

# SKIN TYPE DETECTION SYSTEM

AI-Powered Analysis using MobileNetV2

**Submitted By:** Anushka Rathour | 12200884

**Submitted To:** Ms Sangeeta Bhatti



# The Evolution of Diagnostics

## ✗ Manual Analysis

Traditional skin assessment relies heavily on subjective human perception. Factors like ambient lighting and lack of expertise often lead to misclassification (e.g., confusing 'Oily' with 'Combination'), resulting in ineffective treatments.

## ✓ AI Precision

Our solution leverages Computer Vision to eliminate bias. By analyzing pixel-level texture patterns using deep learning, we provide an instant, consistent, and objective skin profile.

# Dataset Analytics

2,250

Total Samples

5

Classes (Dry, Oily, etc)

450

Images Per Class



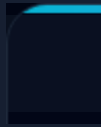
Perfectly Balanced Dataset

# | Technology Stack



**Python**

Core Logic & Data  
Processing



**TensorFlow**

MobileNetV2 Implementation



**OpenCV**

Real-time Face Extraction

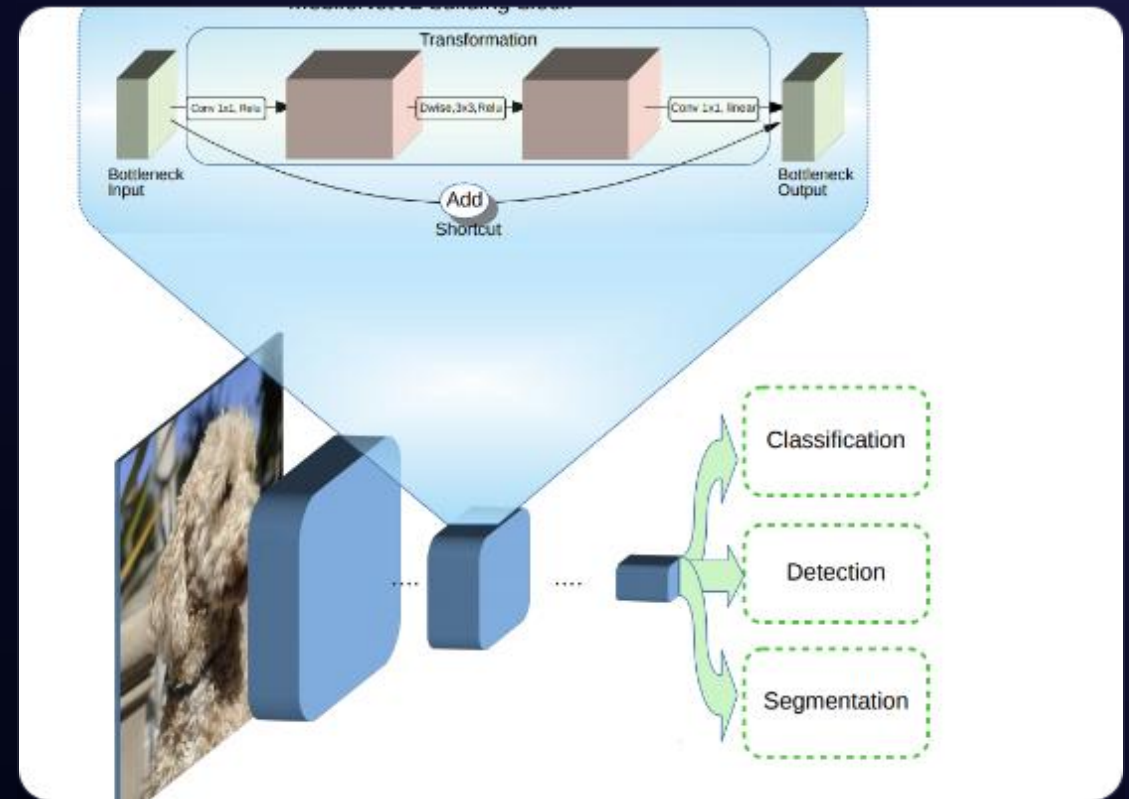


**FastAPI**

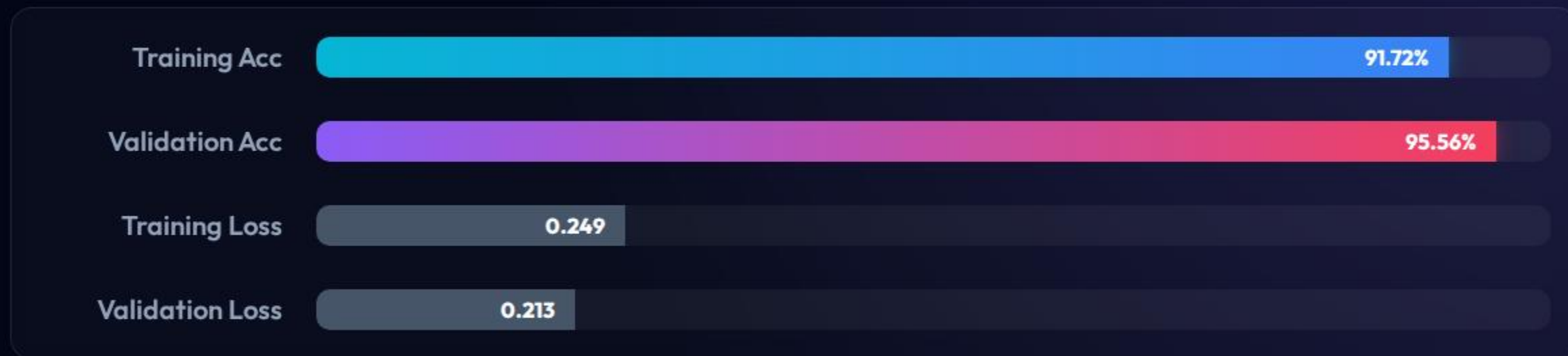
High-Performance REST  
API

# Engine: MobileNetV2

- > **Transfer Learning:** Pre-trained on ImageNet to inherit feature extraction capabilities.
- > **Lightweight Design:** Uses depth-wise separable convolutions for speed, crucial for real-time webcams.
- > **Custom Head:** Removed top layers; added GlobalAveragePooling2D and a dense Softmax layer (5 units).
- > **Optimization:** Trained using Adam optimizer with Categorical Crossentropy loss.

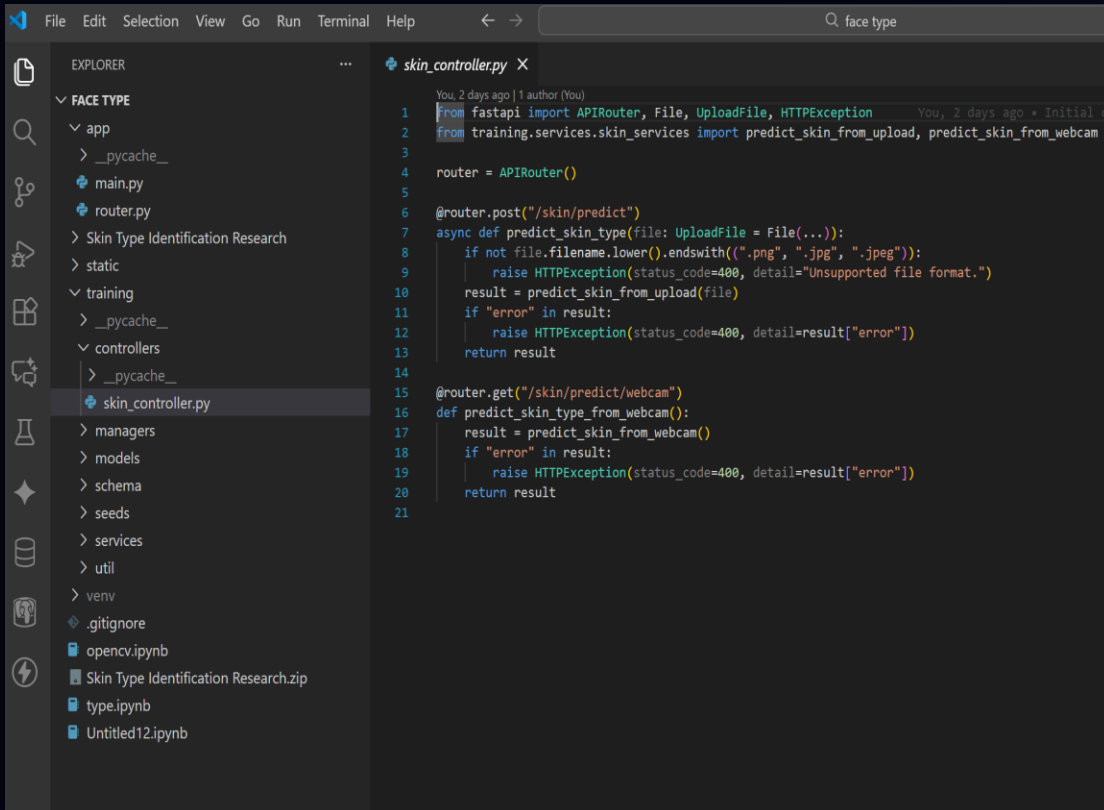


# Training Results



Validation accuracy exceeds training accuracy, indicating excellent generalization.

# Backend Architecture



The screenshot shows a VS Code editor with a project structure on the left and a Python file open in the center. The project structure includes a 'FACE TYPE' folder with subfolders 'app', 'training', 'controllers', and 'services'. The 'controllers' folder contains 'skin\_controller.py'. The 'services' folder contains 'skin\_services.py'. The 'skin\_controller.py' file is open and shows the following code:

```
1 from fastapi import APIRouter, File, UploadFile, HTTPException
2 from training.services.skin_services import predict_skin_from_upload, predict_skin_from_webcam
3
4 router = APIRouter()
5
6 @router.post("/skin/predict")
7 async def predict_skin_type(file: UploadFile = File(...)):
8     if not file.filename.lower().endswith((".png", ".jpg", ".jpeg")):
9         raise HTTPException(status_code=400, detail="Unsupported file format.")
10    result = predict_skin_from_upload(file)
11    if "error" in result:
12        raise HTTPException(status_code=400, detail=result["error"])
13    return result
14
15 @router.get("/skin/predict/webcam")
16 def predict_skin_type_from_webcam():
17     result = predict_skin_from_webcam()
18     if "error" in result:
19         raise HTTPException(status_code=400, detail=result["error"])
20     return result
21
```

## Modular Design Pattern

**Controllers:** Handle HTTP requests (e.g., /predict).

**Services:** Encapsulate ML inference logic.

**Utils:** Helper functions for image preprocessing.

## Endpoints

POST /predict/image

POST /predict/webcam

# Development Roadmap

## Phase 1

Data Collection &  
Preprocessing (Balancing  
Classes)

## Phase 2

Model Training (MobileNetV2  
Transfer Learning)

## Phase 3

Backend Development  
(FastAPI Integration)

## Phase 4

Testing & Validation (Real-  
time Webcam)



# | Live Inference Flow

## 1. Capture

Webcam stream is captured frame-by-frame. OpenCV detects the face region.

## 2. Process

Face is cropped, resized to 224x224, and pixels are normalized [0,1].

## 3. Predict

Model outputs confidence scores. Highest probability determines the label.

# QUESTIONS?

Ready to deploy AI-driven dermatology solutions. ([Nepika | Face Skin Analysis](#))



[rathour-anushka/skin-type-  
detection-using-face-recognition](https://github.com/rathour-anushka/skin-type-detection-using-face-recognition)



[anushkarathour1111@gmail.com](mailto:anushkarathour1111@gmail.com)