration and organization of unlabeled visual data, enabling applications such as image categorization, anomaly detection, and pattern recognition.

2.3 Front-End Development Technology

Front-end development technology is primarily concerned with the construction of user interfaces (UI) and the enhancement of user experiences (UX) for web and mobile applications. Over time, the architecture of front-end development has undergone a significant transformation, progressing from rudimentary implementations using HTML, CSS, and JavaScript to more advanced, component-based frameworks. Contemporary front-end development practices adhere to established design patterns such as the Model-View-Controller (MVC) and Component-Based Architecture (CBA), which are instrumental in improving the maintainability, scalability, and overall robustness of applications.

Design Patterns and Architectures:

The most prevalent architecture in modern front-end development is the CBA, which structures the user interface into discrete, reusable, and modular components. This paradigm is foundational to popular frameworks like React.js, Vue.js, and

Angular. Each component operates within a well-defined lifecycle and can be developed, tested, and managed independently, thereby promoting reusability, simplifying state management, and facilitating collaborative development efforts.

Another pivotal design pattern is Flux and its implementation, Redux, which are specifically designed to handle global application states in complex, large-scale applications. By enforcing a unidirectional data flow, these patterns mitigate the intricacies associated with state management, enhance predictability, and streamline debugging processes. This approach ensures that state changes are transparent and traceable, thereby reducing the likelihood of errors and improving the overall stability of the application.

These architectural patterns and frameworks collectively empower developers to build dynamic, scalable, and maintainable front-end systems that meet the evolving demands of modern web and mobile applications.

Cutting-Edge Technologies and Frameworks:

React.js: A widely used JavaScript library for building interactive UIs. React employs a virtual DOM to improve performance and maintainability.

Vue.js: A progressive JavaScript framework that is lightweight and highly adaptable for small and large-scale applications.

Angular: A TypeScript-based framework developed by Google, featuring two-way data binding, dependency injection, and modular development.

WebAssembly: A low-level bytecode format that enables high-performance execution of languages like Rust and C++ on the web.

Tailwind CSS and Bootstrap: Utility-first and component-based CSS frameworks that streamline UI development with responsive designs.

Modern front-end development also embraces Progressive Web Apps (PWAs), which enhance the user experience by offering offline functionality, push notifications, and improved performance. Additionally, Server-Side Rendering (SSR) and Static Site Generation (SSG) are techniques that improve performance and SEO, implemented in frameworks like Next.js and Nuxt.js.

2.4 Back-End Development Technology

Back-end development serves as the foundational infrastructure of an application, responsible for managing business logic, data storage, authentication, and the seamless interaction between the client and server. Contemporary back-end architectures emphasize microservices, serverless computing, and containerization to achieve superior scalability, flexibility, and operational efficiency.

Design Patterns and Architectures

The progression of back-end systems has given rise to a variety of architectural paradigms aimed at enhancing maintainability, scalability, and performance:

Monolithic Architecture: A conventional approach where all application components are tightly integrated into a single codebase. While straightforward to develop and deploy, this architecture often struggles with scalability and adaptability as the application grows.

Microservices Architecture: A decentralized model where the application is divided into smaller, independent services that communicate via APIs. Each service can be developed, deployed, and scaled independently, fostering agility and resilience in complex systems.

Serverless Architecture: A cloud-native paradigm where developers write and deploy individual functions, which are automatically managed and scaled by cloud providers such as AWS Lambda, Azure Functions, and Google Cloud Functions. This model eliminates the need for server management, allowing developers to focus solely on code.

Event-Driven Architecture: A design pattern where microservices interact asynchronously through event-driven messaging systems like Apache Kafka and RabbitMQ. This approach ensures high scalability, responsiveness, and loose coupling between services.

Cutting-Edge Technologies and Frameworks:

Modern back-end development is powered by a suite of advanced technologies and frameworks designed to meet the demands of scalable, high-performance applications:

Node.js: A JavaScript runtime built on Chrome's V8 engine, renowned for its efficiency in handling asynchronous, event-driven programming, making it ideal for real-time applications.

Django and Flask: Python-based frameworks that offer rapid development capabilities, scalability, and robust support for building RESTful APIs and web applications.

Spring Boot: A Java-based framework tailored for microservices development, featuring built-in dependency injection, cloud integration, and simplified configuration.

GraphQL: A query language for APIs that enables clients to request precisely the data they need, reducing over-fetching and under-fetching issues commonly associated with REST APIs.

Containerization and Orchestration: Technologies like Docker and Kubernetes streamline the deployment, scaling, and management of applications in cloud environments, ensuring consistency across development and production stages.

NoSQL Databases: Databases such as MongoDB and Cassandra excel at handling unstructured or semi-structured data, offering greater flexibility and scalability compared to traditional relational databases.

CHAPTER 3. METHODOLOGY

3.1 Problem

In the realm of modern deep learning research, a persistent and critical challenge, despite the extraordinary progress in the design of neural network architectures, is the intrinsic opacity of model decision-making processes. Although deep convolutional neural networks (CNNs) have exhibited unparalleled efficacy in applications such as facial recognition, the mechanisms underlying their predictions remain predominantly inscrutable. Conventional methodologies depend on feature extraction achieved through successive layers of convolutions and non-linear activations; however, the pathway by which a model derives its ultimate classification decision remains elusive and difficult to interpret.

The central objective of this study is to advance the interpretability of facial classification models, particularly within the domain of deep learning-driven identity recognition. Moving beyond the conventional treatment of neural networks as opaque entities, this research aims to dissect the decision-making framework by scrutinizing intermediate feature representations and examining clustering behaviors embedded within the model's learned feature spaces. The FaceScrub dataset forms the cornerstone of this exploration, comprising facial images of 530 distinct individuals. After rigorous preprocessing and quality assurance procedures, the dataset is refined to a final working corpus of 488 unique identities, ensuring a robust foundation for analysis.

Processing workflow of this project:

- Obtain FaceScrub dataset (530 people)
- Preprocess dataset - Filter valid images (488 people)
- Train CNN model (MobileNetV2, EfficientNetB0, ResNet-50) on face classification
- Extract 2D feature maps from the convolutional layer

- Apply K-Means/MiniBatchKMeans/GMM clustering (k = 2 to 16 or 32)
- Select optimal k (16 per identity)
- Compute centroids (488 * 16 = 7808)
- Convert samples to distance vectors (samples, 7808)
- Train classification models (KNN, Logistic Regression, MLP, Random Forest, Naïve Bayes)
- Evaluate performance on the test set
- Deploy model in a mobile application
- Detect face in real-time
- Extract feature maps, compare to centroids
- Generate heatmap for explainability
- Display most similar identities

3.2 Models

3.2.1 MobileNetV2

MobileNetV2, an advancement over its predecessor MobileNetV1, is a deep learning model designed to optimize computational efficiency while maintaining high accuracy in image classification and feature extraction tasks. This model, developed by Google, introduces a significant architectural innovation known as the inverted residual block with linear bottlenecks, which enhances representational power while mitigating memory and processing demands. The primary objective of MobileNetV2 is to achieve superior performance on mobile and edge devices by leveraging depthwise separable convolutions and lightweight neural structures.

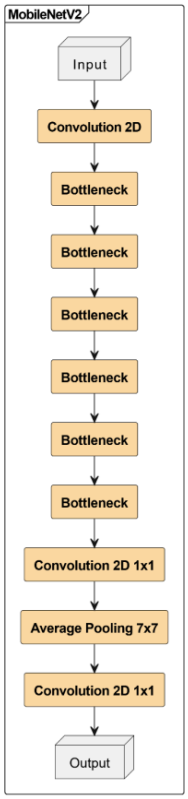

Figure 3.1 Architecture of MobileNetV2

Processing Workflow:

The underlying mechanism of MobileNetV2 is driven by a structured sequence of computational steps that ensure efficient information propagation while maintaining the expressive power of deep convolutional neural networks. The model processes an input image by passing it through an initial convolutional layer, which

extracts fundamental low-level features such as edges and textures. The resultant feature maps are subsequently refined through multiple iterations of inverted residual blocks, each containing an expansion phase, depthwise convolution, and linear projection. The utilization of ReLU6 activations enhances robustness against numerical instabilities, particularly in lower-precision computations. The final layers incorporate global average pooling to condense spatial information, followed by a fully connected layer and a softmax activation function, which facilitates categorical predictions.

3.2.2 EfficentNetB0

EfficientNetB0 represents a paradigm shift in convolutional neural network architectures, wherein model scaling is systematically optimized using compound coefficient-based scaling strategies. Introduced by Google, EfficientNetB0 employs a novel methodology for balancing depth, width, and resolution in a structured and computationally efficient manner. This model leverages MBConv blocks, a variant of depthwise separable convolutions, which significantly enhances performance while maintaining a reduced computational footprint.

The foundational structure of EfficientNetB0 is derived from the MobileNetV2 framework but extends its capabilities through an enhanced compound scaling approach. The model is architected with multiple MBConv blocks, each of which integrates an expansion phase, depthwise convolution, and squeeze-and-excitation (SE) attention mechanism. The SE blocks dynamically recalibrate channel-wise feature responses, thereby improving the model's capacity to capture intricate patterns and dependencies.

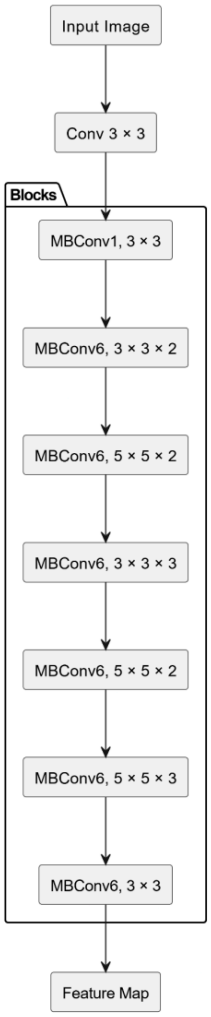

Figure 3.2 Architecture of EfficientNetB0

Input Layer: The architecture starts with an input image, typically of size (H × W × 3), where 3 represents the RGB color channels.

Initial Convolution: A 3×3 convolutional layer extracts basic features from the input image.

MBConv Blocks: The architecture is divided into several blocks:

Block 1: A single MBConv1, 3×3 layer.

Block 2: Two MBConv6, 3×3 layers.

Block 3: Two MBConv6, 5×5 layers.

Block 4: Three MBConv6, 3×3 layers.

Block 5: Two MBConv6, 5×5 layers.

Block 6: Three MBConv6, 5×5 layers.

Block 7: A single MBConv6, 3×3 layer.

Feature Map Output: The processed feature maps are passed to the final stage, which can be used for classification or further downstream tasks.

3.2.3 ResNet-50

The Residual Network (ResNet-50) is a convolutional neural network (CNN) that has emerged as one of the most robust architectures in deep learning, owing to its sophisticated design incorporating residual learning. Unlike conventional CNNs, which often suffer from the vanishing gradient problem in deep architectures, ResNet-50 employs skip connections, allowing gradients to propagate effectively through numerous layers, thereby enhancing convergence stability and optimizing representational learning.

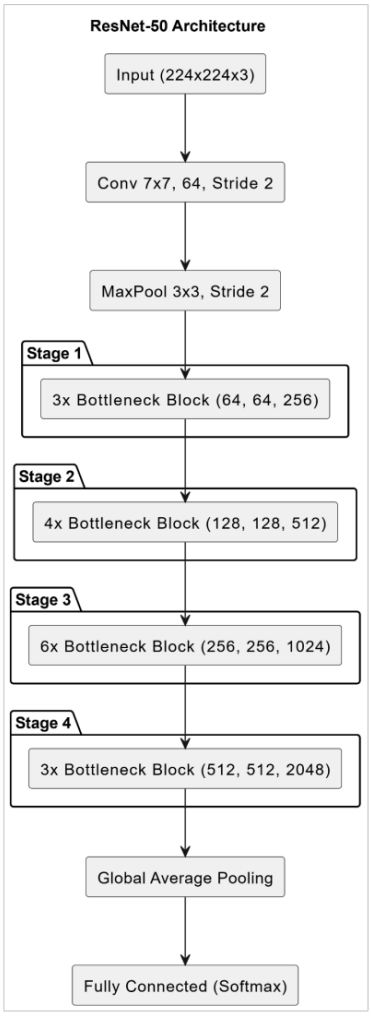

Figure 3.3 Architecture of ResNet-50

ResNet-50 consists of 50 layers, predominantly comprising convolutional layers interspersed with batch normalization, ReLU activation functions, and identity residual connections. The architecture is structured into five convolutional stages, where each stage incorporates multiple residual blocks, enabling feature reusability and facilitating deeper network training.

Processing Workflow:

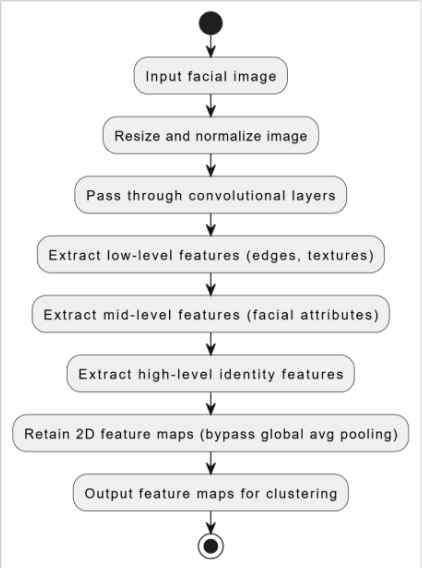

Figure 3.4 Processing steps of ResNet-50

The process of extracting facial embeddings using ResNet-50 follows a well-defined sequence:

1. A given facial image is preprocessed and resized to match the network's input dimensions.

2. The image is propagated through a series of convolutional layers, each extracting hierarchical features ranging from low-level edges to high-level identity characteristics.
3. Instead of applying global average pooling, which collapses spatial information, the feature maps from the last convolutional layer are retained, ensuring the preservation of crucial positional attributes.
4. These 2D feature maps are subsequently utilized for centroid-based representation learning.

3.2.4 KMeans and MiniBatchKMeans

K-Means clustering represents a fundamental unsupervised learning algorithm utilized to partition data into k distinct clusters based on feature similarity. The centroid-based nature of this algorithm renders it particularly suitable for grouping facial embeddings, ensuring that identity representations are compressed into a structured, interpretable format.

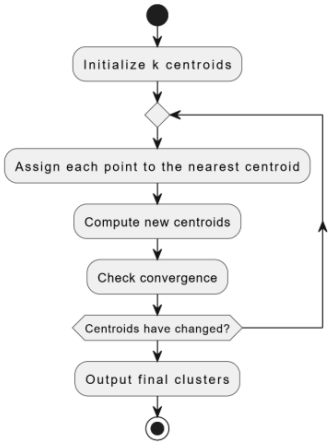

Figure 3.5 Processing steps of Kmeans

Processing Steps:

Initialization: k cluster centroids are randomly selected from the dataset.

Assignment: Each data point is allocated to the closest centroid based on Euclidean distance.

Update: The centroids are recalculated as the mean of all points belonging to the respective cluster.

Iteration: Steps 2 and 3 are repeated until centroids converge, ensuring minimal intra-cluster variance.

Table 3-1 Compare K-Means and MiniBatchKMeans

Aspect	K-Means	MiniBatchKMeans
Algorithmic Paradigm	Operates in a deterministic, full-batch mode wherein each iterative cycle exhaustively reassigns all data points to the nearest centroid, effectuating a global recalibration of cluster centroids based on the entirety of the dataset.	Employs a stochastic, mini-batch strategy that selects random subsets of the data per iteration, thereby incrementally updating centroids through a series of localized, probabilistic adjustments rather than a comprehensive reassessment.
Computational Complexity	Exhibits computational complexity proportional to the total number of data instances (n), clusters (k), and dimensionality (d), typically on the order of $O(n \cdot k \cdot d)$ per iteration, which can be computationally intensive for voluminous datasets.	Significantly reduces per-iteration computational demands by operating on a smaller subset of b instances (with $b \ll n$), yielding an approximate complexity of $O(b \cdot k \cdot d)$; this trade-off accelerates convergence at the potential cost of precision in the final clustering outcome.

Convergence Dynamics	Generally converges to a local optimum via a rigorous, iterative reallocation process, yet this method can incur substantial latency due to the exhaustive data utilization in each iteration.	Converges in a more expeditious, albeit stochastic, manner; while the mini-batch approach typically attains near-optimal solutions more swiftly, it may exhibit greater variability and a marginally inferior attainment of the objective function's minimum.
Memory Footprint	Necessitates the simultaneous in-memory presence of the entire dataset, thereby imposing significant memory constraints and limiting its applicability to extremely large-scale problems.	Demonstrates enhanced memory efficiency by processing discrete, manageable data batches sequentially, which alleviates the constraints associated with large datasets and facilitates scalable implementation in memory-limited environments.

Illustration for this topic:

For a dataset of 488 identities, the optimal k=16 is determined. This results in 7808 centroids, where each individual's feature representation is transformed into a distance vector, indicating proximity to the nearest clusters, thereby creating a highly structured classification-ready dataset.

3.2.5 Gaussian Mixture Models – GMM

A Gaussian Mixture Model (GMM) is a probabilistic model used for clustering and density estimation. It assumes that the data is generated from a mixture of several

Gaussian distributions, each with its own mean and covariance. The probability density function of a GMM with K components is given by:

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$

where:

π_k are the mixture weights (summing to 1),

$N(x|\mu_k, \Sigma_k)$ is a Gaussian distribution with mean μ_k and covariance Σ_k

K is the number of Gaussian components.

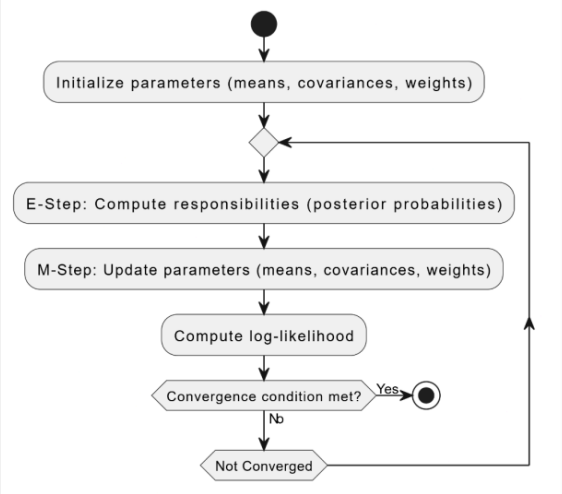

Figure 3.6 Processing steps of GMM

Processing Steps:

Initialization: The model starts by randomly initializing K Gaussian components, each with a mean, covariance matrix, and weight. These parameters are usually initialized using techniques like K-Means clustering or random selection.

Expectation (E-Step): For each data point, the algorithm calculates the responsibility that each Gaussian component has for generating that point. This is done by computing the posterior probability of the data point belonging to each component based on the current parameter estimates.

Maximization (M-Step): The parameters of each Gaussian component (mean, covariance, and weight) are updated using the weighted sum of data points assigned to that component. This step ensures that the estimated parameters maximize the likelihood of the observed data.

Iteration: Steps 2 (E-Step) and 3 (M-Step) are repeated until the algorithm converges. Convergence occurs when the log-likelihood of the data stops improving significantly, indicating that the model has stabilized and found an optimal distribution of clusters.

3.2.6 K-nearest neighbors – KNN

K-Nearest Neighbors operates on a non-parametric classification paradigm, wherein an input sample is classified based on its proximity to labeled training points. It is particularly suitable for this study, as the dataset comprises distance vectors, making similarity-based classification inherently effective.

Processing steps:

Step 1. Determine K value: K value is the number of nearest neighbor data points that we will consider.

Step 2. Calculate distance: Calculate the distance from the new data point to all data points in the training data set. Common distance measures include:

- Euclidean distance (L2 norm):

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

- Manhattan distance (L1 norm): measures the absolute differences between coordinates.

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- Minkowski distance (Generalized form):

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{\frac{1}{p}}$$

where p and q are two points in n-dimensional space.

Step 3. Sort: Arrange the data points in increasing order of distance.

Step 4. Classification or regression:

• Classification: Select the class that appears most frequently in the K nearest data points as the class of the new data point.

• Regression: Calculate the average of the values of the K nearest data points as the predicted value for the new data point.

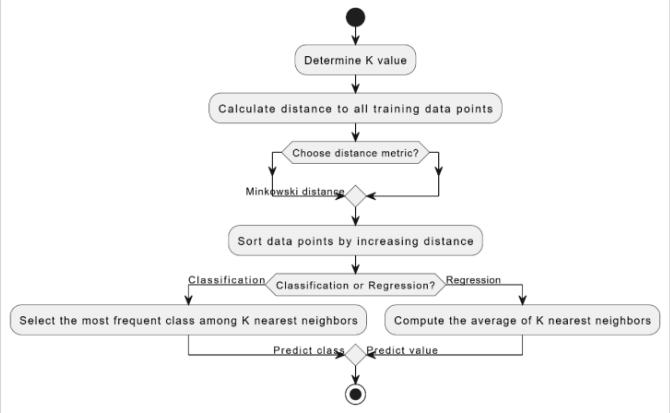

Figure 3.7 Processing steps of KNN

Illustration for this topic:

In the provided KNeighborsClassifier configuration, the parameter n_neighbors=10 specifies that only the 10 nearest neighbors are considered for the final decision. The model assigns weights to these neighbors based on their distance, giving more influence to closer points. Additionally, it uses all available CPU cores (n_jobs=-1) to speed up computation and applies the Euclidean distance metric (p=2) to measure similarity.

3.2.7 Logistic Regression

Logistic Regression, despite its nomenclature, functions as a linear classification algorithm rather than a regression model. It is built upon the principles of statistical learning, mapping input feature vectors to categorical output labels via a logistic (sigmoid) function, which constrains predictions within the range [0,1]. This characteristic makes it particularly suitable for binary classification, though it can be extended to multi-class scenarios via techniques such as one-vs-rest (OvR) classification.

The hypothesis function of Logistic Regression is defined as follows:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where θ represents the weight parameters of the model, and x is the input feature vector. The model is optimized using maximum likelihood estimation (MLE), with the cost function formulated as the log loss:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

where m represents the number of training samples, and $y^{(i)}$ denotes the ground-truth label.

Processing workflow:

- The input dataset, comprising distance vectors of shape $(samples, 7808)$, is first normalized to ensure numerical stability.

- The weight vector θ is initialized randomly or using a heuristic strategy.
- Each sample is passed through the sigmoid activation function, generating probability estimates for each identity class.
- The cost function is iteratively minimized using gradient descent or its variants (e.g., stochastic gradient descent).
- The final classification decision is made based on a thresholding mechanism, assigning labels according to the highest probability output.

Illustration for this topic:

Consider an input sample representing a facial image, transformed into a distance vector through deep feature extraction and centroid clustering. The logistic regression model computes a probability distribution over 488 identity labels, assigning the image to the class with the highest probability. If the highest probability is 0.92 for person X, the model confidently classifies the image as belonging to that individual.

3.2.8 Support Vector Machine

Support Vector Machines (SVMs) belong to the family of discriminative classifiers that construct optimal hyperplanes for decision boundaries. Unlike Logistic Regression, which models probabilities, SVMs operate by identifying a maximum margin between class-separated data points, ensuring robust generalization.

For a given dataset $D = \{(xi, yi)\}_{i=1}^{m}$, where x_i represents the feature vector and y_i is the corresponding class label, the optimal hyperplane is determined by maximizing the margin:

$$\max_{w,b} \frac{2}{\|w\|}$$

subject to the constraint:

$$y_i(w^T x_i + b) \geq 1, \forall i \in [1, m]$$

where w is the normal vector to the hyperplane, and b represents the bias term. In cases where data is non-linearly separable, a kernel trick is employed, transforming input space into a higher-dimensional feature space.

Operational Mechanism:

- Each input sample is mapped to a high-dimensional space using either a linear or kernel-based transformation.
- A hyperplane is identified that maximizes the margin between identity classes.
- Misclassified samples are penalized using a hinge loss function, which ensures robustness against outliers.
- The optimal hyperplane is iteratively adjusted using quadratic programming or stochastic gradient descent.

3.2.9 Decision Tree

Decision Trees (DTs) constitute a rule-based classification algorithm that recursively partitions input space into hierarchical decision nodes. Unlike linear classifiers such as Logistic Regression and SVM, Decision Trees create branching structures where decisions are made based on thresholded feature values.

Structural Overview and Algorithmic Process

A Decision Tree is composed of three fundamental components:

- Root Node: Represents the entire dataset, initiating recursive splitting.
- Internal Nodes: Intermediate decision points where feature-based comparisons are performed.
- Leaf Nodes: Terminal nodes corresponding to final class labels.

The algorithm employs an information gain-based splitting criterion, selecting features that maximally reduce entropy:

$$IG = H(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} H(D_i)$$

where $H(D)$ represents the entropy of dataset D and D_i denotes the partitioned subset.

Processing Workflow:

- The dataset is analyzed to determine the most informative feature using information gain.
- A decision threshold is established, splitting data into branches.
- Recursive partitioning is performed until homogeneity is achieved in each leaf node.
- A test sample is classified by traversing the tree from root to leaf.

3.2.10 Multi-layer Perceptron – MLP

The Multi-Layer Perceptron (MLP) is a fully connected neural network that maps input features to output labels through multiple hidden layers, leveraging non-linear transformations to improve decision boundaries.

This multilayer perceptron (MLP) architecture is meticulously designed for a classification task spanning 488 distinct categories. It commences with a 512-neuron dense layer, activated by ReLU, ensuring efficient feature extraction while mitigating gradient vanishing. Dropout (30%) follows to enhance generalization by preventing overfitting. A subsequent 256-neuron dense layer, also activated by ReLU, refines feature hierarchies, followed by another Dropout layer to maintain robustness. The 128-neuron dense layer further condenses representations, preserving non-linearity through ReLU activation. Ultimately, a 488-neuron softmax layer transforms logits into a probability distribution, facilitating categorical differentiation. This architecture harmonizes depth, regularization, and activation choices to optimize performance in high-dimensional classification.

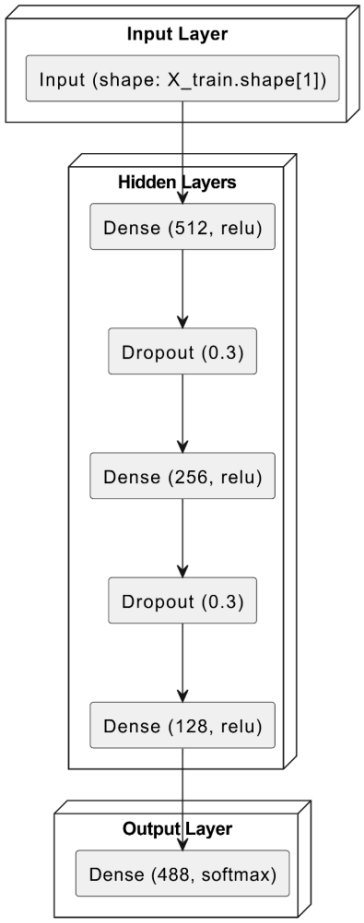

Figure 3.8 Architecture of MLP

3.2.11 Random Forest

Random Forest is an ensemble learning algorithm that constructs multiple decision trees, averaging their predictions to enhance classification robustness.

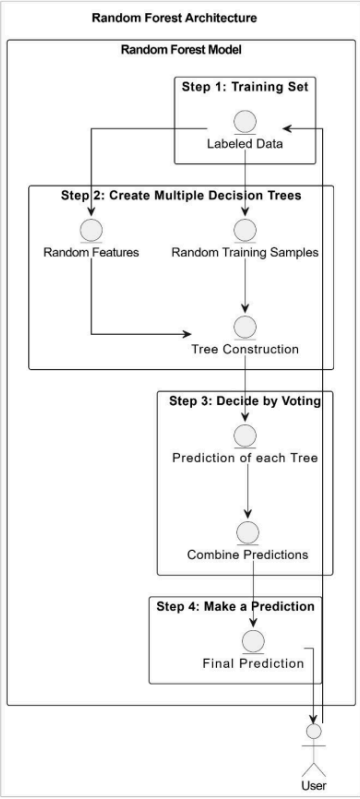

Figure 3.9 Processing workflow of Random Forest

Processing Workflow:

Step 1. Training Set

Data Set: This is a large set of labeled data (e.g. cat/dog images, spam/non-spam emails), which is used to train the model.

Step 2. Create Multiple Decision Trees

Random Training Samples: From the initial training set, we randomly take a portion of the data to train each decision tree. This random sampling increases the diversity of the trees.

Random Features: When building each tree, we randomly select only a subset of the features to split the data at the nodes. This helps to reduce the possibility of overfitting and increase the generalization ability of the model.

Tree Construction: Each decision tree is built using the traditional decision tree algorithm, by finding the best split points to separate the data.

Step 3. Decide by Voting

Prediction of each tree: When there is a new data to predict, each decision tree will make a separate prediction.

Combine predictions: Based on the predictions of all trees, we use the voting method to make the final decision.

- Classification: The class predicted most by the trees will be the final result.
- Regression: The average value of the predictions of the trees will be the final result.

Step 4. Make a prediction

Result: The final result is the prediction of the Random Forest model for the new data.

Illustration for this topic:

For an input distance vector, 100 decision trees independently classify the identity, and the mode of their predictions determines the final class label.

3.3 Dataset – FaceScrub

3.3.1 Overview about dataset

The dataset utilized in this study is the FaceScrub dataset, a meticulously assembled repository of facial imagery that has gained widespread recognition within the domain of computer vision, especially for applications involving facial recognition and classification. This dataset encompasses a diverse array of images featuring actors and actresses, each annotated with comprehensive metadata. Such metadata includes unique image identifiers, precise bounding box coordinates delineating facial regions, cryptographic hash values for data integrity verification, and the corresponding URLs indicating the origins of the images.

Figure 3.10 Request for file facescrub_actors.txt and facescrub_actress.txt

To obtain the dataset, a formal request was submitted to the custodians of FaceScrub, upon which two text files, facescrub_actors.txt and facescrub_actress.txt, were received. These files serve as structured indices for the dataset, enumerating a comprehensive list of images with corresponding attributes, as illustrated in the following structure:

Table 3-2 Structure of facescrub_actors.txt and facescrub_actress.txt

Name	image_id	face_id	url	Bbox	sha256
Alley Mills	114640	51940	http://images.xxx.	219,195,376,352	cb0c0897adfeac43656a

Each entry within these files consists of six fundamental components:

- name: The full name of the celebrity.
- image_id: A unique numerical identifier assigned to the image.
- face_id: A distinct identifier associated with the detected face in the image.
- url: The web link to the image's original source.
- bbox (bounding box coordinates): The pixel coordinates (x_min, y_min, x_max, y_max) that define the region enclosing the detected face.
- sha256: A SHA-256 cryptographic hash value that ensures image integrity verification post-download.

3.3.2 Data Collection via Web Scraping

To systematically retrieve and store the images specified in the dataset files, a Python script was developed. This script performs the following key operations:

Establishing an HTTP session with controlled request headers to simulate a legitimate web browser and mitigate potential access restrictions imposed by servers.

Reading the dataset metadata files to extract URLs and relevant attributes for each image.

Sending HTTP requests to fetch images, ensuring appropriate error handling mechanisms to address scenarios such as broken links, timeouts, and redirections.

Validating the integrity of downloaded images by computing their SHA-256 hash and comparing them with the expected values to detect corruption or unauthorized modifications.

Storing images in a structured directory format, organizing them based on actor/actress names while preserving metadata for further processing.

Figure 3.11 A raw sample image

Optionally cropping face regions using bounding box coordinates if a facial recognition-based experiment is required.

Figure 3.12 A cropped image

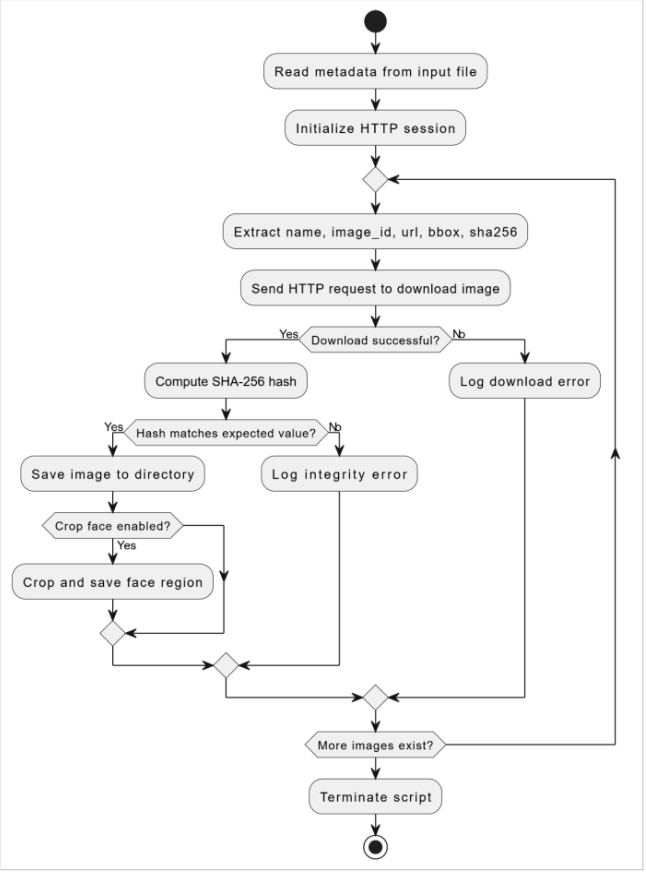

Figure 3.13 Flow of crawling FacesScrub dataset

Following the data acquisition phase, the initial dataset encompassed 530 actors and actresses, totaling over 43,000 images. However, a significant imbalance was observed in the distribution of images per individual, with some subjects having as few as 10 to 20 images. To ensure a more uniform representation and improve the

quality of the dataset, a rigorous data preprocessing and filtering procedure was conducted. This process involved retaining only individuals with an image count ranging between approximately 40 and 130.

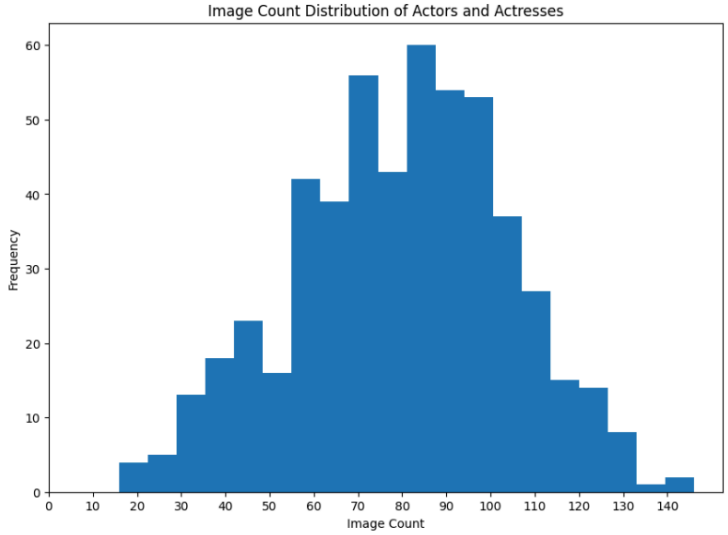

Figure 3.14 Distrubution of the number of images per individuals before processing

As a result of this refinement, the final curated dataset consists of 488 individuals, collectively amounting to approximately 32,000 images. Each individual is now represented by a balanced subset, containing between 50 and 70 images, thereby enhancing the dataset's suitability for subsequent analysis and model training.

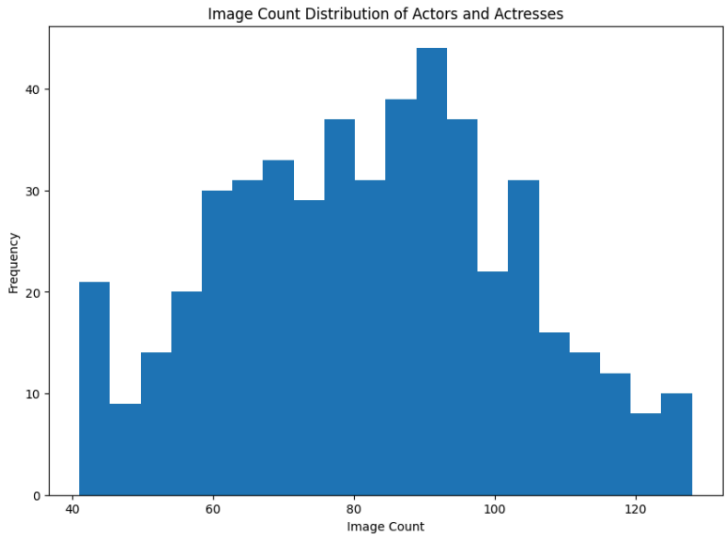

Figure 3.15 Distrubution of the number of images per individuals after processing

3.4 Experiment

3.4.1 Training Methodology with CNN models

Data Preprocessing & Augmentation:

- Rescaling pixel values to [0,1] for normalization.
- Augmentations: rotation, translation, shear, zoom, flipping, brightness variation, and channel shifting.

Model Architecture & Transfer Learning:

- Uses ImageNet-pretrained weights for feature extraction.
- Keeps convolutional base frozen in initial phases.
- Adds Global Average Pooling (GAP), dense layers (ReLU), and a softmax classifier for multi-class classification.

Optimization & Regularization:

- Optimizer: Adam (learning rate = 0.001).
- Loss function: Categorical cross-entropy.
- Regularization: Dropout (0.5) to prevent overfitting.
- Early stopping & ReduceLROnPlateau for adaptive learning rate adjustments.

Training & Evaluation:

- Batch size: 32, epochs: 20.
- Dynamic steps per epoch to match dataset constraints.
- Evaluates train, validation, and test accuracy.

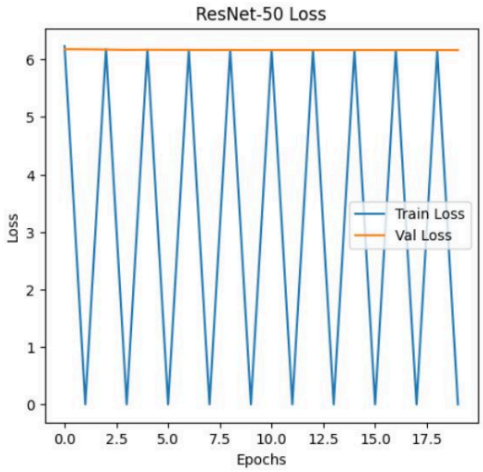

Figure 3.16 Loss training of EfficientNetB0

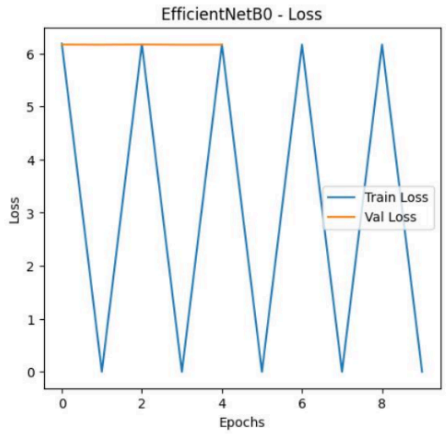

Figure 3.17 Loss training of EfficientNetB0

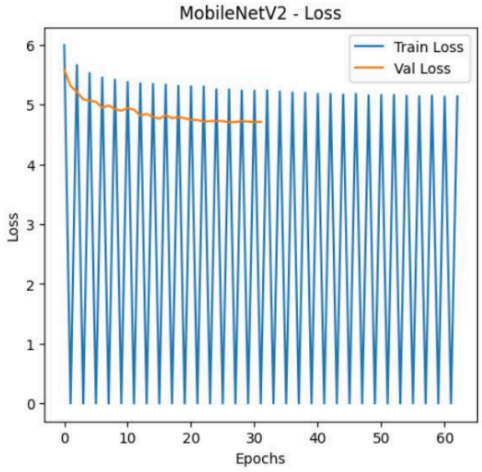

Figure 3.18 Loss training of MobileNetV2

Table 3-3 Results of training CNN models

Model	Train Accuarcy (%)	Val Accuarcy (%)	Test Accuarcy (%)
EfficientNetB0	2.1	3.1	3.3
ResNet-50	2.2	3.2	3.4
MobileNetV2	5.87	9.4	9.13

3.4.2 Clustering with Kmeans, MiniBatchKmean and GMM

Data Preprocessing

- Load the feature maps and labels extracted from CNN models.
- Extract labels and determine the number of unique classes (persons).
- Reshape the feature maps into a 2D array for clustering.
- Apply Principal Component Analysis (PCA) to reduce dimensionality while preserving important information.

Finding the Best k for Each Class

- Define a range of k values (e.g., $k \in [2, 16]$).
- For each unique class (person):
    - Extract feature maps corresponding to the person.
    - Iterate through different values of k and fit clustering models using:
        - K-Means: Compute average intra-cluster distance (lower is better).
        - MiniBatchKMeans: Similar to K-Means but processes data in batches.
        - Gaussian Mixture Model (GMM): Compute average Mahalanobis distance or log-likelihood (higher is better).
    - Select the best k that minimizes the intra-cluster distance (for K-Means/MiniBatch) or maximizes the log-likelihood (for GMM).

Clustering Using the Best k

- For each class, fit a clustering model with the selected best *k*:
- K-Means: Assign each data point to the nearest centroid.
- MiniBatchKMeans: Efficiently update centroids in mini-batches.
- GMM: Assign each data point based on the probability of belonging to a Gaussian component.
- Store the trained models for future inference.

Compute Distance Vectors

- Extract the cluster centroids (for K-Means and MiniBatchKMeans) or Gaussian means (for GMM).
- Compute the Euclidean distance between each feature vector and all centroids.
- Store these distance vectors for further classification.

Evaluation and Visualization

- Plot the average distance to centroids vs *k* to validate clustering quality.
- Compare inertia (for K-Means/MiniBatchKMeans) and log-likelihood (for GMM) across different *k* values.
- Visualize the distribution of distances for different classes.

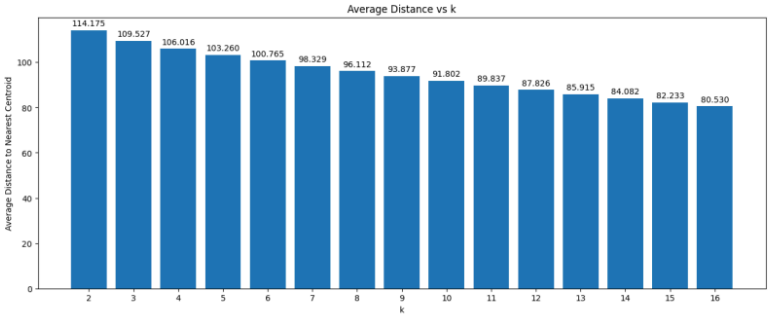

Figure 3.19 MiniBatchKMeans with k from 2 to 16

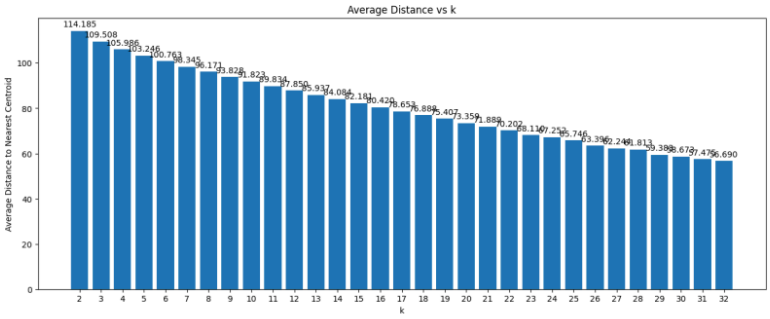

Figure 3.20 MiniBatchKMeans with k from 2 to 32

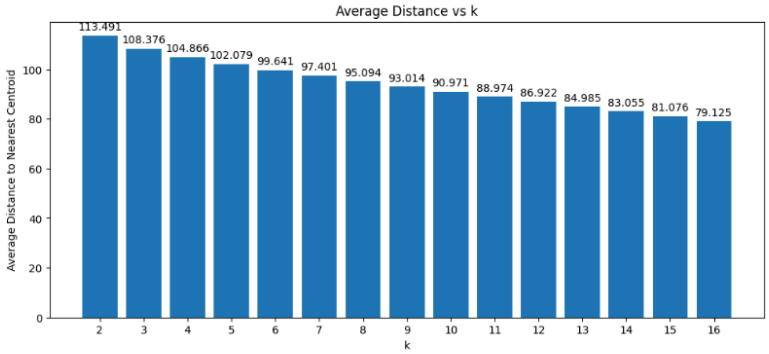

Figure 3.21 KMeans with k from 2 to 16

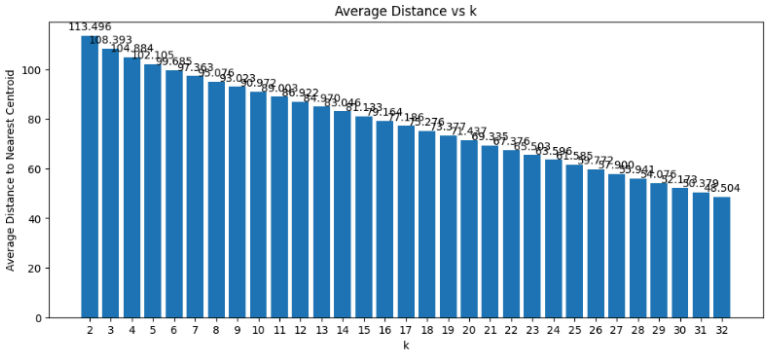

Figure 3.22 KMeans with k from 2 to 32

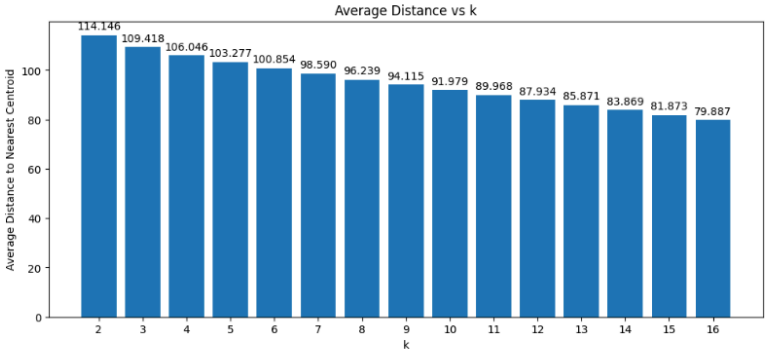

Figure 3.23 GMM with k from 2 to 16

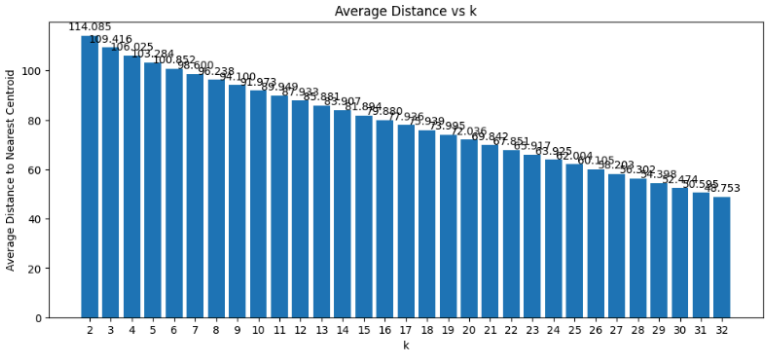

Figure 3.24 GMM with k from 2 to 32

Table 3-4 Results of K-Means, MiniBatchKMeans, and GMM

Clustering Methods	k Range (2-16) Best k	Avg Distance (2-16)	k Range (2-32) Best k	Avg Distance (2-32)
K-Means	16	79.125	32	48.504
MiniBatchKMeans	16	80.530	32	46.690
GMM	16	79.887	32	48.753

3.4.3 Classification Models

Feature Representation: Each data point is represented as a distance vector, capturing the distances to a set of centroids derived from feature extraction.

Preprocessing:

- Standardization ensures that all features have a uniform scale.
- Principal Component Analysis (PCA) reduces dimensionality, preserving important information while improving computational efficiency.
- Synthetic Minority Over-sampling Technique (SMOTE) addresses class imbalances by generating synthetic samples.

Model Selection & Training:

- Various machine learning models are trained using the processed distance vectors.
- Hyperparameter tuning (GridSearchCV) is performed to optimize performance.
- Models are evaluated on training, validation, and test sets to assess generalization.

Table 3-5 Results of Classification Models trained by distance vectors

Models	Train Accuarcy (%)	Val Accuarcy (%)	Test Accuarcy (%)
KNN	31.67	18.88	19.13
Random Forest	36.35	12.98	13.21
MLP	69.52	33.20	34.08
Naïve Bayes	3.13	2.50	2.33
Logistic Regression	74.80	33.13	33.10
SVM	21.27	11.70	11.69
Decision Tree	18.27	3.26	3.49

Training Accuracy (%): Measures how well the model fits the training data.

Validation Accuracy (%): Assesses the models' ability to generalize to unseen validation data, used for hyperparameter tuning.

Test Accuracy (%): Evaluates the models' performance on completely unseen data.

An alternative approach – training directly on feature maps instead of distance vectors – yielded superior performance. This suggests that preserving spatial feature relationships contributes to better classification results, highlighting the importance of feature map structures in deep learning interpretability.

Table 3-6 Results of Classification Models trained by feature maps

Models	Train Accuarcy (%)	Val Accuarcy (%)	Test Accuarcy (%)
KNN	52.78	36.34	36.31
Random Forest	66.35	31.92	33.41
MLP	77.93	56.09	56.72
Logistic Regression	85.48	67.53	67.85
SVM	87.12	49.98	49.95
Decision Tree	29.80	10.70	12.81

Conclusion:

Training with **distance vectors** boosts traditional models by structuring CNN features for better interpretability. However, **feature maps** preserve spatial information, leading to superior performance in deep learning. The choice depends on the balance between interpretability and accuracy.

CHAPTER 4. FRONT-END APPLICATION

4.1 Architecture and Technology

The front-end architecture of this project is meticulously crafted in alignment with contemporary web and mobile development paradigms, prioritizing scalability, maintainability, and operational efficiency. Employing a component-based architecture, the system is built using React Native as the foundational framework. This modular design philosophy decomposes the application into discrete, reusable components, fostering a streamlined development workflow, enhanced code reusability, and a clear separation of concerns.

4.1.1 Component-Based Architecture

A component-based architecture represents a paradigm in software engineering wherein the user interface is deconstructed into discrete, self-sufficient components. Each constituent element encapsulates its structural framework, functional logic, and behavioral attributes, thereby fostering reusability and autonomy from other segments of the application. This modular design philosophy augments scalability, maintainability, and the efficiency of debugging processes.

Within the framework of React Native, individual user interface elements—such as interactive buttons, input fields, and navigational menus—are conceptualized and implemented as distinct components. These components are subsequently orchestrated to construct intricate user interfaces. The architecture adheres to a hierarchical organization, wherein smaller, subordinate components are embedded within larger, parent components, thereby ensuring a lucid demarcation of responsibilities and concerns.

This architectural methodology empowers developers to implement targeted modifications to specific components without necessitating alterations to the broader application infrastructure. Furthermore, it optimizes performance by leveraging React's virtual DOM to facilitate efficient rendering and re-rendering mechanisms. The inherent modularity of this approach also enhances collaborative workflows

within development teams, as contributors can concurrently develop and refine independent components without interdependencies.

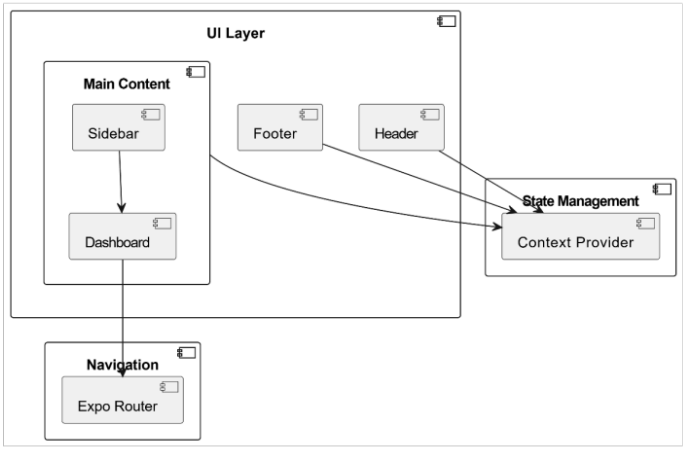

Figure 4.1 Diagram of Component-Based Architecture

The image above illustrates the structure of a component-based architecture. At the core, individual components are responsible for specific functionalities, while higher-order components manage state and interactions. This approach promotes reusability and maintainability by ensuring that each component has a single responsibility.

4.1.2 React Native

React Native serves as the foundational framework for this project, facilitating the creation of cross-platform applications through a unified codebase. Rooted in the principles of React, React Native empowers developers to utilize JavaScript for coding while rendering native user interface components, thereby delivering a performance experience akin to that of fully native applications. A pivotal strength of React Native lies in its capacity to produce genuinely native applications while

preserving the adaptability and productivity characteristic of web development methodologies.

Figure 4.2 React Native Logo

This project capitalizes on React Native's declarative UI paradigm, wherein the interface is constructed using components that dynamically adapt in response to changes in the application's state. The framework's support for hot reloading substantially streamlines the development process, enabling developers to observe modifications instantaneously without the need to recompile the entire application.

Context API for State Management

Effective state management is a cornerstone of front-end development, and this project employs React's Context API to establish a centralized and streamlined state management mechanism. The Context API circumvents the inefficiencies of prop drilling, permitting components to access shared state without the necessity of propagating props through multiple layers of the component hierarchy. This methodology simplifies the handling of global states, including user authentication, theme preferences, and application-wide configurations.

By encapsulating state logic within context providers, the application retains a coherent and modular architecture, ensuring that components remain dedicated to their specific functionalities without incurring extraneous dependencies. Moreover, this strategy enhances performance by refining state updates and reducing superfluous re-renders.

Expo Router for Navigation

Navigation is a critical determinant of user experience, and this project incorporates Expo Router to orchestrate navigation with precision and fluidity. Expo Router extends the capabilities of React Navigation by introducing a file-based routing system reminiscent of Next.js, thereby offering a more intuitive framework for structuring screens and delineating navigation pathways.

With Expo Router, each screen is conceptualized as an independent route, simplifying the organization and maintenance of the application's navigation hierarchy. The inclusion of dynamic routing, nested routes, and deep linking further enriches the navigational experience. By harnessing the Expo ecosystem, the application achieves a seamless development and deployment workflow, thereby amplifying overall efficiency.

4.2 Virtual device

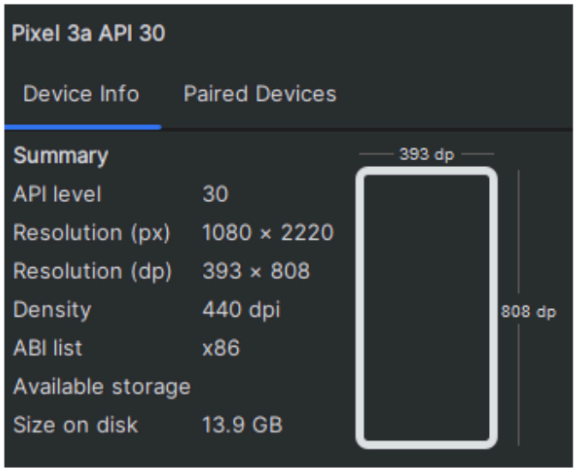

Figure 4.3 Configuration of Virutal device

The Pixel 3A API 30 virtual device is an emulator designed to replicate Google's Pixel 3A smartphone, allowing developers to test Android applications in a controlled environment. Running on Android API level 30, which corresponds to

Android 11 (R), this emulator provides access to the latest system features and enhancements.

4.3 User Interface

4.3.1 List of actors/actresses

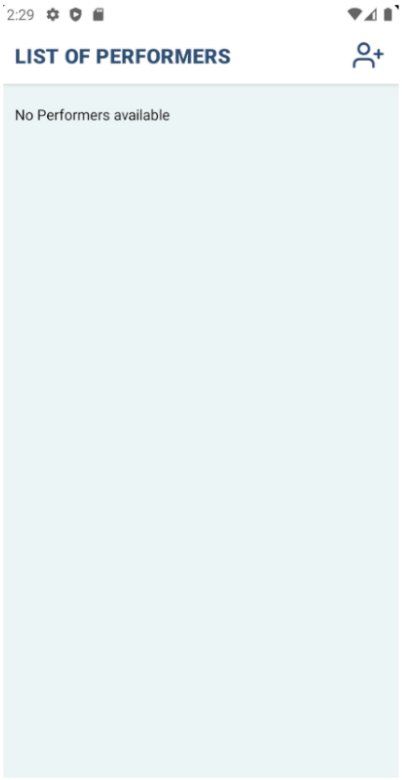

Figure 4.5 Empty List

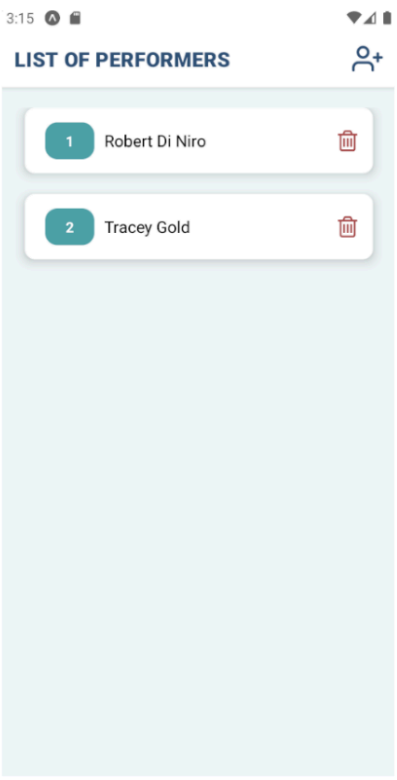

Figure 4.4 List of actors/actresses

In Figure 4.5, the list is empty, displaying a message that reads "No Performers available." The interface is minimalistic, featuring a clean background with a top

navigation bar that includes a user icon and an add (+) button, presumably for adding new performers.

In Figure 4.4, the second image the list populated with two performers: "Robert Di Niro" and "Tracey Gold." Each performer is displayed within a rounded card-style container, numbered sequentially

4.3.2 UI for uploading

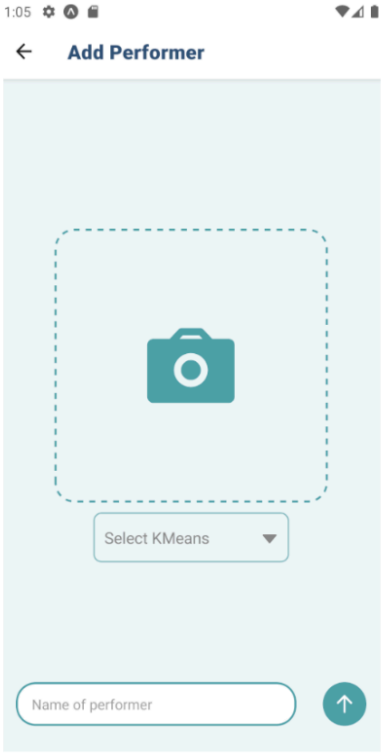

Figure 4.7 Before upload image

Figure 4.6 After upload image

In the **Error! Reference source not found.**, the interface prompts the user to add a performer by displaying a large dashed-bordered box with a camera icon, suggesting an option to upload a photo

In the **Error! Reference source not found.**, the user has added a performer. A photo of Robert Di Niro is displayed within the dashed-bordered box, now with a small red "X" in the top-right corner, indicating an option to remove or change the image.

4.3.3 UI for result

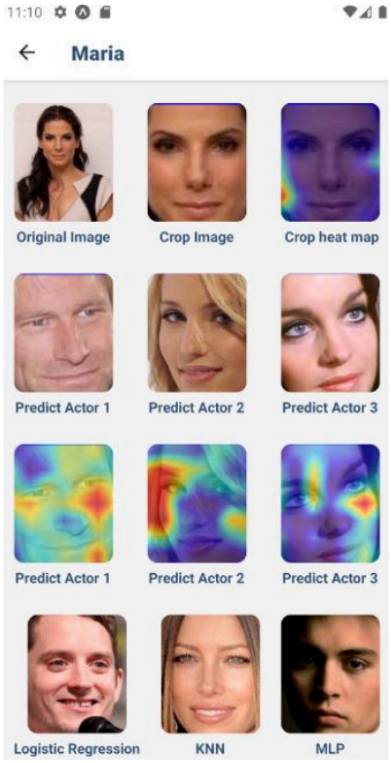

Figure 4.8 Result for input image

The image shows a mobile application screen displaying facial recognition or classification results for Robert Di Niro. At the top, the original image of the actor is shown alongside a heatmap version labeled "Input Image heat map," likely highlighting key facial features used for recognition.

Below, three additional images labeled "Predict Actor 1," "Predict Actor 2," and "Predict Actor 3" are also shown with heatmaps applied.

CHAPTER 5. BACK-END APPLICATION

5.1 Technology and Architecture

The Django Rest Framework, an extension of Django, is a versatile and efficient toolkit designed for constructing Web APIs. It streamlines the development of RESTful services by offering essential features like serialization, authentication, and viewsets. These tools empower developers to create scalable and robust APIs with ease, significantly reducing development time and effort.

Figure 5.1 Django Logo

OpenCV (Open Source Computer Vision Library) is an open-source library specializing in computer vision and machine learning. It includes a comprehensive collection of over 2,500 optimized algorithms for tasks such as face detection, object recognition, and motion tracking. In this application, OpenCV is leveraged to process images and detect faces, enabling functionalities like actor recognition and detailed image analysis.

Figure 5.2 OpenCV Logo

SQLite is a lightweight, file-based relational database management system, well-suited for smaller applications or development environments. It is renowned for

its simplicity and minimal setup requirements, making it an excellent choice for projects that do not demand the advanced features of larger database systems. In this context, SQLite is employed to store actor information and associated images, ensuring efficient data storage and retrieval. Its ease of integration and low resource consumption make it a practical solution for managing data in smaller-scale applications.

Figure 5.3 SQLite Logo

In Django, migrations are a mechanism for applying changes to database schemas based on modifications to models, such as adding or removing fields. They play a critical role in managing the evolution of the database structure, enabling developers to version-control database changes and apply them consistently across various environments.

Serializers in Django Rest Framework facilitate the conversion of complex data types, like Django models, into native Python data types that can be easily rendered into formats such as JSON or XML. They also support deserialization, allowing validated data to be converted back into complex types. This bidirectional functionality is essential for handling data input and output in APIs effectively.

Cross-Origin Resource Sharing (CORS) Middleware is a component in Django that manages CORS headers in responses. CORS is a browser security feature that restricts web pages from making requests to domains other than the one that served the page. By configuring CORS Middleware, the application can define which external domains are allowed to access its resources, enabling secure and controlled interactions with external clients or services.

5.2 Sequential diagram

5.2.1 Get list of actors and actresses

URL: /facedetect/api/v1:

Method: GET

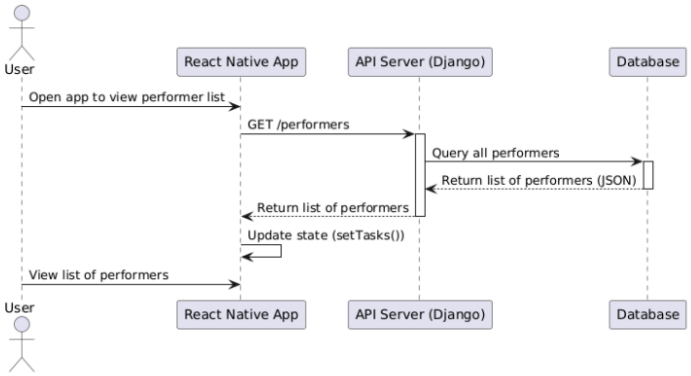

Figure 5.4 Sequential diagram for get list of actors and actresses

The sequential flowchart depicts the interaction of a React Native app, a Django-based API server, and a database. The procedure begins when a user opens the app and selects a list of performers. To query all performers, the React Native app sends a GET request to the API server's /performers endpoint. The Django server handles this request, obtains the list of performers from the database, and provides it in JSON format.

When the React Native app receives the JSON response, it modifies its state with a method like setTasks(), which stores the list of performers. Finally, the modified state is utilized to generate and present the list of performers to the user via the app interface. This sequence guarantees that the viewer may read the most recent list of performers saved in the database.

5.2.2 Get a specific actor/actress

URL: /facedetect/api/v1/{id}:

Method: GET

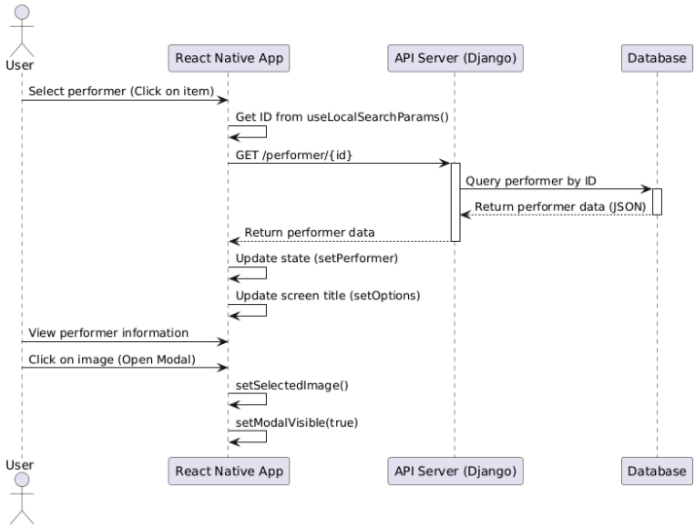

Figure 5.5 Sequential diagram for get a specific actor/actress

The sequential flowchart depicts the process of accessing extensive information about a certain performance in a React Native app while communicating with a Django API server and a database. The procedure begins when a user picks a performer by clicking on an item within the app. The app uses a method like useLocalSearchParams() to get the performer's ID and then sends a GET request to the API server at the /performer/(id) endpoint, where (id) is the exact performer's ID.

The Django server searches the database for the performer's information using the given ID and delivers the results in JSON format. The React Native app gets this data, adjusts its state with a method like setPerformer(), and modifies the screen title with setOptions() to match the performer's details.

The user may then see the performer's details on the screen. When the user clicks on a picture, the app sets the selected image using setSelectedImage() and displays a modal with setModalVisible(true), enabling the user to view the image in a bigger size. This sequence provides a seamless and engaging user experience while viewing extensive performer information and photographs.

Figure 5.6 Test GET method

5.2.3 Creat and save a new actor/actress

URL: /facedetect/api/v1/{id}:

Method: POST

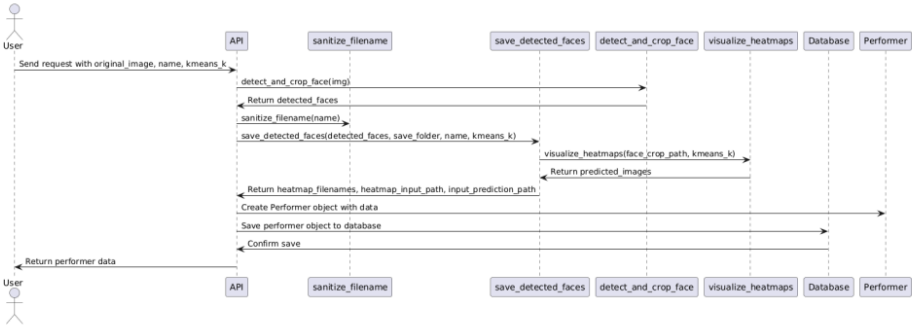

Figure 5.7 Sequential diagram for create and save a new actor/actress

The sequential flowchart depicts the process of adding a new performer to a system using a React Native app, an API server built with Django, and a database. The procedure starts when the user enters the performer's name and picks a photograph from the app. The React Native app prepares the performer's data and image in JSON format before sending a POST request to the API server via the /facedetect/api/v1/ endpoint.

The Django server verifies incoming requests to verify that the data is proper. The image is then processed to find faces using a method called detect_faces. Following that, the server extracts features from the picture using ResNet50, PCA, and KMeans algorithms. The performer's information is recorded to the database, while the photograph is saved in a specific directory. The server sends the newly minted performer ID to the React Native app.

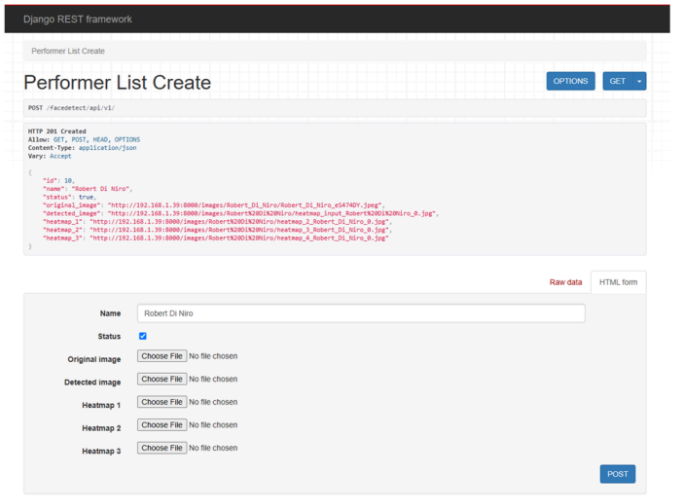

Figure 5.8 Test POST method

When the app receives the performer's ID, it updates the performer list by adding the new performer via a function such as addTask. The user is subsequently

shown the updated list, which includes the new performer. This process guarantees that fresh performer data and photographs are seamlessly integrated into the system, improving both the app's functionality and user experience.

5.2.4 Delete a specific actor/actresses

URL: /facedetect/api/v1/{id}:

Method: DELETE

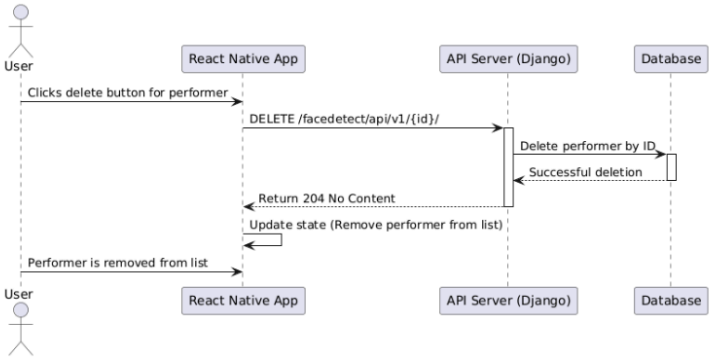

Figure 5.9 Sequential diagram for delete a specific actor/actress

The sequential flowchart depicts the process of eliminating a performer from a system using a React Native app, an API server built with Django, and a database. The procedure begins when a user taps the delete button for a certain performer in the app. The React Native app makes a DELETE request to the API server via the /facedetect/api/v1/(id)/ endpoint, where (id) is the performer's unique identification.

The Django server executes the request by removing the performer's record from the database using the given ID. After a successful deletion, the server sends a 204 No Content response to indicate that the action was successful. The React Native app then modifies its state to remove the removed performer from the list, ensuring that the change is reflected in the UI.

Finally, the user gets an updated list without the deleted performer, indicating that the performer was successfully removed from the system. This sequence guarantees that the app's performer data is managed efficiently and intuitively.

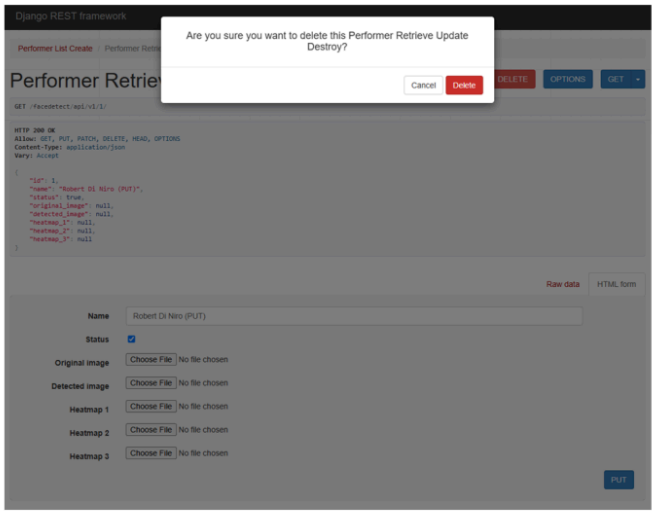

Figure 5.10 Test DELETE API

5.3 Performance

The mobile application was developed to perform face detection and classification, leveraging extracted feature maps to identify individuals based on their proximity to centroid representations. Performance evaluation was conducted on a device with the following configuration: Windows 11, GeForce RTX 3050, 16GB RAM, and a 512GB SSD. While the model inference primarily occurs on mobile hardware, preliminary testing was performed using a PC-based emulator to measure execution efficiency.

To assess the system's responsiveness and computational efficiency, the following performance metrics were measured under different scenarios:

Table 5-1 Benchmark Testing

Test Case	Metric	Result
Face Detection	Inference Time	65ms
Memory Usage	RAM Consumption	320MB
Model Loading Time	Cold Start	2.5s

CHAPTER 6. SUMMARY AND DIRECTIONS IN THE FUTURE

6.1 Summary

This study proposed an enhanced approach to face classification by leveraging centroid-based representation, thereby improving model interpretability in deep learning applications. The research utilized the FaceScrub dataset, preprocessed to include 488 individuals, and employed ResNet-50, MobileNetV2, EfficientNetB0 to extract high-dimensional feature maps without applying global average pooling, preserving spatial information. A clustering-based approach was implemented, determining optimal centroid representations per individual through K-means, leading to the construction of distance vectors used for classification. Various traditional machine learning models—including k-nearest neighbors, logistic regression, multi-layer perceptron, and random forest—were trained on these distance vectors to evaluate classification performance. Despite the methodological rigor and the integration of a mobile application for real-time face similarity detection, the model's accuracy remains suboptimal, suggesting the need for further refinements.

According to classification results in Table 3-3, CNN accuracy did not exceed 10%. In Table 3-5, MLP and LR demonstrated significantly higher accuracy with distance vectors, while Table 3-6 showed the highest results using feature maps. Thus, when the number of images per individual is limited (< 60, for example), CNNs tend to overfit, resulting in poor classification performance. The distance vector approach offers better interpretability and improved classification outcomes, while CNN feature classification markedly enhances accuracy. This issue falls within the context of limited data

Training with distance vectors boosts traditional models by structuring CNN features for better interpretability. However, feature maps preserve spatial

information, leading to superior performance in deep learning. The choice depends on the balance between interpretability and accuracy

Table 6-1 Self-assessment

Task	Percentage of complete
Finding and Crawling Data	100%
Processing raw data	100%
Research Deep Learning Model	100%
Finetuning Deep Learning Model	100%
Clustering	100%
Train Classification Model	100%
Improvement Model	100%
Design Architecture for Front-End	100%
Design Architecture for Back-End	100%
Write API Documents	100%
Summary and write a report	100%

6.2 Future Directions

To enhance performance, future work will explore advanced clustering techniques, such as hierarchical or density-based methods, to refine centroid selection and better capture intra-class variations. Additionally, integrating contrastive or self-supervised learning mechanisms may improve feature representations before clustering, reducing intra-class variance while maintaining inter-class separability. Optimizing feature map extraction by incorporating multi-scale information from different convolutional layers could also be beneficial. Beyond performance improvements, this framework has significant implications for national-scale applications, such as secure identity verification and forensic investigations, where explainable and reliable face classification models are imperative. Further research will also focus on expanding the system's capabilities to handle real-world

conditions, including variations in illumination, pose, and occlusion, ensuring robustness in practical deployments.

REFERENCES

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., & Zhai, X. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv. https://doi.org/10.48550/arXiv.2010.11929

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., & Khan, F. S. (2022). Transformers in Vision: A Survey. ACM Computing Surveys. https://doi.org/10.48550/arXiv.2101.01169

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv preprint arXiv:1905.11946. https://doi.org/10.48550/arXiv.1905.11946

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778. https://doi.org/10.1109/CVPR.2016.90

baocao.docx

BÁO CÁO ĐỘC SÁNG

3%	2%	3%	1%
CHỈ SỐ TƯƠNG ĐỒNG	NGUỒN INTERNET	ẤN PHẨM XUẤT BẢN	BÀI CỦA HỌC SINH

NGUỒN CHÍNH

1	Submitted to Ton Duc Thang University Bài của Học sinh	1%
2	Bui Thanh Hung, M. Sekar, Ayhan Esi, R. Senthil Kumar. "Applications of Mathematics in Science and Technology - International Conference on Mathematical Appplications in Science and Technology", CRC Press, 2025 Xuất bản	1%
3	web.realinfo.tv Nguồn Internet	1%
4	Arpita Nath Boruah, Mrinal Goswami, Manoj Kumar, Octavio Loyola-González. "Embedded Artificial Intelligence - Real-Life Applications and Case Studies", CRC Press, 2025 Xuất bản	1%

Loại trừ Trích dẫn	Mở	Exclude assignment template	Mở
Loại trừ mục lục tham khảo	Mở	Loại trừ trùng khớp	< 1%