

# A Low-Latency Mixture-of-Experts Orchestrator for Domain-Specific Mental Health LLMs

Mohammad Faiz and Vaibhav Bajaj

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

{cs5210128,cs5210126}@iitd.ac.in

## Abstract

Model Repo: <https://huggingface.co/reaper-113/Llama-3.2-1B-Instruct-MOE-Finetuned>

We present a practical Mixture-of-Experts (MoE) system for domain-specialized mental-health dialogue on resource-constrained hardware. Our architecture combines (i) five small expert language models, each fine-tuned on a distinct mental-health disorder, with (ii) a lightweight orchestrator that routes user queries to the most appropriate expert. Because the provided data consisted of full textbooks rather than ready-made question-answer (QA) pairs, we first construct a high-quality synthetic supervision set by prompting a large external model (Gemini-2.5-Flash-Lite) to generate QA pairs from book chunks. Each expert is then trained as a parameter-efficient LoRA adapter on top of Llama-3.2-1B-Instruct. We additionally design a TF-IDF + Logistic Regression router trained first on instruction texts and later on raw book chunks, achieving a macro-F1 of 0.83 on millions of labeled spans. To evaluate end-to-end latency under realistic multi-client load, we replay Azure-server traces against a custom inference server that supports batching, LRU-based model management, PEFT adapter switching, and quantized GGUF backends via llama.cpp. Our experiments show that (1) domain experts provide more clinically grounded answers than the base model despite slightly worse cross-entropy, and (2) replacing HF CPU inference with 8-bit GGUF models reduces end-to-end latencies by roughly 2–3× while keeping routing accuracy high.

## 1 Introduction

Mixture-of-Experts (MoE) architectures offer a principled way to scale language models by activating only a small subset of parameters per input (?). In practice, however, most MoE systems assume GPU-rich environments and do not directly address low-latency inference on commodity CPUs.

This work targets a concrete and constrained setting: a mental-health assistant that must route user queries to one of several small, domain-specific experts (e.g., anxiety, depression, bipolar disorder) and run end-to-end on CPU with tight latency constraints. Our goal is not to push SOTA benchmarks, but to design a *full pipeline*—from data creation to training, routing, and deployment—that is technically sound, latency-aware, and easily reproducible.

The assignment constraints created several challenges: (i) training data was provided as five long PDF textbooks (one per disorder), not as curated question-answer pairs; (ii) the base model was fixed to Llama-3.2-1B-Instruct; and (iii) the system needed to be evaluated under realistic, asynchronous client load emulating Azure traces.

We address these by: (1) generating high-quality QA pairs from each textbook via Gemini-2.5-Flash-Lite; (2) training LoRA experts on the synthetic instruction-answer pairs; (3) designing a text-based orchestrator trained first on instructions and later on raw book chunks; and (4) implementing a custom inference server with model caching, batching, and quantized GGUF backends using llama.cpp.

## Contributions.

- A data-generation pipeline that converts domain textbooks into thousands of clinically-focused QA pairs using a strong external LLM.
- Five domain experts trained as LoRA adapters on Llama-3.2-1B-Instruct, with qualitative analysis of specialization vs. overfitting.
- A TF-IDF + Logistic Regression orchestrator trained on millions of book-derived spans, achieving 0.83 macro-F1 across five disorders.
- A realistic evaluation harness using Azure-style traces, including an inference server with

083	batching, LRU model management, adapter	3.2 Chunking and Prompting	126
084	switching, and GGUF-based quantized ex-	Each PDF is first read via ‘pymupdf’, and then is	127
085	perts.	split into contiguous chunks of 800–1000 words	128
086		using simple heuristics based on paragraph bound-	129
087	<b>2 System Overview</b>	aries. For each chunk, we call Gemini-2.5-Flash-	130
088	Our system is a sparse MoE architecture with ex-	Lite with the following prompt template:	131
089	PLICIT routing. Figure 1 illustrates the main compo-		
090	nents.		
091	At a high level, the pipeline is:	You are an expert in {domain_label}.	132
092		Read the text below and generate 5	133
093	1. <b>Data construction:</b> Each disorder-specific	unique Question-Answer pairs. Focus	134
094	PDF is chunked into 800–1000 word spans.	on clinical/therapeutic advice found	135
095	Using Gemini-2.5-Flash-Lite, we generate	in the text. Format strictly as a	136
096	multiple QA pairs per span, yielding ~750	JSON list: [{ "instruction": "User	137
097	QA pairs per disorder.	Question", "output": "Expert Answer"	138
098		}]\n\nTEXT:{text_chunk}	139
099	2. <b>Expert training:</b> For each of the five do-		
100	main (anxiety, bipolar, depression, OCD,	We found this task to be well within the capabil-	140
101	schizophrenia), we fine-tune a LoRA adapter	ities of a small, fast model like Gemini-2.5-Flash-	141
102	on top of Llama-3.2-1B-Instruct using next-	Lite; the generated QA pairs were generally well-	142
103	token prediction over instruction + output text.	formed, clinically sensible, and closely tied to the	143
104		underlying text.	144
105	3. <b>Orchestrator training:</b> We initially train a	After filtering malformed JSON and very short	145
106	TF-IDF + Logistic Regression classifier on	answers, we obtained roughly 750 QA pairs per dis-	146
107	the synthetic instruction texts. Observing	order, for a total of ~3,750 high-quality instruction-	147
108	some qualitative mismatch, we retrain the	answer examples.	148
109	orchestrator directly on book chunks (sub-		
110	spans), yielding ~ 19.9M labeled training	<b>3.3 Text Cleaning and Tokenization</b>	149
111	examples across domains.	Before training, we apply light preprocessing and	150
112		concatenate instruction and output with a sim-	151
113	4. <b>Deployment and evaluation:</b> We imple-	ple chat-like format for language model training.	152
114	ment a multi-client setup replaying Azure-like	We use the tokenizer associated with Llama-3.2-	153
115	traces. A central server hosts the orchestrator	1B-Instruct and cap sequence length at 512 tokens,	154
116	and a pool of experts, manages LRU-based	truncating long outputs as needed.	155
117	model loading, batching, and inference, and		
118	logs end-to-end latencies. We support both	<b>4 Expert Model Training</b>	156
119	HF-based LoRA experts and quantized 8-bit		
120	GGUF experts via llama.cpp.	<b>4.1 Base Model and Parameter-Efficient</b>	157
121		<b>Fine-Tuning</b>	158
122	<b>3 Data Construction</b>	All experts share a common base model:	159
123		meta-llama/Llama-3.2-1B-Instruct. To keep	160
124	<b>3.1 Source Textbooks</b>	memory and disk footprint small, we train domain-	161
125	We received five textbooks in PDF form, one for	specific <i>LoRA</i> adapters on top of the base model,	162
	each disorder: anxiety, bipolar disorder, depres-	rather than fully fine-tuning all parameters.	163
	sion, obsessive-compulsive disorder (OCD), and	We use PEFT-style LoRA with rank $r = 16$ , $\alpha =$	164
	schizophrenia. These are long-form clinical and	32, and dropout 0.05, targeting attention and MLP	165
	therapeutic texts targeted at practitioners and ad-	projection layers: {q_proj, k_proj, v_proj,	166
	vanced students, not directly usable as supervised	o_proj, gate_proj, up_proj, down_proj}}.	167
	dialogue data.	<b>4.2 Training Objective and Setup</b>	168
		We use a simple next-token prediction objective	169
		(causal LM loss) over the concatenated instruction	170
		+ output texts, without any special RL or prefer-	171
		ence optimization. Hyperparameters are chosen	172
		conservatively:	173

174	• sequence length: 512 tokens;	Based on this, we treat perplexity as a secondary	217
175	• batch size: 8 per device (with gradient accu-	metric and consider 3 epochs to be a “sweet spot”	218
176	• learning rate: $2 \cdot 10^{-4}$ for LoRA parameters;	that balances specialization, readability, and over-	219
177	• epochs: 1, 3, and 5 (for ablations);	fitting. Figure 2 (placeholder) will visualize the	220
178	• optimizer: AdamW with weight decay 0.01.	trade-off.	221
179			
180	We train on a single A100 40GB GPU. Each	<b>5 Orchestrator (Router) Design</b>	222
181	domain-specific expert trains in under a minute for	<b>5.1 Initial Instruction-Level Router</b>	223
182	a few epochs, thanks to the small model size and	We first trained an orchestrator using only the	224
183	LoRA parameterization.	instruction fields from our synthetic QA dataset.	225
184		We model routing as a 5-way text classification	226
185	<b>4.3 Base vs. Expert Validation Loss</b>	problem over disorders, using a lightweight scikit-	227
186	Somewhat counterintuitively, we observe that the	learn pipeline:	228
187	base model sometimes achieves slightly lower vali-	• <b>Features:</b> TF-IDF vectors with unigrams and	229
188	dation cross-entropy on our small domain datasets	bigrams, <code>max_features = 20000</code> , <code>min_df =</code>	230
189	than the LoRA experts. This is expected:	2, English stopwords.	231
190	• the base model is trained on massive, di-	• <b>Classifier:</b> Logistic Regression with L2	232
191	verse corpora and is a strong general language	penalty, <code>max_iter = 1000</code> , and class-	233
192	model;	balanced weights.	234
193	• our domain-specific datasets are small ( $\sim 750$	On a stratified 70/15/15 train/validation/test split	235
194	examples per disorder) and narrow, so addi-	(over $\sim 750$ instructions per class), we compared	236
195	tional finetuning can degrade global perplex-	Logistic Regression and Multinomial Naive Bayes.	237
196	ity;	Logistic Regression outperformed Naive Bayes:	238
197	• our validation loss is computed over both user	• val accuracy: 89.6%;	239
198	and assistant tokens, which may penalize do-	• macro-F1: 0.897 vs. 0.861 for Naive Bayes.	240
199	main specialization.	Per-class F1 scores for Logistic Regression	241
200		were:	242
201	However, qualitative inspection shows that the	• anxiety: 0.93,	243
202	experts produce <i>more domain-specific, clinically</i>	• bipolar: 0.85,	244
203	<i>relevant</i> responses. For example, given “ <i>What are</i>	• depression: 0.81,	245
204	<i>common physical sensations someone with anxiety</i>	• OCD: 0.91,	246
205	<i>might experience?</i> ”, we observe:	• schizophrenia: 0.97.	247
206	• <b>1 epoch:</b> answers remain relatively generic,	Most misclassifications occurred between bipo-	248
207	describing “feeling drained”, “overwhelmed”,	lar and depression, and between anxiety and	249
208	and generic stress responses.	OCD, which is clinically plausible given overlap-	250
209	• <b>3 epochs:</b> responses reference domain-	ping symptom vocabulary (mood swings, rumina-	251
210	specific constructs such as interoceptive cues,	tion, worry, compulsive checking). Schizophrenia	252
211	differentiation from cardiovascular disease,	queries were almost perfectly separated due to dis-	253
212	and cognitive-behavioural strategies (e.g.,	distinctive language about hallucinations, delusions,	254
213	CBT, relaxation training).	and thought disorder.	255
214	• <b>5 epochs:</b> answers become more textbook-	We then retrained Logistic Regression on the	256
215	like, summarising research findings and ex-	combined train+val set and serialized the full	257
216	perimental paradigms (e.g., voluntary hyper-	pipeline (TF-IDF + classifier) via <code>joblib</code> for use	258
	ventilation studies) in a way that is less con-	in the inference server.	259

Class	P	R	F1
anxiety	0.91	0.79	0.85
bipolar	0.89	0.76	0.82
depression	0.87	0.88	0.87
OCD	0.67	0.91	0.77
schizophrenia	0.91	0.76	0.83
overall acc		0.82	
macro-F1		0.83	

Table 1: Router performance on book-chunk test set (3.99M examples). Training on raw textbook spans improves qualitative routing on free-form queries, at the cost of slightly lower but still strong macro-F1.

## 5.2 Book-Chunk Router for Better Generalization

Although the instruction-level router performed well on held-out instructions, we observed that it sometimes misrouted *free-form user queries* that resembled the textbook text more than our synthetic questions. To address this, we re-trained the orchestrator directly on book chunks.

For each PDF, we:

- chunked the text into spans (as in QA generation);
- further subdivided each chunk into smaller spans (e.g., sentences or overlapping windows);
- labeled each span with its disorder domain (anxiety, bipolar, depression, OCD, schizophrenia).

We sampled 50 base chunks per book for Gemini QA, but we used many more spans for router training. In total, we constructed 19,976,983 labeled training examples across the five domains, and held out 3,995,397 examples for testing:

Split: 15,981,586 train / 3,995,397 test.

We trained the same TF-IDF + Logistic Regression pipeline on this much larger dataset. Table 1 reports test performance.

Compared to instruction-only training, overall accuracy and macro-F1 are slightly lower (0.82 vs. 0.90), but this is expected: book spans contain more ambiguous, mixed-topic content than synthetic questions. In qualitative testing on user-style prompts, the book-chunk router produced more robust and intuitive route decisions, so we use it as our final orchestrator.

## 6 Inference Server and Load Generation

### 6.1 Azure-Inspired Client Traces

To evaluate latency under realistic multi-client load, we follow Azure-server traces commonly used for LLM service benchmarking. We take a week-long trace of function invocation times and restrict to the first 100 seconds for convenience.

Each trace entry includes:

- a function ID (used to map to an expert domain in round-robin fashion),
- a relative start time,
- a unique request ID.

We implement a trace-driven client that:

1. sleeps until the relative start time,
2. issues an asynchronous HTTP request to /infer on the server,
3. records send timestamp and end-to-end latency.

Prompts for these experiments are sampled from our QA instructions (and additional free-form queries), with ground-truth labels taken from the underlying dataset for routing evaluation.

### 6.2 Server Architecture

The server exposes a single /infer endpoint. Upon receiving a request, it:

1. runs the orchestrator to obtain a domain label and expert identifier;
2. checks whether the corresponding expert model is already loaded;
3. if loaded, it enqueues the request into that expert’s queue;
4. if not loaded:
  - if the number of loaded models is below a configured maximum, it loads the expert;
  - otherwise, it identifies the least-recently-used (LRU) expert, evicts it, and loads the new expert.
5. after loading, the server waits up to 100 ms for additional requests targeting the same expert to arrive, forming a batch;



335	6. it performs batched inference on the CPU	7	<b>Device Specifications</b>	378
336	backend;		• Memory - 16.0 GB; Speed: 7467 MT/s	379
337	7. finally, it sends responses back to clients and		• CPU - Intel(R) Core(TM) Ultra 7 155H; Base	380
338	logs timing information.		speed: 3.80 GHz;	381
339	We run separate logging threads on both server		• Cores: 16; Logical processors: 22;	382
340	and clients to collect trace-aligned metrics: se-		• L1 cache: 1.6 MB; L2 cache: 18.0 MB; L3	383
341	lected experts, queue times, load times, inference		cache: 24.0 MB	384
342	times, and end-to-end latencies.	8	<b>Experimental Results</b>	385
343	<b>6.3 Optimizations</b>	8.1	<b>End-to-End Latency</b>	386
344	We implemented several optimization passes:		We evaluate three server variants under the same	387
345	<b>Naive version.</b> Each expert is a full HF model		Azure-style client trace (replaying timestamps and	388
346	(base + LoRA weights) loaded and unloaded in-		function IDs, mapping functions to experts in	389
347	dependently. Loading a 1B model on CPU can		round-robin fashion):	390
348	take several seconds, and repeated loads dominate		1. <b>Baseline HF server:</b> a straightforward HF	391
349	end-to-end latency.		deployment that loads and unloads full FP16	392
350	<b>Base Optimisations.</b> Due to memory limits of		experts on demand.	393
351	our device, we needed to create a strict check for		2. <b>Dual-LM server (adapter-switch):</b> keeps	394
352	the number of models loaded at a time in the mem-		one or more base models resident and	395
353	ory to not cause BSOD errors in Windows, along		switches LoRA adapters instead of unloading	396
354	with LRU eviction policy. Apart from this, we also		whole models.	397
355	implemented batching support in the server.		3. <b>GGUF server:</b> uses 8-bit GGUF experts via	398
356	<b>Adapter-switch version.</b> Instead of unloading		llama-cpp-python, with quantized experts	399
357	the entire model, we keep the <i>base</i> model resident		loaded on demand.	400
358	in memory and only switch PEFT adapters when		Each client issues asynchronous POST /infer	401
359	changing domains. This greatly reduces load time,		requests after sleeping to match the original trace	402
360	since adapter weights are small. We further extend		timestamps. The server logs for every request:	403
361	this to support multiple bases (e.g., one base for		• <i>e2e_time_ms</i> : client send to client receive;	404
362	some disorders and another for others), each with		• <i>server_total_time_ms</i> : server receive to re-	405
363	its own set of adapters.		sponse completion;	406
364	<b>GGUF + llama.cpp version.</b> For maximum		• <i>queue_time_ms</i> : time spent in the expert-	407
365	CPU efficiency, we:		specific queue before batching;	408
366	1. merge each LoRA adapter into the base		• <i>load_time_ms</i> : time spent loading or attach-	409
367	Llama-3.2-1B-Instruct weights to obtain a full		ing a model / adapter;	410
368	FP16 expert;		• <i>inference_time_ms</i> : pure generation time in-	411
369	2. convert the merged HF model to GGUF using		side the backend;	412
370	<code>convert_hf_to_gguf.py</code> ;		• <i>orchestrator_time_ms, preprocess_time_ms</i> :	413
371	3. quantize to 8-bit (q8_0) GGUF format;		router and tokenization overheads.	414
372	4. serve these experts via llama-cpp-python,		The major bottleneck here was the Memory	415
373	which provides a fast CPU backend with		Limit of 16 Gb imposed by our device. Memory	416
374	batched inference.		was quickly filled, as shown in Figure 4 in the base	417
375	This pipeline preserves the semantics of the			
376	trained experts while reducing memory footprint			
377	and improving inference throughput.			

server, where only 2 models could be loaded and inferred at a time. CPU usage was always 100% when requests came in bursts as shown in Figure 5. The best part with using quantized GGUF models was that we were able to load all the five models at once on the CPU, which reduced the queue time and repeated load-unload times by a lot, as shown in Figure 6

### 8.1.1 Latency over Time and Load

Figure 7 shows end-to-end latency over wall-clock time for all three techniques, aligned by client timestamps. GGUF exhibits the tightest latency band with relatively low variance. The Baseline HF server shows frequent spikes under bursty load, and the Dual-LM server experiences prolonged periods of extremely high latency when queues back up.

To contextualize these spikes, Figure 8 plots requests-per-second (RPS) over time. Peaks in RPS align closely with queue build-up and e2e latency spikes, especially for Baseline and Dual-LM, while GGUF remains substantially more stable.

### 8.1.2 Latency Component Breakdown

We further decompose server time into queue, load, and inference components. Figure 9 shows a stacked area view over time; queue time dominates the tail, followed by inference. GGUF makes load time almost negligible, while the Baseline HF server frequently spends several seconds loading models.

Figure 10 shows the e2e latency histogram. GGUF has a relatively tight unimodal distribution with a much shorter tail. Baseline and Dual-LM exhibit heavy right tails, with Dual-LM particularly pathological under sustained load.

A correlation heatmap (Figure 11) confirms that *queue\_time\_ms* is the primary driver of e2e latency across techniques, followed by *inference\_time\_ms*. Load time correlates strongly with latency for the Baseline HF server but is negligible for GGUF.

### 8.1.3 Per-component Comparison Across Techniques

Table 2 summarizes the p95 latency for each major component across the three server variants.

From Table 2 we observe:

- **Queue time dominates tail latency.** All three systems are ultimately bottlenecked by queuing, but GGUF reduces p95 queue time by

Component	GGUF	Baseline HF	Dual-LM
Queue time (ms)	44,820	170,717	193,563
Load time (ms)	800	118,972	—
Inference time (ms)	31,172	62,417	40,996
Total time (ms)	57,959	198,099	214,102

Table 2: p95 latency (in milliseconds) for key components across techniques. Dual-LM has no explicit model load (only adapter switches), hence “—” for load time.

roughly 3–4× compared to Baseline and Dual-LM (44.8s vs. 170.7s / 193.6s).

- **Baseline HF suffers catastrophic load time.** Its p95 load time (118.9s) is two orders of magnitude larger than GGUF (0.8s), and directly amplifies both queue and total latency.
- **GGUF is the fastest inference backend.** GGUF achieves the lowest p95 inference time (31.2s), Dual-LM is moderate (41.0s), and Baseline HF is slowest (62.4s).
- **Dual-LM collapses under load despite zero load time.** Removing model-load cost is not enough: p95 total time for Dual-LM (214.1s) is actually worse than Baseline HF (198.1s) because queues explode under bursty traffic.
- **Overall ranking.** Combining all components, GGUF is clearly the most robust deployment strategy (p95 total 57.9s), while Baseline HF and especially Dual-LM are not viable under sustained load.

## 8.2 Qualitative Case Studies

To complement the aggregate metrics, we inspect individual routed conversations. Below we show representative examples where the router and experts behave as intended.

### Bipolar: mood swings between extremes.

**User:** “My mood keeps swinging between feeling like a genius and feeling like a complete failure.”

**Bipolar expert:** “It’s not uncommon for people experiencing bipolar disorder to have pronounced mood swings. These shifts can be intense and affect many aspects of life, including relationships, work, and daily activities. The feelings of being a ‘genius’ and a ‘failure’ are often linked to the manic and depressive phases, respectively. It’s important to

504	seek professional help to manage these	A therapist can work with you to under-	552
505	fluctuations and develop effective coping	stand your thought patterns, gradually	553
506	strategies. A therapist can help you iden-	reduce reliance on rituals, and develop	554
507	tify patterns in your mood, work through	more adaptive coping strategies.”	555
508	these feelings, and create a personalised	The OCD expert explicitly names OCD, cor-	556
509	plan to improve daily functioning.”	rectly links rituals to distress-reduction, and ref-	557
510	In this case, the router assigns the query to the	erences CBT/ERP as gold-standard interventions.	558
511	bipolar expert, which in turn correctly frames the	<b>OCD: checking compulsion.</b>	559
512	symptom pattern in terms of manic and depres-	<b>User:</b> “I have to check the door multiple	560
513	sive episodes and recommends appropriate clinical	times before I can leave, or I feel some-	561
514	follow-up.	thing bad will happen.”	562
515	<b>Anxiety: social/performance anxiety.</b>	<b>OCD expert:</b> “Needing to check the	563
516	<b>User:</b> “My heart races, my hands shake,	door repeatedly, combined with a fear	564
517	and I can’t focus whenever I have to talk	that something bad will happen if you	565
518	to someone new.”	do not, is a common presentation of	566
519	<b>Anxiety expert:</b> “These symptoms sug-	Obsessive–Compulsive Disorder (OCD).	567
520	gest a form of social anxiety, sometimes	Intrusive thoughts about danger can	568
521	called social phobia. It involves an in-	trigger compulsive checking as a way	569
522	tense fear of social or performance sit-	to temporarily feel safer. Cognitive–	570
523	uations where you might feel judged,	behavioural therapy and exposure-based	571
524	evaluated, or embarrassed. Physical sen-	approaches can help you gradually re-	572
525	sations such as a racing heart, shaking	duce the checking and tolerate the anx-	573
526	hands, and difficulty focusing are com-	iety that comes up when you resist the	574
527	mon manifestations of this anxiety. If	urge to check.”	575
528	these experiences are affecting your daily	This example shows that the system also handles	576
529	life, it can be helpful to speak with a men-	canonical checking-type compulsions in a clini-	577
530	tal health professional. They can work	cally plausible way.	578
531	with you to understand your fears, de-	<b>Schizophrenia: auditory hallucinations.</b>	579
532	velop coping strategies, and gradually	<b>User:</b> “I hear voices commenting on	580
533	build confidence in social situations.”	what I’m doing, but nobody else seems	581
534	Here the system correctly interprets the prompt	to hear them.”	582
535	as social anxiety, explains the fear-of-judgement	<b>Schizophrenia expert:</b> “What you de-	583
536	mechanism, and provides standard psychoeduca-	scribe sounds like auditory hallucina-	584
537	tional guidance.	tions, which can be a challenging part of	585
538	<b>OCD: obsessions, rituals, and ERP.</b>	conditions such as schizophrenia. These	586
539	<b>User:</b> “My mind gets stuck on certain	voices can be intrusive and may make	587
540	thoughts and I can’t move on until I re-	comments or criticisms about you or	588
541	peat a ritual.”	your actions, even though other people	589
542	<b>OCD expert:</b> “That pattern is com-	cannot hear them. It is important to talk	590
543	mon in Obsessive–Compulsive Disorder	with a mental health professional, such	591
544	(OCD). Repeating rituals can temporar-	as a psychiatrist or psychologist, who can	592
545	ily reduce the distress caused by un-	help you understand what is happening	593
546	wanted thoughts, but over time they can	and discuss treatment options to reduce	594
547	also intensify anxiety and interfere with	the impact of the voices on your daily	595
548	daily life. Treatments such as cognitive–	life.”	596
549	behavioural therapy (CBT) and expo-	The router correctly assigns this prototypical psy-	597
550	sure and response prevention (ERP) are	chotic symptom to the schizophrenia expert, which	598
551	specifically designed to help with this.	responds with appropriate framing and a recom-	599
		mendation for specialist care.	600

**Optional examples.** For completeness, we also observe more generic but still reasonable responses:

- *Bipolar fatigue:* prompts such as “I feel like I’m constantly tired, even when I sleep enough.” are routed to the bipolar expert, which provides a generic discussion of fatigue and encourages medical evaluation. This is sensible but less clearly disorder-specific.
- *Worthlessness thoughts:* prompts like “I keep thinking that everyone would be better off without me, even if I don’t want to do anything extreme.” are often routed to the anxiety expert, which responds with reassurance, self-worth framing, and suggestions for seeking support. These cases illustrate grey areas where content overlaps with depression, anxiety, and suicidality.

## 9 Discussion

Our results illustrate several trade-offs in building a practical MoE system under resource constraints.

**Data generation vs. quality.** Using a strong external LLM (Gemini-2.5-Flash-Lite) to generate QA pairs from textbooks yields high-quality supervision with relatively little engineering effort, but inherits both the biases and stylistic preferences of the external model and the source books.

**Perplexity vs. specialization.** Domain-specific fine-tuning on small datasets slightly worsens global cross-entropy but improves task-specific quality. For mental-health dialogue, we explicitly prioritize clinically relevant content and routing behaviour over raw perplexity, choosing 3-epoch experts as a compromise between generalization and specialization.

**Router training data.** Training the router on synthetic instructions yields excellent held-out accuracy but does not fully capture the diversity of real user queries. Training on raw textbook spans lowers macro-F1 slightly but produces more intuitive routing in practice. This highlights a common pattern: router performance on synthetic benchmarks is not always predictive of MoE behaviour in the wild.

**Latency bottlenecks.** Across all techniques, queue time is the dominant contributor to tail latency, with inference as the second-largest factor.

Baseline HF deployment suffers from catastrophic load times ( $p_{95} \approx 119$  s), which amplify queueing and lead to  $p_{95}$  total latency near 200 s. The Dual-LM variant removes load time but still collapses under load because queue time becomes extreme ( $p_{95} \approx 194$  s). GGUF, by contrast, reduces both load and inference times and keeps queueing under control, achieving  $p_{95}$  total latency of only 57,959 ms, a  $3\text{--}4\times$  improvement over the alternatives.

**Deployment choices.** LoRA experts are convenient during training and experimentation, but HF CPU inference is slow and tuning eviction/loading logic is painful. Merging and quantizing experts to GGUF allows us to leverage `llama.cpp`’s optimized CPU kernels, yielding both lower average latency and vastly better tail behaviour under realistic Azure-style traces. In short, **quantization and runtime choice are as important as model architecture** for practical MoE systems.

## Limitations

This work has several limitations:

- **Synthetic data dependence.** All training data for experts is generated by Gemini from textbooks, which may limit diversity and propagate biases from both sources.
- **No real user evaluation.** We do not conduct studies with actual users or clinicians; all qualitative assessments are based on the authors’ judgments.
- **Safety and scope.** The system is not intended for deployment as a real mental-health assistant. It does not implement safety filters, crisis detection, or referral mechanisms.
- **Limited model size and techniques.** We restrict ourselves to 1B-parameter experts and simple linear routing. Larger experts or more sophisticated neural routers (or multi-expert selection) might yield better performance but were out of scope for this assignment.

### 9.1 Failure Cases and Limitations

While the system behaves plausibly on many prompts, qualitative inspection also reveals important failure modes.

**Misrouting delusional content to anxiety.** Some prompts are linguistically mild but clinically closer to psychosis. For example:



694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
  
719  
720  
721  
  
722  
723  
  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
  
737  
738  
739  
740  
741  
742

**User:** “I get the sense that messages on TV or online are secretly directed at me.”

**Routed expert (anxiety):** “The idea that messages on TV or online are aimed directly at you can feel unsettling. In most cases, the information we encounter is not personal but created for a broad audience to inform or entertain. Sometimes this feeling comes from misinterpreting the context or intention behind what you see. It can help to pause, consider the bigger picture, and check whether there is any concrete evidence that these messages are specifically about you before drawing conclusions.”

Clinically, this kind of “messages are secretly directed at me” content is much closer to delusions of reference and would typically raise concern about psychotic processes. The router, however, assigns it to the anxiety expert, which treats it as a media-literacy or misinterpretation issue and does not surface the potential for psychosis or recommend urgent evaluation. This highlights a safety-relevant limitation of a purely text-based, TF-IDF router without explicit risk-awareness.

**Anhedonia routed to OCD expert.** We also observe confusion between affective symptoms and OCD. For instance:

**User:** “Lately nothing feels enjoyable, even the things I used to love.”

**Routed expert (OCD):** “A loss of pleasure in activities that used to feel rewarding is often associated with depression. People sometimes describe it as a sense of detachment or not getting anything out of things they once enjoyed. If this persists or starts to interfere with daily life, it can be important to talk with a mental health professional. They can help you explore possible causes, including depression, and work with you on steps to gradually re-engage with activities and rebuild a sense of enjoyment.”

Here the router sends a classic anhedonia prompt to the OCD expert. The expert’s answer itself frames the issue in terms of depression and recommends appropriate next steps, but the misrouting indicates that the router struggles to reliably distinguish between depressive and OCD-like language

when overlap in vocabulary is high. This aligns with our quantitative confusion patterns and underscores that routing errors can occur even when individual experts behave reasonably.

## 10 Conclusion

We presented an end-to-end MoE-style system for mental-health dialogue on resource-constrained hardware. By combining synthetic QA generation from textbooks, LoRA-based domain experts, a TF-IDF + Logistic Regression router trained on book spans, and a quantized GGUF deployment via llama.cpp, we obtain:

- strong routing performance (macro-F1 0.83 on 3.99M test spans, with clinically meaningful confusions),
- domain-specialized experts that produce more clinically grounded responses than the base model, and
- a GGUF-based server whose p95 latency is 3–4× lower than Baseline HF and Dual-LM variants under Azure-style traces.

Although designed as a course project, the architecture and experimental setup highlight several generally useful patterns: using external LLMs for targeted data creation, preferring simple but robust routers, treating quantization and runtime as first-class design decisions, and evaluating MoE systems under realistic load patterns rather than only static benchmarks. Future work could explore more principled preference optimization for expert responses, richer safety mechanisms, and learned neural gating networks, while preserving the lightweight, latency-aware deployment philosophy demonstrated here.

## References

743  
744  
745  
746  
  
747  
  
748  
749  
750  
751  
752  
753  
754  
  
755  
756  
757  
  
758  
759  
760  
  
761  
762  
763  
  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
  
777

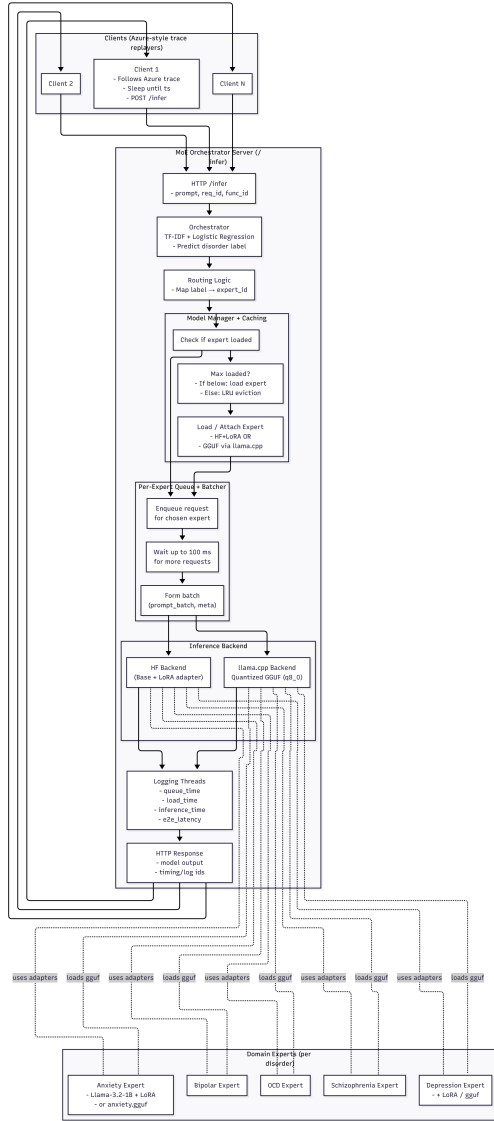


Figure 1: Overall architecture of the MoE system. A TF-IDF + Logistic Regression orchestrator routes each query to one domain expert. Experts are LoRA- or GGUF-based variants of Llama-3.2-1B-Instruct. A custom server manages model loading, batching, and logging under Azure-style traces.

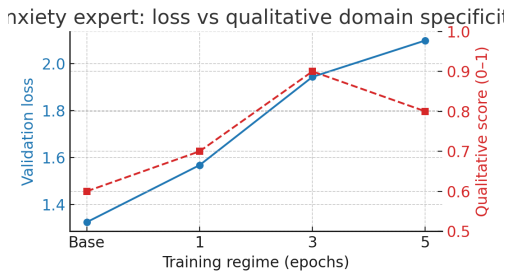


Figure 2: Illustrative trade-off between epochs and behaviour of a single expert (anxiety). While cross-entropy is best for the base model, 3 epochs produce the most clinically grounded yet conversational responses.

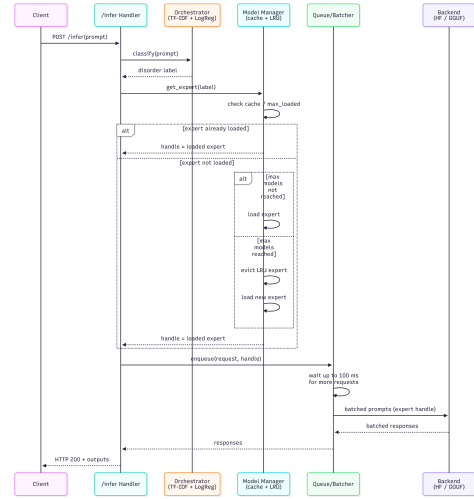


Figure 3: server timeline / state diagram, showing routing, model loading/eviction, batching and inference.

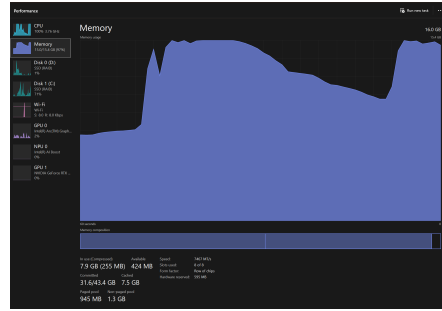


Figure 4: Base Server: Memory usage lowers after the model is loaded, but still we can only load 2 models.

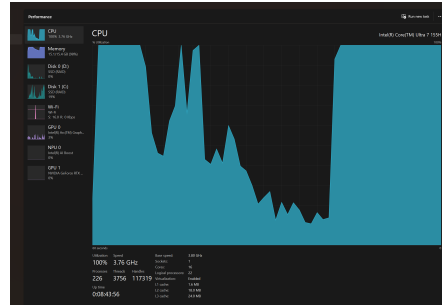


Figure 5: CPU Usage reaches 100% when requests come in bursts

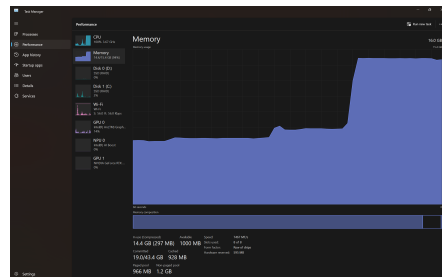


Figure 6: Memory Usage in Llama-CPP and Quantized GGUF models where all the 5 models were loaded at once

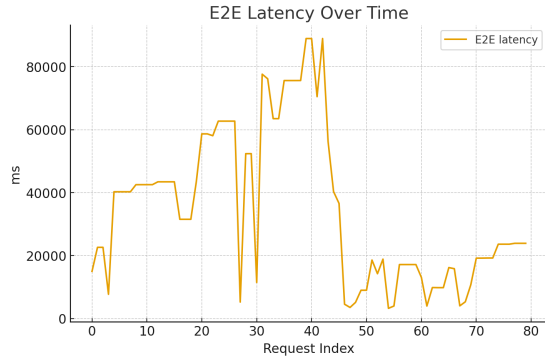


Figure 7: End-to-end latency over time under Azure-style traces for three server variants. GGUF maintains the tightest band; Baseline and Dual-LM exhibit large spikes.

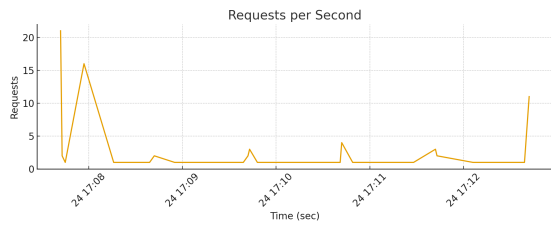


Figure 8: Requests per second (RPS) over time. Burst regions correlate with queue-time spikes in Baseline and Dual-LM, whereas GGUF absorbs load more gracefully.

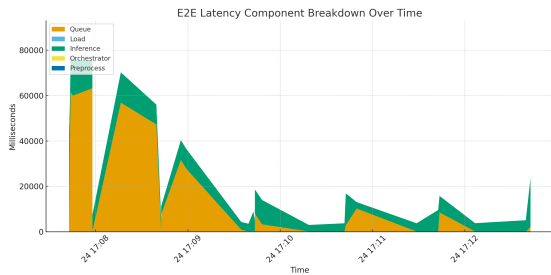


Figure 9: Stacked area breakdown of server time into queue, load, and inference. Queue time drives most latency spikes; load time is substantial only in the Baseline HF server.

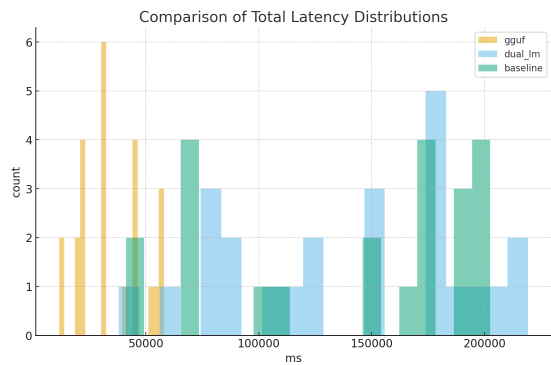


Figure 10: Distribution of end-to-end latency across all requests. GGUF has the tightest distribution; Baseline and especially Dual-LM show heavy long tails.

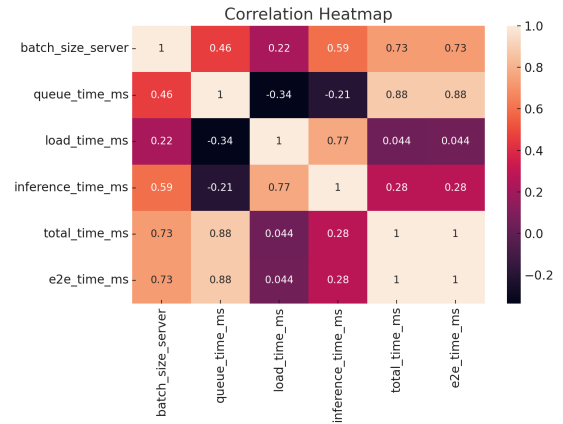


Figure 11: Correlation between latency components and total end-to-end latency. Queue time is the strongest predictor, followed by inference time.