Coding Nexus · Follow publication

✦ Member-only story

How to Fine-Tune Gemma 3 270M and Run It On-Device

5 min read · Oct 15, 2025

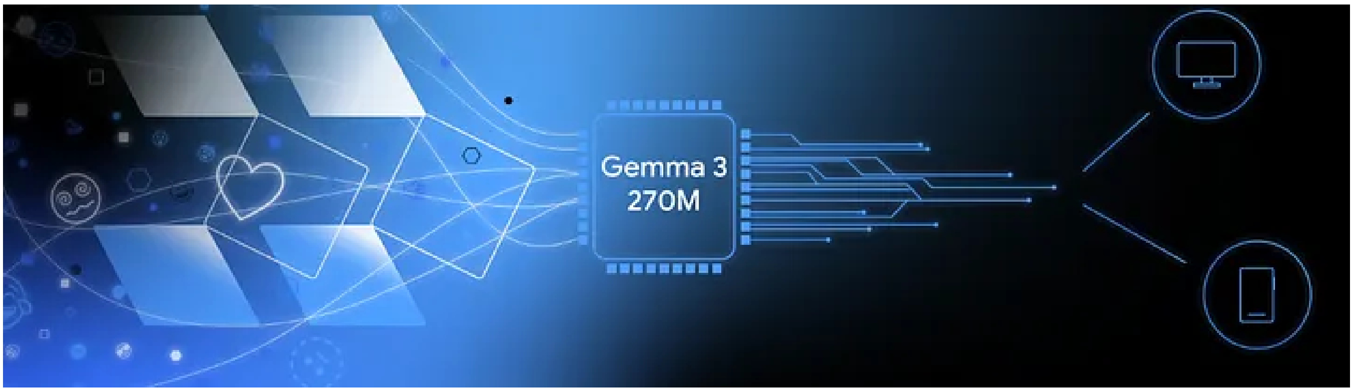
Civil Learning Following

If you've ever wished to run your own small AI model on your laptop or phone — without setting up servers or paying for GPUs — I have some good news.

Meet **Gemma 3 270M** — a small but surprisingly capable open model from Google. It's part of the Gemma family, which essentially brings the same technology used in Gemini models into a lightweight, customizable form.

And here's the fun part: you can **fine-tune it in less than an hour**, reduce its size to under **300MB**, and run it directly in your browser.

In this post, I'll show you how I created my own **emoji translator** with Gemma — a small model that converts text into emojis and runs locally.



. . .

Step 1: Teaching Gemma to “Think in Emojis”

Out of the box, Gemma is a generalist. Ask it to translate text into emojis, and it’ll be a bit too polite about it.

Prompt:

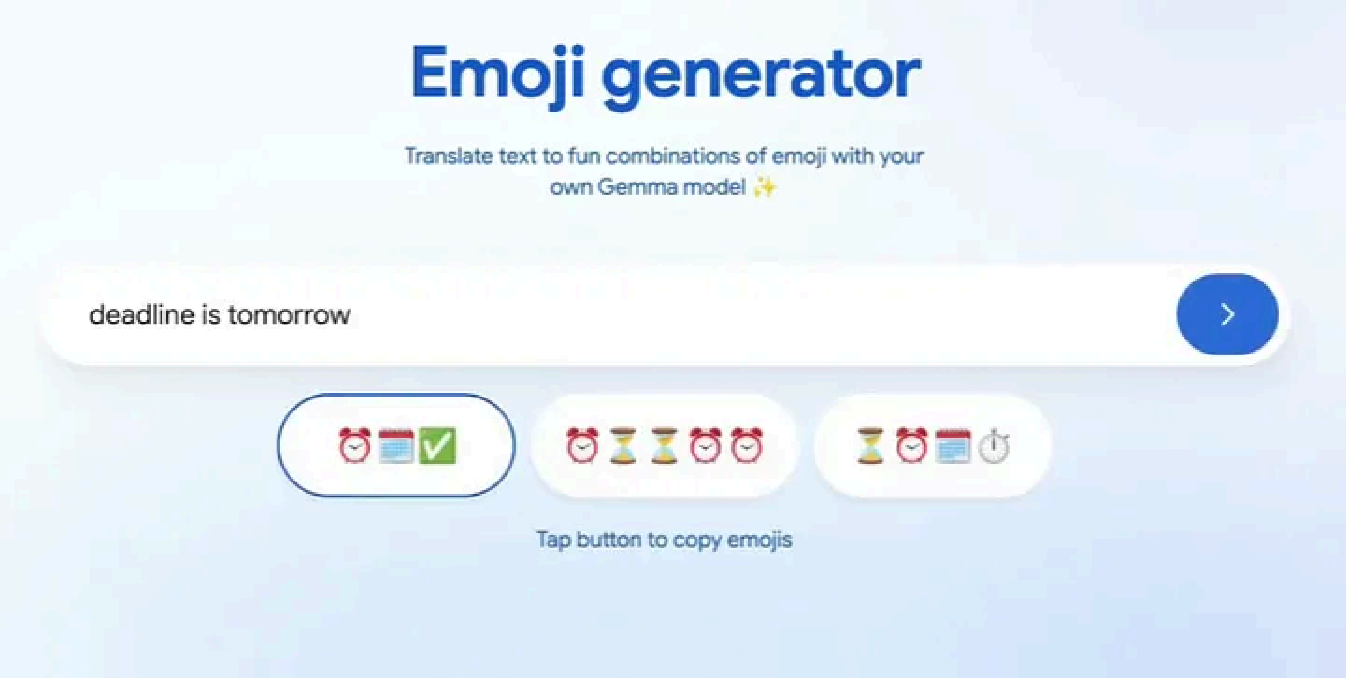
> “Translate this text into emojis: what a fun party”

Model:

> “Sure! Here is your emoji: 🥳🎉🎈 ”

Close, but not exactly what I was aiming for. For my app, I wanted **only emojis** — no words, no “sure!”, just the fun stuff.

I decided to fine-tune it.



Emoji generator

Translate text to fun combinations of emoji with your own Gemma model ✨

deadline is tomorrow

Building a tiny dataset

I began with a straightforward JSON file — text in, emoji out.

```
[
  { "input": "what a fun party", "output": "🥳🎉🎈 " },
  { "input": "good morning sunshine", "output": "☀️🌻😊 " },
  { "input": "so tired today", "output": "😴💤" }
]
```

I also had some fun with this — I asked ChatGPT (ironically) to generate different phrases for the same emoji sets. That helped the model learn variations like "that was lit 🔥" → "🎉🔥🙌".

Fine-tuning in Colab

Fine-tuning used to require an A100 GPU and patience. Not anymore. Using **QLoRA**, which updates only a few parameters, I fine-tuned the model on a **free T4 GPU** in Google Colab.

Here's an approximate overview of what that looked like:

```
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    Trainer,
    TrainingArguments,
    DataCollatorForLanguageModeling
)
from peft import LoraConfig, get_peft_model
from datasets import load_dataset

model_name = "google/gemma-3-270m"
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Critical: Set pad_token for Gemma
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",  # Optional: Use auto dtype for efficiency
    device_map="auto"    # Optional: Auto-map to GPU if available
)

dataset = load_dataset("json", data_files="emoji_dataset.json")

# Optional: Pre-tokenize and truncate if sequences are long (Trainer can handle
# def tokenize_function(examples):
#     return tokenizer(examples["text"], truncation=True, max_length=512)
# dataset = dataset.map(tokenize_function, batched=True, remove_columns=dataset

lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=[
        "q_proj", "k_proj", "v_proj", "o_proj",  # Attention layers
        "gate_proj", "up_proj", "down_proj"     # MLP layers
    ],
    lora_dropout=0.05,
    task_type="CAUSAL_LM"  # Explicit for clarity
)
model = get_peft_model(model, lora_config)

training_args = TrainingArguments(
    output_dir="./gemma-emoji",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    save_steps=100,
    logging_steps=10,      # Optional: Log more frequently
    evaluation_strategy="no",  # Add eval_dataset if you have one
    # group_by_length=True,  # Optional: Group similar lengths for efficiency
    # max_steps=-1,          # Optional: Run for full epochs
)

# Critical: Proper collator for CLM
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False  # Causal LM, not masked
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset["train"],
    tokenizer=tokenizer,       # Enables auto-tokenization if not pre-tokenized
    data_collator=data_collator
)
trainer.train()
```

That’s it — after training, my model began producing **only emojis**.

If you want the model to memorize specific emoji, provide more examples in the dataset.

. . .

Step 2: Making It Tiny Enough for the Web

After fine-tuning, the model remained about 1GB — small by LLM standards, but large for the browser.

To run it locally, I quantized it to 4-bit with **LiteRT** (you could also choose the **ONNX** route if you prefer Transformers.js).

Quantisation is simply advanced math for “store numbers with fewer bits.” The result: a model that takes up less space with almost the same accuracy. Mine ended up under **300MB**.

This smaller version is ideal for **MediaPipe** or **Transformers.js**, both of which utilize **WebGPU** to access your device’s hardware. So yes — it literally runs in your browser.

. . .

Step 3: Running the Model in the Browser

Here’s the fun part — no servers, no APIs, no waiting.

I used **MediaPipe's GenAI Tasks** to load and run my model directly in the browser.

```
const genai = await FilesetResolver.forGenAiTasks(
  'https://cdn.jsdelivr.net/npm/@mediapipe/tasks-genai@latest/wasm'
);

const llmInference = await LlmInference.createFromOptions(genai, {
  baseOptions: { modelAssetPath: 'path/to/yourmodel.task' }
});
const prompt = "Translate this text to emoji: what a fun party!";
const response = await llmInference.generateResponse(prompt);
console.log(response);
```

Once cached, it operates entirely offline. Zero latency. Complete privacy. It works even in airplane mode.

Smaller models result in a faster-loading app and better experience for end users.

. . .

Why This Matters

This is where AI is headed — **small, private, personal models** that you manage.

You don't need large GPUs or cloud APIs. You can fine-tune, compress, and deploy a model that fits within your app and performs exactly what you trained it for.

Gemma 3 270M is an excellent platform for that — fast, lightweight, and friendly enough to experiment with.

. . .

Final Thoughts

This project took me less than an hour from start to finish, and the result was a model that felt *mine*. It even uses my favourite emojis when I test it.

So if you've been wanting to experiment with local AI, start small. Choose a simple task, fine-tune Gemma, quantize it, and let it run right in your browser.

Because the future of AI isn't just large models in the cloud — it's the small ones that reside in your pocket.

Gemma 3 · AI · Google · Ai Agent · Llm

Follow

Published in Coding Nexus

8.2K followers · Last published just now

Coding Nexus is a community of developers, tech enthusiasts, and aspiring coders. Whether you're exploring the depths of Python, diving into data science, mastering web development, or staying updated on the latest trends in AI, Coding Nexus has something for you.

Following

Written by Civil Learning

3.2K followers · 6 following

We share what you need to know. Shared only for information.

Responses (2)

Bgerby

What are your thoughts?

asier etxebeste

Oct 26

I can't run the code.

Reply

Fraorchome

Oct 24 (edited)

I can't get the code to work. It's probably a problem with the prompt composition. How do you construct the prompt in the two phases (fine-tuning and inference)? The model's answers are always strange and have nothing to do with emojis.

Reply

More from Civil Learning and Coding Nexus

In Coding Nexus by Civil Learning

MarkItDown: Convert Anything into Markdown — the Smart Way to Feed LLMs

You know that feeling when you’re trying to feed a PDF or a Word document into an LLM, and it just doesn’t understand what’s going on…

✦ Oct 15 262 4

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You’ll Install This Year

I didn’t expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

✦ Oct 23 👏 686 💬 29

In Coding Nexus by Civil Learning

The Guy Who Let ChatGPT Trade for Him — and Somehow It Worked

You know how everyone says, “Don’t let AI touch your Money”? Well, someone on Reddit decided to ignore that.

✦ Oct 8 👏 273 💬 8

In Coding Nexus by Civil Learning

Google’s New LLM Runs on Just 0.5 GB RAM — Here’s How to Fine-Tune It Locally”

A few days ago, Google quietly released a little AI model called Gemma 3 270M.

✦ Aug 15 2.1K 30

See all from Civil Learning

See all from Coding Nexus

Recommended from Medium

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You’ll Install This Year

I didn’t expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

✦ Oct 23 686 29

In Towards Deep Learning by Sumit Pandey

LEANN: The World’s Smallest Vector Index That Could Redefine RAG

LEANN: The smallest vector index in the world. 97% storage savings, no accuracy loss. RAG just got lighter

✦ Sep 8 195 1

In Towards AI by Gao Dalie (高達烈)

RAG is Not Dead! No Chunking, No Vectors, Just Vectorless to Get the Higher Accuracy

Over the past two years, I have written numerous articles on how Retrieval-Augmented Generation has become a standard feature in nearly all…

✦ Oct 17 795 9

In Level Up Coding by Gaurav Shrivastav

I tuned a 7B Model That Outperforms GPT-4 (Here’s How You Can Too)

A practical guide to understanding and implementing model specialization for real-world applications

✦ Jul 13 320 1

Dr. Shouke Wei

oLLM vs Ollama: Democratizing Local AI Inference in 2025

Comparing Two Powerhouses for Running LLMs on Everyday Hardware

✦ Oct 6 100 2

In AI Software Engineer by Joe Njenga

Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild

Cursor has released version 2.0 , bringing the most powerful agentic AI we have seen yet, more autonomous than ever before,here’s what’s…

✦ 6d ago 543 12

See more recommendations