ESP32-CAM Face Mask Detection System

Project: Industrial AI - Edge AI in Industrial Applications (AI5252)
Semester: Winter Semester 25/26

Submitted By: Kushal Kattel
Matriculation No: 1568514

Submitted To: Professor Ivan Jursic

1. Project Overview

This project builds a real-time face mask detection system running entirely on an ESP32-CAM microcontroller. The system captures camera frames, runs a CNN classifier on-device, and triggers a visual alert (LED + dashboard) when no mask is detected. No image data is ever sent to a cloud server or stored persistently. The model was trained via Edge Impulse and deployed as a TinyML application.

The system is designed as a workplace entry-point screening tool. It performs binary classification: mask or no mask. A person wearing a mask incorrectly (e.g. below the nose) is classified as “mask”.

2. Dataset

2.1 Source

The dataset used is from Kaggle, published by **vijaykumar1799**: https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection

License: CC0 — Public Domain

2.2 How the Dataset Was Built (from the Kaggle author)

The author states the following in the dataset description:

The dataset is a combination of two other Kaggle datasets that were cleaned and rebalanced:

Source 1: ashishjangra27/face-mask-12k-images-dataset
- Contains ~12K images total
- The "with mask" images (~6K) were scraped from Google search
- The "without mask" images were preprocessed from the **CelebFaces dataset** by Jessica Li (https://www.kaggle.com/jessicali9530)
- License: CC0

Source 2: andrewmvd/face-mask-detection
- Contains 853 images with bounding boxes in PASCAL VOC format
- Three classes: with mask, without mask, mask worn incorrectly
- License: CC0

The author took images from both sources, manually removed noisy/outlier images, and augmented the data so each class has equal distribution.

2.3 Final Dataset Composition

The dataset contains **3 folders**, each with exactly **2,994 images. Total: 8,982 images**:

Folder	Count	Description
with_mask	2,994	People wearing masks correctly
without_mask	2,994	People not wearing masks

Folder	Count	Description
mask_weared_incorrect	2,994	People wearing masks incorrectly (below nose, below chin, etc.)

2.4 What Was Used for Training vs Testing
This is a critical point for understanding the model's behaviour.

Training set (uploaded to Edge Impulse): Only **with_mask** and **without_mask** folders. The mask_weared_incorrect folder was **excluded** from training entirely.

Test set: All three folders were included. This means during model testing, the system is evaluated on incorrectly worn masks as well, even though it was never trained on that class.

2.5 File Renaming

Before uploading to Edge Impulse, all images were renamed using a Python script to standardize the naming convention:

```
import os
folder = "./Dataset/mask_weared_incorrect"
for i, file in enumerate(os.listdir(folder)):
    if file.endswith(".png"):
        os.rename(
            os.path.join(folder, file),
            os.path.join(folder, f"maskincorrect-{i}.png")
        )
```

This was run on all three folders, resulting in:

- mask-0.png, mask-1.png, ... mask-2993.png
- nomask-0.png, nomask-1.png, ... nomask-2993.png
- maskincorrect-0.png, maskincorrect-1.png, ... maskincorrect-2993.png

Edge Impulse uses the filename prefix before the hyphen as the class label, so this renaming ensured clean, consistent labelling on upload.

2.6 Bias and Demographic Limitations

The dataset does **not** include any demographic metadata (ethnicity, age, gender). The Kaggle author does not provide this information. The upstream sources also do not document it.

What we can state honestly:

- The "without mask" images originate primarily from the CelebFaces dataset, which is known to be skewed toward certain demographics
- The "with mask" images were scraped from Google, which introduces unknown selection bias depending on what search queries were used

- No demographic audit of this specific dataset has been published

This is a known limitation. For a production deployment, fairness testing on diverse subjects would be mandatory before go-live. For this academic project, we acknowledge the gap and document it explicitly rather than fabricating demographic percentages.3. Edge Impulse Configuration (Impulse #22)

3.1 Impulse Design

From the Edge Impulse interface:

Block	Setting
Input	Image, 96×96, Resize mode: Fit shortest axis
DSP	Image — Grayscale
Learning	Transfer Learning(Images)
Output	2 classes: mask, no-mask

3.2 Why Grayscale

Switching from RGB (3 channels) to Grayscale (1 channel) reduces the input data per frame from 27,648 bytes (96×96×3) down to 9,216 bytes (96×96×1). This is a 3× reduction.

For mask detection this trade-off works because the model needs to detect **shape and texture** — whether the nose/mouth region is covered by fabric — not colour. A surgical mask's edges, the contour of the face, the presence or absence of fabric across the lower face — these are all structural features that survive the conversion to grayscale. The EON Tuner results confirm this: the top grayscale model achieves 95% accuracy, which is competitive with or better than the earlier RGB models.

3.3 Why Inference Runs on an Interval, Not Continuously

The inference task runs every **5 seconds** (INFERENCE_INTERVAL = 5000). It does not process every camera frame.

This is intentional for three reasons. First, the ESP32 is dual-core: Core 0 runs inference, Core 1 runs the web server (dashboard + video stream). If inference ran continuously on Core 0, it would consume all available CPU cycles and the dashboard/stream would freeze. Second, a single inference cycle takes roughly 600–2000ms depending on the model. Running back-to-back would leave no breathing room. Third, for an entry-point screening scenario, 5-second intervals are sufficient — a person walking through a doorway is in the camera's field of view for 2–4 seconds, so they will be captured.

Thermal Management: Beyond CPU scheduling, the 5-second inference interval serves as a critical thermal management strategy. Continuous high-speed inference on the ESP32 (running at 240MHz) alongside active Wi-Fi transmission generates significant heat. This interval prevents the hardware from thermal throttling or frequency drifting, ensuring long-term stability within an industrial enclosure.

4. Incorrect Mask Handling — Design Decision

4.1 What Happens

The model was trained on only two classes: mask and no-mask. When it encounters a person wearing a mask incorrectly (below the nose, around the chin), it classifies them as **"mask"**. No alert is triggered.

This was verified during testing: the mask_weared_incorrect test images are classified as "mask" by the model.

4.2 Why This Is Acceptable

This is called **lenient mask detection**, and it is a common and deliberate approach in specific deployment contexts:

Entry-point screening (not enforcement): The system's role is to catch people who are completely bare-faced. People whose masks have slipped slightly are still attempting to comply. Flagging them would create friction and false-positive fatigue — security staff would start ignoring alerts because too many are borderline cases.

High-traffic flow management: At a busy entry point, a strict model that flags every slightly-off mask would create a bottleneck. People would queue up waiting for re-checks. The lenient model lets the flow continue while still filtering out the most obvious non-compliance.

Binary classification is the correct starting point: Adding a third class ("incorrect") would require retraining with the mask_weared_incorrect data and would complicate the decision logic. The current binary approach proves the core capability — distinguishing "something on the face" from "nothing on the face" — before adding positioning precision later.

The test confirms the behaviour is consistent: By keeping mask_weared_incorrect in the test set, we can measure exactly how the model behaves on these edge cases. It doesn't randomly flip between mask and no-mask for incorrectly worn masks — it consistently classifies them as mask. This predictability is important for documentation and for setting user expectations.

4.3 Future Path

If strict detection is needed later, the mask_weared_incorrect folder (2,994 images) is already cleaned and renamed. It can be added back to the training set to create a 3-class model. This would be a separate impulse design with a 3-output classifier.

5. EON Tuner Results — All Runs

The EON Tuner was run with the objective set to **minimize validation error**. Target hardware: Espressif ESP-EYE (ESP32 240MHz). Target latency: 100ms (this is Edge Impulse's default benchmark target — it does not reflect our actual requirement, which is <2 seconds per inference cycle given the 5-second interval).

Target RAM: 4096 kB. Target ROM: 4096 kB.

All models tested used **Grayscale** as the DSP type.

5.1 EON Tuner Config

Two search tracks were configured:

Track 1 — Custom Conv2D:

- Learning rate: 0.0005
- Training cycles: 15
- Architecture search: Conv2D layers [2, 3, 4], base filters [8, 16, 32], kernel size 3×3, dropout [0.25, 0.5]
- Input dimensions tested: 96×96, 128×128, 160×160

Track 2 — Transfer Learning:

- Learning rate: 0.0005
- Training cycles: 20
- Models tested: MobileNetV2 α=0.35, MobileNetV2 α=1.0, MobileNetV2 α=0.5, MobileNetV1 α=0.25
- Dense neurons: [16, 64], dropout: [0.1, 0.5]
- Augmentation: all
- Input dimensions tested: 96×96, 128×128, 160×160

5.2 Complete Results (Test Set, sorted by accuracy)

All results from the EON Tuner screenshots, compiled into one table:

Rank	Model	Accuracy	Latency (ms)	RAM (kB)	ROM (kB)	Input	Mask F1	No-Mask F1
1	grayscale-conv2d-9be	**95 %**	650	97	44	96×96	.96	.96
2	grayscale-mobilenetv2-e	94 %	2060	454	196	128×128	.96	.95
3	grayscale-conv2d-868	93 %	1150	167	51	128×128	.95	.94
4	grayscale-mobilenetv2-6f5	93 %	1959	444	150	128×128	.95	.94
5	grayscale-mobilenetv2-f	93 %	1173	285	196	96×96	.95	.95
6	grayscale-mobilenetv1-4	90 %	1110	129	287	96×96	.92	.93
7	grayscale-mobilenetv1-0	91 %	1184	129	287	96×96	.94	.92
8	grayscale-mobilenetv1-b	88 %	516	88	287	64×64	.9	.92
9	grayscale-mobilenetv2-2	88 %	600	163	196	64×64	.91	.9

5.3 Confusion Matrix Detail — Top 3 Models

grayscale-conv2d-9be (95% — BEST):

	Predicted: mask	Predicted: no-mask	Predicted: uncertain
Actual: mask	**95**	3	2
Actual: no-mask	3	**95**	2

F1: mask = 0.96, no-mask = 0.96. Perfectly balanced performance.

grayscale-mobilenetv2-e(94%):

	Predicted: mask	Predicted: no-mask	Predicted: uncertain
Actual: mask	**96**	2	2
Actual: no-mask	4% error (predicted mask)	**91**	3

The tooltip in the screenshot shows: "Error: 4%, Actual label: no-mask, Predicted label: mask". This is a false positive — the model incorrectly called a no-mask person as mask. F1: mask = 0.96, no-mask = 0.95.

grayscale-mobilenetv2-6f5 (93%):

	Predicted: mask	Predicted: no-mask	Predicted: uncertain
Actual: mask	**95**	4	1
Actual: no-mask	5	**91**	4

F1: mask = 0.95, no-mask = 0.94.

5.4 Why grayscale-conv2d-9be Is the Best Choice

This is the model to deploy. Here is why, point by point:

Highest accuracy: 95% on the test set — 1% above the next best model.

Lowest RAM: 97 kB. The MobileNet variants use 129–454 kB. On a device with 520 kB total SRAM, this matters. It leaves more headroom for the blur buffer, the web server, and the video streaming buffer.

Lowest ROM: 44 kB. The smallest model footprint of all candidates.

Fastest latency of the high-accuracy models: 650ms. The two MobileNet models at 93–94% take 1959–2060ms. The conv2d model is 3× faster.

Best balanced F1 scores: 0.96 for both mask and no-mask. This means it performs equally well on both classes — no class is being favoured.

Why MobileNet lost despite being designed for mobile: MobileNet uses depthwise separable convolutions, which are optimised for ARM Cortex-M processors with SIMD instructions. The ESP32 uses an Xtensa LX6 processor, which does not have the same SIMD acceleration. Standard convolutions (what the custom Conv2D model uses) actually execute more efficiently on this architecture. This is a key insight from the EON Tuner — the "best" architecture depends entirely on the target hardware.

5.5 Architecture of grayscale-conv2d-9be

From the EON Tuner classification block details:

Layer	Filters	Kernel	Dropout
conv2d	8	3×3	—
conv2d	16	3×3	—
conv2d	32	3×3	—
dropout	—	—	0.25

Learning rate: 0.0005. Training cycles: 15. Validation accuracy: 95%.

6. Algorithm Selection Justification

6.1 Why CNN (not MLP, RNN, or Transformer)

The input is a 2D image. CNNs are designed specifically for this type of data.

vs MLP (Multi-Layer Perceptron): An MLP would flatten the 96×96 image into a 9,216-element vector and connect every input to every neuron in the first layer. This destroys spatial relationships — it treats a pixel in the top-left corner as equally related to a pixel in the bottom-right. A CNN preserves spatial structure by applying small filters (3×3) that slide across the image, detecting local patterns like edges and textures. For face/mask detection, spatial relationships are everything — where the nose is relative to the mouth, whether there is fabric covering that region.

vs RNN (Recurrent Neural Network): RNNs are designed for sequential data (time series, text). A single camera frame has no temporal sequence to model. Each inference is independent — we are not tracking a person across frames or predicting what happens next. RNN is the wrong tool.

vs Transformer (Vision Transformer): Transformers can work on images by splitting them into patches, but they require massive amounts of training data and compute to work well. A ViT model small enough to run on an ESP32 would not have enough capacity to learn useful features. The custom 3-layer CNN achieves 95% with only 15 training epochs on ~6,000 images. A transformer would need orders of magnitude more data and training time for comparable results on this task.

6.2 Why This Specific CNN Architecture

The EON Tuner selected 3 convolutional layers with filter counts 8 → 16 → 32. This is a classic progressive architecture:

- **Layer 1 (8 filters):** Detects basic features — edges, simple textures. 8 filters is enough to capture the variety of edges in a face/mask boundary.
- **Layer 2 (16 filters):** Combines layer 1 features into more complex patterns — curves, partial shapes. Doubles the filter count because there are more mid-level patterns than basic edges.

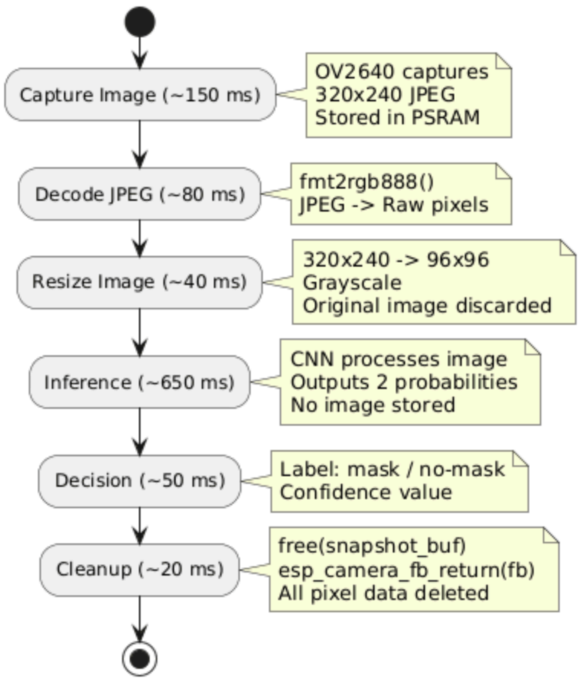
- **Layer 3 (32 filters):** Detects high-level features — "mask present across nose/ mouth" or "bare skin visible". 32 filters capture the variety of mask types, skin tones, and lighting conditions.

The dropout layer (0.25) after the dense layers prevents overfitting. With only ~6,000 training images, overfitting is a real risk. Dropout randomly drops 25% of neurons during training, forcing the network to learn redundant representations.

7. Privacy and Data Handling

7.1 What Happens to the Image

This is the key privacy argument. The image exists on the ESP32 for less than 2 seconds, then it is deleted. Here is the exact sequence:



Activity Diagram

What remains after cleanup: a string ("mask"/"no-mask"), a float (confidence 0.0–1.0), and a timestamp. This is 42 bytes of data. No face. No image. No biometric template.

7.2 What Is Transmitted

The ESP32 runs in **WiFi Access Point mode**. It does not connect to any external network. It creates its own local network that devices connect to.

The dashboard receives two things:

- **JSON status** (~200 bytes): Contains the classification result, confidence scores, and counters. No image data.

- **MJPEG video stream**: This is the live camera feed shown on the dashboard. However, every frame is **Gaussian blurred with a 15-pixel kernel** before being sent. This blur is applied by the blurJPEGFrame() function. The blur is strong enough to make facial recognition impossible — individual features like eyes, nose, and mouth are completely smeared.

7.3 GDPR Relevance

The system processes **biometric data** (facial images), which is a special category under GDPR Article 9. The key principles that apply:

Data Minimisation (Article 5(1)(c)): Only the classification result is retained. No names, no identifiers, no location data, no health data beyond "mask present or not."

Storage Limitation (Article 5(1)(e)): Image lifetime is under 2 seconds. There is no persistent storage on the device — no SD card is connected, no EEPROM is used for images, no cloud upload occurs.

Privacy by Design (Article 25): The architecture itself enforces privacy. The ESP32 hardware has no SD card slot connected. The memory is volatile — power off the device and everything disappears. The blur is applied in firmware before any data leaves the device. These are not policy decisions that could be bypassed — they are hardware and software constraints.

Lawful Basis (Article 9(2)): For a workplace deployment, the lawful basis would be either explicit consent from employees or legitimate interest (workplace safety), supported by a Data Protection Impact Assessment (DPIA).

7.4 EU AI Act Relevance

Under the EU AI Act (Regulation 2024/1689), this system would be classified as **high-risk** because it performs biometric categorisation in a workplace setting (Annex III, Category 1).

The key requirements for high-risk systems that this project addresses:

- **Article 9 (Risk Management):** Risks are documented — bias, false positives/negatives, hardware failure.
- **Article 10 (Data Governance):** The training dataset source is documented with its license and provenance.
- **Article 11 (Technical Documentation):** This document.
- **Article 14 (Human Oversight):** The system triggers an LED alert. A human (security staff) must visually verify before any action is taken. The system does not autonomously enforce compliance.
- **Article 15 (Accuracy):** Performance metrics are documented from Edge Impulse testing.

8. MLOps Workflow

MLOps (Machine Learning Operations) covers the full lifecycle from data to deployment to monitoring. Here is how each stage applies to this project:

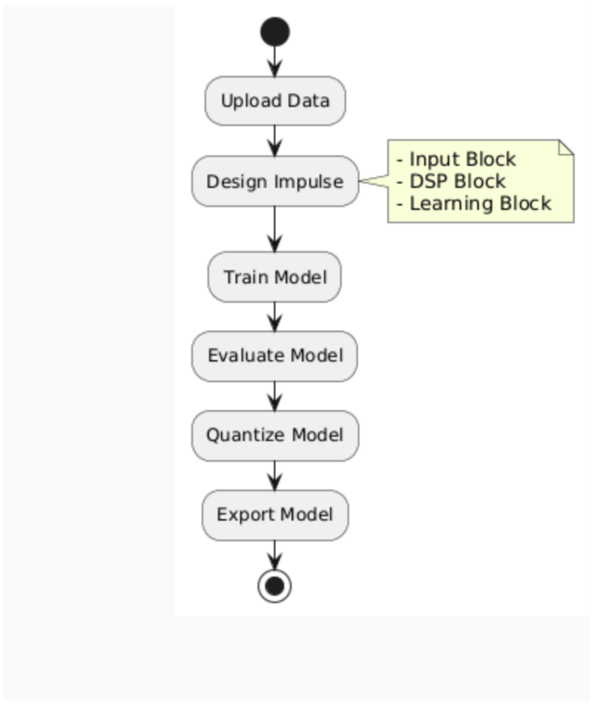
8.1 Data Management

- **Source:** Kaggle dataset, CC0 license, publicly available

- **Versioning:** The dataset was downloaded, renamed (standardised filenames), and uploaded to Edge Impulse. Edge Impulse tracks the dataset version internally per project.
- **Integrity:** The rename script ensures consistent labelling. Edge Impulse validates image format on upload.
- **Split:** Edge Impulse handles the train/validation/test split. The default split ratios apply.

8.2 Model Training

- **Platform:** Edge Impulse Studio (cloud-based training)
- **Process:**



Development Lifecycle diagram

- **Quantization:** The model is quantized from FP32 to INT8 by Edge Impulse's EON Compiler. This reduces model size and speeds up inference on the ESP32. The accuracy impact is tracked by comparing validation vs test set performance.
- **Versioning:** Edge Impulse assigns version numbers to each impulse. The EON Tuner runs are logged with timestamps (visible in the screenshots: 1/31/2026, 10:03:00 PM etc.).

8.3 Deployment

- **Export:** Edge Impulse exports the model as an Arduino library (C++ header files with the INT8 weights embedded)
- **Target:** ESP32-CAM (AI Thinker board)
- **Upload:** Via Arduino IDE, flashed over USB-UART
- **Verification:** Serial monitor confirms "SYSTEM READY", dashboard is accessible, inference runs every 5 seconds

8.4 Monitoring

The system logs the following to the serial console in real time:

- Every inference result (class, confidence, mask/no-mask probabilities)
- Every 10 inferences: a summary block
- Every 15 seconds: a violation count report
- Every 5 minutes: a system health check (free heap memory, error counts, uptime)

The dashboard provides a live view of:

- Current detection and confidence
- Confidence bar chart (mask vs no-mask probabilities)
- Violation counter (resets every 15 seconds)
- Countdown to next report
- Live blurred video stream
- Connection status indicator (green dot = connected, red = disconnected)

8.5 Retraining Triggers

In a production MLOps pipeline, retraining would be triggered when:

- Accuracy drops below a threshold (e.g., <90% on spot checks)
- False positive or false negative rates increase significantly
- The confidence score distribution shifts (model becomes less certain on average)
- The deployment environment changes (different lighting, different distances)

For this project, the Edge Impulse versioning system supports retraining: upload new data, retrain, compare against previous version, deploy if better.

9. System Architecture and Inference Pipeline

9.1 Hardware

Component	**Specification**	**Function / Usage**
Brain	Espressif ESP32-D0WDQ6	Dual-core Xtensa® LX6 @ 240MHz
Memory (RAM)	520 KB SRAM + **4 MB PSRAM**	PSRAM handles the high-res image buffers
Storage (Flash)	4 MB SPI Flash	Holds the firmware and ML model weights
Sensor	OV2640 (2 Megapixel)	Standard lens; supports JPEG/Grayscale/RGB
Connectivity	2.4 GHz Wi-Fi & Bluetooth	Set to **AP Mode**
Onboard LED	GPIO 4 (High-power Flash)	Used as a status/alert indicator
External Storage	MicroSD Slot	Support for FAT/FAT32

9.2 Dual-Core Task Split

Core	**Tasks**	**Why**
Core 0	CNN inference (every 5s)	Inference is CPU-intensive (~650ms). Isolating it on one core prevents it from blocking other operations.

Core 1	Web server, dashboard, MJPEG stream, status API	These need to respond quickly to HTTP requests. They run independently of inference timing.

A mutex (detectionMutex) protects the shared detection result variables. The inference task writes the result; the status API reads it. The mutex ensures the API never reads a half-updated value.

9.3 Inference Timing (grayscale-conv2d-9be)

Stage	Time	Notes
Camera capture	~150 ms	JPEG from OV2640
JPEG decode	~80 ms	fmt2rgb888
Grayscale conversion + resize to 96×96	~40 ms	Single channel, smaller target
CNN inference	~650 ms	INT8 quantized, per EON Tuner
Post-processing	~50 ms	Parse results, LED control
Cleanup	~20 ms	free() buffers
Total	**~990 ms**	Well within 5s interval

The remaining ~4 seconds in each cycle are available for streaming video frames and serving dashboard requests.

9.4 Memory Footprint

Component	Size
Model weights (INT8, in flash)	44 kB ROM
Runtime RAM during inference	97 kB
Camera frame buffer (in PSRAM)	~15–20 kB per frame
Blur buffers (temporary)	~230 kB peak (RGB decode of 320×240)
Web server + dashboard string	~8 kB

Peak RAM usage stays well under the 520 kB SRAM limit. PSRAM handles the large frame buffers.

10. Code Structure Summary

The production code (ESP32_Mask_Detection_Working.ino) is organised as follows:

Configuration block: Camera pin definitions, system constants (blur strength, confidence threshold, inference interval), WiFi credentials.

Blur functions: applyGaussianBlur() performs a separable 2-pass box blur on a single channel. blurJPEGFrame() decodes a JPEG frame to RGB, separates into R/G/B channels, blurs each, merges back, and re-encodes to JPEG. All temporary buffers are freed after use.

Edge Impulse integration: ei_camera_get_data() feeds pixel data to the Edge Impulse classifier. ei_camera_capture() captures a frame and resizes it to the model's input dimensions.

Core detection logic: runMaskDetection() orchestrates one inference cycle — allocate buffer, capture, run classifier, parse results, update shared state, control LED, free buffer.

Inference task: inferenceTask() runs on Core 0. It checks the elapsed time and triggers runMaskDetection() every 5 seconds.

Web server handlers:

- handleDashboard() - serves the HTML/CSS/JS dashboard page
- handle_jpg_stream() - MJPEG stream with blur applied to every frame
- handleStatus() - JSON API returning current detection state (no mutex needed, reads cached values)

Setup and loop: setup() initialises camera, WiFi AP, web server, mutex, and creates the inference task. loop() handles web client requests and prints the 15-second violation report.

11. Known Limitations

Limitation	Severity	Details
Demographic bias	Medium	Training data source (CelebFaces + Google scrape) likely overrepresents certain ethnicities. No demographic audit exists for this dataset.
Incorrect mask not flagged	By design	Masks worn below nose/chin are classified as "mask". This is intentional for lenient screening.
Distance	Medium	Accuracy degrades beyond ~2m as the face occupies fewer pixels in the frame.
Lighting	Medium	Very low light (<50 lux) or strong backlighting degrades accuracy. Training data is primarily well-lit indoor scenes.
Beards	Low-Medium	Dense dark beards can visually resemble a dark mask. The model may misclassify.
Single face	Low	The model classifies the dominant face in the frame. Multiple faces are not individually tracked.