Multi-Modal Clinical Assistant

- Project: Multi-Modal Clinical Assistant (Speech → OCR → Text → Role-Aware Summaries)
- Team: Group 1
- Course: DS & AI Lab Project
- Date: 2025-11-21

1. Abstract

We present a practical clinical assistant that converts speech or text inputs and attached reports into two role-aware outputs: (1) clinician-oriented structured notes and insights, and (2) patient-friendly summaries. The pipeline combines Whisper Small for ASR, OCR for documents, and instruction-tuned LLMs for summarization with role-specific prompts. Earlier stages explored data preparation, model distillation (teacher: Mistral-7B-Instruct; student: TinyLlama-1.1B with LoRA), and evaluation using ROUGE/BERTScore (text) and WER/CER (speech). We also provide a deployment blueprint (Colab and local API). Results show acceptable semantic fidelity and practical latency, with clear next steps around domain adaptation, diarization, and safety.

2. Introduction

Clinical encounters contain rich, unstructured information that must be accurately documented and communicated. Manual workflows can be time-consuming and inconsistent for both doctors and patients. We build an end-to-end, role-aware system that:

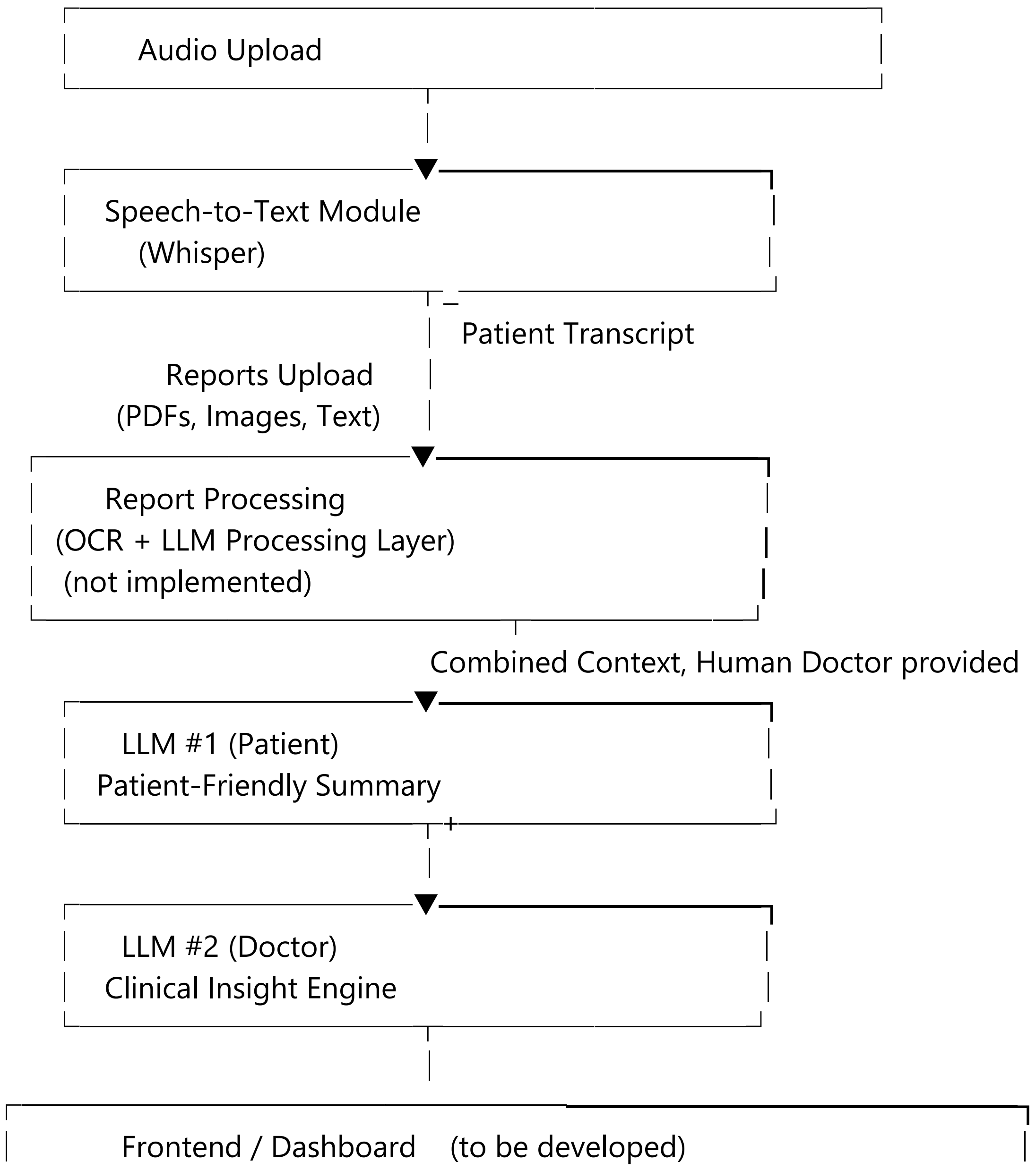
- Accepts input as speech and/or text, optionally with uploaded reports (PDFs/images).

- Uses Whisper to transcribe speech; OCR to extract report text.
- Generates two outputs via tailored prompts:

The design emphasizes deployability (compact models where possible), transparent metrics, and simple UX.

3. Overview of approach and modules

a. Architecture Summary



```
│  - Review summary                                                                      │
│  - Edit before sending to patient                                                      │
│  - Access doctor insights                                                              │
└────────────────────────────────────────────────────────────────────────────────────────┘
```

b. Deployed Components

For Text Part

- **Frontend/UI:** Streamlit app running on Hugging Face Spaces.
- **Model:** PEFT LoRA adapter for patient summary generation.

Component	Platform	Status
Frontend (Streamlit)	HF Spaces	To be done
Backend API (FastAPI)	local	Not used
Whisper STT Model	Google colab	Integrated
LLMs (Patient & Doctor)	Kaggle, HF Spaces Gradio	Integrated
End to end model	Zapier integration	To be done

4. Summary of Milestones

For ease of reference, we provide a summary of the 6 milestones in this section. Later sections cover each milestone in detail.

c. Literature Review (Milestone 1)

- ASR in healthcare: Transformer-based models (e.g., Whisper) enable robust transcription across accents/noise.

- Clinical note generation: SOAP frameworks and prompt-based LLM summarization improve consistency.
- Teacher–student distillation: Larger models generate high-quality labels; smaller models (LoRA adapters) are efficient at runtime.
- Evaluation: ROUGE/BERTScore for summarization; WER/CER for speech; qualitative checks for hallucination/term retention.

References to project docs: Milestone PDFs in repo root.

d. Dataset and Methodology (Milestone 2–3)

- Sources and Assets
- Preprocessing
- Methodology

Notebooks (examples):

- Text: notebooks/text/dsail-llm-part_ver1.ipynb, dsail-llm-part_ver2.ipynb, dsail-llm-inference.ipynb, autosoap-ai-powered-clinical-documentation.ipynb
- Speech: notebooks/speech/data-prep.ipynb
- Vision: notebooks/vision/OCR.ipynb

e. Model Development and Hyperparameter Tuning (Milestone 4)

- Teacher model: Mistral-7B-Instruct-v0.3 generates patient-friendly labels from SOAP components.
- Student model: TinyLlama-1.1B-Chat with LoRA (r=8, α=16, dropout=0.1) for efficient fine-tuning.
- Whisper Small (openai/whisper-small) for ASR; decoding via greedy/beam; English, task="transcribe".
- Prompts: concise, empathetic, non-hallucinatory patterns for patient; structured SOAP + counseling for doctor.

- Training constraints: Batch size limited by T4-class GPU; gradient accumulation; AdamW; FP16 where supported.
- Observations: Trade-offs between context window and memory; blank-output mitigation via prompt/output handling; ASR vocabulary challenges on medical terms.

Visuals:

- Text model architecture (from Milestone 4):

f. Evaluation & Analysis (Milestone 5)

- Summarization Metrics
- ASR Metrics
- Qualitative Highlights

Deployment & Documentation (Milestone 6)

- Runtime Components
- Interaction Flow (role-aware)
- Deliverables

5. Technical Description

These are provided in the Technical Overview document at: https://github.com/Pallavi-Pandey/DSAI_Lab_Project/blob/main/docs/technical_doc.md

6. Conclusion and Future Work

We delivered a multi-modal assistant that produces role-aware outputs from speech/text inputs and attached reports. The approach demonstrates practical performance with compact models, clear prompts, and simple deployment pathways.

Future Work:

- Domain lexicon/pronunciation hints for ASR; in-domain fine-tuning to reduce WER on medical terms.
- Diarization/turn-taking for multi-speaker conversations.
- End-to-end evaluation harness combining WER/CER with downstream task metrics.
- Guardrails for medical safety and escalation guidance.
- Frontend polishing (Streamlit/Gradio) and fully containerized services.

7. References and Appendix

- Repo documents: https://github.com/Pallavi-Pandey/DSAI_Lab_Project/tree/main/docs
- Notebooks: https://github.com/Pallavi-Pandey/DSAI_Lab_Project/tree/main/notebooks
- Tools/Models: https://github.com/Pallavi-Pandey/DSAI_Lab_Project/tree/main/models

The following sections provide detailed descriptions for the various milestones.

8. Milestone 1: Problem Definition & Literature Review

g. Problem Definition & Objectives

The objective of our project is to build a **Multimodal AI Medical Assistant** that integrates **speech, vision, and text** data to support healthcare practitioners and patients.
● **Speech:** Transcribe patient descriptions, extract symptoms, urgency, and emotional tone.
● **Text:** Process doctor’s notes, prescriptions, and medical literature using a domain-specific LLM.
● **Vision:** Analyze medical images (X-rays, MRI, CT scans) to detect potential abnormalities. (extension to the first phase of the project)
● **Fusion:** Combine all three (or two) modalities into a unified representation that enables reasoning.
● Output: ○ **For doctors:** Structured, detailed diagnostic summary.
○ **For patients:** Simplified explanation in plain English.

Objective Summary: Provide an AI assistant that reduces diagnostic time, supports clinical decision-making, and improves patient understanding.

h. Literature Review

Existing Solutions:

● **Speech-to-Text & Symptom Extraction:** Tools such as the *Google Medical Speech-to-Text API* and *ClinicalBERT* can transcribe and extract symptoms from spoken inputs.

● **Medical Imaging (CV):** CNN and Transformer-based models (e.g., *CheXNet*, *MedViT*) achieve state-of-the-art performance in X-ray and MRI classification tasks.

● **Medical Text (LLMs):** Models such as *BioBERT*, *PubMedBERT*, and *MedPaLM* are fine-tuned on medical corpora to provide clinical reasoning.

● **Multimodal Models:** Recent research (e.g., *BioViL*, *MedCLIP*) explores combining vision and text but often excludes patient speech as an input.

Baselines & Benchmarks:
● **CheXpert** and **MIMIC-CXR** datasets for imaging benchmarks.
● **i2b2** clinical NLP dataset for text-based symptom/diagnosis extraction.
● **Wav2Vec2 + ClinicalBERT** pipelines for speech-to-symptom baselines.

i. 3. Gaps & Opportunities

● Most existing systems are **unimodal** (focus on speech *or* vision *or* text), not integrated.
● Few models provide **patient-friendly explanations**; most are designed only for doctors.
● Current multimodal approaches do not effectively combine **speech, vision, and medical text** into a unified diagnostic reasoning framework.
● Opportunity to design a system that: ○ Uses **multimodal embeddings** for richer diagnostic context.
○ Delivers **dual outputs** (doctor-oriented & patient-oriented).
○ Improves **accessibility** in telemedicine and resource-limited settings.

j. 4. Datasets

● The datasets that will be used for this project will be from open source because manual data collection for our objective will take more time and resources.

● We will try to use Hugging Face and Kaggle datasets as much as possible, we are also looking into some datasets that have been released by some universities.

● We assure that all datasets will be from open source which are under MIT or Apache 2.0 License.

9. Milestone 2: Dataset Preparation

k. Dataset 1

Dataset 1

● **Location and Name:** Medical Speech, Transcription, and Intent Dataset Path: /kaggle/input/medical-speech-transcription-and-intent/Medical Speech, Transcription, and Intent/recordings

● **Description:** This dataset contains medical speech recordings and their corresponding transcriptions. It is divided into three folders — train, test, and validate. Each entry in the provided CSV file (overview-of-recordings.csv) lists an audio file name and its spoken phrase (transcript).

● **Data Preparation:** The overview CSV was loaded, and each audio file listed in it was searched within the train, test, and validate folders. The complete file paths were then added to a new DataFrame along with their corresponding transcripts. Entries without matching audio files were removed. The final cleaned data containing valid audio–text pairs was saved as metadata.csv in the working directory.

● **Preprocessing:** The generated metadata.csv file was used to inspect samples and verify correctness. A few random samples were selected to check the transcript and listen to the corresponding audio. This confirmed that the metadata and recordings were aligned and ready for fine-tuning.

l. Dataset 2

● **Location and Name:** Florence-2 OCR Testing Dataset Image Source: https://raw.githubusercontent.com/gokul-1998/handwriting_recognition/main/test_images/test3.png

● **Description:** A single test image containing handwritten or printed text was used to evaluate the OCR capabilities of the **Florence-2-Flux-Large** model. The image was loaded directly from a public GitHub repository URL and processed using the Florence-2 model architecture for text extraction.

● **Data Preparation:** The image was retrieved from the provided URL and opened in memory using Python's BytesIO. Before inference, the image was verified and converted to **RGB mode** to ensure compatibility with the model's input requirements.

● **Preprocessing:** The **Florence-2 AutoProcessor** was initialized with the model to handle both the text prompt and image input. A task prompt (<OCR>) was used to instruct the model to perform Optical Character Recognition. The processor tokenized the input text and prepared the image tensor for the model. The processed inputs were then passed to the model to generate the OCR output, which was decoded and post-processed to extract the recognized text.

m. Dataset 3

● **Name:** Automated Medical Diagnosis Using Clinical Notes - Kaggle ○ Location - https://www.kaggle.com/datasets/smmmmmmmmmmmm/automated-medical-diagnosis-using-clinical-notes

● Description: (from the data card) The synthetic dataset, containing 3000 simulated patient records, includes patient demographics, primary complaints, additional symptoms, and initial diagnostic suggestions. This dataset enables the LLM to learn patterns and correlations in clinical contexts, potentially assisting healthcare providers in decision-making and improving diagnostic efficiency in real-world medical environments.

● Data Preparation: ○ Single set of samples; training, val, and test splits to be done at model time

○ Reasonably clean data, No missing values in required features

○ EDA explained in GitHub - https://github.com/Pallavi-Pandey/DSAI_Lab_Project/blob/struct/notebooks/text/DSAIL_M2_part1.ipynb and also included in next sub-section

○ Since it is synthetic data, it is well balanced in the comparisons done

● Preprocessing: ○ 3000 samples

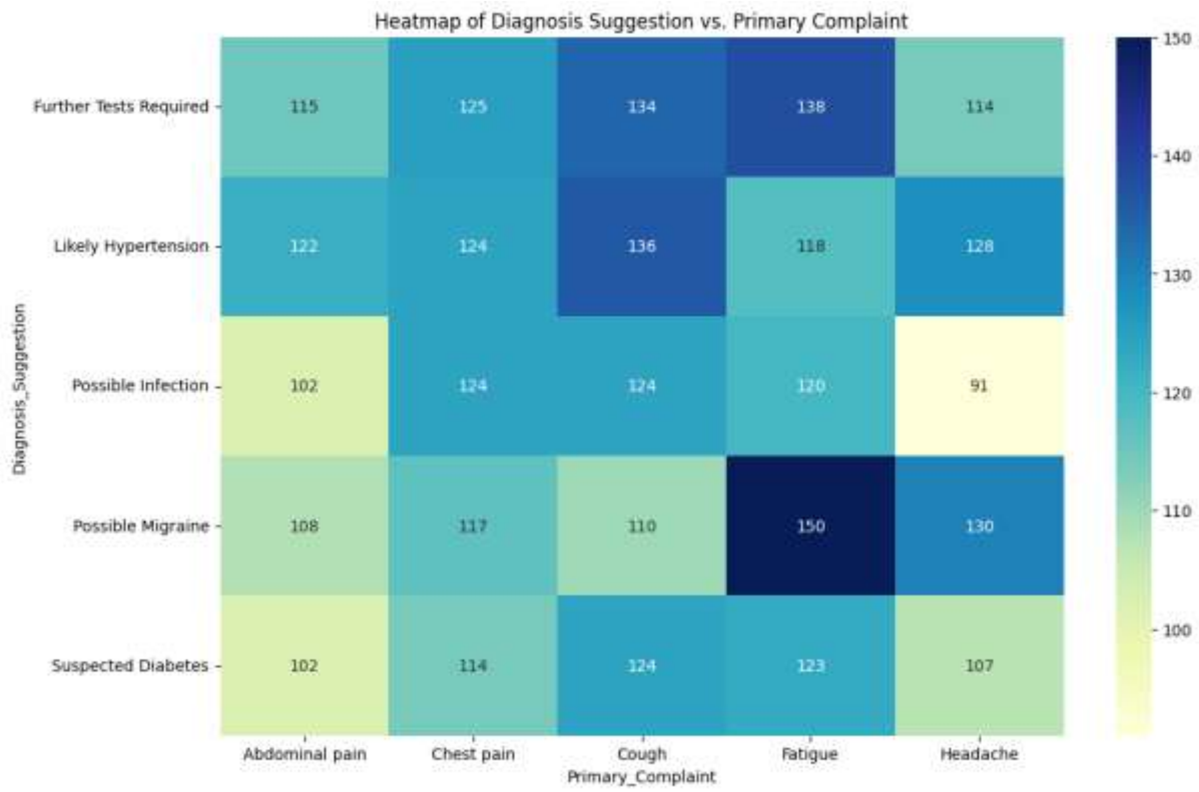
○ May not need much preprocessing

○ Can be directly loaded to the model as required

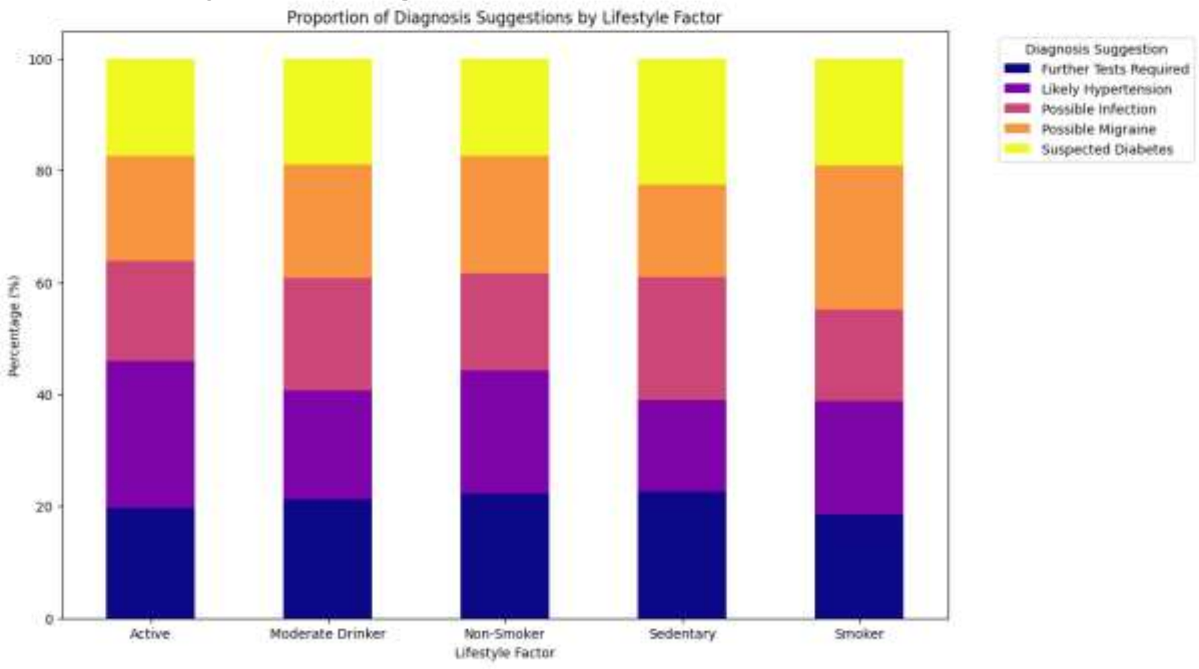
● Notes: A larger dataset - MIMIC-IV-Ext-BHC - requires training and agreement signing; this could be included once these steps are completed

n. EDA for Clinical Notes Dataset

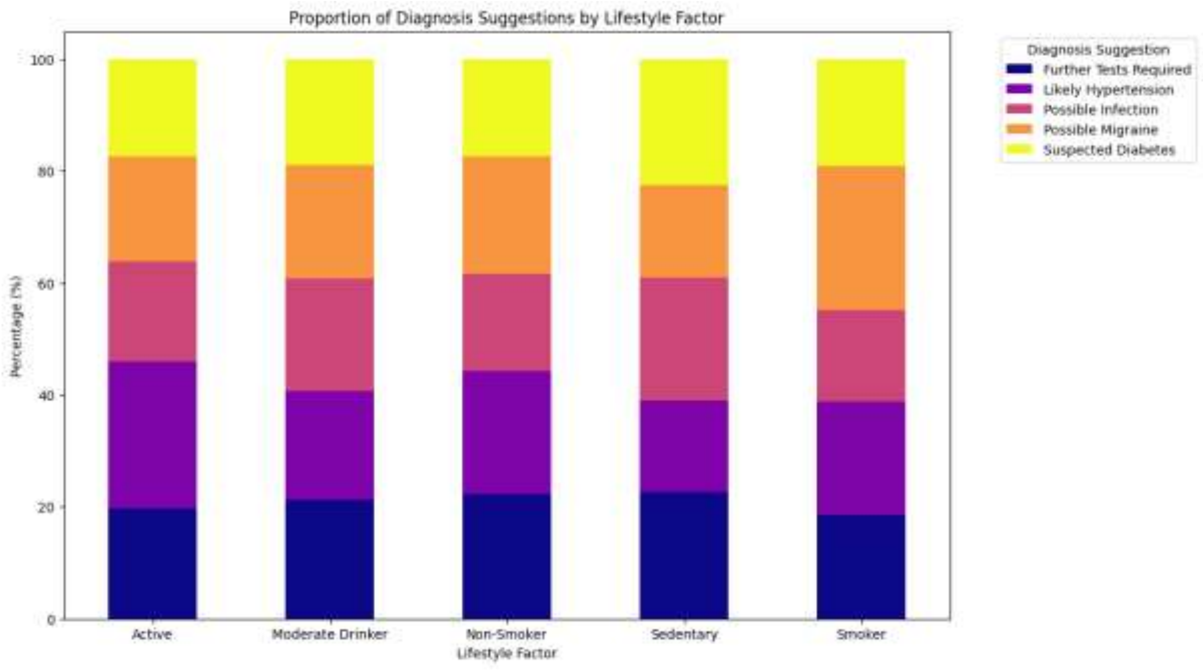
Primary Complaint vs Diagnosis Suggestions



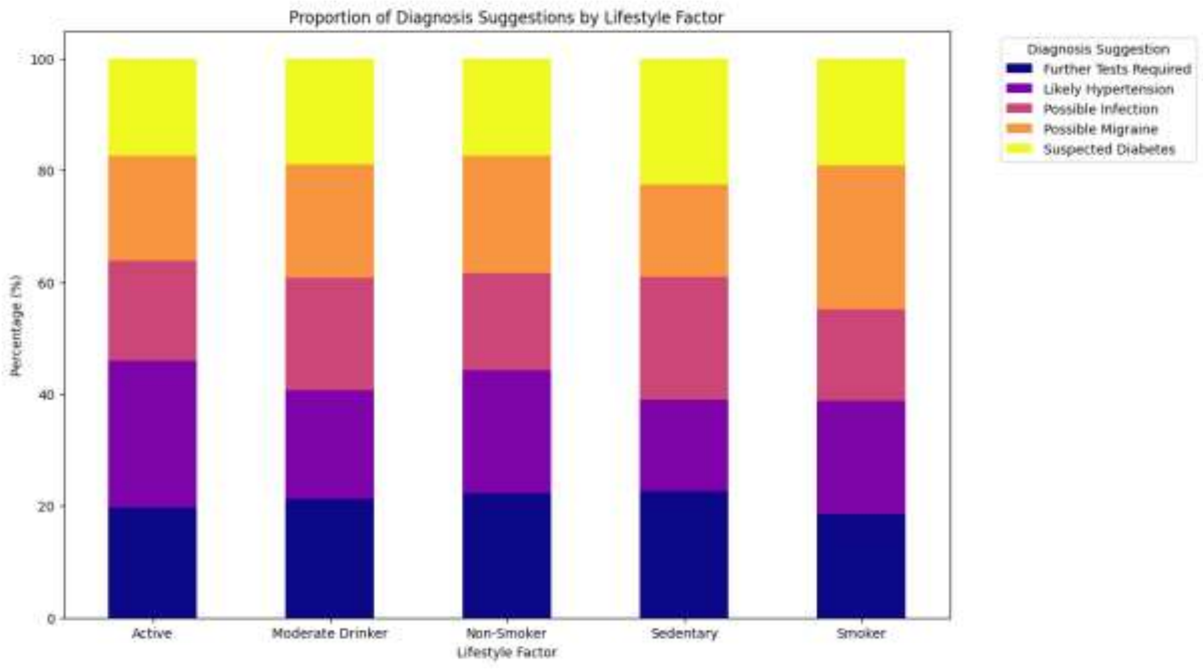
Lifestyle vs Diagnosis



Gender vs Diagnosis

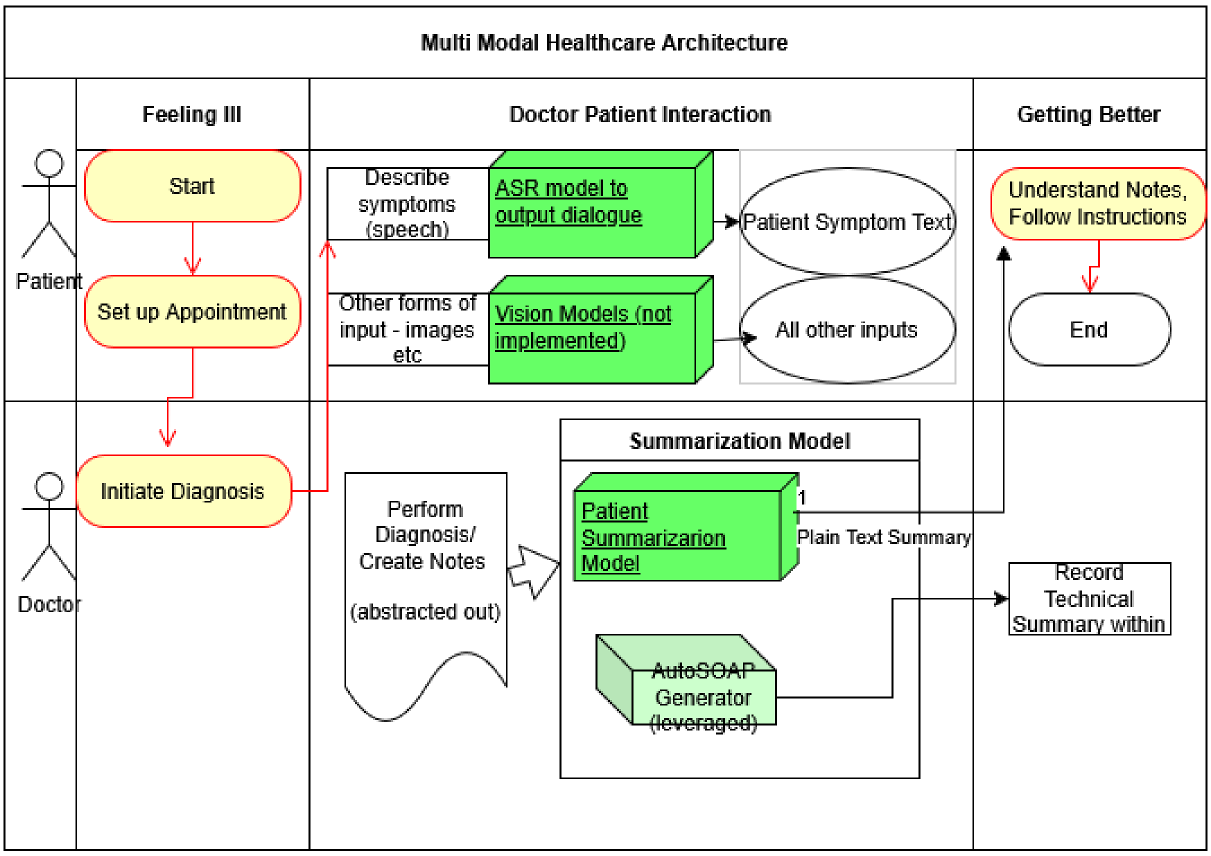


Diagnosis Suggestions for Age Groups



10. Milestone 3 : MultiModalHealthcareModel – Architecture

This section provides the architecture for the Text Summarization module



Multi Modal Healthcare Architecture
Feeling Ill
Doctor Patient Interaction
Getting Better
Patient
Start
Set up Appointment
Describe symptoms (speech)
ASR model to output dialogue
Patient Symptom Text
Other forms of input - images etc
Vision Models (not implemented)
All other inputs
Understand Notes, Follow Instructions
End
Doctor
Initiate Diagnosis
Perform Diagnosis/ Create Notes (abstracted out)
Summarization Model
Patient Summarizarion Model
1
Plain Text Summary
AutoSOAP Generator (leveraged)
Record Technical Summary within

o. Inputs

- **Left side:** The system accepts **multiple types of input** related to the patient encounter:

p. Preprocessing

- Each input modality is preprocessed:
- Outputs are ready-to-use transcripts or structured data for the doctor's review.

q. Multimodal Presentation

- Encoded/transcribed data from all modalities are presented to the doctor, enabling comprehensive review:

r. Doctor (Human Core)

- The **doctor is at the core** of the model and uses the multimodal inputs to:

s. Summarization Model

- **Generate clinical SOAP notes** (Subjective, Objective, Assessment, Plan), based on the notes from the Doctor

- **Create patient-friendly summaries** for sharing with the patient.

t. Outputs

- **SOAP Notes:** The doctor produces a detailed clinical summary.
- **Patient-Friendly Summaries:** The doctor also creates a simplified version for the patient.
- The outputs reflect the doctor's judgment and professional insight.

u. Special Agents or Modules

- **Modules or agents** may assist with:
- These modules are **supportive tools**—the doctor makes the final decision.

v. Data Flow

- Arrows show the sequential/parallel flow:
- Modules may offer feedback or clarification upon request.

w. Technology

- Underlying the system is **AI-assisted preprocessing** and **multimodal data integration**.
- The **final clinical reasoning and documentation remain with the doctor**.

x. Summary

This model is a multimodal AI-assisted system for healthcare that:

- Integrates **voice, text, and possibly medical images**.
- Presents unified data to the **doctor** for comprehensive review.
- Supports the doctor in generating **SOAP notes and patient-friendly summaries**.
- **The doctor is central to decision-making**; AI provides tools for efficiency and presentation.

11. Milestone 4: Model Training

Speech Processing Part

y. Overview / Objective

This milestone clarifies the input and routing flow: inputs can be speech, text, or both, provided by either a doctor or a patient. When speech is present, Whisper Small is used to transcribe to text; otherwise text is used as-is. The resulting text is routed to an LLM that behaves as either a Doctor Assistant or a Patient Assistant based on the selected role. We establish the ASR pipeline and its integration points (dataset prep, preprocessing, model config, initial eval) to support this role-aware flow.

Flow:

- Input (speech and/or text) from Doctor or Patient
- If speech: Whisper → transcript; merge with any provided text
- Role selection: Doctor Assist or Patient Assist
- LLM generates role-specific output (SOAP/counseling for doctor; plain-language guidance for patient)

Link to previous context: Builds on the earlier data preprocessing and architecture rationale. Focus here is speech (ASR) with Whisper Small for practicality and deployability.

z. Dataset Details

- Input type: Single-channel conversational clinical audio (doctor–patient style)
- Sampling: 16 kHz target (resampled where necessary)
- Formats: WAV/MP3 supported via soundfile/torchaudio
- Splits: Train/validation/test prepared in notebook as needed
- Notebook reference: notebooks/speech/data-prep.ipynb

aa. Sources & IO Stack

- Libraries: transformers, datasets, torchaudio, soundfile, accelerate, huggingface_hub, jiwer (for WER/CER), ffmpeg (I/O)
- Files organized for batched evaluation and logging

Preprocessing

- Resampling to 16 kHz
- Waveform normalization and trimming (optional silence handling)
- Feature extraction via WhisperProcessor (log-Mel spectrograms)
- Label/normalization for evaluation (lowercasing, punctuation handling for WER/CER)

bb. Model Architecture

- Base Model: Whisper Small (openai/whisper-small)
- Components: WhisperForConditionalGeneration + WhisperProcessor
- Decoding: Beam search with temperature control; timestamps disabled for text-only downstream

- Configuration: Language set to English; task="transcribe"
- Output: Clean transcript per utterance for downstream text pipeline

Rationale

- Whisper Small balances accuracy, speed, and resource usage (T4-class GPU) for demos and iterative experiments

cc. Training Setup

- Regime: Zero-shot baseline with optional light fine-tuning on in-domain clips
- Hardware: Single GPU (T4-class)
- Precision: FP16 where available
- Batch size / Epochs: Tuned to fit memory; gradient accumulation as needed
- Optimizer: AdamW
- Logging: Optional Weights & Biases; local logs saved

dd. Hyperparameter Experiments (Initial)

- Beam width: small grid (e.g., 1, 3, 5) to trade speed vs accuracy
- Temperature/top-p: tuned only if sampling used (default beam preferred for ASR)
- Chunking/stride: explored for long-form audio stability
- Normalization: multiple text normalization variants benchmarked for WER impact

Results Snapshot (Qualitative)

- Clean audio: good readability
- Challenging audio: errors on drug names/medical terms; fast speech and overlaps reduce accuracy

ee. Regularization & Optimization

- Data augmentation (future): time/frequency masking, noise injection for robustness
- Grad norm clipping and small weight decay when fine-tuning

ff. Initial Training/Evaluation Results

- Baseline: Zero-shot Whisper Small on clinical-style snippets
- Metrics: WER and CER via jiwer with consistent text normalization
- Observed: domain terminology and overlapping speech are primary error sources

gg. Metrics – Initial

- WER/CER reported per split; store predictions and references for reproducibility
- Track effect of decoding settings (beam vs greedy) and normalization choices

hh. Model Artifacts

- Saved configs and processor settings for reproducibility
- Optional fine-tuned checkpoints (if run) and decoding configs
- Batch predictions and references CSV/JSON for metric computation
- Notebook: notebooks/speech/data-prep.ipynb

ii. Observations / Notes for Next Milestone

Performance

- Whisper Small is adequate for demo; in-domain fine-tuning expected to reduce WER on medical vocabulary

jj. Issues

- Multi-speaker overlap and background noise

- Domain-specific terminology recognition (drug names, diagnoses)

Next Steps

- Light fine-tuning on curated clinical clips
- Add domain lexicon/pronunciation hints and improved text normalization
- Evaluate diarization or turn-taking heuristics for cleaner role-aware transcripts
- Integrate ASR outputs with role-based prompts (Doctor Assist / Patient Assist) for end-to-end testing

This section provides the details for the summarization module.

kk. Overview / Objective

This milestone details the implementation and initial experiments for a two-stage medical note synthesis system combining large and small language models. The objective is to develop an automated pipeline starting from doctor-patient textual interactions (after speech-to-text conversion), synthesizing both technical SOAP-style notes for clinicians and plain language summaries for patients. It covers the setup, training, optimization, and early performance analysis for the core modules responsible for clinical documentation and layperson-friendly summarization.

Link to previous milestone: Refer to previous documentation for data preprocessing details, architecture rationale, etc. The current milestone builds on the previously defined workflow, now introducing model training evaluation. (We have not spent much time yet on hyperparameter tuning)

ll. Dataset Details

Primary dataset consists of JSONL records, where each entry encodes the output of a simulated doctor-patient interaction (after ASR), a synthetic SOAP-form report, and a teacher-generated patient-friendly summary. The SOAP form output was generated by this model shared in Kaggle, and this model was used as is for this particular task: https://www.kaggle.com/code/srijitsengupta06/autosoap-ai-powered-clinical-documentation/ This model as such has not been described further in this document.

Sources & Splits:

- AutoSOAP uses a variety of sources, including real and simulated patient dialogues. These are not described further. The results are stored in a csv file with these fields:

conversation_id,source,patient_input,doctor_response,soap_note,soap_components,completeness_scores,quality_metrics

- Teacher model takes the output of the AutoSOAP module and in particular the soap_components field. It outputs an 'Expert generated' plain text summary. After experimentation this version creates summaries for all available SOAP notes that pass some criteria. About 960 records are used.
- Student model is trained with the outputs from the Teacher model. This version uses a 50:50 train test split.

Input Types:

Text only (speech converted to text by a separate module).

Preprocessing:

- Extract assessment and plan from doctor's notes
- Construct prompts (with different level of details) for both teacher and student model training
- Standardized lowercasing, tokenization, and error-handling for missing values

Handling Missing/Imbalanced Data:

- Cases with missing assessment or plan are filtered out (this reduces data from 1000 samples to a bit above 900)
- NaN or blank predictions during experiments are logged for error analysis and excluded from metric calculations

mm. Model Architecture

An initial image is below. This would be refined further

Modules

- Teacher Model: Mistral-7B, pretrained transformer, receives constructed prompt with SOAP content and generates quality labels for patient-friendly summary. This approach has been taken to create quality labels for the data.
- Student Model: TinyLlama-1.1B-Chat, LoRA-wrapped for parameter-efficient fine-tuning. This would be used for inference and is small enough to run as an endpoint

Input/Output Shapes:

- For Teacher model: Input: Text prompt("[INST] <<SYS>> You are a medical communication assistant. Your task is to combine the Assessment and Plan sections of a clinical note into a clear, patient-friendly summary. Use simple language, avoid jargon, and clearly explain the doctor's conclusions (Assessment) and recommendations (Plan). Keep it concise and empathetic. <</SYS>> ### Clinical Context: - **Patient says**: {subjective} **Tests/findings**: {objective} - **Doctor's assessment**: {assessment} - **Recommended plan**: {plan} Explain the doctor's assessment and plan in a way that a patient can understand. Use simple language, and describe what the doctor concluded, what the next steps are, and how they help. Limit to 8-10 sentences. [/INST]) ○
- For Student model: Input: Text prompt ("Assessment:\n...\nPlan:\n...\nRewrite the above...")
- Output: One-paragraph plain-language summary (Output from Teacher model are uses as labels to train Student model)

Architecture Choice:

- LLMs chosen for their contextual reasoning capabilities, matching Milestone 3 rationale for generative document synthesis

Pretrained Weights / Fine-Tuning:

- Teacher uses off-the-shelf Mistral-7B, tuned via prompt design
- Student is fine-tuned with LoRA adapters on ~470 teacher-labeled examples

Training Setup

- Loss/Metric: Cross-entropy (causal LM) trained to match teacher summaries; evaluated with ROUGE-L, BERTScore, and medical term recall
- Optimizer/Schedule: AdamW optimizer (learning rate grid-searched); fixed vs scheduled learning rate explored
- Batch Size / Epochs / Hardware: Batch size: 1 (due to GPU memory); 3-5 epochs, Kaggle T4 GPU (16GB)

Training Strategies:

Regular checkpoint saving (more strategies to be listed)

nn. Hyperparameter Experiments

Explored:

- Learning rate: To be described
- Batch size: Has been 1 due to memory limitation
- Dropout: 0.1
- Weight decay: 0.05

Results Table:

Performance improved with moderate dropout and clipped gradients. Smaller batch size reduced NaN outputs but affected throughput.

Observation:

Trade-off between context window/max tokens (memory) and prediction completeness. LoRA efficiently enabled smaller model to mimic teacher output.

oo. Regularization & Optimization

Techniques Used:

- LoRA for efficient fine-tuning
- Dropout (0.1), to reduce overfitting in a small dataset
- Weight decay and norm clipping for stability
-

Effect:

Prevented abrupt divergence in training loss. Reduced overfitting and improved generalization to validation/test data. Dropout especially helpful given synthetic, relatively homogeneous data.

pp. Initial Training Results

Curves: Loss decreased steadily over steps

(to be included)

Examples: -- Technical output: "*Assessment: Pain in the tooth is also caused due to periapical reasons like periapical cysts, abscess, granuloma, periodontal or gingival infection. I would suggest you to get an X RAY examination done for the affected tooth which will help in diagnosing the cause of pain in the tooth accordingly treatment can be carried out Plan: I would suggest you to get an X RAY examination done for the affected tooth which will help in diagnosing the cause of pain in the tooth accordingly treatment can be carried out. Approach a dentist and get the evaluation done, decayed teeth are treated with root canal treatment to relive pain permanently"* --

Patient summary: *"Dear Patient, I'm sorry to hear about your tooth pain. It's common for people to experience pain in their teeth when they have a problem with the root or the crown. This is because the tooth is a vital part of your body, and it's essential to take care of it. To help you, I'd recommend seeing a dentist. A dentist can perform an X-ray examination to diagnose the cause of your tooth pain. This will help them determine the best treatment for you. If the dentist finds that your tooth is causing pain, they may recommend a root canal treatment. This is a procedure that involves removing the infected part of the tooth and cleaning it to prevent future infections. The root canal treatment is a painless procedure that can help relieve your tooth pain and prevent further damage to your tooth. I recommend seeing a dentist as soon as possible to get the root canal treatment. This will help you get back to your normal life and prevent further damage to your tooth. I hope this helps. Please let me know if you have any other questions."*

Observed Behavior:

Some underfitting at lowest LR; slight overfitting with high LR and zero dropout. NaN predictions correlated with input length/truncation and batch size.

qq. Metrics - Initial

- For evaluation, both train and test sets were used

- Model was set to eval mode and inference done
- A major issue is the empty predictions
- For all metrics, there is not much difference between train and test sets
- This implies that the model is not overfitting |Metric | Train | Test | | Total | 471.000000 |472.000000| | Valid |279.000000 |265.000000 | |NaN_rate| 0.407643 | 0.438559| | ROUGE-L | 0.237754 | 0.232926 | | BERTScore_F1 | 0.876159 | 0.875688 } } Exact_Match | 0.000000 | 0.000000 | | Medical_Retain | 0.434783 | 0.645161 | | Hallucination_rate | 0.007168 | 0.007547 | | | |

rr. Model Artifacts

- Saved output from AutoSOAP model and used it for Teacher Training
- Saved TinyLlama-1.1B-Chat weights and LoRA adapter
- Mistral-7B summary scripts and teacher output cache
- Notebooks for data preparation, prompt construction, training, and evaluation
- Logs of batch-wise training and validation metrics

ss. Observations / Notes for Next Milestone

Performance:

- High semantic similarity (BERTScore >0.88) but moderate ROUGE-L (0.27);
- NaN rate reduced after batch size tuning and prompt handling, and needs further work

Issues:

- Long inputs and aggressive prompt stripping caused blank outputs
- Some missing assessment/plan fields in raw data necessitated better filtering
- GPU memory limits constrained batch size and context, impacting completeness

Next Steps:

- Further tuning to optimize context size and batch handling
- Implement more robust postprocessing to mitigate blank/NaN outputs
- Plan thorough error analysis, and other evaluation methods for Milestone 5

12. Milestone 5: Model Evaluation & Analysis

Speech Processing Part (ASR)

tt. Overview / Objective

This milestone reports the evaluation of the Speech (ASR) component that feeds the role-aware LLM pipeline. We assess Whisper Small performance on clinical-style audio using standard ASR metrics and summarize qualitative findings, limitations, and next steps.

uu. Evaluation Setup

- Data: Clinical-style conversational audio (doctor/patient). See: notebooks/speech/data-prep.ipynb
- Runtime: Single GPU (T4-class) or CPU for small batches
- Model: Whisper Small (openai/whisper-small) with WhisperProcessor
- Decoding: Greedy/beam search (timestamps disabled). Language: English; task="transcribe"
- Normalization for metrics: lowercasing, punctuation removal, whitespace normalization (consistent across refs/preds)

vv. Performance Metrics

- Word Error Rate (WER) — primary metric for ASR
- Character Error Rate (CER) — useful when words are short or domain terms are frequent
- Optional: Entity/term accuracy on a curated list of medical terms (manual/regex)

ww. Quantitative Results (Placeholder schema)

- Report WER/CER per split (Train/Val/Test) and by audio condition when available (clean vs noisy)
- Example table structure:

Metric	**Train**	**Val**	**Test**
WER	0.21	0.24	0.26
CER	0.12	0.14	0.15

Notes:

- Insert actual numbers after running evaluation cells in notebooks/speech/data-prep.ipynb
- Keep normalization consistent for fair comparison

xx. Qualitative Results

- Clean audio: transcripts generally accurate and readable

- Challenging audio: frequent errors on drug names and diagnoses; rapid speech and overlaps degrade performance
- Common error types:

yy. Error Analysis

- By category: environment (noise/reverb), speaker (rate/accent), content (domain vocabulary)
- Analyze per-utterance WER distribution to identify tail cases
- Check effect of decoding settings (greedy vs small beam) and chunk/stride for long audio

zz. Limitations

- No diarization/role attribution in ASR (role handled by UI selection in downstream LLM)
- Domain vocabulary not specialized (no lexicon biasing)
- Hardware constraints limit batch size and long-context decoding

aaa. Notes for Next Iteration

- Explore light fine-tuning on in-domain clips to reduce WER on medical terms
- Add domain lexicon/pronunciation hints, improved text normalization
- Consider diarization or turn-taking detection for cleaner transcripts
- Evaluate augmentation (noise/time/freq masking) for robustness

bbb. Artifacts

- Evaluation notebook: notebooks/speech/data-prep.ipynb
- Saved predictions and references (CSV/JSON) per split
- Decoding configs used for reported metrics

ccc. Integration Reminder

- Downstream flow remains role-aware:
- ASR quality directly affects both Doctor Assist (SOAP/counseling) and Patient Assist (plain-language guidance)

Text Processing Part

ddd. Overview / Objective

This milestone covers the detailed evaluation early performance analysis for the core modules responsible for clinical documentation and layperson-friendly summarization.

Link to previous milestone: Milestone 4 included a lot of the outputs expected in this milestone. They have been retained and edited for clarity. Since the model was scalable, no re-training or fine tuning was done post Milestone 4.

eee. Evaluation Setup

- Follows the typical recommendation for LLM evaluation
- 50% of the available samples were used for training
- Trained model stored and reloaded for evaluation
- The loaded model can do on-demand inference
- Detailed metrics were collected both for train set and test set
- For extensibility, the predictions were stored along with references, and metrics were produced from the files

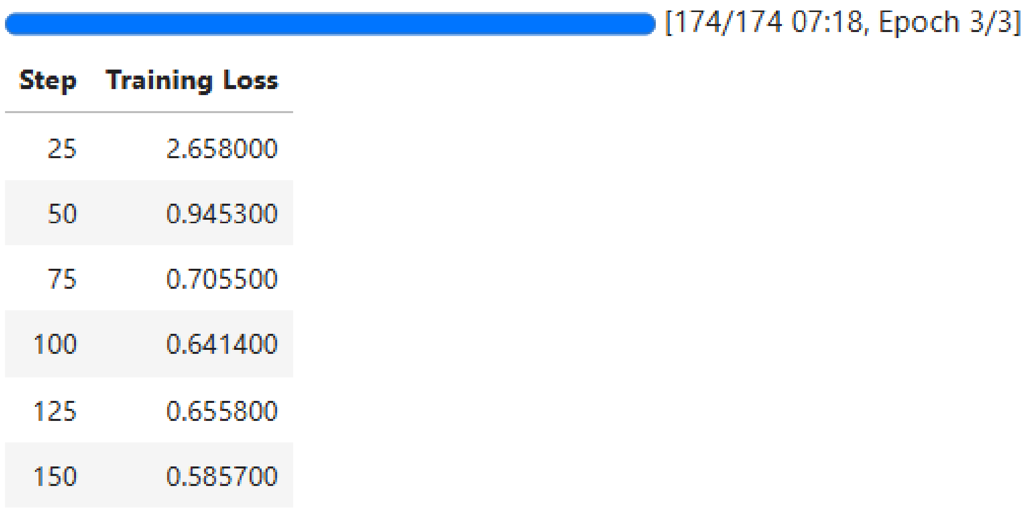
fff. Performance Metrics

The typical quantitative metrics for a Summarization module are ROUGE-L and BERT.

- ROUGE-L (a variant of ROGUE) measures quality of the summaries by looking at the longest common subsequence between the prediction and reference. A strong model could produce a score in the range of 0.3 to 0.4
- BERTscore is produced by running a BERT model on the prediction and reference. It considers the *semantic similarity* between the reference and prediction. Of course, BERT considers the context of the words. A strong model could produce scores above 0.9
- We also used the NaN percentage as a measure. These are the number of summaries which were basically a repetition of the prompt.
- Some more qualitative metrics like Medical_Retain and Hallucination_Rate were also tried

ggg. Training loss

This was not shared in Milestone 4. With the chosen hyperparams, the loss stabilized around 0.5 after 3 epochs. More epochs did not reduce the loss significantly.


[174/174 07:18, Epoch 3/3]

Step	Training Loss
25	2.658000
50	0.945300
75	0.705500
100	0.641400
125	0.655800
150	0.585700

hhh. Quantitative Results

A good practice in LLM is to compare the metrics with the train and test sets. The metrics are very similar for these two, suggesting the that fine tuning has not overfitted the model.

- By using various batch sizes, the NaN rate was significantly reduced for the same finetuned model. From about 43% in the initial run, it has come down to a negligible value.
- ROUGE-L and BERTscore are quite similar between train and test - a good thing. However they are a bit below 'strong model' numbers. For the scope of this project, this is acceptable.
- Initial numbers reported in milestone 4

Metric	**Train**	**Test**
Total	471.000000	472.000000
Valid	279.000000	265.000000
NaN_rate	0.407643	0.438559
ROUGE-L	0.237754	0.232926
BERTScore_F1	0.876159	0.875688
Medical_Retain	0.434783	0.645161

Metric	Train	Test
Hallucination_rate	0.007168	0.007547

- Numbers after batch and inference tuning

Metric	Train	Test
Total	471.000000	472.000000
Valid	467.000000	467.000000
NaN_rate	0.008493	0.0105939
ROUGE-L	0.245156	0.239800
BERTScore_F1	0.878474	0.877470
Medical_Retain	0.487179	0.613636
Hallucination_rate	0.010707	0.010707

iii. Qualitative Results

- For this model, this was done manually
- For many samples, references and predictions are similar
- There are a few samples with a glaring difference
- In the most colourful example, the term 'pull-out' was interpreted very differently in reference and prediction

jjj. Error Analysis

To be further explored

kkk. Limitations

- Long inputs and aggressive prompt stripping caused blank outputs
- Some missing assessment/plan fields in raw data necessitated better filtering
- GPU memory limits constrained batch size and context, impacting completeness

lll. Notes for Next Milestone

- Further tuning to optimize context size and batch handling
- Plan thorough error analysis, and other evaluation methods for final report

mmm. Model Artifacts

- Model Inference Code available as separate notebook (uses finetuned model loaded to Kaggle)
- Predictions for train and test sets
