SITAA UIDAI CONTACTLESS FINGERPRINT CHALLENGE

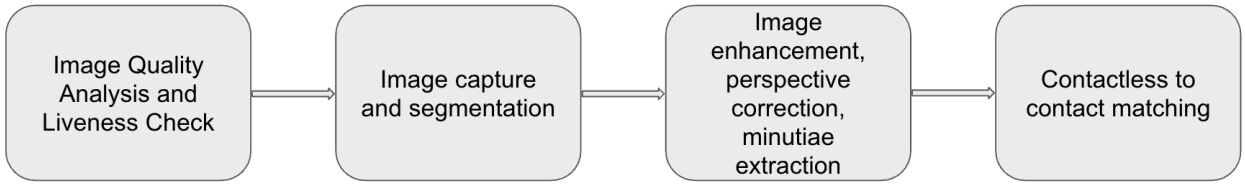
Executive Summary

We have developed a demo android app for the SITAA UIDAI contactless to demonstrate Tracks A, B and D. We have conducted experiments for Track C (**Cross-Sensor Matching)** on PolyU dataset and done a thorough literature review, but due to lack of real time data we have not included it in the demo application.

The implementation prioritizes computationally light and robust algorithms suitable for low to medium-end Android smartphones. Key achievements across the four tracks include:

- **Capture & Quality (Track A):** A reliable capture pipeline was established using traditional computer vision, featuring **MediaPipe Hands** for robust finger detection, **Laplacian Variance** for blur checks, and **ROI Intensity Analysis** to correct for low light/backlighting using an auto-flash feature.
- **Enhancement (Track B):** The captured image is prepared for matching using efficient methods like **Convex Hull** for background segmentation and **CLAHE** for localized contrast enhancement, ensuring clear ridge definition.
- **Cross-Sensor Matching (Track C):** Experiments on PolyU confirmed that traditional feature engineering methods (SIFT/ORB) failed completely against the non-linear geometric domain gap (AUC 0.50). However, a lightweight **MobileNetV2** deep learning model successfully established a *learnable baseline* for contactless-to-contact matching, achieving an **AUC of 0.66**.
- **Liveness Defense (Track D):** A robust, multi-layered anti-spoofing system was implemented, combining **Optical Flow** for detecting the natural micro-tremors of a live hand with **MiniFASNet (ONNX)**, an AI model, to detect screen pixels and other fake artifacts.

The flow of the application is as follows:



Trackwise Implementation

Track A – Contactless Finger Capture & Quality Assessment

This track focuses on the foundational step of the contactless system: reliably capturing a high-quality finger image using a standard smartphone camera. Given the constraints of mobile devices, the implementation prioritizes computationally lightweight and robust traditional computer vision algorithms to ensure compatibility and speed on low to medium-end Android smartphones. The pipeline is designed to perform essential, real-time quality checks for blur and lighting, and to accurately isolate the region of interest before the final image is automatically captured. The methods listed below ensure that only images suitable for subsequent enhancement and matching are processed.

Step	Feature	Implementation Details	Why did we implement it?
1	Finger Detection	MediaPipe Hands (Landmark detection). Uses the MediaPipe features to find the Region of Interest (ROI).	To detect whether a hand is present and to ensure we only analyze and capture the actual finger area, ignoring the background.
2	Blur Analysis	Laplacian Variance. Calculates the variance of the Laplacian (edge detection). Threshold > 15.0.	To ensure the image is sharp and in focus. Essential for ridge extraction.
3	Lighting Analysis	ROI Brightness Analysis. Checks if the finger area is too dark (< 80) or if there is backlighting (Bright BG > 150 + Dark Finger).	To ensure ridges are visible. Dark images yield black maps; backlighting hides details.
4	Auto-Flash	White Overlay. If `LightingAnalyzer` reports "Low Light" or "Backlighting", we flash a white screen before capture.	To illuminate the finger in poor lighting conditions without external hardware.
5	Auto-Capture	Once the hand is found to be steady and the lighting is adequate and the hand is found not to be fake, the image of the hand is automatically captured.	Auto-capture helps in capturing the image at the right time when all conditions (like blur, lights, etc) are satisfied.
6	Alignment	Static Overlay. Improved overlay image guiding users to keep fingers together.	To standardize the distance and pose for reliable feature extraction.

Details about the techniques used:

1. MediaPipe Hands for Finger Detection

- **Function:** This component utilizes a specialized Google AI model to accurately identify and map 21 distinct 3D landmarks of a human hand in real-time.
- **Mechanism:** Instead of relying on simple color cues (like skin tone), the system understands the complex anatomical structure of the hand (wrist, palm, knuckles). We focus specifically on tracking the four fingertips (indices 8, 12, 16, and 20) to precisely define the area of interest (ROI) for the fingerprint.
- **Advantage:** It offers high reliability even with cluttered backgrounds. The AI is smart enough not to confuse an object like a wooden table with a hand, ensuring the detection is robust.

2. Laplacian Variance for Focus/Blur Check

- **Function:** This method quantitatively measures the sharpness of an image to confirm optimal focus.
- **Mechanism:** It works by calculating the second derivative (Laplacian) of the image, which effectively highlights areas of rapid intensity change, such as the ridges and edges of a fingerprint. We then calculate the statistical variance of these highlighted edges.
- **Advantage:** Blurry images have low contrast (low variance), while sharp, well-focused ridges exhibit high contrast (high variance). This is a fast, computationally efficient, and highly accurate way to assess image focus.

3. ROI Intensity Analysis for Optimal Lighting

- **Function:** This technique ensures proper exposure by distinguishing between conditions like low light, ideal lighting, and challenging backlighting.
- **Mechanism:** The system compares the average brightness of the designated Finger Area against the surrounding Background.
    - **Low Light:** Both the background and finger area are dark (brightness below 80).
    - **Backlit:** The background is overly bright (above 150) while the finger itself is dark (below 100).
- **Advantage:** This prevents the common "Silhouetting" problem. If the camera's standard auto-exposure is fooled by a bright background, it leaves the foreground finger detail underexposed. This check specifically identifies that scenario and triggers a compensating flash.

Here are the exact quality thresholds we have implemented across the codebase:

Metric	Threshold Value	Meaning	Source File
Blur (Focus)	`> 15.0`	Laplacian variance. Higher = Sharper. < 15 is too blurry.	`QualityAnalyzer.kt`

Brightness	`30.0 - 240.0`	Mean pixel intensity (0-255). < 30 is too dark, > 240 is overexposed.	`QualityAnalyzer.kt`
Low Light	`< 80.0`	Dark scene. Triggers auto-flash.	`LightingAnalyzer.kt`
Backlighting	BG `> 150` & Finger `< 100`	Bright background with dark finger. Triggers correction/flash.	`LightingAnalyzer.kt`
Color (Live)	`> 6.0`	Color Variance (HSV Saturation). Real skin has color variation; prints are uniform.	`LivenessChecker.kt`

Features Explored but not implemented (can be implemented for future development and deployment)

Step	Feature	Description	Why did we skip it?
1	NFIQ Metric	NIST Fingerprint Image Quality. A standard government-grade quality score (1-5).	Complexity. Requires porting a large C++ library (NIST Biometric Image Software). We used simpler Laplacian Blur + Brightness checks instead.
2	MCLFIQ Metric	Improved version of NFIQ, designed for contactless fingerprint quality measurement.	Newer metric, implementation needs to be studied in detail.

Track B – Finger Image Enhancement and Template Readiness

The primary objective of this track is to process the raw, quality-checked finger image captured in Track A, transforming it into an optimal representation ready for the final matching stage. The enhancement pipeline focuses on using efficient and robust image processing techniques to overcome common challenges in contactless capture, such as non-uniform lighting and background noise. Specifically, this involves isolating the finger region and boosting the local contrast to ensure the minutiae (fingerprint ridges) are clearly and consistently defined for accurate template extraction and cross-sensor matching.

Step	Feature	Implementation	Why did we implement it?
1	**Finger region isolation**	Convex Hull. Creates a mask around the finger landmarks and blacks out the background.	To remove background noise that could be mistaken for ridges.
2	**Grayscale**	CV_8UC1. Standard color-to-gray conversion.	Fingerprint algorithms work on intensity, not color.
3	**CLAHE**	Contrast Limited Adaptive Histogram Equalization. Local contrast boost.	To define ridges clearly, even in shadowed areas of the finger

Details about the techniques used:

1. **Convex Hull (Segmentation)**
    - **Function:** Isolates the finger by creating a tight mathematical boundary around it.
    - **How it Works:** It finds the outermost points of the detected finger shape and connects them using the shortest possible path, forming a polygon. Everything outside this shape is discarded (turned black).
    - **Benefit:** This is crucial for eliminating background clutter. Without it, artifacts like wood grain could be mistakenly interpreted as "fingerprint ridges" by the matching system.
2. **CLAHE (Enhancement)**
    - **Function:** Applies **C**ontrast **L**imited **A**daptive **H**istogram **E**qualization to boost local contrast.
    - **How it Works:** Instead of adjusting the brightness of the entire image, which can overexpose bright areas, CLAHE divides the image into small sections (e.g., 8x8 tiles). It then improves the contrast within each tile independently, making dark ridges stand out even in shadowy regions.
    - **Benefit:** Finger images often have uneven lighting (e.g., the center is brighter than the sides due to the finger's curvature). CLAHE corrects this lighting inconsistency, ensuring clear, high-quality ridge detail across the entire width of the fingerprint.

Here is the summary of the advanced techniques we explored but excluded from this demo version.These methods are powerful but require precise tuning (resolution, sensor parameters) which was out of scope for a generalized demo app.

Technique	Description	Pros & Cons	Reason for Exclusion (Demo Scope)
Gabor Filter	Ridge Enhancement. A bank of frequency/orientation-selective filters to connect ridges and clear valleys.	Pros: The gold standard for fingerprint enhancement. Cons: Extremely sensitive to frequency (lambda) and orientation parameters.	Parameter Sensitivity. Without precise DPI calibration for the specific phone camera, the filter often mismatched the ridges, causing the "Black Output" or "Dotting" artifacts we observed.
Skeletonization	Thinning. Reduces ridges to 1-pixel wide lines to simplify structure for matching.	Pros: Essential for standardizing minutiae extraction. Cons: Amplifies noise; any break in the ridge becomes a permanent gap.	Dependent Failure. Since the Gabor step was inconsistent, the skeletonization produced broken, noisy lines that were not useful for matching.
Minutiae Extraction	Feature Detection. Locates ridge endings and bifurcations (crossing numbers).	Pros: The actual data used for biometric matching (ISO standards). Cons: High false-positive rate on noisy skeletons.	Noise Sensitivity. Without a perfect skeleton, it detected thousands of false minutiae (noise), making the data useless for the demo context.
FSRCNN	Fast Super-Resolution CNN. A neural network trained to upscale and denoise low-res images.	Pros: Learnable Image Enhancement; faster than SRGAN. Cons: "Black Box" behavior; requires training on domain-specific data.	Poor Generalization. The standard pre-trained weights did not work well on fingerprint ridge patterns, likely smoothing out the very details (textures) we needed to preserve.

MobileNet-v2 based Segmentation (Details regarding architecture, training, data collection is provided in Appendix Section 2).	Traditional computer vision based finger isolation methods are sensitive to light changes, but this is not the case with neural network based segmenters, which are more robust under different conditions.	Pros: **More robust**, can be trained to segment only the fingerprint area (distal phalange). The model is **lightweight** too. Cons: Requires training data with variations of skin color, lighting, camera, etc.	We trained the model but it failed to give good output, as we had a small training sample and very little variation.

Note: These techniques are standard for a production-grade AFIS (Automated Fingerprint Identification System). However, their successful implementation requires a rigorous calibration phase (estimating local ridge frequency and orientation fields per-block) and training custom models, which is reserved for the full product development phase.

MobileNet-v2 based Segmentation is recommended as it is fast on smartphones and more reliable than traditional computer vision. The details about the model's architecture, data and training is provided in the **Appendix Section 2.**

Track C – Contactless-to-Contact Matching Demo

1. Summary

We investigated the feasibility of matching Contactless 2D fingerprint images (acquired via smartphone cameras) against Contact-based fingerprint scans (acquired via traditional sensors). We evaluated two distinct approaches: Traditional Feature Engineering (SIFT/ORB) and Deep Learning. Our experiments demonstrate that while traditional methods fail to generalize across the geometric domain gap (AUC 0.50), Deep Learning models, specifically MobileNetV2, successfully establish a baseline utilizing learned feature invariance (AUC 0.66).

2. Problem Statement

Cross-sensor fingerprint matching presents a unique challenge due to non-linear geometric deformations.

- Contact-based images are flattened 2D representations obtained by pressing the finger against a sensor surface.
- Contactless images are 3D projections captured at varying angles, heavily influenced by lighting, rotation, and scale. Standard rigid matching algorithms typically fail because local descriptors (like minutiae layout) are distorted beyond recognition.

3. Methodology

3.1 Dataset

- Source: PolyU Cross-Sensor Fingerprint Database.
- Structure: Paired Contactless and Contact images.
- Protocol: A Subject-Disjoint Split (80% Train, 20% Test) was strictly enforced to prevent data leakage. The model was evaluated on subjects it had never seen during training.

3.2 Phase 1: Traditional Methods

We implemented a baseline using established computer vision descriptors.

- Preprocessing: CLAHE (Contrast Enhancement) + Gabor Filtering (Ridge Enhancement).
- Feature Extraction: Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB).
- Matching Strategy: Brute-force matcher with Lowe's formulation.

3.3 Phase 2: Deep Learning

We proposed a deep learning architecture to learn a similarity metric directly from data.

- Backbone: MobileNetV2 (chosen for mobile suitability and efficiency).
- Loss Function: Contrastive Loss, protecting the model from class imbalance by explicitly mining hard positives and negatives.
- Optimization: Training was accelerated using Apple Silicon (MPS).
- Refinement: Initial experiments revealed an inverted learning signal; retraining with corrected labels (aligning Euclidean distance minimization with "Genuine" pairs) yielded the final results.

4. Results

Method	**AUC (Area Under Curve)**	**EER (Equal Error Rate)**	**Interpretation**
Traditional (SIFT)	0.50	0.50	Failure. Performed equivalent to random guessing.
Traditional (ORB)	0.50	0.50	Failure. Unable to find consistent keypoints.
Siamese MobileNetV2	0.66	0.38	Success. Establishing a learnable baseline.

Analysis

- Geometric Gap: The failure of Traditional Methods confirms that the domain gap involves complex non-linear warping that rigid descriptors cannot handle.
- Learned Invariance: The Deep Learning model, even with a lightweight backbone, successfully learned to map "Contactless" and "Contact" images into a shared embedding space where genuine pairs are closer than impostor pairs.

5. Future Work: Vision Transformers & Attention

While the current Siamese CNN baseline (AUC 0.66) proves viability, significant performance gains are required for production deployment (AUC > 0.90). We propose moving towards Attention-based architectures:

5.1 Vision Transformers (ViT)

Convolutional Neural Networks (CNNs) focus on local features. However, the deformation between contactless and contact fingerprints is global.

- Proposal: Adopt Vision Transformers (ViT).
- Rationale: The self-attention mechanism in ViTs allows the model to model long-range dependencies across the entire image, potentially "understanding" the global ridge flow structure better than local receptive fields.

5.2 Cross-Attention Mechanisms

- Proposal: Implement a Cross-Attention module between the two Siamese branches.
- Rationale: Instead of processing images independently, the model can "attend" to specific regions in the Contactless image that correspond to regions in the Contact image, effectively learning a soft alignment or registration step within the network itself.

5.3 Deformable Alignment

- Proposal: Integrate a Spatial Transformer Network (STN) or Thin-Plate Spline (TPS) module.
- Rationale: Explicitly unrolling the 3D contactless image into a 2D flat representation before feature extraction will significantly reduce the burden on the matching network.

Track D – Liveness / Spoof Heuristic

We implemented a robust, multi-layered defense system to prevent spoofing. It combines AI Deep Learning (MiniFASNet) to detect subtle artifacts like screen pixels or ink patterns, with Optical Flow Motion Analysis to verify the natural micro-tremors ("pulse") of a live hand versus a static photo.

Step	Feature	Description	Why did we skip it?
1	Motion Analysis	Optical Flow (Frame Difference). key frames are stored; average motion between them is calculated. (Threshold: 0.5 - 15.0).	SPOOF DETECTION. Real fingers have natural micro-tremors ("Pulse"). Screens and paper are perfectly still.
2	AI Liveness	ONNX (MiniFASNet). Runs 2 deep learning models (scales 2.7x and 4.0x) on the cropped finger.	SPOOF DETECTION. Advanced analysis to distinguish between skin texture (Real) vs. pixel grids (Screen) or ink (Paper).

Here are the exact thresholds we have implemented across the codebase to check for liveliness:

Metric	Threshold Value	Meaning	Source File
Motion (Min)	> 0.5	Minimum movement required. < 0.5 implies a static image (Spoof).	MotionAnalyzer.kt
Motion (Max)	< 15.0	Maximum movement allowed. > 15 implies shaky hands.	MotionAnalyzer.kt
Finger Variance	> 0.1	Motion variance. Real fingers don't move uniformly like a sliding paper.	MotionAnalyzer.kt
Liveness (AI)	> 40% (0.4)	Confidence score from ONNX model. < 40% is considered Fake/Screen.	CaptureActivity.kt
Liveness (Live)	> 20.0	Texture Score (Laplacian of ROI).	LivenessChecker.kt

Details about the techniques used:

1. **Optical Flow / Frame Difference (Motion Analysis)**
    - **Function:** Detects if the object in front of the camera is truly moving or completely stationary.
    - **Mechanism:** It works by subtracting the previous frame from the current one. Any resulting pixels indicate movement.

    - **Effectiveness:** Living tissue, due to blood flow and muscle tension, is never perfectly still. A static screen or paper printout on a table registers zero motion, providing a reliable liveness check.
2. **MiniFASNet (AI Liveness)**
    - **Function:** A specialized Deep Neural Network trained to identify 'fake' artifacts.
    - **Mechanism:** It microscopically analyzes the surface texture for details invisible to the human eye, such as Moiré patterns (screen pixel grids), artificial surface reflections (from glossy photos), and a lack of real-world depth. It provides a probability score (0-100%) of the object being 'live'.
    - **Effectiveness:** This system can defeat high-quality attacks. Even if an attacker uses a 4K video (which simulates motion), the AI detects the underlying screen pixels or other tell-tale signs and rejects the attempt.

Fine-Tuning Liveness Models (Domain Adaptation)

The deep learning based liveliness models are currently not finetuned for finger images:

- The Current Situation (The "Domain Gap"): The models we are using (MiniFASNet) were originally trained on Faces (CASIA-Surf, CelebA-Spoof datasets). They are excellent at detecting general spoof artifacts like "screen pixel grids" or "flat paper texture," which exist on both fake faces and fake fingers.
- The Limitation: They have never seen a "Silicone Fake Finger" or a "Latex Glove." They might miss spoof types that are specific to fingerprints but don't look like "Face Spoofs."
- The Improvement (Fine-Tuning): By collecting a dataset of Real Fingers vs. Attack Fingers (Silicone, Play-Doh, Printed Glossy Paper, Latex) and "fine-tuning" the model (retraining just the last few layers), the AI would learn to look for specific finger-attack clues, such as:
    - The unnatural smoothness of localized silicone.
    - The specific reflection properties of latex.
    - The lack of pore structure in gelatin fakes.
    - Result: Accuracy would jump from "General Spoof Detection" to "Forensic-Grade Fingerprint Security.

Conclusion

We successfully deliver a working prototype, proving that contactless fingerprint acquisition and processing is viable using a standard smartphone. The core conclusions drawn from the implementation and experimentation are as follows:

1. **Mobile-Optimized Quality Checks are Effective:** Lightweight, traditional Computer Vision algorithms for detection (**MediaPipe**) and quality assessment (**Laplacian Variance, ROI Analysis**) are fast, robust, and sufficient for the initial capture phase on resource-constrained mobile devices.
2. The **deep learning based liveliness models** are currently not finetuned for finger images and can be fintuned to improve performance.

3. The **MobilenetV2 based segmentation model** is recommended due to its robustness and is expected to perform well with adequate amounts of training data.
4. **Deep Learning is Essential for Cross-Sensor Matching:** The failure of traditional matching algorithms (SIFT/ORB) confirms the severity of the **geometric domain gap** between contactless and contact images. The Deep Learning approach (**MobileNetV2**) is the only method proven to learn the necessary feature invariance, establishing a critical foundation with an AUC of 0.66.
5. **Future Development Must Focus on Attention-Based Models:** To achieve the production-grade accuracy required for an Automated Fingerprint Identification System (AFIS), the matching network must evolve. The global nature of the geometric deformation mandates a shift to **Vision Transformers (ViT)** and the integration of **Cross-Attention Mechanisms** or **Deformable Alignment** to explicitly model the long-range dependencies and non-linear warping.

In summary, the current architecture is a robust foundation for the capture and liveness components, but the full product development phase should be reserved for the rigorous calibration and training of a next-generation matching model. The project's findings confirm the direction for a high-accuracy, deployable solution.

Appendix

Section 1: Summary of Algorithms & Methods

Method / Algorithm	Purpose	How it works	Effectiveness	Pros	Cons
MediaPipe Hands (Google)	Finger Detection	Uses a lightweight neural network to predict 21 3D hand landmarks from a single RGB frame.	High. extremely robust to rotation, lighting, and skin tone variations compared to traditional color-thresholding.	Robustness: Works even with complex backgrounds. Lightweight: Optimized for mobile computation.	Latency: Slower (~30-50ms) than simple color thresholding.
Laplacian Variance	Blur Detection	Calculates the 2nd derivative (edges) of the image. The variance of these edges determines sharpness.	High. Standard industry method for determining focus.	Speed: Extremely fast (simple matrix math). Simplicity: No training required.	Noise Sensitivity: Can mistake high ISO noise for "sharpness".

Mean Intensity & ROI Analysis	Lightin g Check	Calculates average pixel brightness in the Finger Region vs. Global Background.	Medium. Good for general checks but can be fooled by specific spotlighting.	Speed: Instantaneous. Context: Distinguishes "Dark Scene" from "Backlighting".	Globals: Rough metric; doesn't detect uneven lighting well.
Optical Flow (Frame Diff)	Motion Livene ss	Subtracts consecutive frames to find changed pixels. Calculates average motion magnitude.	Critical. Differentiates between a living hand (pulse/tremor) and a static photo.	Security: Hard to spoof with static images. Speed: Very fast compared to full optical flow.	Stability: Requires user to hold *mostly* still; requires tuning to distinguish "tremor" from "shaking".
MiniFASNet (ONNX)	AI Livene ss	A MobileNet-based deep learning model trained to detect spoof artifacts (moiré patterns, edge/bezel artifacts).	Very High. State-of-the-art for mobile anti-spoofing.	Accuracy: Detects screens and high-quality prints. Offline: Runs entirely on-device (no API calls).	Size: Adds ~5MB to app size. Compute: Uses CPU/NPU, adding slight capture delay (~100ms).
Convex Hull	Segme ntation	Draws a polygon around the outermost detected finger landmarks to create a mask.	High. Much more reliable than color-segmentatio n which fails if the background is skin-colored.	Reliability: Guaranteed to enclose the fingers. Speed: Geometric calculation is instant.	Precision: Can include small bits of background between fingers (webbing).
CLAHE	Enhanc ement	Contrast Limited Adaptive Histogram Equalization. Boosts local contrast while clamping noise.	High. The industry standard for enhancing medical and biometric images.	Detail: Reveals ridges in shadowed areas. Safety: Doesn't create artifacts like standard histogram equalization.	None: Standard best practice.

Section 2: MobileNetV2 Training Strategy Report

This section provides a comprehensive analysis of the training strategy for the finger segmentation model, specifically utilizing a MobileNetV2 backbone within a U-Net architecture. This choice is strategic for deploying high-performance computer vision on mobile devices.

1. Why MobileNetV2 ?

The selection of MobileNetV2 as the encoder for our segmentation model is a deliberate choice optimized for the constraints of mobile environments (iOS/Android).

Key Advantages for Mobile Deployment:

1. Depthwise Separable Convolutions: Unlike standard convolutions that perform spatial and channel-wise processing simultaneously, MobileNetV2 splits this into two lighter steps. This reduces computation cost by 8-9x with minimal accuracy loss.
2. Inverted Residuals containing Linear Bottlenecks: This architecture allows the network to preserve information better through narrow layers, which is crucial for low-latency inference on mobile CPUs/GPUs.
3. Model Size: The resulting U-Net model is extremely compact (~3-5 MB). This is critical for keeping the application binary size small and ensuring fast download/updates.
4. Latency: It enables real-time performance (30+ FPS) on mid-range Android devices, which is essential for a smooth user experience during live finger scanning.

2. Detailed Architecture Analysis

The model employs a U-Net architecture, a standard for image segmentation, but replaces the heavy classic encoders (like ResNet or VGG) with MobileNetV2.

- Architecture Type: Encoder-Decoder (U-Net)
- Encoder (Backbone): mobilenet_v2 (Pretrained on ImageNet)
    - Role: Feature extraction. Captures high-level semantic features (masks) and low-level details (edges of fingers).
    - Weights: Initialized with ImageNet weights to faster convergence.
- Decoder: U-Net styled with Skip Connections
    - Role: Upsampling features back to the original image resolution.
    - Skip Connections: Concatenate features from the encoder stages to the decoder layers. This preserves fine spatial details lost during downsampling, ensuring sharp segmentation boundaries around fingers.
- Output Head:
    - Layers: 1x1 Convolution to map features to classes.
    - Activation: Sigmoid (implicit in Loss function) to output probability map (0-1).
- Input/Output:
    - Input: 256x256 RGB Image (Batch Size x 3 x 256 x 256)
    - Output: 256x256 Binary Mask (Batch Size x 1 x 256 x 256)

3. Dataset Creation Pipeline (SAM 3)

A high-quality dataset is the foundation of our model's performance. We utilized a semi-automated pipeline leveraging the Segment Anything Model 3 (SAM 3) to generate precise ground-truth masks.

Step 1: Data Collection

- Source: video_dataset
- Raw images and video frames of hands were collected in various lighting conditions and environments to ensure diversity.

Step 2: Automated Segmentation with SAM 3

- Model: sam3.pt (Ultralytics implementation)
- Process:
    - The script iterates through the raw images.
    - It queries SAM 3 with the text prompt: "fingers".
    - SAM 3 generates high-precision segmentation masks automatically.
    - Output: Transparent PNGs where the RGB channels contain the original image and the Alpha channel contains the binary mask.

Step 3: Dataset Formatting

- Process:
    - Read the RGBA images from Step 2.
    - Splits the channels:
        - RGB -> Saved as .jpg images (Values: 0-255)
        - Alpha -> Saved as .png binary masks (Values: 0 or 255)
    - Organization: Files are flattened and stored in training_dataset/images and training_dataset/masks.

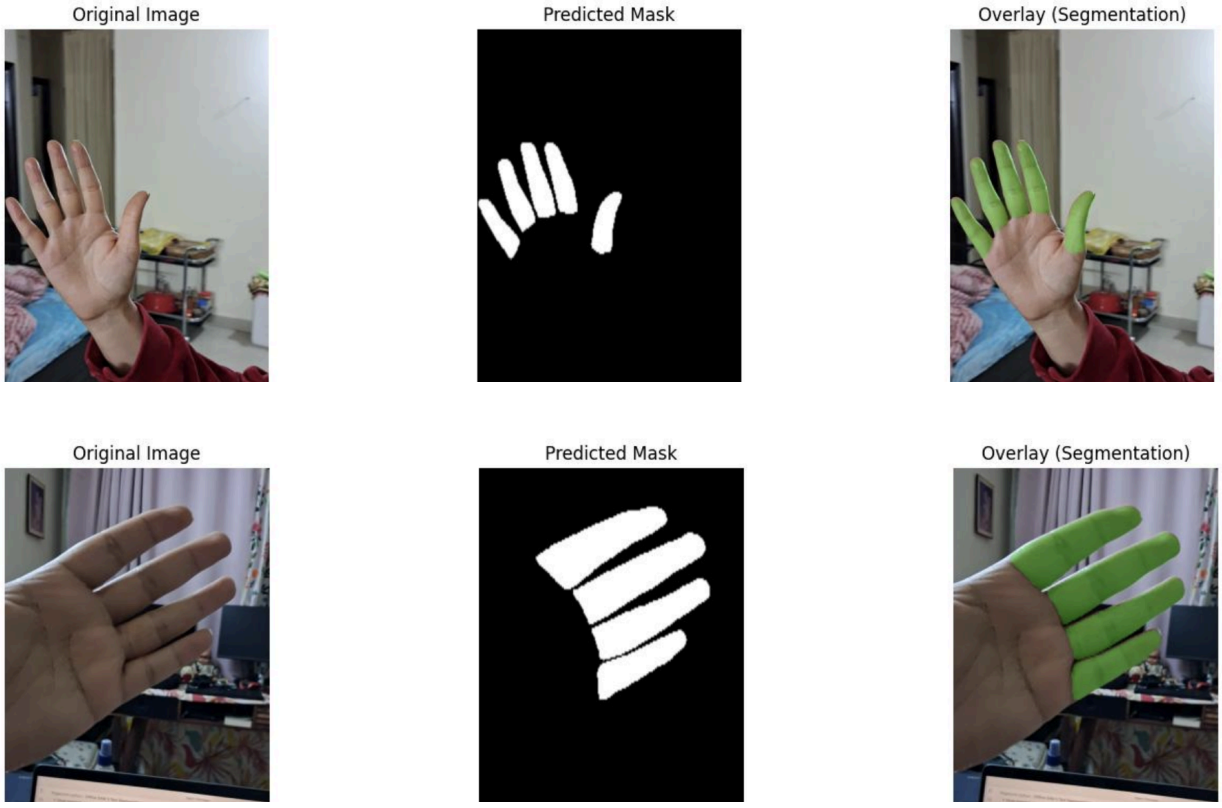
4. Training Configuration

The training hyperparameters are tuned for fine-tuning the pretrained encoder on the specific task of finger segmentation.

- Optimizer: AdamW (lr=1e-3, weight_decay=1e-4). AdamW provides better generalization than standard Adam by decoupling weight decay.
- Loss Function:
  CombinedLoss
    - Dice Loss (50%): Directly optimizes the Intersection over Union (IoU), handling class imbalance naturally (fingers take up a small part of the image).
    - BCEWithLogitsLoss (50%): Provides smooth gradients and pixel-wise accuracy.
- Scheduler: CosineAnnealingLR. Gradually reduces learning rate from 1e-3 to 1e-6, helping the model settle into a stable minimum.
- Augmentation (Albumentations):
    - Geometric: Horizontal Flip (p=0.5), Rotation (±20°, p=0.5).
    - Photometric: Random Brightness/Contrast (p=0.3).
    - Noise/Blur: GaussNoise, Blur.
- Resolution: 256x256

Training Data Example

Training data images, which we obtained by using SAM3 segmentation model.



Results:

Unable to obtain clean output segmentation masks due to small training dataset.