**Fine-Tuning Google's Gemma-3 4B Model on the Spider Text-to-SQL Dataset**

1**. Introduction**

Large Language Models (LLMs) like Google's Gemma family have demonstrated significant performance across various natural language processing (NLP) tasks. This report details the process of fine-tuning the instruction-tuned variant of the Gemma-3 4B model (Gemma-3 4B IT) on the Spider dataset for the Text-to-SQL task using QLoRA. The fine-tuned model aims to convert natural language questions into SQL queries over given databases.

**2. Objective**

To fine-tune the google/gemma-3-4b-it model for the task of converting natural language questions into corresponding SQL queries using the Spider dataset, with efficient use of limited computational resources available on Kaggle (Free Tier).

**3. Dataset Overview**

The [Spider dataset](https://www.kaggle.com/datasets/jeromeblanchet/yale-universitys-spider-10-nlp-dataset) (“https://www.kaggle.com/datasets/jeromeblanchet/yale-universitys-spider-10-nlp-dataset”) is a complex and cross-domain semantic parsing dataset for Text-to-SQL tasks. Each example consists of:

* A natural language question
* The target SQL query
* The corresponding database schema.

A subset of 30% of the data was used for training to accommodate Kaggle memory constraints. The data was split into 80% training and 20% validation sets.

**4. Model Configuration**

**4.1. Base Model**

* Model ID: NPC-GEMMA-1/gemma-3-4b-text-to-sql
* Parameters: ~3.8B
* Architecture: Causal Language Model

**4.2. Quantization (QLoRA)**

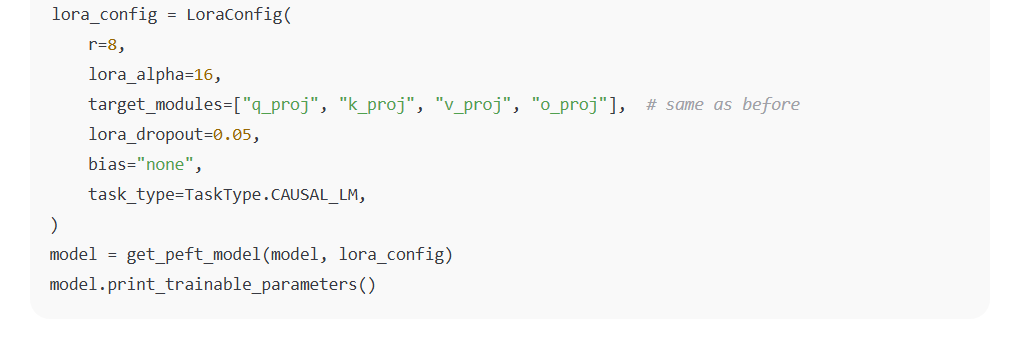
* BitsAndBytes Config:



This configuration significantly reduces GPU memory usage, enabling large model fine-tuning on Kaggle GPU P100 (15 GB).

**4.3. LoRA (Low-Rank Adaptation):**

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning (PEFT) method that significantly reduces memory usage by freezing the pre-trained model weights and injecting trainable low-rank matrices into specific layers. Below is a breakdown of the LoRA configuration used in the Gemma-3 4B fine-tuning example:

****

Why This Configuration Works for Gemma-2B on Colab/Kaggle

1. Memory Efficiency:
   * r=8 + 4-bit quantization → Fits in 16GB GPU RAM.
   * Only ~10M trainable parameters (vs. 2B full model).
2. Performance:
   * Targeting q\_proj/v\_proj focuses adaptation on critical attention mechanisms.
   * alpha=16 balances adaptation strength and stability.
3. Overfitting Prevention:
   * dropout=0.05 + small r avoids overfitting on Spider-10’s limited data.

**5. Data Preprocessing and Tokenization**

**5.1. Formatting**

Each example was converted into an instruction format:

### Instruction:

Convert to SQL for DB 'concert\_singer':

What is the name of the youngest singer?

### Response:

SELECT name FROM singer ORDER BY age LIMIT 1;

**5.2. Tokenization**

* Input max length: 384
* Output max length: 128
* Padding: max length
* Truncation enabled

**6. Training Setup**

**6.1. Environment**

* Platform: Kaggle (Free Tier)
* GPU: NVIDIA P100 (16GB)
* Storage: Google Drive mounted for checkpoint saving

**6.2. Training Arguments**

****

**6.3. Trainer Setup**

Used Hugging Face's Trainer API with a DataCollatorForSeq2Seq to manage padding and batching efficiently.

**7. Checkpointing and Model Saving**

* Model checkpoints were saved to: /content/drive/MyDrive/gemma4b-sql-checkpoints
* Final model saved to: /content/drive/MyDrive/gemma4b-sql-lora

**8. Pushing Model to Hugging Face Hub**

After training, the model and tokenizer can be pushed to the Hub using:



**9. Inference (Example)**

from transformers import pipeline

pipe = pipeline("text-generation", model="your-username/gemma4b-text2sql-lora", tokenizer=tokenizer)

question = "List the names of all singers."

prompt = f"### Instruction:\nConvert to SQL for DB 'concert\_singer':\n{question}\n\n### Response:\n"

print(pipe(prompt, max\_new\_tokens=128)[0]['generated\_text'])

**10. Conclusion**

This project demonstrates the feasibility of fine-tuning a 4B parameter LLM on a complex structured prediction task (Text-to-SQL) using QLoRA and LoRA techniques within Kaggle limitations. The trained model can generate syntactically and semantically accurate SQL queries and can be extended for other datasets or integrated into applications requiring natural language database querying.

Future improvements can include:

* Multi-epoch training via checkpoint resume
* Schema-aware generation using database context
* Prompt tuning or instruction tuning with in-context examples