**Real-Time Age and Gender Prediction Using Lightweight Multi-Task Convolutional Neural Networks**

### INTRODUCTION

**1.1.** **Abstract**

This project presents a deep learning-based system capable of predicting the **age** and **gender** of a person from an image using Convolutional Neural Networks (CNNs). We utilize the publicly available **UTKFace dataset**, which contains over 20,000 images labeled with age and gender. The model is designed as a multi-output CNN that predicts age as a regression task and gender as a classification task. Once trained, the model is used in real-time through a webcam to infer age and gender.

**1.2. Motivation**

The ability to estimate demographic attributes such as age and gender from facial images has wide-ranging applications in fields like surveillance, human-computer interaction, targeted marketing, and more. Traditional machine learning approaches often struggle with such tasks due to complex facial variations across age, gender, and ethnicity. Deep learning, especially CNNs, has emerged as a powerful tool in visual recognition tasks due to its ability to automatically learn and generalize features from raw images.

1. **PROJECT DESCRIPTION AND GOALS**
   1. **Survey on Existing System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sn** | **Paper Name** | **Remarks** | **Year** | **Reference** |
| 1. | A systematic review and research perspective on recommender systems | This research paper presents a systematic review of recent advancements in recommender systems, focusing on applications like books, movies, and products. Despite the precision of current systems, scalability, cold-start, and sparsity challenges persist. The study analyzes applications, conducts algorithmic assessments, and establishes a taxonomy for effective recommender systems. Evaluation criteria include datasets, simulation platforms, and performance metrics, providing a concise overview of the field's current state, highlighting gaps and challenges for future development. | 2022 | Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5 |
| 2. | Prioritizing tasks in software development: A systematic literature review | This work focuses on task prioritization in software development, aiming to identify effective tools and techniques. Conducting a systematic literature review following the PRISMA statement, the study reveals that most existing approaches concentrate on bug prioritization. Notably, recent works highlight the significance of "pull request prioritization" and "issue prioritization," reflecting the growing influence of version control and issue management software systems. The study also notes common metrics for evaluating prioritization models, such as f-score, precision, recall, and accuracy. This research provides valuable insights for IT practitioners, offering a comprehensive overview of the current state of task prioritization in the Software Engineering domain. | 2024 | Bugayenko Y, Bakare A, Cheverda A, Farina M, Kruglov A, Plaksin Y, et al. (2024) Prioritizing tasks in software development: A systematic literature review. PLoS ONE 18(4): e0283838. https://doi.org/10.1371/journal.pone.0283838 |
| 3. | Prioritizing Use Cases: A Systematic Literature Review | This research highlights the importance of use-case-based prioritization in software development. A systematic literature review of 40 approaches over the past two decades reveals a focus on user-centric requirements in areas like IoT and mobile development. Notably, only 32.5% considered scenario-based prioritization, and the majority were semiformally developed (53.8%). The findings underscore the need for new approaches addressing gaps in strategic goal inclusion and criteria like risks and quality-related requirements, providing insights for practitioners and researchers aiming to enhance prioritization practices. | 2023 | Odeh, Y.; Al-Saiyd, N. Prioritizing Use Cases: A Systematic Literature Review. *Computers* **2023**, *12*, 136. https://doi.org/10.3390/computers12070136 |
| 4. | Web Scraping as a Data Collection Strategy: The Perils and Pitfalls | This article explores the integration of artificial intelligence, specifically web scraping, into discourse analysis in the social work domain. Focusing on a study of blogging discourse on Black women's mental health during the dual pandemics of COVID-19 and anti-Black racism, the research leverages Python coding to automatically extract information from Medium.com. The study highlights obstacles, resolutions, and key recommendations for future web scraping studies, emphasizing the effectiveness and efficiency of this method with careful planning, preparedness for challenges, and resource considerations. Barriers such as expertise and technology resources are addressed, along with considerations for virtual work environments and managing hardware and software demands. | 2024 | DeVance Taliaferro, Jocelyn DeVance and Hedadji, Fatima and Duling, Emma, Web Scraping as a Data Collection Strategy: The Perils and Pitfalls. Available at SSRN: https://ssrn.com/abstract=4479267 or http://dx.doi.org/10.2139/ssrn.4479267 |
| 5. | WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis | This paper addresses the Implicit Aspect Identification task in sentiment analysis, a crucial element for real-world applications in e-commerce and manufacturing. The proposed approach enhances K-Nearest Neighbors (KNN) by incorporating WordNet semantic relations for improved distance computation. Empirical evaluations on electronic products and restaurant review datasets demonstrate the effectiveness of the approach, considering factors such as KNN distance, the number of nearest neighbors (K), and behavior towards Overfitting and Underfitting. The results indicate that the proposed method enhances KNN performance, particularly in Implicit Aspect Identification tasks, offering valuable insights for sentiment analysis applications. | 2023 | Benarafa, H., Benkhalifa, M. & Akhloufi, M. WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis. *Int J Comput Intell Syst* **16**, 3 (2023). https://doi.org/10.1007/s44196-022-00164-8 |
| 6. | Sentiment Analysis “Using SVM, KNN and SVM with PCA” | This paper explores the vast potential of sentiment analysis in natural language processing, emphasizing its ability to decipher human sentiments in various contexts. Focusing on restaurant reviews, the study compares different sentiment analysis techniques—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Support Vector Machine with Principal Component Analysis (PCA). The goal is to identify the most effective technique for accurately classifying reviews as positive or negative. Automation of sentiment analysis proves valuable for restaurant owners, offering insights into customer preferences and areas for improvement. The study achieves a notable 96% accuracy using the SVM model, showcasing its efficacy compared to other classification models. | 2023 | Verma, P., Bhardwaj, T., Bhatia, A., Mursleen, M. (2023). Sentiment Analysis “Using SVM, KNN and SVM with PCA”. In: Bhardwaj, T., Upadhyay, H., Sharma, T.K., Fernandes, S.L. (eds) Artificial Intelligence in Cyber Security: Theories and Applications. Intelligent Systems Reference Library, vol 240. Springer, Cham. https://doi.org/10.1007/978-3-031-28581-3\_5 |
| 7. | YouTube Sentimental Analysis Using a Combined Approach of KNN and K-means Clustering Algorithm | This paper addresses the necessity of sentiment analysis in comprehending user opinions on YouTube, a widely used video-sharing platform. With the surge in comments on popular channels, dealing with this vast and unstructured data requires effective applications or methods. The study employs sentiment analysis, combining K-Nearest Neighbor (KNN) and K-means clustering approaches to categorize YouTube comments. The proposed technique is compared with SVM classifier and Naive Bayes for accuracy, showcasing its effectiveness in analyzing sentiments on a larger platform like YouTube. | 2025 | Adhikari, S., Kaushik, R., Obaid, A.J., Jeyalaksshmi, S., Balaganesh, D., Hanoon, F.H. (2023). YouTube Sentimental Analysis Using a Combined Approach of KNN and K-means Clustering Algorithm. In: Peng, SL., Jhanjhi, N.Z., Pal, S., Amsaad, F. (eds) Proceedings of 3rd International Conference on Mathematical Modeling and Computational Science. ICMMCS 2023. Advances in Intelligent Systems and Computing, vol 1450. Springer, Singapore. https://doi.org/10.1007/978-981-99-3611-3\_4 |
| 8. | Sentiment Analysis of Review Datasets Using Naive Bayes and K-NN Classifier | This project addresses the surge in sentimental content on the web, focusing on movie and hotel reviews in social media. The sentiment-focused web crawling framework utilizes statistical methods to capture subjective style and sentence polarity. The study employs two supervised machine learning algorithms, K-Nearest Neighbour (K-NN) and Naive Bayes, for sentiment analysis. In movie reviews, Naive Bayes outperforms K-NN, while both algorithms show comparable accuracies for hotel reviews. The project emphasizes the importance of timely discovery of sentimental web content for applications like contextual advertisements, recommendation systems, and market trend analysis in the Web 2.0 era. | 2016 | [Dey, L., Chakraborty, S., Biswas, A., Bose, B., & Tiwari, S. (2016). Sentiment Analysis of Review Datasets Using Naïve Bayes‘ and K-NN Classifier. International Journal of Information Engineering and Electronic Business, 8(4), 54–62. MECS Publisher. ISSN: 2074-9031. DOI: 10.5815/ijieeb.2016.04.07.](http://dx.doi.org/10.5815/ijieeb.2016.04.07) |
| 9. | Sentiment Analysis in the Era of Large Language Models: A Reality Check | The paper explores the application of large language models (LLMs), exemplified by ChatGPT, in sentiment analysis (SA). While LLMs show promising results in simpler tasks, their performance lags in more complex sentiment analysis tasks requiring a deeper understanding or structured sentiment information. The study encompasses 13 tasks on 26 datasets, comparing LLMs with small language models (SLMs) trained on domain-specific datasets. Notably, LLMs exhibit significant advantages in few-shot learning scenarios, indicating their potential when annotation resources are limited. The paper also highlights the limitations of current evaluation practices and introduces a new benchmark, SENTIEVAL, to assess LLMs' sentiment analysis capabilities comprehensively. | 2023 | Zhang, W., & Deng, Y. (2023). Sentiment Analysis in the Era of Large Language Models: A Reality Check. Retrieved from https://synthical.com/article/85237ecb-ae59-47ec-9c7c-c26866cf9cfa. arXiv preprint arXiv:2305.15005. |
| 10. | Modelling Sentiment Analysis: LLMs and augmentation techniques | This paper explores various approaches for binary sentiment classification on a limited training dataset, utilizing large language models (LLMs) like BERT, RoBERTa, and XLNet, known for delivering state-of-the-art results in sentiment analysis. Additionally, the paper introduces diverse data augmentation techniques to address the challenges posed by the small training dataset. | 2023 | Guillem Senabre Prades. "Modelling Sentiment Analysis: LLMs and Data Augmentation Techniques." arXiv preprint, 2023. |

* 1. **Research Gap**

Scalability Issues: Many existing systems struggle with scalability when handling large volumes of data. The increase in user interactions on platforms such as YouTube, Amazon, and Flipkart has led to a data overload that many current feedback management systems are not equipped to handle efficiently.

Despite significant advancements in facial attribute recognition, there remains a notable gap between research-focused models and practical, real-time implementations. Most existing works either rely on large-scale, pre-trained models like VGGFace, DEX, or IMDB-WIKI-based systems that demand high computational resources, or they focus solely on either age or gender prediction, lacking the efficiency of multi-task learning. Commercial APIs such as Face++ and Amazon Rekognition offer high accuracy but operate as black-box systems, limiting transparency and customization while also raising privacy concerns. Furthermore, academic research often employs complex architectures like ResNet or Inception, which, although accurate, are not well-suited for real-time deployment on consumer-grade hardware. In contrast, this project addresses these limitations by introducing a lightweight, end-to-end CNN model capable of performing both age estimation and gender classification simultaneously. The model is trained on the UTKFace dataset, designed for real-time inference, and implemented with open-source tools, making it accessible, efficient, and suitable for local deployment without reliance on external APIs or high-end hardware.

* 1. **Problem Statement**

In today's rapidly advancing digital world, the ability to automatically estimate demographic attributes such as age and gender from facial images has become increasingly valuable. Applications span across various domains including targeted advertising, human-computer interaction, biometric security systems, digital content personalization, and surveillance. Despite this demand, accurately predicting age and gender from real-world facial images remains a challenging problem due to a variety of factors.

Firstly, facial appearances vary significantly due to lighting conditions, facial expressions, head pose, image resolution, race, and other environmental factors. These variations can significantly reduce the performance of traditional machine learning techniques that rely on handcrafted features. Furthermore, while many deep learning approaches have improved prediction accuracy, they are often computationally expensive, utilize massive pre-trained models, or rely on cloud-based services, which may not be suitable for real-time or privacy-sensitive applications.

Additionally, most existing systems treat age estimation and gender classification as separate problems, leading to increased computational cost and complexity. Multi-task learning, where a single model is trained to perform both tasks simultaneously, has shown promise but is still underutilized in practical applications, especially in lightweight, real-time systems.

Moreover, there is a lack of open-source, beginner-friendly implementations that integrate end-to-end training, testing, and live webcam-based inference using accessible frameworks such as TensorFlow and OpenCV. This limits adoption and further research, especially for students and developers without access to high-end GPUs or large datasets.

Therefore, there is a clear need for a lightweight, efficient, and accessible solution that can predict both age and gender from facial images in real-time, without relying on external APIs or cloud services. This project aims to bridge that gap by developing a compact, multi-output Convolutional Neural Network trained on the UTKFace dataset, capable of performing real-time inference using a standard webcam and personal computer.

1. **TECHNICAL SPECIFICATION**
   1. **Requirements**

**3.1.1.** Functional

**Data Gathering:**

Broad Coverage: The system should also be capable of extending its data collection capabilities to other emerging platforms and social media to ensure comprehensive market feedback is captured.

Real-time Data Streaming: Implement real-time data gathering capabilities to allow immediate analysis of fresh data, which is critical for time-sensitive feedback.

**Data Preprocessing:**

Language Detection: Incorporate a language detection step to handle multi-language content appropriately, ensuring that the preprocessing tools used are language-specific.

Error Handling: Develop robust error handling mechanisms to manage incomplete or corrupt data entries without disrupting the preprocessing workflow.

**Comment Categorization:**

Scalable Clustering: Ensure the clustering algorithm can scale with increased data volume without sacrificing processing speed or accuracy.

Dynamic Model Updating: Implement mechanisms for the periodic retraining of the clustering model to adapt to new trends and changes in user interaction patterns.

**Priority Assignment:**

User Impact Analysis: Integrate an analysis layer to assess the potential impact of each comment based on the commenter's influence and historical engagement metrics.

Contextual Awareness: Develop the system’s ability to consider the context of the discussion when assigning priorities, recognizing urgent issues even in seemingly neutral comments.

**Output Generation:**

Visualization Tools: Incorporate visualization tools in the output to present the prioritized comments and insights in an intuitive and easy-to-understand manner for quick decision-making.

Feedback Loop: Establish a feedback loop mechanism where the output results can be evaluated and refined based on user or moderator input to continually enhance the accuracy of the prioritization process.

**3.1.2.** Non-Functional

**Performance:**

Adaptive Performance Optimization: Implement adaptive algorithms to optimise resource allocation based on current load and data complexity, ensuring consistent performance under changing situations.

Parallel Processing: Use parallel processing techniques to manage and analyse data simultaneously, dramatically lowering the time required to provide outputs.

Reliability:

Failover Mechanisms: Include automatic failover capabilities to switch to a backup system in the event of a failure, assuring uninterrupted operation.

Regular System Audits: Schedule regular audits and updates to ensure that all components work properly and that any possible problems are addressed promptly.

**Scalability:**

Cloud Integration: Enable dynamic resource allocation depending on the system's present requirements by integrating with cloud services to promote scalability.

Load Balancing: Employ load balancing strategies to divide network traffic and data processing equally among servers.

**Security:**

Encryption Protocols: To prevent unwanted access to sensitive data, use robust encryption algorithms for both data in transit and data at rest.

Frequent Security Updates: To guard against fresh vulnerabilities, keep up a routine of updating security software and protocols.

**Usability:**

Interactive Tutorials: To help new users navigate and operate the system efficiently, include interactive tutorials and help sections on the dashboard.

Customizable User Interfaces: Increase comfort and personalisation by giving users the ability to alter the interface to suit their preferences and usability requirements.

**Maintainability:**

Automated Testing: Implement automated testing frameworks to regularly check and ensure that all parts of the system are functioning as intended.

Version Control Integration: Use version control systems to manage changes and updates to the software, ensuring that maintenance and upgrades are systematically tracked and implemented.

**Compatibility:**

Multi-platform Support: Ensure the system is compatible across various operating systems and devices, including mobile platforms, to enhance accessibility.

Standards Compliance: Adhere to international web standards and protocols to ensure compatibility and interoperability with a wide range of web technologies.

**Accuracy:**

Continuous Learning: Integrate continuous learning mechanisms that allow the system to learn from past decisions and refine its algorithms over time, improving accuracy.

Quality Assurance Metrics: Establish and monitor key quality assurance metrics that track the accuracy of data categorization and prioritization, ensuring the system meets high standards.

* 1. **Feasibility Study**

A feasibility study evaluates the practicality of a proposed project or system in terms of its technical, economic, operational, and time-based aspects. This section assesses whether the development and deployment of the **Age and Gender Detection System using CNN** is viable given the available resources and constraints.

### ****1. Technical Feasibility****

The project utilizes well-supported, widely-used technologies such as Python, TensorFlow/Keras, NumPy, and OpenCV, all of which are open-source and compatible with various platforms. The UTKFace dataset, used for training and evaluation, is publicly available and provides a diverse range of facial images labeled with age and gender.

The model employs a custom-built, multi-output Convolutional Neural Network (CNN), which is computationally efficient and does not require extensive hardware resources. Training and deployment are feasible on standard computing hardware, such as a personal laptop with 8–16 GB of RAM and, optionally, GPU acceleration for faster processing. Real-time webcam inference is facilitated through OpenCV, allowing live video capture and prediction visualization.

**Conclusion**: The system is technically feasible using readily available tools and typical hardware configurations.

### ****2. Economic Feasibility****

All software components and tools used in the development of the system are open-source, thereby eliminating licensing costs. The UTKFace dataset is also freely accessible for academic and research purposes. Since the model operates locally and does not depend on third-party cloud APIs or paid services, there are no recurring costs associated with its operation.

The development and deployment process does not necessitate specialized or expensive hardware, making it suitable even for low-budget environments such as academic institutions or small-scale startups.

**Conclusion**: The system is economically feasible, requiring minimal to no financial investment.

### ****3. Operational Feasibility****

The proposed system is designed to be user-friendly, with minimal installation and setup requirements. It offers real-time predictions through a webcam interface and does not require prior expertise in deep learning or computer vision to operate once deployed. Additionally, the local execution of the model ensures data privacy and operational independence from external networks.

The modular structure of the codebase allows easy integration with other systems such as security applications, smart kiosks, or attendance monitoring systems. This enhances its potential for real-world adoption.

**Conclusion**: The system is operationally practical and suitable for deployment in various real-time environments.

### ****4. Time Feasibility****

The estimated timeline for project completion is between four to six weeks. This includes time for literature review, data preprocessing, model development, training, evaluation, and integration of webcam-based real-time inference. The relatively short development cycle makes the project ideal for academic coursework, research projects, or prototype-level development.

**Conclusion**: The system is time-feasible and can be completed within a standard academic or project development timeline.

* 1. **System Specification**

**3.3.1.** Hardware Specification

**AWS EC2 Instance:** Using an AWS EC2 instance to host the backend is a smart decision that leverages Amazon's cloud infrastructure to assure scalability and stability. EC2 instances can be dynamically modified to match the system's computational demands, giving administrators greater flexibility in managing computational resources.

**High Capacity GPU (e.g., NVIDIA GeForce RTX 3090):** A high-performance GPU, such as the NVIDIA GeForce RTX 3090, is required for processing big datasets and completing complicated calculations quickly. GPUs are especially useful for machine learning and data analysis activities, where parallel computing can drastically cut data processing and model training time.

**3.3.2.** Software Specification

**Python Environment with Anaconda Navigator:** Using Anaconda Navigator to manage the Python environment is a practical option because it simplifies the installation and management of various Python packages and their dependencies. Anaconda includes a comprehensive set of tools and libraries required for data science and machine learning projects, such as MatPlotLib for data visualisation and NumPy for numerical computations, among others.

**Clustering Calculations:** The software must perform clustering computations, which are critical to the system's capacity to categorise and prioritise user comments. These calculations demand strong computing assistance, necessitating the employment of powerful hardware and efficient software libraries.

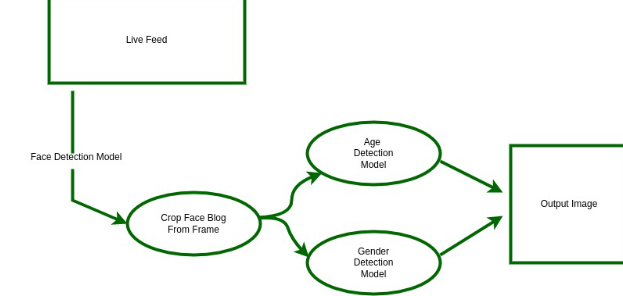
**Flask for Backend API:** Flask is a Python web framework that is both lightweight and powerful, making it ideal for developing APIs. It is chosen to function as a microservice backend, processing API requests between the frontend and the server. Flask's simplicity and versatility make it ideal for projects that demand a clean and efficient approach to deploy web services without the expense of larger frameworks such as Django.

**Integration and Implementation :**

Integration: By combining high-performance hardware with advanced software settings, the system is better able to handle intense tasks such as real-time data processing and machine learning model deployments.

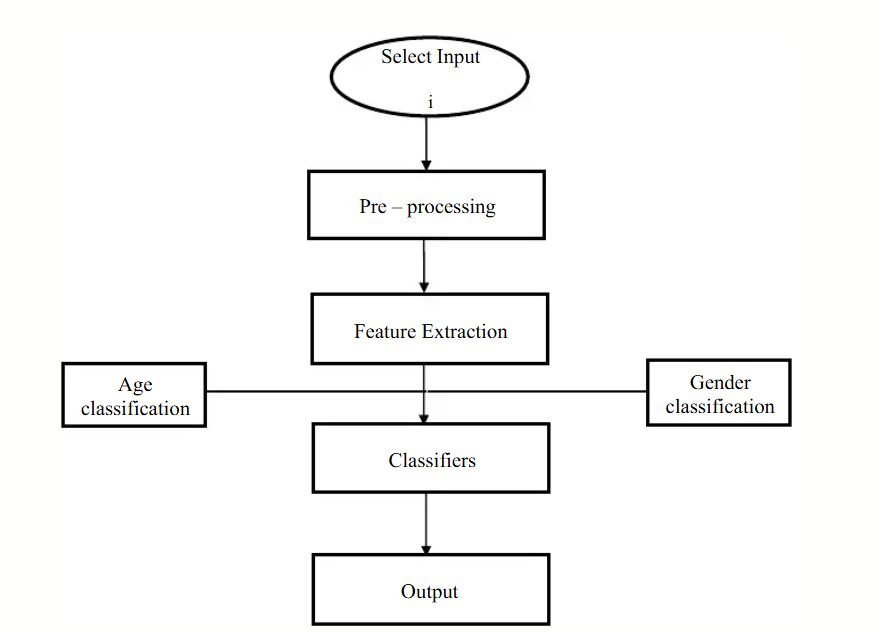
Implementation: These standards must be carefully planned and configured to guarantee that all components work together seamlessly. This includes creating EC2 instances, configuring GPUs for peak performance, establishing a Python environment with Anaconda, and deploying Flask to handle API calls**.**

1. **DESIGN APPROACH AND DETAILS**
   1. **System Architecture**

****

* 1. **Design**

**4.2.1**. **Data Flow Diagram**

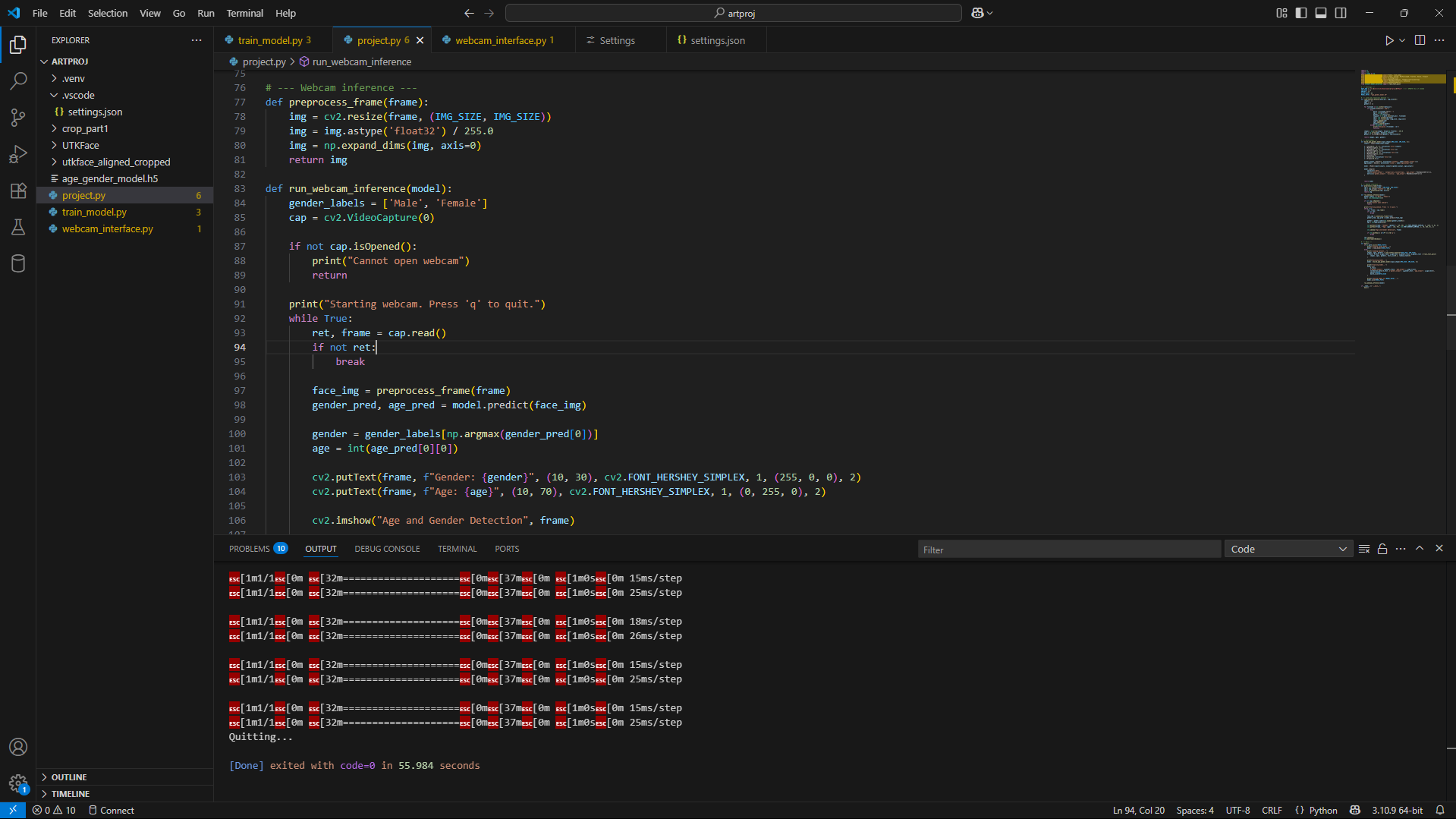


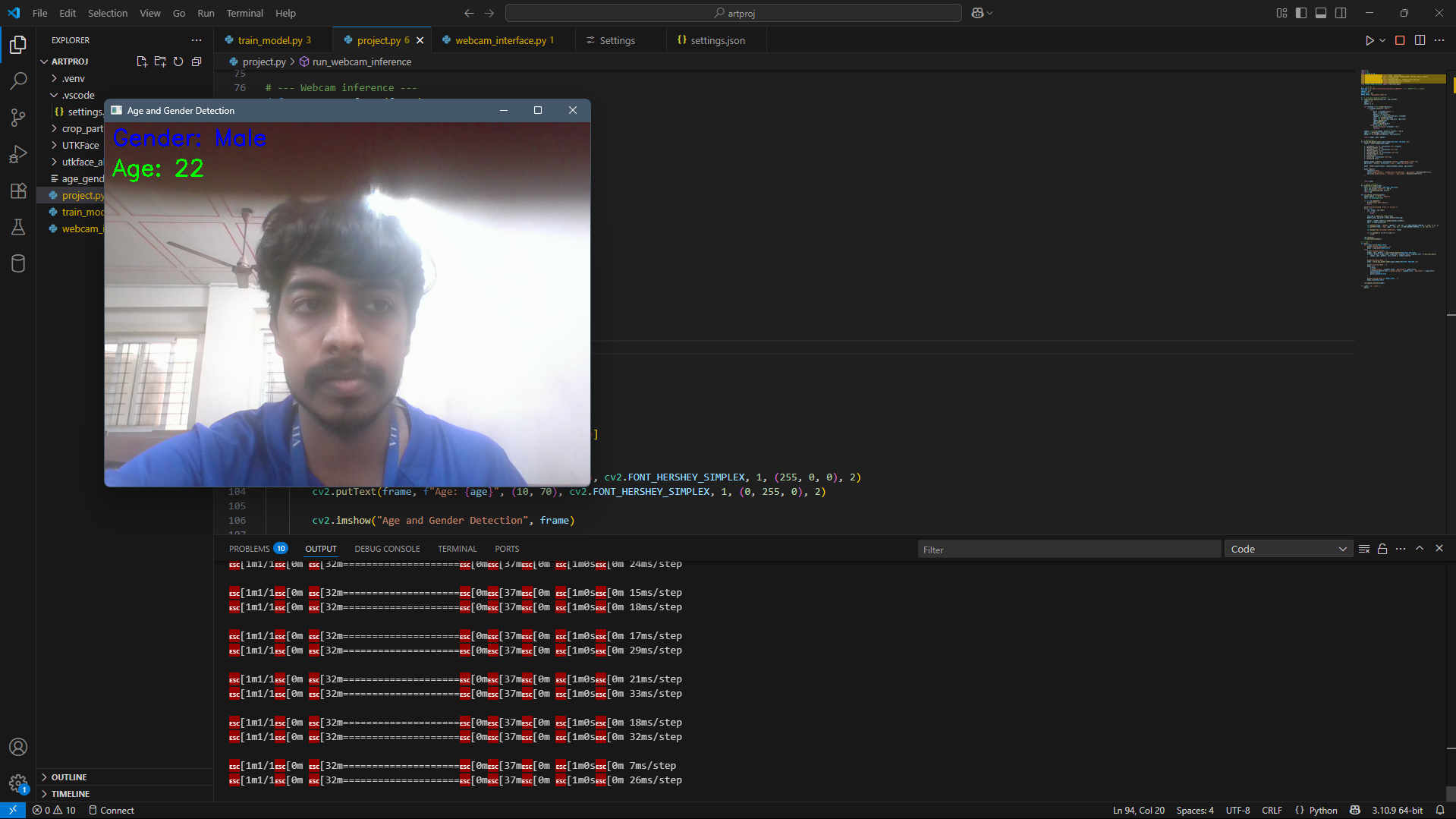
4.2. Dataflow diagram illustrating the automated processing and analysis of YouTube comments for strategic insight generation.

* 1. Role-based use case diagram showing the process of managing and analyzing user comments to enhance user interactions.

The diagram depicts a role-based workflow for managing and analyzing user comments, with specific tasks such as fetching, scraping, and clustering comments, as well as sentiment analysis, keyword identification, and topic modelling, assigned to different roles: Developer, Administrator, and Data Analyst. The outcomes of these procedures converge to prioritize clusters and deliver actionable insights that directly affect the user experience.

1. **PROJECT DEMONSTRATION**





1. **COST ANALYSIS / RESULT & DISCUSSION**
   1. **Cost Analysis**

**The overall cost of developing and deploying the Age and Gender Detection system is minimal, primarily because it leverages open-source software tools and publicly available datasets. The programming language Python and essential libraries such as TensorFlow, Keras, OpenCV, NumPy, and Pandas are all free to use, eliminating any software licensing expenses. Development environments like Visual Studio Code or the Community edition of PyCharm are also freely available, further reducing costs. On the hardware front, the project can be executed on a standard personal computer or laptop equipped with at least 8 GB of RAM and a mid-range processor such as Intel i5 or AMD Ryzen 5. Such hardware typically costs between ₹45,000 to ₹60,000 (approximately $600 to $800), which is a one-time investment if the developer already owns such a machine, effectively making the hardware cost negligible for many users. Additionally, training the model can be accelerated using a GPU, but this is optional, and cloud-based resources like Google Colab offer free GPU access, avoiding the need for additional hardware expenses. Since the system performs all inference locally, there are no ongoing costs related to cloud services or API subscriptions. Operationally, the cost is also minimal as the system requires no special maintenance or additional infrastructure. Therefore, the cost analysis reveals that the project is highly economical and accessible, making it feasible for students, researchers, and small-scale developers.**

1. **SUMMARY**

This project presents a lightweight, multi-output convolutional neural network (CNN) designed for real-time age estimation and gender classification from facial images. Utilizing the publicly available UTKFace dataset, the model simultaneously predicts both attributes, improving efficiency compared to separate models. The system is implemented using open-source tools such as TensorFlow and OpenCV, enabling webcam-based live inference without relying on cloud services or external APIs. Designed to run on standard personal computers, the project addresses key challenges including variability in facial features, lighting, and pose while maintaining user privacy by processing data locally. The model’s balance between accuracy and computational efficiency makes it suitable for practical applications in security, marketing, and human-computer interaction. Overall, the project demonstrates an accessible and cost-effective approach to demographic attribute prediction with real-time capabilities.

1. **REFERENCES**

Web Links:

[1] Cloudflare. "Under Attack." [Online]. Available: https://www.cloudflare.com/under-attack. Accessed: Dec. 2014.

[2] The Daily Beast. "Hackers' 10 Most Famous Attacks: Worms and DDoS Takedowns." [Online]. Available: http://www.thedailybeast.com/articles/2010/12/11/hackers-10-most-famous-attacks-wormsandddos-takedowns.html. Accessed: Dec. 2014.

Journals:

[3] G. Zacharia, "Trust Management through Reputation Mechanisms," in Workshop in Deception, Fraud and Trust in Agent Societies, Third International Conference on Autonomous Agents (Agents’99), ACM, 1999. International Journal of Security and Its Applications, vol. 9, no. 9, 2015, pp. 210. [Online]. Available: [Source Link].

[4] L. Eschenauer, V. D. Gligor, and J. Bara, "On Trust Establishment in Mobile Ad Hoc Networks," in Security Protocols, Springer, 2004, pp. 47-66.

[5] N.Ch. S.N. Iyengar, Gopinath Ganapathy, P.C. Mogan Kumar, and Ajith Abraham, "A multilevel thrust filtration defending mechanism against DDoS attacks in cloud computing environment," International Journal of Grid and Utility Computing, vol. 5, no. 4, 2014, pp. 236-248.

Book:

[6] A.P. Malvino and D.P. Leach, "Digital Principles and Applications," Tata McGraw Hill, 2014, Special Edition – 2009.

1. **APPENDIX A – SAMPLE CODE**

**Train\_model.py**

|  |
| --- |
| **import os**  **import cv2**  **import numpy as np**  **from tensorflow.keras.models import Model, load\_model**  **from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout**  **from tensorflow.keras.utils import to\_categorical**  **from tensorflow.keras.losses import MeanSquaredError, CategoricalCrossentropy**  **from tensorflow.keras.metrics import MeanAbsoluteError, Accuracy**  **from tensorflow.keras.losses import MeanSquaredError**  **from sklearn.model\_selection import train\_test\_split**  **# --- Settings ---**  **DATA\_DIR = r"C:\Users\rishi\Downloads\artproj\UTKFace"  # <-- UPDATE this if needed**  **IMG\_SIZE = 64**  **EPOCHS = 15**  **BATCH\_SIZE = 64**  **MODEL\_PATH = "age\_gender\_model.h5"**  **# --- Load and preprocess dataset ---**  **def load\_utkface\_dataset(data\_dir, img\_size=64):**  **images = []**  **ages = []**  **genders = []**  **for filename in os.listdir(data\_dir):**  **if filename.endswith(".jpg"):**  **try:**  **parts = filename.split('\_')**  **age = int(parts[0])**  **gender = int(parts[1])**  **img\_path = os.path.join(data\_dir, filename)**  **img = cv2.imread(img\_path)**  **img = cv2.resize(img, (img\_size, img\_size))**  **images.append(img)**  **ages.append(age)**  **genders.append(gender)**  **except Exception as e:**  **print(f"Skipping {filename}: {e}")**  **continue**  **images = np.array(images, dtype=np.float32) / 255.0**  **ages = np.array(ages, dtype=np.float32)**  **genders = to\_categorical(genders, num\_classes=2)**  **return images, ages, genders**  **# --- Build model ---**  **def build\_age\_gender\_model(input\_shape=(IMG\_SIZE, IMG\_SIZE, 3)):**  **inputs = Input(shape=input\_shape)**  **x = Conv2D(32, (3, 3), activation='relu')(inputs)**  **x = MaxPooling2D(2, 2)(x)**  **x = Conv2D(64, (3, 3), activation='relu')(x)**  **x = MaxPooling2D(2, 2)(x)**  **x = Conv2D(128, (3, 3), activation='relu')(x)**  **x = MaxPooling2D(2, 2)(x)**  **x = Flatten()(x)**  **x = Dense(256, activation='relu')(x)**  **x = Dropout(0.5)(x)**  **gender\_output = Dense(2, activation='softmax', name='gender\_output')(x)**  **age\_output = Dense(1, activation='linear', name='age\_output')(x)**  **model = Model(inputs=inputs, outputs=[gender\_output, age\_output])**    **model.compile(**  **optimizer='adam',**  **loss={'gender\_output': 'categorical\_crossentropy', 'age\_output': MeanSquaredError()},**  **metrics={'gender\_output': 'accuracy', 'age\_output': MeanAbsoluteError()}**  **)**    **return model**  **# --- Webcam inference ---**  **def preprocess\_frame(frame):**  **img = cv2.resize(frame, (IMG\_SIZE, IMG\_SIZE))**  **img = img.astype('float32') / 255.0**  **img = np.expand\_dims(img, axis=0)**  **return img**  **def run\_webcam\_inference(model):**  **gender\_labels = ['Male', 'Female']**  **cap = cv2.VideoCapture(0)**  **if not cap.isOpened():**  **print("Cannot open webcam")**  **return**  **print("Starting webcam. Press 'q' to quit.")**  **while True:**  **ret, frame = cap.read()**  **if not ret:**  **break**  **face\_img = preprocess\_frame(frame)**  **gender\_pred, age\_pred = model.predict(face\_img)**  **gender = gender\_labels[np.argmax(gender\_pred[0])]**  **age = int(age\_pred[0][0])**  **cv2.putText(frame, f"Gender: {gender}", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2)**  **cv2.putText(frame, f"Age: {age}", (10, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)**  **cv2.imshow("Age and Gender Detection", frame)**  **if cv2.waitKey(1) & 0xFF == ord('q'):**  **break**  **cap.release()**  **cv2.destroyAllWindows()**  **# --- Main ---**  **def main():**  **if os.path.exists(MODEL\_PATH):**  **print("Loading saved model...")**  **model = load\_model(MODEL\_PATH)**  **else:**  **print("Loading dataset...")**  **images, ages, genders = load\_utkface\_dataset(DATA\_DIR, IMG\_SIZE)**  **X\_train, X\_test, y\_age\_train, y\_age\_test, y\_gender\_train, y\_gender\_test = train\_test\_split(**  **images, ages, genders, test\_size=0.2, random\_state=42**  **)**  **print("Building model...")**  **model = build\_age\_gender\_model(input\_shape=(IMG\_SIZE, IMG\_SIZE, 3))**  **print("Training model...")**  **model.fit(**  **X\_train,**  **{'gender\_output': y\_gender\_train, 'age\_output': y\_age\_train},**  **validation\_data=(X\_test, {'gender\_output': y\_gender\_test, 'age\_output': y\_age\_test}),**  **epochs=EPOCHS,**  **batch\_size=BATCH\_SIZE**  **)**  **print(f"Saving model to {MODEL\_PATH}...")**  **model.save(MODEL\_PATH)**  **run\_webcam\_inference(model)**  **if \_\_name\_\_ == "\_\_main\_\_":**  **main()** |

**Webcam\_interface.py**

|  |
| --- |
| import cv2  import numpy as np  from tensorflow.keras.models import load\_model  IMG\_SIZE = 64  MODEL\_PATH = "age\_gender\_model.h5"  def preprocess\_frame(frame):      img = cv2.resize(frame, (IMG\_SIZE, IMG\_SIZE))      img = img.astype('float32') / 255.0      img = np.expand\_dims(img, axis=0)      return img  def run\_webcam\_inference():      model = load\_model(MODEL\_PATH)      gender\_labels = ['Male', 'Female']      cap = cv2.VideoCapture(0)      if not cap.isOpened():          print("Cannot open webcam")          return      else:          print("Webcam opened successfully")      print("Starting webcam. Press 'q' to quit.")      while True:          ret, frame = cap.read()          if not ret:              print("Failed to grab frame")              break          img = preprocess\_frame(frame)          gender\_pred, age\_pred = model.predict(img)          gender = gender\_labels[np.argmax(gender\_pred)]          age = int(age\_pred[0][0])          cv2.putText(frame, f"Gender: {gender}", (10, 30),                      cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2)          cv2.putText(frame, f"Age: {age}", (10, 70),                      cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)          cv2.imshow("Age and Gender Detection", frame)          if cv2.waitKey(1) & 0xFF == ord('q'):              print("Quitting...")              break      cap.release()      cv2.destroyAllWindows()  if \_\_name\_\_ == "\_\_main\_\_":      run\_webcam\_inference() |