Fine-Tuning Qwen2.5-3B for Function Calling

Achieving 85% Tool-Call Accuracy on a Free T4 GPU using QLoRA

technical specs

Model: Qwen2.5-3B-Instruct (4-bit Quantized)

Technique: 🔧 QLoRA (LoRA r=16)

Platform: Google Colab Free Tier

GPU: 📟 NVIDIA Tesla T4

Project by Mustafa Haybat Gözgöz | www.linkedin.com/in/mustafa-haybat-gozgoz35

Turning Text into Actionable Data

Function calling transforms an LLM from a chatbot into an agent that can interact with APIs.

>_

Standard LLM

User: What is the weather like in Istanbul?

Model: It is usually sunny in Istanbul this time of year...

>_

Fine-Tuned Agent

User: What is the weather like in Istanbul?

```
Model: {
  "name": "get_weather",
  "arguments":
  "arguments": '{"city": "Istanbul"}'
}
```

TOOL CALLED

High Efficiency with QLoRA

Full fine-tuning requires enterprise compute.
QLoRA trains only adapters, freezing the main model.

```
Total Parameters: 3.1 Billion
```

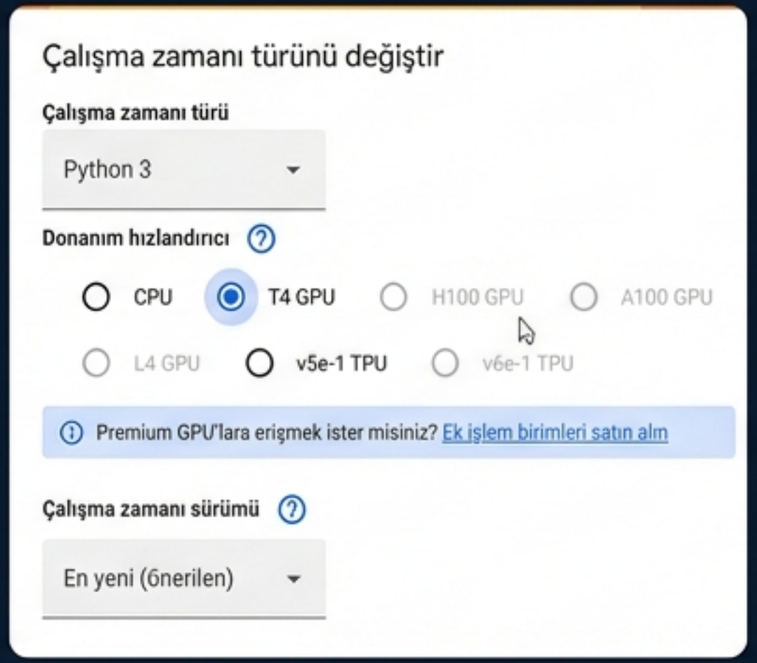
```
Trainable Parameters: 29.9 Million (0.96%)
```

```
>_
Trainable parameters: 29,933,568 of 3,115,872,256 (0.96% trained)
```

The Hardware Constraint: Zero Cost

Environment: Google Colab Free Tier | Framework: Unsloth + TRL

Hardware Verified



GPU: **NVIDIA Tesla T4**

VRAM: **15.6 GB**

Quantization: **4-bit**

Cost: **$0.00**

Data Strategy & Evaluation Balance

Dataset: glaive-function-calling-v2

Challenge: Initial eval sets often lack tool-call examples, leading to misleading metrics.

Solution: Curated a Balanced Eval Set.

`100` `Tool-Call Examples`

`100` `Plain-Text Examples`

```
================================================
TOKEN LENGTH STATS (TOKEN UZUNLUĞU İSTATİSTİKLERİ)
================================================
Mean (Ortalama) : 526 tokens
Median (Medyan) : 421 tokens
Max (Maksimum)  : 2579 tokens
Min (Minimum)   : 109 tokens

💡 If Max >> 2048, consider reducing MAX_SEQ_LENGTH
(Eğer Max >> 2048 ise, MAX_SEQ_LENGTH'i azaltmayı düşün)

> 
```

Training Configuration

Batch Size: **1 (per device)**

Grad Accumulation: **8 steps**

Effective Batch: **8**

Max Seq Length: **1024**

Steps: **120**

Total Time: **~13 mins**

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 2,500 | Num Epochs = 1 | Total steps = 120
0^0/ \_/ \    Batch size per device = 1 | Gradient accumulation steps = 8
\        /    Data Parallel GPUs = 1 | Total batch size (1 x 8 x 1) = 8
 "-____-"     Trainable parameters = 29,933,568 of 3,115,872,256 (0.96% trained)
|                                                  | [120/120 13:00, Epoch 0/1]
```

Step	Training Loss	Validation Loss
60	0.442100	0.385183
120	0.443100	0.369496

```
Unsloth: Not an error, but Qwen2ForCausalLM does not accept `num_items_in_batch`.
Using gradient accumulation will be very slightly less accurate.
Read more on gradient accumulation issues here: https://unsloth.ai/blog/gradient
wandb: WARNING URL not available in offline run
```

Analyzing the Loss Curve

Analysis: Sharp drop in first 20 steps indicates rapid syntax adaptation.

Convergence: Loss stabilizes ~0.44 at Step 60.

Decision: Continued to Step 120 to ensure robust JSON syntax generalization.

From 29% to 85% Accuracy

Strict First-Turn Accuracy (Function Name Match)

```
========================================
BREAKDOWN:
  Tool-call : 23/27 = 85.2%
  Plain-text: 70/73 = 95.9%
========================================
```

Base Model: **29.6%**

Tuned Model: **85.2%**

Improvement: **+55.6%**

Overall Accuracy: **93.0%**

Experiment: 60 Steps vs. 120 Steps

Hypothesis: Loss plateaued at step 60. Is shorter training sufficient?

STEPS	LOSS	TOOL-CALL ACCURACY
Steps=60	Loss: 0.396	Tool-Call Accuracy: 70.4% (Underfit Syntax)
Steps=120	Loss: 0.521	Tool-Call Accuracy: 85.2% (Optimal)

```
============================================================
🏆 HYPERPARAMETER COMPARISON (HİPERPARAMETRE KARŞILAŞTIRMASI)
============================================================
Base model (Base model) : 80.0%
steps=120 (120 adım)    : 93.0% | Loss: 0.521
steps=60 (60 adım)      : 90.0% | Loss: 0.396
============================================================
-> Similar accuracy but lower loss — more efficient!
(Benzer doğruluk ama daha düşük loss — daha verimli!)
```

Conclusion: 120 steps required for robust JSON generation despite higher loss value.

Engineering Challenges Solved

CUDA Out of Memory

↘ Reduced batch size to **1**.

- Increased gradient accumulation to **8**.
- Enabled `expandable_segments`.

The “100% Accuracy” Trap

- Initial metric checked only for JSON existence.

↘ Rewrote metric to validate exact function name match.

Data Imbalance

- Eval set had zero tool calls.
- Manually curated a 50/50 split (**100 Tool** / **100 Text**) for fair testing.

Project Summary

- **Efficiency:** 0.96% parameter training yielded a +55.6% accuracy boost.
- **Viability:** 3B parameter models are highly effective agents when fine-tuned.
- **Rigor:** A balanced evaluation set is critical for measuring real-world performance.

```
================================================================
🎯 PROJECT SUMMARY (PROJE ÖZETİ)
================================================================
   Model   : Qwen2.5-3B-Instruct (QLoRA, r=16)
   Dataset: glaive-function-calling-v2
   Train   : 2,500 examples | Eval: 200 balanced
================================================================
Config                      Overall  Tool-call     Loss
----------------------------------------------------------------
Base model                    80.0%      29.6%        -
steps=120, lr=2e-4            93.0%      85.2%    0.521
steps=60,  lr=2e-4          90.0%              ?    0.396
================================================================

📈 Best improvement (En iyi iyileşme):
   Tool-call accuracy: 29.6% → 85.2% → +55.6% 🚀

💡 Hyperparameter finding (Hiperparametre bulgusu):
   steps=60 gives lower loss (0.396 vs 0.521)
   (steps=60 daha düşük loss veriyor)
   Suggests model converges quickly on this dataset.
   (Model bu veri setinde hızlı yakınsıyor.)
================================================================
```

Mustafa Haybat Gözgöz

AI / ML Engineer specializing in LLMs and Fine-Tuning

Open to opportunities in LLM Fine-Tuning and Agentic AI.

www.linkedin.com/in/mustafa-haybat-gozgoz35