GSoC 2026 Proposal
(NeoSmartML Foundation Model for Neonatal ICU Monitor Data)

Applicant: Anagha Pradeep
Organization: LibreHealth
Project Size: Large (~350 hours)
Mentors: Saptarshi Purkayastha, Sherri Bucher

ABOUT ME

I am a master's student in computer science who is presently interning at IntelinAir. I work on weather and crop disease prediction systems, developing pipelines that integrate sensor and satellite data to provide actionable forecasts for farmers. Prior to pursuing my master's degree, I worked as a software engineer for two years on backend systems and infrastructure. I am now shifting to AI/ML, and when I saw the NeoSmartML project, I was immediately pulled to it since it combines edge deployment, multimodal learning, and real clinical impact, which is exactly the type of topic I want to work on. Contributing to an open-source project of this size, under the supervision of experienced mentors, is exactly the type of learning environment I've been seeking.

That background influences how I approach new codebases. When I work on an unfamiliar project, I utilize a combination of thorough reading, documentation, and review to quickly develop a mental model of the system, knowing not only what the code does but also why it is built that way, where the risks are, and what is missing. Every discovery, change, and architectural decision I propose can be fully explained and defended

I'm still honing my ML skills, but I bring to this project something that many other ML candidates don't, two years of production software engineering experience, familiarity with embedded systems and firmware, and the habit of considering correctness, security, and maintainability in addition to model accuracy. I feel that combination is what NeoSmartML, a clinical edge-deployment project, requires, and I am delighted to make a major contribution to it.

PRELIMINARY WORK

I completed both preliminary tasks listed in the project description.

Task 1: Dynamic Quantization of CheXNet

I used PyTorch dynamic quantization (torch.ao.quantization.quantize_dynamic) on the CheXNet DenseNet-121 model, targeting both nn.Linear and nn.Conv2d layers.

The results:

- Original checkpoint (model.pth.tar): **84 MB**
- Quantized model (chexnet_dynamic_int8.pth): **approx. 28 MB** (approx. 3x compression)
- Verification: max |float − quantized| = **0.39** (expected for int8 dynamic quantization)

An important finding from this work: quantizing only nn.Linear layers (as many tutorials suggest) produces almost no compression for DenseNet-121, since the model is dominated by nn. Conv2d layers. The final classifier is a single Linear layer, so Linear-only quantization reduces size by less than 0.2%. Including Conv2d in the quantization set is essential for meaningful compression on CNN architectures. I documented this in my write-up as a deliberate finding rather than hiding it.

The script supports both flat (torchvision-style) and CheXNet-master checkpoint layouts, with auto-detection from state dict keys and a architecture override flag.

Task 2: Rotation Sensor Integration + Bulk Download Endpoint

Rotation sensor (ESP32 firmware): I added MPU6050 gyroscope support to esp_camera_ble_control.ino. Rather than using the Adafruit MPU6050 library (which caused IRAM overflow when combined with BLE + Camera + WiFi on the AI Thinker ESP32-CAM), I read the sensor directly via raw I2C register access, resulting in a much-reduced footprint. If no sensor is found, the code uses simulated values. Gyro readings are sent over BLE every 500ms in X:val,Y:val,Z:val format.

Bulk download endpoint (app_httpd.cpp): I also looked into issue #4 which needed a way to download multiple images from the ESP32 at once. I added a /download?count=N endpoint that reads N images from the SD card and streams them as a gzip file. While working on it I found a small bug where the endpoint would 404 if no count was given, I fixed it to default to 10 images. I am still testing and refining it before submitting as a merge request.

Code review and additional MRs: I went through the full neosmartml-main codebase and noted a few issues worth fixing an unsafe strcpy in main.c, a path traversal risk in the FastAPI upload handler, WiFi credentials accidentally committed to the repo, unauthenticated Bluetooth Low Energy configuration, and outdated pandas method that breaks on pandas 2.x. I documented each one with the file and line number and suggested fixes

Beyond the preliminary tasks, I also submitted two merge requests addressing open issues:

- **!23 - OCR Code Refactor** (closes #9): Replaced 8 nearly identical extract_* and plot_* functions with a single generic extract_vital_value() and plot_vital_results(), driven by a VITAL_CONFIGS dictionary. Reduces ~300 lines to ~100 and makes adding new vital signs require only one dict entry. Includes passing tests.
- **!24 - CI Pipeline Fix** (closes #10): Fixed a pre-existing yaml invalid error in .gitlab-ci.yml that was causing 0 jobs to run across all MRs. The sast job was defined twice

manually and inside the included SAST template causing a conflict. Removing the redundant definition restored the pipeline for all contributors.

PROJECT UNDERSTANDING

The NeoSmartML project uses ESP32-CAM to capture NICU monitor images and OCR to extract vital signs. Its goal is to build a foundation model that can handle both image and temporal data for multiple clinical prediction tasks such as risk classification, apnea/bradycardia event prediction, treatment response, and physiological state assessment. The primary technical difficulty is developing a hierarchical multimodal architecture that integrates:

1. A vision encoder processes raw monitor images.

2. Added an OCR extraction layer for structured vital signs text.

3. A temporal sequence model for analyzing trends over multiple time increments.

4. Multi-task prediction for various clinical outcomes.

The second difficulty is edge deployment: the model must run on or near ESP32-class hardware, so extreme compression (quantization, pruning, distillation) is not a possibility.

PROPOSED ARCHITECTURE

Based on my review of the project description and codebase, I propose the following architecture for the foundation model. This is my view of how to fulfill the mentors' goals, and I am willing to refine it depending on their comments.

Stage 1: Multimodal Encoder.

Vision branch: A MobileNetV3 or EfficientNet-Lite backbone that has been pre-trained on ImageNet and fine-tuned using NICU monitor images. For the foundation model, I pick MobileNetV3-Small over DenseNet-121 because its depthwise separable convolutions are better suited to static quantization and have a smaller IRAM footprint on ESP32.

The present EasyOCR pipeline pulls structured vital sign readings (HR, SpO2, RR, and temperature). A modest learnt projection layer is used to embed these in a fixed-dimension vector.

Fusion: I propose a cross-attention fusion layer in which the OCR embedding is used as a query against the visual feature map. This addresses the primary difficulty of asynchronous, poorly correlated sensor streams: the attention weights learn which visual regions are semantically significant to each extracted vital sign, rather than just concatenating modalities.

Stage 2: Temporal Sequence Modeling

A lightweight transformer encoder (4 layers, 128 hidden dims) processes fused embedding sequences over a sliding window (e.g., 30 seconds). This detects trend patterns rising/falling

trajectories, oscillations that forecast occurrences like as apnea and bradycardia even before they occur.

For edge deployment, a GRU (smaller, quicker) can be used with low accuracy loss. I would benchmark both and use the GRU as the default edge variant.

Stage 3: Multi-Task Prediction Heads.

Each clinical task is assigned a lightweight adapter head (2-layer MLP) that connects to the temporal encoder output. The foundation model is trained using a multi-task loss that includes all heads. For downstream fine-tuning, the backbone is frozen and only the adapter head is modified this is the "fine-tuning" approach described in the project description.

For prompt engineering style adaptation, I would consider adding a tiny collection of learnable task-specific tokens prepended to the temporal sequence, akin to prefix tuning, to avoid any weight updates to the backbone.

Edge Compression Pipeline

Based on the preliminary quantization work, the entire compression pipeline would be:

Post-training dynamic quantization (int8 Linear + Conv2d) has been shown to reduce size by around three times.

Static quantization with calibration set from real NICU picture sequences expected additional reduction.

Structured pruning of attention heads with low average activation magnitude

Distilling knowledge from the whole model to a smaller simpler model, with a goal size of <5MB for ESP32 deployment.

TIMELINE:

Period - week	Work
Community bonding	Study codebase, set up environment, discuss plan with mentors
1–2	Vision encoder + OCR embedding layer
3-4	Fusion layer + unit tests
5-6	Temporal sequence model
7-8	Prediction heads + training loop
9-10	Evaluation and metrics
11	Model compression (quantization + pruning)
12	ESP32 integration + documentation
13	Buffer week — feedback and fixes

WHY I'M THE RIGHT CANDIDATE

I have worked on several parts of this project already and feel ready to take on the full scope.

- I completed the CheXNet quantization task and understand how and when to apply different compression techniques on CNN architectures
- I added MPU6050 sensor support and a bulk download HTTP endpoint to the ESP32 firmware, working through real hardware constraints like IRAM overflow and BLE conflicts
- I refactored the OCR pipeline reducing around 300 lines to 100 and fixed a CI issue that was blocking every contributor from running pipelines
- I reviewed the full codebase for security issues and documented each one with file references and suggested fixes

I work best in writing and async communication which fits the remote GSoC format well. I am an early riser and tend to do my most focused and careful work in the early morning hours.

I am someone who takes full ownership of the work I submit. I may not have years of ML experience yet but I have strong software engineering foundations, I ask good questions, and I push through problems until I understand them properly. I think that mindset matters especially in a project that handles real clinical data from neonatal patients.