# **Gemma-Kavach**

## **Day 1: Setup up backend server**

I built a **production-ready backend server** using **FastAPI** to serve the **multimodal Gemma 3n model** from Google. This server supports:

* 🔤 **Text-based queries** via /generate
* 🖼️ **Image + prompt reasoning** via /ask\_image
* 🎤 **Voice/audio transcription + reasoning** via /ask

The server enables secure, real-time, and **offline-first multimodal inference**, designed to support use cases like **crisis response** and **privacy-preserving assistants**.

**🧠 Key Learnings**

1. **Stick to Official Google Docs**  
   Trying to outsmart the official guidance led to unnecessary errors. Following Google’s documentation for Gemma 3n (especially around loading AutoProcessor and AutoModelForImageTextToText) ensured compatibility and feature completeness.
2. **Correct Library Installation is Crucial**
   * Mismatched or outdated transformers, accelerate, or torch versions caused slowdowns and even runtime errors.
   * Up-to-date, version-pinned installations prevent debugging nightmares.
3. **Model Parameters Directly Impact Speed**
   * Failing to disable torch.compile, or setting the wrong attn\_implementation, can **drastically slow down token generation**.
   * Using torch\_dtype=torch.bfloat16, disable\_compile=True, and attn\_implementation="eager" gives you a huge speed boost for inference.
4. **Every Developer Should Know How to Serve Models Locally**  
   Hosting your own inference backend is critical for:
   * 💼 **Enterprise deployments** with data security needs
   * 🔐 **Offline/private use** (like in crisis zones or field missions)
   * 🛠️ Learning how open-source LLMs actually integrate with production apps

## **Day 1: Feature Planning: Gemma Kavach Vision**

Simple Objective: Let people upload a video and get analysis and otherwise I real time stream.

**This is an AI-powered real-time crowd safety monitoring system designed to prevent stampede disasters at large events.** The system continuously captures video frames from a webcam, analyzes each frame using computer vision AI to detect signs of crowd panic (pushing, falling, overcrowding), and immediately alerts security personnel through multiple channels when dangerous situations are identified. It operates like an automated safety watchdog that never gets tired or distracted - constantly scanning crowds and providing instant warnings with sound alerts, email notifications, and visual indicators on the live feed. The system is particularly valuable for festivals, religious gatherings, concerts, and any large public events where crowd management is critical, as it can detect early warning signs of stampedes before they become fatal incidents and provide documented evidence with timestamped alerts, risk scores, and visual summaries for emergency response teams.

**Key Value:** Transforms any standard camera into an intelligent crowd safety monitor that can potentially save lives by providing early detection and rapid alerting of dangerous crowd conditions, making it an essential tool for event organizers and security teams managing large gatherings.

## **Day 2 Feature Planning and implementation: Gemma Kavach Vision**

So we are fine for demo of feature 1 i.e Gemma Kavach Vision but it is like a script in my laptop so no one apart from me can use it we need to migrate this to a backend server and then make a ui so that any one can use it like that is very important from myend.

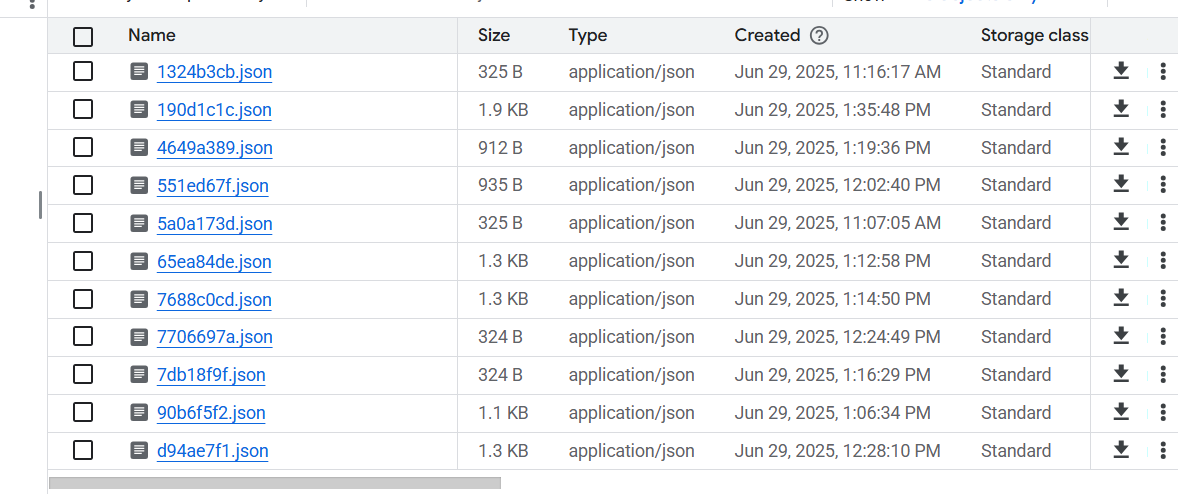
**Planning Backend for this feature.**

Gemman Kavach Vision will be a seprate backend, running so some port, the ui will seprate so we do design a full backend gemma kavach vision.

Also we need a object storage location for now using gemma3n-raw a goole coud storage engine

**1.@router.post("/session/create", response\_model=SessionResponse)**

Create Session on every new run of the app we will create session



File storage is not the best place to store this type of information but fine fow done.



2. **@router.post("/session/{session\_id}/frame", response\_model=FrameAnalysisResponse)**

The frontend will keep sending images in here

1.Get the session context

2.Make a call to gemma to see if the frame is risky or not.

3.Save the flagged image to google storage

4.Append session data for that frame

5.Send email via background tasks if we are good i.e

# Alert thresholds

MIN\_FRAMES\_FOR\_ALERT = 3

RISK\_THRESHOLD\_FOR\_ALERT = 75.0

**3. @router.get("/session/{session\_id}", response\_model=SessionStatusResponse)**

Get simple session status works fine

## **Day 3: Gemma Kavach Vision key thoughts**

Ideally would have loved the system to run completely offline but do not have compute, so in report we need to mention this

**While the demo runs on RunPod to simulate GPU-enabled edge deployment, Gemma Kavach is fully designed to run offline on devices like Jetson, Ollama, or any CUDA-compatible laptop. The model server is containerized and portable — no internet required**

**Mostly edge deployment is stimulated with runpod**

**The only thing preventing us from running it offline right now is hardware access — not software design.**

At a base level this feature looks okay

Now we need sort to think to improve it

Can we break this into parts and be like ok two passes?

One crowd density

One Flow of people  
and if both yes we sort flag?

**Day 4 : Can Gemma Kavach be improved for a accuracy standpoint?**

->From a accuracy standpoint and better explainabilty standpoint I will be using **Crowd Density and Crowd Motion, this will us improve a lot with results accuracy.**

So a major portion of time went is debug why css /js changes weren’t reflected in the app so not sure but yes with runpod it an issue I guess something to do the way proxy server handles static files we can skip we need to wrap our system in one command or move to docker since this would be really helpful .

Now we need to sit and think more about what needs to be done from a Product and feature standpoint.

(The ui looks okay but it too flashy need to get the design right)

## **Day 5 :What more can be done in Gemma Vision Kavach**

**Thinking can something more be done?**

So based on juding criterion this seems fine

* **Impact & Vision (40 points)**: As demonstrated in your video, how clearly and compellingly does your project address a significant real-world problem? Is the vision inspiring and does the solution have a tangible potential for positive change?
* **Video Pitch & Storytelling (30 points)**: How exciting, engaging, and well-produced is the video? Does it tell a powerful story that captures the viewer's imagination? Does it clearly and effectively demonstrate the product in action, showcasing a great user experience? Does it have viral potential?
* **Technical Depth & Execution (30 points)**: As verified by the code repository and writeup, how innovative is the use of Gemma 3n's unique features (on-device performance, multimodality, mix'n'match, etc.)? Is the technology real, functional, well-engineered, and not just faked for the demo?

We need work one and two but yes that will come later

**Next Feature Planning?**

**So sometimes the backend server faces this issue**

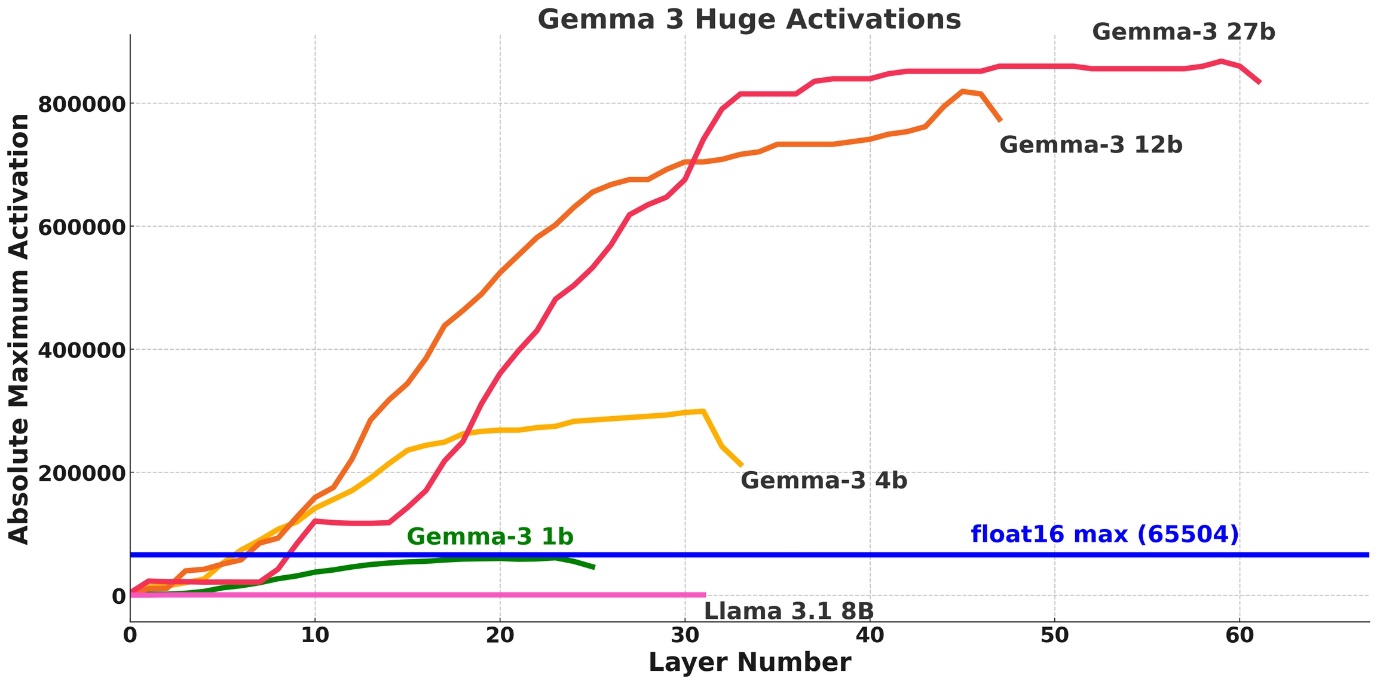
**/pytorch/aten/src/ATen/native/cuda/TensorCompare.cu:112: \_assert\_async\_cuda\_kernel: block: [0,0,0], thread: [0,0,0] Assertion `probability tensor contains either `inf`, `nan` or element < 0` failed.**

**♾️Infinities and NaN gradients and activations**

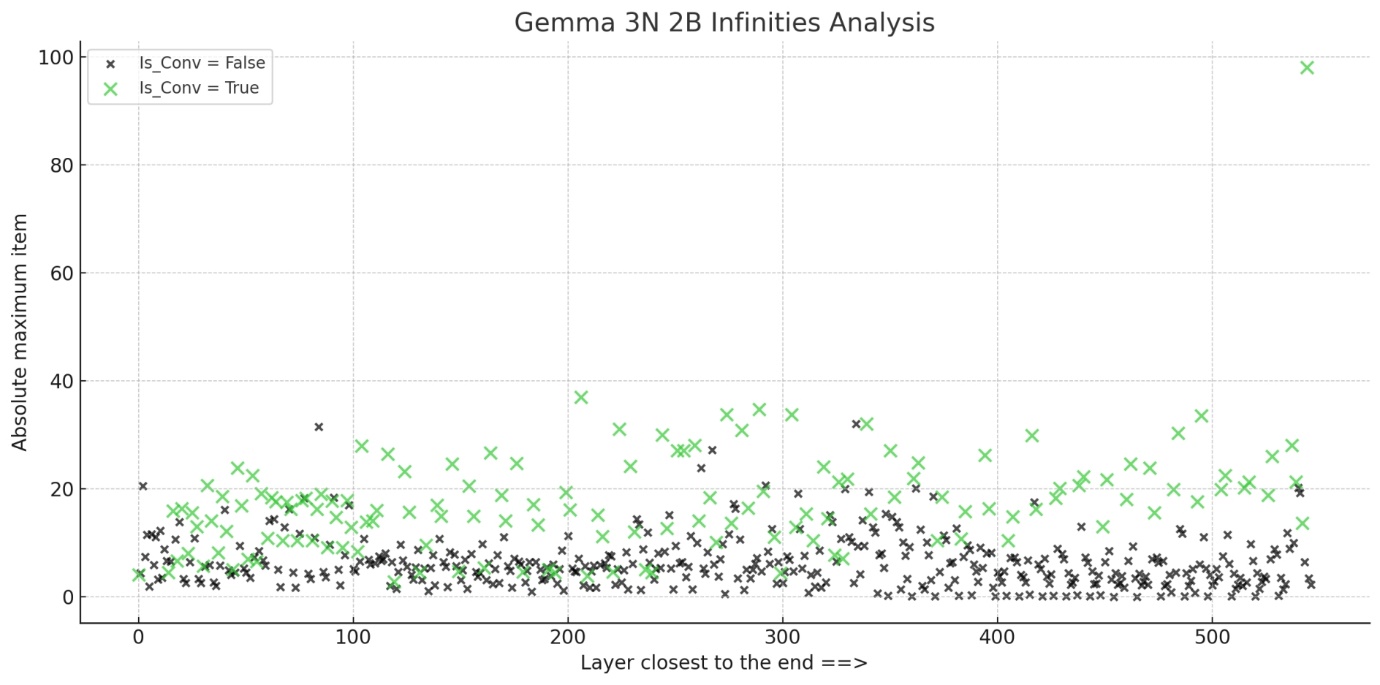
Gemma 3n, like Gemma 3, has issues running on FP16 GPUs (e.g., Tesla T4s in Colab).

We [discussed it here](https://docs.unsloth.ai/basics/gemma-3-how-to-run-and-fine-tune). For Gemma 3, we found that activations exceed float16's maximum range of **65504.**

**Gemma 3N removed the activation issue, but instead we still encountered infinities!**



We instead plotted the absolute maximum weight entries for Gemma 3N, and we see the below:



We note that the green crosses are convolutional weights. You can see the magnitude is much larger than other weights. And if we inspect the activations, they go to infinity!

Below is a table for Conv2D weights which have large magnitudes. Essentially during a Conv2D operation, large weights multiply and sum together, and unluckily exceed float16's maximum range of **65504.** Bfloat16 is fine, since it's maximum range is 10^38.

Name

So it’s best we migrate our server to unsloth only reasons-

**1.Will fix the nan and infi error**

**2.Will be little optimized**

**Note sine we r using rtx 4090 we need to can use**

**RTX 4090 VRAM Analysis**

**RTX 4090 specs:**

* **24GB VRAM - plenty of headroom**
* **Native bfloat16 support - optimal precision**
* **High memory bandwidth - can handle full precision efficiently**

**Memory Usage with load\_in\_4bit=False:**

**Gemma 3N 4B model in bfloat16:**

* **Model weights: ~8GB (4B parameters × 2 bytes)**
* **Activations + KV cache: ~3-6GB (depends on sequence length)**
* **Total estimated: ~12-14GB peak usage**
* **Your available: 24GB**
* **Headroom: ~10GB free (plenty of safety margin)**

**4. load\_in\_4bit=True**

* **What**: Quantizes model weights from 16-bit to 4-bit
* **Memory savings**: ~75% reduction (4B model: ~16GB → ~4GB VRAM)
* **Accuracy**: Minimal loss with Unsloth's optimized quantization
* **Your benefit**: RTX 4090 has 24GB, so this gives you plenty of headroom
* **Alternative**: load\_in\_4bit=False for maximum accuracy but higher VRAM

A lot of issues in migration

1.Environement setup

2.4 bit model not working well

**For Multimodal Model Deployment:**

* **Framework abstractions can break modality-specific processing**
* **Always validate end-to-end multimodal pipelines**, not just text components
* **Token alignment is critical** for proper attention computation in multimodal transformers

This turned to big task than thought of .

Some times for me I should first be sure at function level in a jupyter notebook than go for a api server (ai models can do it) **But I should test things in components so any component is breaking I can be sure of rather not burn so much tokens , money , energy and bigest thing (TIME)**

## **Day 6 : Ground work on Gemma Vision Commander**

Comments –

1.The native voice capibilty of the system is not good

model, tokenizer = FastModel.from\_pretrained(

model\_name="unsloth/gemma-3n-E4B-it",

dtype=None, # Auto detection (tutorial setting)

max\_seq\_length=1024,

load\_in\_4bit=True, # Tutorial setting (works with multimodal)

full\_finetuning=False,

trust\_remote\_code=True,

)

It can’t understand my Hindi nor English in a place where there is so much voice is likely to be distored we would need a strong model

?

Let’s try to make this parameter load\_in\_4bit=True as False and see how our server breaks or makes

If we increase the precision –

1.Will the code work as intended?

2.Will we see any improvement in accuracy?

Results –

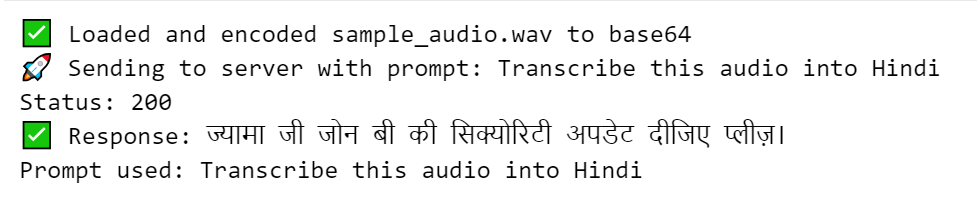
1.Works

2.Doesn’t work something seems off , I guess the model ability to nativly understand audio is poor (for Indian acent and the way we speak)

So the model doesn’t work great for speech reconginition

**According to google offcial docs**

**Tip: For best performance with AST tasks, provide a separate instruction to transcribe the audio into the original language, with a follow-on instruction to translate the text into a target language.**



**Something strange is happening**

**Offical docs**

**according to offcial docs When encoding audio data with your own code implementation for use with Gemma 3n, you should follow the recommended conversion process. If you are working with audio files encoded in a specific format, such as MP3 or WAV encoded data, you must first decode these to samples using a library such as ffmpeg. Once the data is decoded, convert the audio into mono-channel, 16 kHz float32 waveforms in the range [-1, 1]. For example, if you are working with stereo signed 16-bit PCM integer WAV files at 44.1 kHz, follow these steps: Resample the audio data to 16 kHz Downmix from stereo to mono by averaging the 2 channels Convert from int16 to float32, and divide by 32768.0 to scale to the range [-1, 1] Note: When resampling audio to 16 kHz, you should use a Fourier method for best results, such as scipy.signal.resample or librosa.sample(res\_type ='scipy').**

**So we done the following transformations in audio input**

 **Resampled to exactly 16kHz** - Gemma is trained on this sample rate

 **Converted stereo → mono** - Reduced noise and confusion

** Normalized volume - Prevented quiet audio issues**

 **Removed DC offset** - Eliminated static/hum

 **Used Fourier resampling** - High quality conversion

**So after preprocessing audio as par offical docs we have a improvement on a scale of 10 records**

**1.No processing gets – 1 correct**

**2.Ater processing gets – 5 correct**

**Expirement 2:So improved the audio recording used a mike getting shocking results**

**OLD Method (No Preprocessing):**

* **8/10 correct transcriptions - Got "Zone B ki security update dijiye" right!**
* **7 unique responses = Better consistency**
* **Only 2 failures (#5, #9)**

**NEW Method (With Preprocessing):**

* **5/10 correct transcriptions - Actually WORSE than old method!**
* **8 unique responses = More inconsistent**
* **5 failures with error messages**

**Shocking Discovery! 😱**

**With good quality audio, the OLD method actually performed BETTER**

**You need adaptive processing - detect audio quality first, then decide:**

* **Poor quality audio → Enable preprocessing**
* **Good quality audio → Skip preprocessing**

**Bad audio improves because you're fixing issues. Good audio degrades because you're over-processing it.**

**So the results are extremely wired which means it the model is still not able to understand stuff,niether adaptive processing is helping too much**

**So two options:**

**1.One is let us try to give it better quality audio for test and demo and be like using llm as a judge so If the output doesn’t make sense it reasks the user to send it**

**2.Fine tune gemma model audio**

**SO unsloth says**

**“You can finetune the vision and text parts for now through selection - the audio part can also be finetuned - we're working to make it selectable as well!”**

**So its ideal we wait for unsloth since unsloth is the only infrence provided for making sure**

**Unsloth is the only framework which works in float16 machines for Gemma 3n inference and training our engine is bulit on it so it’s best we wait and try to use llm as a judge**

**SO high level design**

**1.Voice ->Transcribe if Zone details available then we r good for downstream pipeline , rest we will ask the user the audio was not clear pls re upload it**

**Just thinking load voice can be a modality to do stuff , we should explore on how can be fine tune voice models and make them on consumer grade hardware like (rtx 4090) and stuff should be fun and a not of exploration**

**Day 7 Working and setup a retry pipeline (rough implementation of llm as a judge)**

So for voice still the audio quality is not high,our infrence provided “Unsloth doesn’t yet allow us to fine tune the audio layer”.

So yes we can try to fine tune it using hugging face but the model somes suffer’s from NAN and infi issue with fp16

So voice

Extarct zone from zone query that zone get the results and show the user those results , that ;s flow we can do from my end

Out of cursoity is there an opportunity for a light weight audio to text hindi model

**So yes the flow is user -> input->transcribe->extarct zone->search that zone ->pass it to an llm translation (response)**

**So yes our audio is feature is also running)**

**Day 8 Audio Feature properly**

So we need voice as output , gemma3n doesn’t allow us it gives us only text so we can I guess google’s api for this.

So google text to speech api is doing a good job

New flow

**Query->Transcribe->extract zone ->Make a database query (info about the zone)->results pass to an llm->message->text to speech api and get results**

**Issues in deployment**

**1.Voice mode is not working on phones**

Also despiting delecting the folder from terminal run pod still doesn’t update my folder so I have to create a new one this is super tough.

1.We need a little different type of design for iphone in general due to permission issue , but yes in laptop at least the flow works well.

Also audio to text is coming out as average not bad at all.

SO working now issue fixed-

1.Frontend voice handeling for IOS

**So a nice citizen app will be also super useful**

| **Situation** | **What User May Do** | **What AI Should Trigger** |
| --- | --- | --- |
| 🧍‍♂️ **Person injured/fainted** | Send photo + message “Need help” | Trigger **medical alert** to doctor booth, show route to ambulance |
| 🚸 **Child lost** | Text: “A small boy is missing in Zone D” + photo | Trigger **public announcement** or staff ping |
| 🔥 **Small fire** | Photo + “Smoke near food stalls” | Notify **fire safety team** if severity is high |
| 🧠 **Crowd panic** | “People are pushing, very congested” | System checks recent Gemma analysis → triggers alert if crowd is “High + Chaotic” |
| 🧳 **Lost item** | Message: “I lost my bag” | Log it into a lost & found dashboard (low-priority route) |
| 🚧 **Path blocked** | “Exit near Zone C blocked” + image | Security gets route-adjust suggestion from map overlay |
| 🗣️ **Need interpreter** | “Tourist can’t find their group” | Send to nearby volunteer or support station |

We need to plan the third feature well.Becuase it is important

Important links which might help

<https://github.com/google-gemini/gemma-cookbook/blob/main/Gemma/%5BGemma_3n%5DMatFormer_Lab.ipynb>

<https://docs.unsloth.ai/basics/gemma-3n-how-to-run-and-fine-tune>

<https://developers.googleblog.com/en/introducing-gemma-3n-developer-guide/>

<https://github.com/huggingface/huggingface-gemma-recipes/tree/main/notebooks>

So will be fine tuning a model possibly “unsloth/gemma-3n-E2B-it-unsloth-bnb-4bit”

For our classification use this and use that in our third app, this will open more doors and we can try multiomodal fine tunning as well if it supported in unsloth

The thrid app is consumer grade we should be good .(we will start with text only to see how nice it is then take a call)

So generaeted datasets manually could have an api but I wanted system to understand hinglish

Hugging face is slighty more complicated yet stable than unsloth but yes unsloth is great

Open source models are a great opporutnity

So trainning run is ready lets see how much can juice out in 2 hours

**Day 9 Fine tunning run**

So we will be fine tunning the model with unsloth so that it can act better on hinglish classification task will be fun

Also we just using a 2Billion Para (not literally) but I terms of vram consumption makes it smaller faster and extermely cheaper.

The idea for using SLM on ur task is fun ,fast and extremely cost effective.

Notes for understanding the fine tunning job better-

1. **Triton** is a low-level **GPU kernel programming language** developed by **OpenAI**.  
It allows developers (and libraries like Unsloth) to write **custom GPU-accelerated operations** that are often **faster** than PyTorch's default ones — especially for:

* Matrix multiplications
* Attention mechanisms
* Transformer layers

2. TORCH\_USE\_CUDA\_DSA = 1  
→ **Enables Dynamic Shared Allocation (helps catch unaligned memory issues or race conditions early — useful for fine-tuning stability)**

Many LLM libraries (like **Unsloth**, **FlashAttention**, **xFormers**) use Triton to:

* Speed up **forward/backward passes**
* Reduce **VRAM usage**
* Avoid bottlenecks of generic PyTorch CUDA kernels

3. datasets.Dataset — from 🤗 Hugging Face, wraps your data into a format SFTTrainer can use

|  |
| --- |
| **max\_seq\_length=1024** |

|  |
| --- |
| **Sets the maximum length of input the model can handle (standard for 2B models)** |
| **(for light weight tasks)**   |  |  | | --- | --- | | load\_in\_4bit=True | Loads the model in **QLoRA 4-bit mode** = massively reduced VRAM, same quality | |

**✅ Step 5: Apply Chat Template for Gemma**

python

CopyEdit

from unsloth.chat\_templates import get\_chat\_template

tokenizer = get\_chat\_template(tokenizer, chat\_template="gemma-3")

While finetunning the connection got stopped automatically hmm so clearly run pod has issue which lets a azure or gcp won’t have but the cost benefit is also there

So at 400 steps loss was 0 I didn’t apply checkpointing so model needs to be retrained I guess only 600 samples are a issue we need more data I guess , but let’s do a 400 step run and then we can improve on data

I think we have a data problem in here let the trainning run be completed

**Again connection error comes in unsloth good this we had checkpointing enabled so we were good, we can restart fine tunning from there**

**So clearly a 2B model is not working well ?**

Will going for a proper fine tuning run

**Trainning data –**

**1.We will use gemini to genrated synthetic trainning data with prompting.**

**(gemini-2.5-flash-lite-preview-06-17)**

**100 api calls in 100 ->10k**

**So I guess 600 api calls we would need**

The next run will be bigger and(bigger not only in terms of steps ) in terms of size as well .

So we started by 600 samples for a 2 billion model it makes sense , but I thought may we can try but now we will be using 60k samples(10k per classes)

**So 600-60000 a 100x jump in a date and we trained for 400 steps we will 11,250 total steps (28x more training!) jump**

As they say train for longer duration and with big /massive data magic happens lets’s see(not algo is important he he he)

Note please use this runpod otherwise we will always have connection error

Once you've launched your training with:

**nohup python job.py >> train.log 2>&1 &**

You are **100% safe to:**

* ❌ Close the terminal
* ❌ Refresh your browser
* ❌ Shut your laptop lid
* ❌ Disconnect your internet

Your training will **keep running in the background** on RunPod.

**🔄 Later, you can reconnect and check:**

**🧾 To confirm it's still running:**

ps aux | grep job.py

**📄 To see your training output:**

tail -f train.log

**🔚 To stop it if needed:**

kill <PID>

**🛠 Pro Tip:**

Put this in a run\_train.sh if you want to reuse it easily:

#!/bin/bash

LOGFILE="logs/train\_$(date +%Y-%m-%d\_%H-%M).log"

nohup python job.py >> "$LOGFILE" 2>&1 &

echo "🚀 Training started. Logs in $LOGFILE"

Let me know if you want me to generate that file for you.

Big lessons on training-

1.Use the above commands else no run will be complete

2.We need proper logging mechanism to track progress

# **Day 10: Improving the fine Tunning Run**

nohup python job.py > train.log 2>&1 &

# Monitor in real-time:

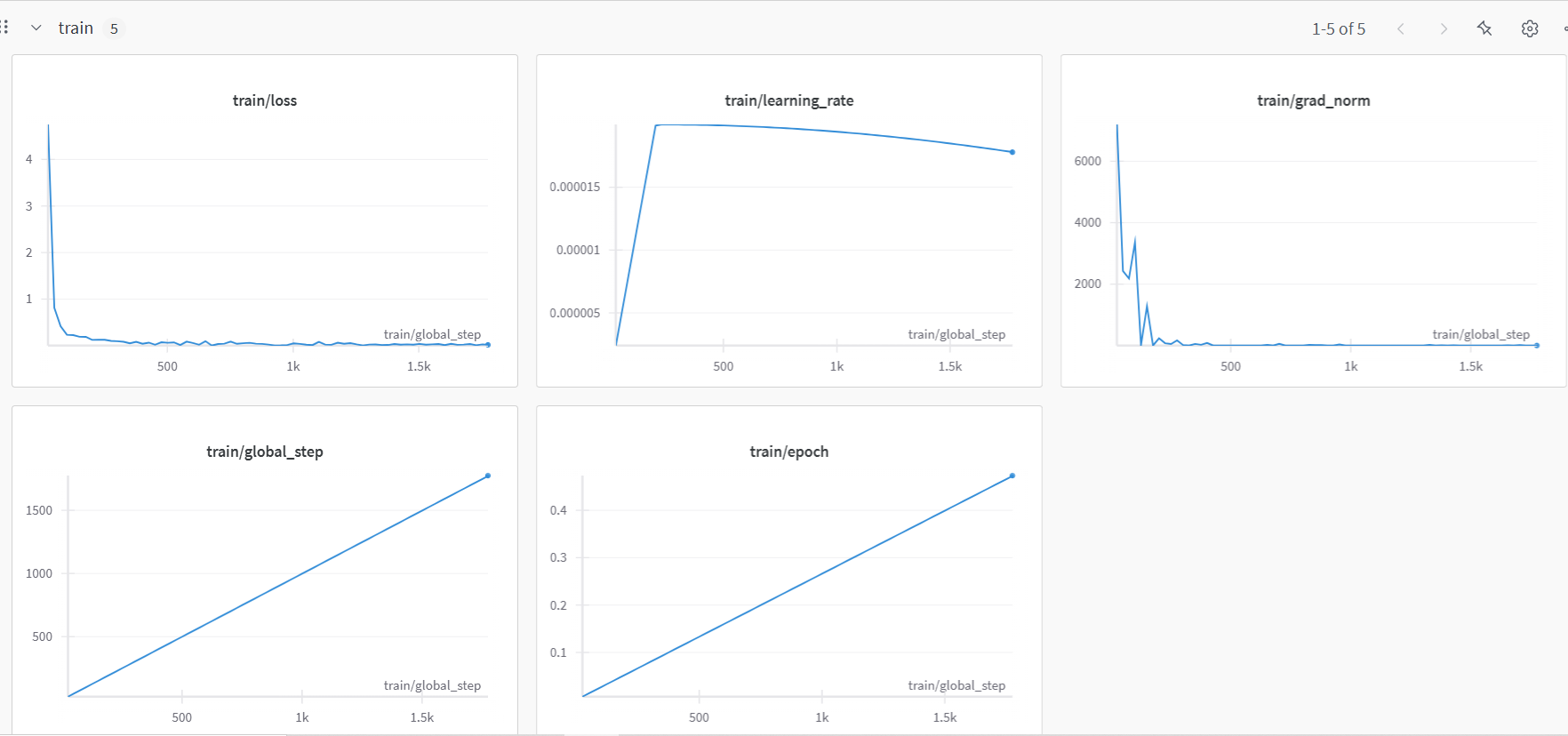
tail -f train.log

tail -f training\_progress.log

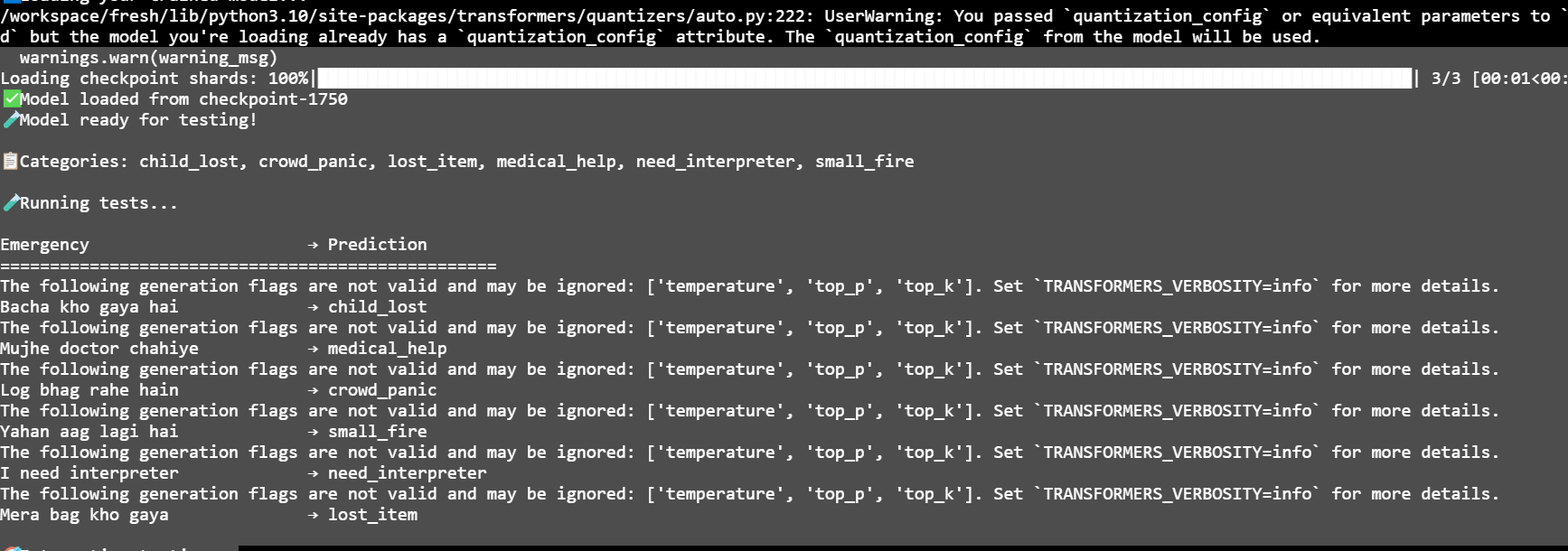
# Check wandb dashboard: <https://wandb.ai>

So the main improvements have nothing much do with neural network(like we changed hyperparams),but now we can setup the run in runpod without any major issues.

Results from W and B run



Also the jupyter lab of the pod is corrupted so we need a new pod



Enter emergency (or 'quit'): Bhaut dekha muke ho rahe hai yahan

The following generation flags are not valid and may be ignored: ['temperature', 'top\_p', 'top\_k']. Set `TRANSFORMERS\_VERBOSITY=info` for more details.

→ crowd\_panic

Just wrapped up a successful fine-tuning run on a 2B parameter small language model. Thought I’d share some hard-earned lessons from the process — things I wish I had known earlier.  
  
1.Started small — just 600 rows.  
 The results weren’t great, but it helped test the pipeline and confirm that the model could begin learning. Even with poor accuracy, I could see signs of signal being picked up, which was encouraging.  
  
2.Scaled up to 60,000 samples.  
 A 100x jump. This changed everything. If you’ve read the Chinchilla paper, you know that for compute-optimal training, data volume matters as much (if not more) than model size. I learned that the hard way — data quality and quantity are non-negotiable.  
  
[**3.Training**](http://3.training/) steps went from 400 to around 1750.  
 Initially I was stopping early, assuming the model was "good enough." Pushing further helped reduce loss significantly, but I also saw diminishing returns beyond a point. Knowing when to stop matters — training endlessly doesn’t always mean better results.  
  
4.Set up Weights & Biases early.  
 Having live dashboards to track loss, learning rate, gradient norms, and step progression made debugging and evaluation 10x easier. It also helped me spot instabilities before they became real issues.  
  
5.Ran the training in the background on Linux.  
 Using nohup python [**job.py**](http://job.py/) > train.log 2>&1 & ensured my process didn’t get killed during minor glitches or session drops. If you’re not using background processes, you’ll likely waste time restarting runs.  
SLMs are incredibly practical.  
  
This wasn’t a 65B foundation model run — it was a fine-tuned small model that now performs well on a real-world emergency classification task. The entire thing ran on an RTX 4090 using RunPod, with help from the Unsloth library. Fast, affordable, and efficient.  
  
All of this is leading up to a product demo that I’ll be sharing soon. If you’re working with small models, want to get more from your data, or just trying to build something meaningful — I hope this helps.  
  
Always happy to talk about fine-tuning, model scaling, and lessons from the trenches.

SO the model is sort of working which is good

Now the next step will be focussing on setting up server for this model to support chat and for image we wise use our old model, so that means we will sort of have 2 servers, yes but that is needed.

Yes for new all actions we will mail only to this

Simple user app will be-  
User enters text ->our fine tunned model

And optinally uploads a image->our normal model

And location

We sort of route to the issue

And then trigger an email

Task was an api server took time-

1.Memory issues

2.Tokenizer issue

But our fine tuned model is now as a endpoint aslo making a 2billon parameter models is super nice