SYNTHEIA

Real-Time Emotion Recognition on the Edge

Gowtham Kumar Kamuni
Sanggil Lee
Illinois Institute of Technology
Instructor: Dr. Jafar Saniie
ECE 501- Artificial Intelligence and Edge Computing
Summer 2025 - June

Group Members

Sanggil Lee

Graduate Student, Electrical and Computer Engineering Illinois Institute of Technology, Chicago

Graduated from SeoulTech in South Korea with a Bachelor's in Electrical and Information Engineering. Currently pursuing a Master's in Electrical and Computer Engineering at Illinois Tech, specializing in computer networks and working under the guidance of Dr. Lin Cai. Passionate about building reliable systems that connect people and machines seamlessly.

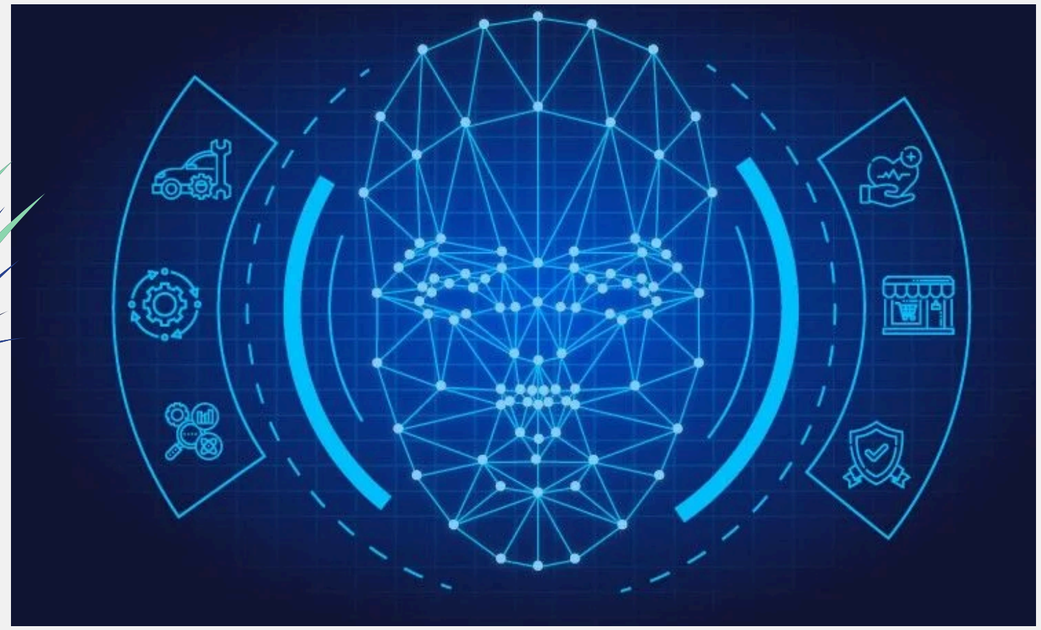
Gowtham Kumar Kamuni

Graduate Student, Artificial Intelligence Illinois Institute of Technology, Chicago

Completed my Bachelor's in Computer Science back in India, where I built my foundation in programming and problem-solving. Now diving deep into the world of AI at Illinois Tech, blending logic with intuition to build systems that don't just compute, they understand. SYNTHEIA is a reflection of that journey.

INTRODUCTION AND MOTIVATION

Machines listen. But they rarely understand.
SYNTHEIA was built to change that, to give technology a way to sense what’s left unsaid.



01.

Emotions don’t always speak, but they show.
People often mask what they feel.
SYNTHEIA picks up on those micro expressions that words miss.

02.

Current systems are overkill.
Most emotion recognition tools rely on cloud servers, voice analysis, or invasive sensors. We’re doing it real-time, offline, and on a single embedded device.

03.

Privacy isn't optional — it's core.
SYNTHEIA doesn’t store data or depend on external connections. It respects the moment and the person in it.

Problem Statement

- Emotions often go unnoticed.
- In classrooms, therapy rooms, or even daily conversations, people mask their feelings. Machines miss that entirely.
- Existing solutions are impractical.
- Cloud-based emotion detectors, multi-modal systems, and sensor-heavy setups don’t work well in real-world, real-time environments.
- There’s no lightweight, private alternative.
- We needed something small, fast, and emotionally aware, without needing the cloud, voice input, or invasive hardware.

Project Objective

What We're Building and Why It Matters

- Our Objectives:
- Detect and classify facial emotions using lightweight deep learning models like Mini-Xception or MobileNet
- Deploy the full pipeline on a Jetson Orin, with no cloud, no external dependencies
- Ensure the system is fast, responsive, and respectful of user privacy at all times

System Architecture and Workflow

01

Camera Input: Live feed captured from USB/CSI camera

02

Face Detection: OpenCV or Dlib isolates face region

03

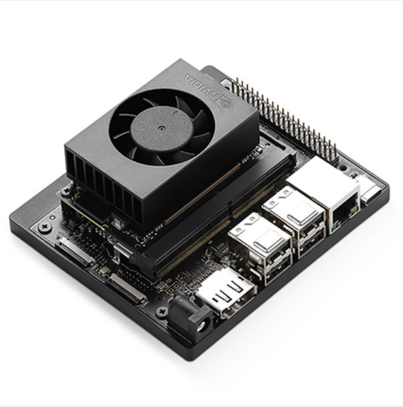
Preprocessing: Resize, normalize, grayscale/RGB conversion

04

Emotion Classification: Mini-Xception or MobileNet CNN model classifies emotion

05

Display Output: Emotion + confidence shown on-screen (Tkinter or Flask GUI)



Hardware Components

All components powered by a compact, portable setup, built for real world deployment and testing.

Jetson Orin Developer Kit

- The brain of our system
- NVIDIA GPU + ARM CPU optimized for AI workloads
- Handles real time emotion detection

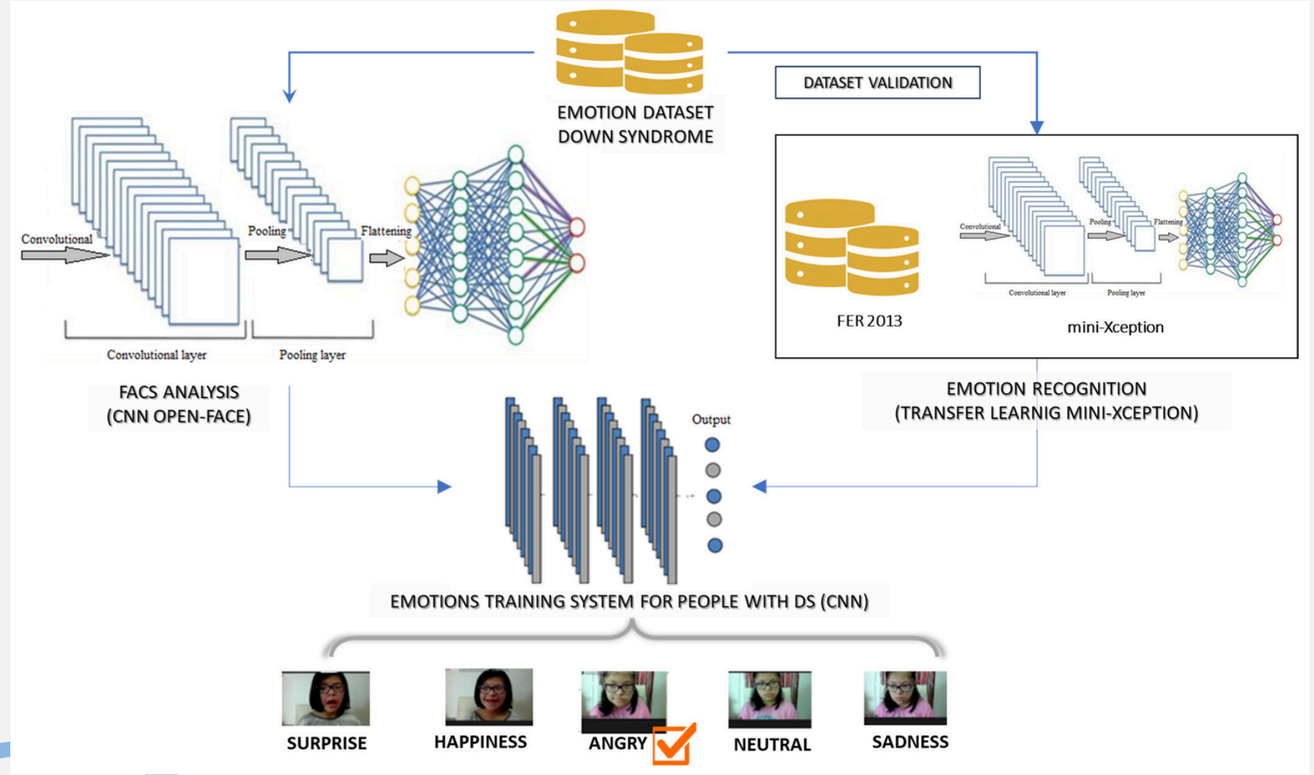
Camera Module

- USB or CSI camera input
- Captures video at 30 FPS
- Ensures facial expressions and micro reactions are recorded clearly

Display / Interface

- Monitor or LCD to show real time emotion predictions
- GUI built with Flask or Tkinter
- Keeps the system user friendly and demo ready

Software Components



Built for Speed, Privacy, and Precision

Operating System & SDK

- JetPack SDK + Ubuntu (Jetson flavored)
- Includes CUDA, cuDNN, TensorRT, and drivers
- Powers GPU accelerated inference

Deep Learning Framework

- TensorFlow / PyTorch
- For model training and prototyping
- Converted to ONNX, optimized using TensorRT for Jetson

Image Processing

- OpenCV for face detection, frame capture, drawing overlays
- NumPy for array manipulations and preprocessing

GUI & Output

- Flask / Tkinter
- Used to display real-time video with emotion predictions and confidence scores
- Optional logging of timestamps + emotion states

Development Tools

- Jupyter, VS Code for development
- TensorBoard for training performance visualization

Dataset & Model Training Strategy

Training the Mind to Read the Face

01.

Dataset: FER-2013

- **~35,000 labeled grayscale images, 48x48 pixels**
- **7 emotion classes: Happy, Sad, Angry, Fearful, Surprised, Disgusted, Neutral**
- **Publicly available, widely used benchmark**
- **Used for training, validation, and testing**

02.

Model Architecture

- **Mini-Xception / MobileNetV2**
- **Lightweight CNNs ideal for low-latency embedded deployment**
- **Trained in TensorFlow or PyTorch**
- **Converted to ONNX, optimized with TensorRT for Jetson**

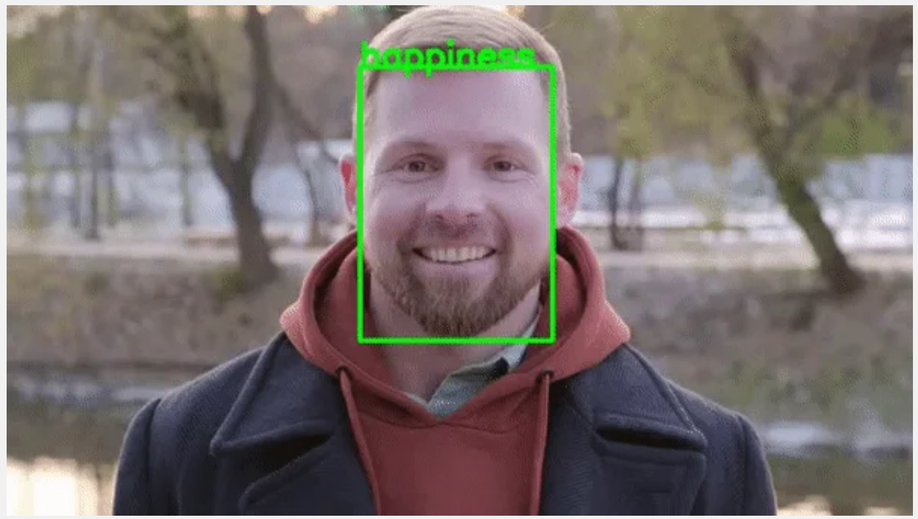
03.

Training Strategy

- **Start with pretrained weights for faster convergence**
- **Fine tune on FER-2013, with optional augmentation**
- **Batch size = 32, Epochs 25-30**
- **Final model benchmarked and deployed on Jetson for live testing**

Expected Output & User Interface

Clear. Fast. Local.



Expected Output

- Real-time video feed with detected face highlighted
- Predicted emotion label shown above or below the face
- Confidence score (%) displayed alongside label
- Frame rate: 10–20 FPS, smooth and responsive

User Interface (GUI)

- Built using Flask or Tkinter
- Display includes:
  - Live video stream
  - Emotion label + confidence
  - Optional logging toggle (for timestamped emotion tracking)
- Runs completely on Jetson, no internet, no cloud

65–75%

Evaluation Plan & Promising Accuracy

Performance Goals

Classification Accuracy:

- ~65–75% on FER-2013
- Up to 80%+ after fine-tuning on real-world images

Latency (inference time):

- <100ms per frame using TensorRT on Jetson Orin

Frame Rate (FPS):

- Target: 10–20 FPS for smooth user interaction

Evaluation Plan

Quantitative:

- Accuracy, Precision, Recall using test set
- Confusion matrix to analyze emotion-specific performance

Qualitative:

- Real-time testing with different lighting & angles
- Feedback from users on GUI clarity and responsiveness

Tools Used:

- TensorBoard, Matplotlib, OpenCV visual logs
- Internal logging for timestamped predictions

Development Timeline

4-Weeks

WEEK 1

Setup & Dataset Prep

- Jetson Orin hardware + camera integration
- Install JetPack SDK, OpenCV, TensorFlow
- Collect & organize FER-2013 dataset
- Run baseline tests on pretrained models (desktop)

WEEK 2

Model Training & Optimization

- Train/fine-tune Mini-Xception or MobileNet on FER-2013
- Convert models to ONNX
- Optimize with TensorRT for edge inference
- Begin testing model on Jetson Orin live input

WEEK 3

GUI Integration & Output

- Build GUI with Flask or Tkinter
- Display emotion predictions & confidence scores
- Implement optional logging system
- Test real-time output in varied lighting/angles

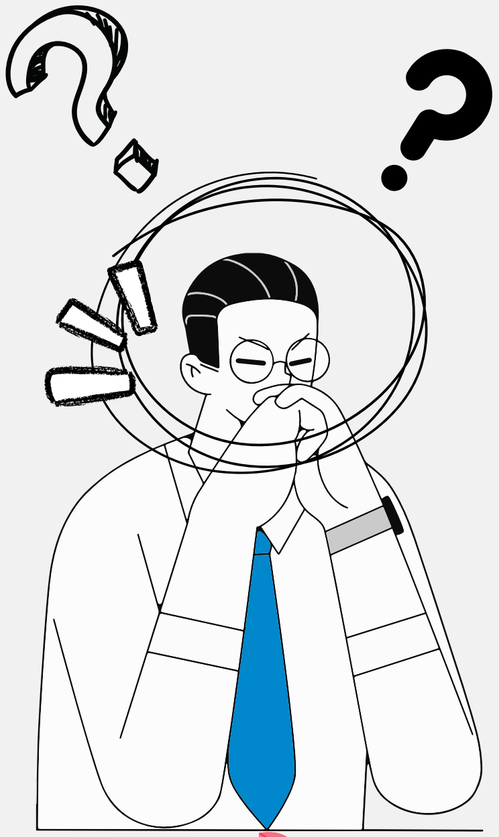
WEEK 4

Testing, Evaluation, and Finalization

- Benchmark FPS, latency, and accuracy
- Debug issues & polish visual feedback
- Record demo video + write final documentation
- Prepare presentation and submit deliverables

Challenges & Limitations

The Gaps We're Aware Of.



Edge Hardware Constraints

- Jetson Orin is powerful, but not limitless, we have to optimize carefully to maintain speed and accuracy under limited resources.

Lighting & Expression Variability

- Emotion recognition can be impacted by poor lighting, unusual angles, or occlusions (glasses, hair). We're training with variety, but edge cases remain.

Model Generalization

- FER-2013 doesn’t fully represent real-world diversity. We may face misclassifications without fine-tuning on more varied or real-time data.

Privacy Ethical Balance

- Even without data storage, the system must respect boundaries. User trust and transparent operation are non-negotiable.

GUI/Inference Integration

- Combining live camera input, deep learning inference, and GUI rendering in real-time, all on embedded hardware, can lead to compatibility or performance hiccups.

Future Scope & Improvements

Where SYNTHEIA is headed: long-term potential and new capabilities.

Voice-Based Emotion Detection
- Extend recognition to vocal tone for deeper emotional context.

Biometric Fusion
- Combine facial data with non invasive sensors like heart rate or skin conductance.

Multi-Face Analysis
- Enable detection and emotional analysis of multiple people at once.

Contextual Intelligence
- Use environmental cues and temporal patterns to enhance prediction accuracy.

Enhancements we aim to implement in upcoming development cycles.

Fine-Tuning on Custom Data
- Collect real-world face data to improve accuracy and generalization.

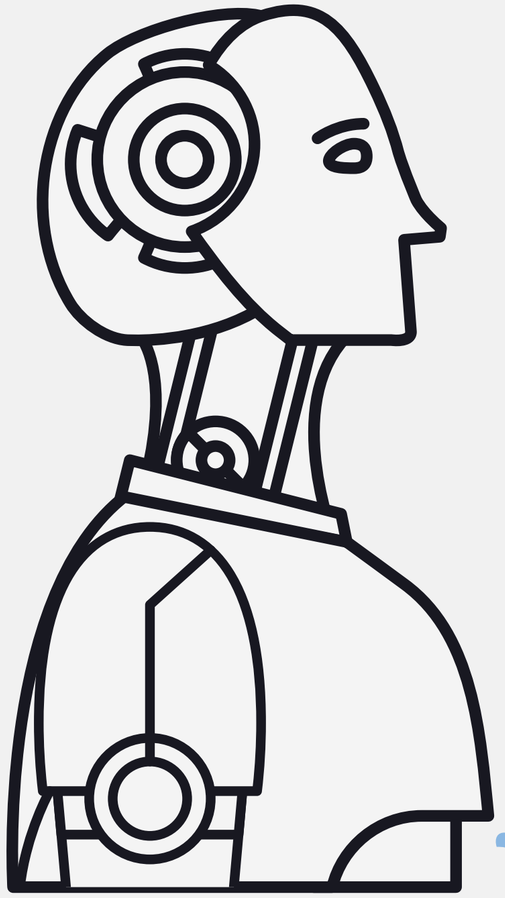
Interface Polish & Feedback Display
- Add smoother transitions, better visual indicators, and customizable display options.

Improved Logging System
- Build structured, exportable logs for long-term analysis and research.

Lighting & Occlusion Handling
- Improve performance under poor lighting, occlusions (e.g., glasses, masks), and side profiles.

BUILT FOR NOW

DESIGNED FOR WHAT'S NEXT



Real-World Applications of SYNTHEIA

Therapy & Mental Health Monitoring

SYNTHEIA can assist therapists by detecting subtle emotional shifts in real time, helping them better understand patient reactions, even when words fall short.

Classrooms & Education Support

Teachers can gain insight into student engagement, stress, or confusion during lessons, enabling more responsive and emotionally aware teaching.

Human-Computer Interaction (HCI) Research

Used in usability labs or product testing to measure emotional responses to interfaces, prototypes, or interactive systems, beyond verbal feedback.

Assistive Technology for Non Verbal Users

SYNTHEIA can help caregivers, educators, or family members better understand the emotional state of individuals who are non verbal or on the autism spectrum.

WHERE SYNTHEIA MAKES A DIFFERENCE

Team Responsibilities

Shared Effort. Aligned Vision.

Gowtham Kumar Kamuni

- Hardware setup: Jetson Orin integration and system deployment
- Model optimization using TensorRT and ONNX
- GUI design and on-device output logic
- Final testing, debugging, and documentation

Sanggil Lee

- Dataset preparation and preprocessing (FER-2013)
- Model training and parameter tuning
- GUI implementation using Flask/Tkinter
- Evaluation metrics and visual analysis (accuracy, confusion matrix)

Joint Contributions

- Ideation, research, and system integration
- Presentation, polishing, and oral delivery
- Final project vision and execution, together

Conclusion

Not perfect. Not finished. But real and ready to evolve.

SYNTHEIA isn’t just a project we built.
It’s something we believed needed to exist.
We didn’t want another cloud-powered, overengineered AI tool.
We wanted something that could quietly understand, without asking, without listening, without storing.
Just… seeing.
In a world that’s loud and fast, SYNTHEIA listens to what’s not said.
It catches a flicker of discomfort, a hidden smile, a moment of silence, and it says,
“I see you.”
From real classrooms to therapy rooms, we hope SYNTHEIA becomes more than just code.
We hope it becomes a reason for someone to feel understood, even if just for a moment.
And we’re just getting started.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016).
   Deep Learning. MIT Press.
2. Arriaga, O., Valdenegro-Toro, M., & Plöger, P. (2017).
   Real-time Convolutional Neural Networks for Emotion and Gender Classification.
   arXiv:1710.07557
3. Barsoum, E., Zhang, C., Ferrer, C. C., & Zhang, Z. (2016).
   FER-2013: A Dataset for Facial Expression Recognition.
4. OpenCV Documentation
   https://docs.opencv.org/
5. NVIDIA JetPack SDK
   https://developer.nvidia.com/embedded/jetpack
6. TensorFlow Documentation
   https://www.tensorflow.org/
7. PyTorch Official Site
   https://pytorch.org/
8. ONNX Runtime
   https://onnxruntime.ai/
9. FER-2013 Dataset – Kaggle
   https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data
10. Mini-Xception Model Architecture – Original GitHub
    https://github.com/oarriaga/face_classification

Thank you very much!