Internship Report

Task 4: Nationality Detection System

A Multi-Task Computer Vision Application for Human Attribute Analysis

Prepared by: Tarrush Saxena

Completed: January 2026

1. Introduction

This report documents my work on Task 4 of the internship program, which involved developing a Nationality Detection System. The project challenged me to build a complete computer vision application that can detect and analyze multiple human attributes from a live camera feed. What started as a straightforward classification task quickly evolved into something much more interesting—a multi-task learning system that detects nationality, emotion, age, and even clothing color, all in real-time. I found myself diving deep into transfer learning, neural network architectures, and designing a user interface that could

display all this information in an intuitive way. The experience really pushed me to integrate everything I had learned about deep learning, image processing, and GUI development into one cohesive project.

2. Background

The idea behind this project comes from the growing need in various industries to understand demographic and emotional patterns. Think about retail analytics where stores want to know who their customers are, or hospitality services that could benefit from understanding guest demographics. Even security systems are starting to incorporate these kinds of technologies. The core problem I tackled was this: given a video stream from a webcam, can we reliably identify a person's nationality or ethnicity, determine their emotional state, estimate their age, and pick up on what color clothes they're wearing? Each of these is a complex problem on its own, but the real challenge was making them work together smoothly and fast enough for real-time processing. I used OpenCV for handling the video feed and face detection, TensorFlow and Keras for the deep learning models, and scikit-learn for the color clustering algorithm. For the user interface, I went with Tkinter because it's reliable and doesn't add unnecessary complexity.

3. Learning Objectives

Going into this project, I set myself some clear goals about what I wanted to learn and accomplish. Here's what I was aiming for:

• Get hands-on experience with transfer learning using pre-trained models like MobileNetV2

• Build and train CNNs for different tasks—classification for nationality and emotion, regression for age

• Figure out how to do real-time face detection and tracking with OpenCV

• Learn K-Means clustering for practical image analysis like color extraction

• Put together a complete end-to-end ML pipeline from data prep to deployment

• Design a graphical interface that updates smoothly with real-time predictions

• Understand how to organize a multi-model system with proper configuration management

4. Activities and Tasks

The project work broke down into several distinct phases, each building on the previous one: **Dataset Preparation:** I spent quite a bit of time gathering and organizing training data. For nationality detection, I worked with the FairFace dataset, which has reasonably balanced representation across different ethnic groups. For emotion recognition, I used FER-2013, which is a classic dataset with seven emotion categories. Age prediction training used the UTKFace dataset. The data setup script I wrote handles all the downloading, extraction, and folder organization automatically—it saves a lot of time when you need to set up the project on a new machine. **Model Architecture Design:** For the nationality model, I used MobileNetV2 as a base and added custom classification layers on top. MobileNetV2 is great because it's efficient enough for real-time inference but still powerful enough to learn complex patterns. The emotion and age models use more traditional CNN architectures since they work with smaller input images and don't need the same level of feature extraction capability. **Training Pipeline:** I wrote separate training scripts for each model, all with argument parsing so you can easily adjust hyperparameters like epochs, batch size, and learning rate from the command line. Each script includes data augmentation, early stopping, and model checkpointing to save the best weights. The training process outputs progress updates so you can monitor how things are going. **Color Extraction Module:** This was an interesting sub-project. Rather than using another neural network, I went with a K-Means clustering approach to find the dominant color in the torso region (below the detected face). The algorithm clusters pixel colors and returns the most common one, which I then map to a human-readable color name using Euclidean distance to a predefined color palette. **Engine Integration:** The NationalityEngine class ties everything together. It loads all the models, handles preprocessing for each one, and provides a clean interface for the GUI to call. The engine also implements a branching logic system—depending on the detected nationality, different attributes might be computed or displayed differently. **GUI Development:** I built the interface using Tkinter with a clean, professional look. The main window shows the live camera feed with bounding boxes and labels overlaid on detected faces. A side panel displays execution logs so you can see what the system is doing. The models load in a background thread so the UI stays responsive during startup.

5. Skills and Competencies

Working on this project really leveled up my skills in several key areas: **Deep Learning:** I got much more comfortable with TensorFlow and Keras, especially when it comes to transfer learning. Understanding how to freeze base layers, add custom heads, and fine-tune models was invaluable. I also learned a lot about handling different model types—classification vs regression—and how the loss functions and output layers differ. **Computer Vision:** OpenCV became second nature by the end of this project. Haar cascades for face detection, image preprocessing, color space conversions, drawing annotations on frames—all of these are now tools I can reach for confidently. **Python Development:** The project reinforced good practices like modular code organization,

configuration management, argument parsing for scripts, and threading for responsive UIs. I also got better at writing code that fails gracefully—the system handles missing models or failed predictions without crashing. **Problem Solving:** There were so many little puzzles to figure out. How do you estimate where someone's torso is based on their face location? How do you make model inference fast enough for real-time video? How do you display multiple attributes on screen without making it look cluttered? Each of these required creative thinking and experimentation.

6. Feedback and Evidence

The finished system works quite well in practice. Here's what I observed during testing: The nationality detection runs at a comfortable frame rate on my laptop—fast enough that there's no noticeable lag when moving in front of the camera. The emotion recognition is surprisingly responsive, picking up changes in facial expressions almost instantly. Age predictions are in the right ballpark, though like most age estimation systems, they sometimes miss by a few years. The color detection feature is probably the most fun to watch. When I change shirts or stand in front of different backgrounds, the system correctly identifies the dominant color most of the time. It occasionally gets confused with very complex patterns or similar shades, but overall performs well. The code is well-documented and organized into clear modules. The configuration file makes it easy to adjust model paths, color detection parameters, and branching logic without touching the core code. The project includes unit tests and a comprehensive README with installation and usage instructions.

7. Challenges and Solutions

Like any substantial project, this one came with its share of headaches: **Model Loading Errors:** Initially, loading saved Keras models would fail with cryptic errors about missing metrics or custom objects. The fix turned out to be using compile=False when loading, then recompiling if needed. This taught me a lot about how Keras serializes model configurations. **Real-Time Performance:** My first implementation was way too slow—predictions were lagging behind the video by several seconds. I solved this by using smaller input sizes where possible, reducing the number of K-Means iterations for color detection, and being smarter about when to run predictions (not every single frame needs full analysis). **Dataset Quality:** Some of the datasets I used have known biases and quality issues. For example, lighting conditions in training images don't always match real webcam conditions. Data augmentation during training helped somewhat, but this remains an area for improvement. **Threading and UI Responsiveness:** The GUI would freeze while models were loading at startup. I fixed this by loading models in a background thread and updating the UI when loading completes. It seems simple now, but getting the threading right without race conditions took some careful thought. **Torso Region Estimation:** Figuring out where to look for clothing color wasn't straightforward. The face

bounding box only tells you where the face is, not where the body is. I developed a heuristic that extends below and slightly to the sides of the face, which works well for typical webcam angles.

8. Outcomes and Impact

By the end of this task, I had built a fully functional multi-attribute detection system that processes live video in real-time. Here's what the final product accomplishes: **Technical Deliverables:** • Three trained deep learning models (nationality, emotion, age) with saved weights • A modular codebase with separate modules for GUI, engine, and color extraction • Training scripts with configurable parameters and automatic model checkpointing • Data preparation utilities that handle dataset download and organization • A complete configuration system for customizing model paths and detection parameters **Practical Capabilities:** • Detects faces in real-time video and draws color-coded bounding boxes • Predicts nationality/ethnicity with confidence scores displayed on screen • Recognizes seven different emotions and updates predictions live • Estimates age using a regression model trained on face images • Identifies dominant clothing color using unsupervised clustering **Learning Outcomes:** This project gave me end-to-end experience with a real ML application. I now understand what it takes to go from raw datasets to a working product that regular users can interact with. The skills I developed—transfer learning, real-time processing, multi-model systems, UI development—are directly applicable to other computer vision projects.

9. Conclusion

Task 4 was probably the most comprehensive project I worked on during this internship. Building the Nationality Detection System pushed me to integrate multiple machine learning models, work with real-time video processing, and create a user-friendly interface—all in one cohesive application. The biggest takeaway for me is understanding how different components of an ML system need to work together. It's not enough to train a good model; you also need efficient inference, clean code architecture, error handling, and a way for users to actually interact with your work. This project covered all of those aspects. Looking forward, there are definitely ways this system could be improved. Better training data, more sophisticated face detection algorithms, and GPU acceleration for inference would all help. But as a learning experience and a demonstration of multi-task computer vision, I'm proud of what I built here. The complete source code, trained models, and documentation are organized in a clean project structure and ready for future development or deployment.