To fine-tune the **TinyLLaMA** model on your **CSV dataset**, you'll need to follow a few structured steps. Here's a clear, high-level guide tailored to your use case:

□ Step-by-Step: Fine-Tune TinyLLaMA on CSV Data

---

1. Understand Your CSV Dataset

- Ensure your CSV has a clear structure. For language models, you typically need:
  - A **prompt** or **instruction** column
  - A **response** or **output** column

Example:

```
prompt,response
"Translate English to French: Hello","Bonjour"
"Summarize: The cat sat on the mat.","The cat rested on the mat."
```

If it's just plain text (e.g., `text` column), you can format it into an instruction-following style.

---

2. Environment Setup

Install necessary libraries:

```
pip install transformers datasets accelerate peft bitsandbytes
```

For training with limited resources (e.g., single GPU), consider using:

- **LoRA** (Low-Rank Adaptation) via `PEFT`
- **8-bit or 4-bit quantization** via `bitsandbytes`

---

3. Load and Preprocess the Dataset

You can use Hugging Face's `datasets` library:

```
from datasets import load_dataset

dataset = load_dataset('csv', data_files='your_file.csv')

def format_sample(example):
    return {
        "text": f"### Instruction:\n{example['prompt']}\n\n###
Response:\n{example['response']}"
    }

formatted_dataset = dataset.map(format_sample)
```

---

4. Load TinyLLaMA Model and Tokenizer

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_name = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto",
load_in_8bit=True)
```

---

5. Apply LoRA for Efficient Fine-Tuning (Optional but Recommended)

```
from peft import get_peft_model, LoraConfig, TaskType

peft_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    r=8,
    lora_alpha=32,
    lora_dropout=0.1,
    bias="none"
)

model = get_peft_model(model, peft_config)
```

---

6. Tokenize the Data

```
def tokenize(example):
    return tokenizer(example["text"], truncation=True, padding="max_length",
max_length=512)

tokenized_dataset = formatted_dataset.map(tokenize, batched=True)
```

---

7. Fine-Tune the Model

Using Hugging Face's `Trainer` API:

```
from transformers import TrainingArguments, Trainer,
DataCollatorForLanguageModeling

training_args = TrainingArguments(
    output_dir="./tinyllama-finetuned",
    per_device_train_batch_size=4,
    num_train_epochs=3,
    save_steps=500,
    save_total_limit=2,
    logging_steps=100,
    fp16=True,
    learning_rate=2e-4,
    report_to="none"
)

data_collator = DataCollatorForLanguageModeling(tokenizer, mlm=False)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)

trainer.train()
```

---

8. Save and Use the Model

```
model.save_pretrained("./tinyllama-finetuned")
tokenizer.save_pretrained("./tinyllama-finetuned")
```

□ Tips

- **GPU Recommended**: At least 16GB VRAM is ideal for full precision or LoRA in 8-bit.
- **Mixed Precision**: Use `fp16=True` or `bf16=True` in `TrainingArguments`.
- **Evaluation**: You can use a held-out split of your CSV for evaluation if needed.

🎯 Goal

Fine-tune `TinyLlama/TinyLlama-1.1B-Chat-v1.0` using this kind of CSV dataset:

Expanded CSV Content (`dataset.csv`):

```
prompt,response
"Translate English to French: Hello","Bonjour"
"Summarize: The cat sat on the mat.","The cat rested on the mat."
"Explain what photosynthesis is.","Photosynthesis is the process by which
green plants use sunlight to make food from carbon dioxide and water."
"Convert 25°C to Fahrenheit.","77°F"
"Write a short poem about the ocean.","The ocean waves crash on the shore,\nA
rhythmic sound forevermore,\nIts depths are vast, its blues so wide,\nA world
of wonder deep inside."
"Translate English to Spanish: Good morning","Buenos días"
"Summarize: Artificial Intelligence is transforming industries by automating
tasks and enhancing decision-making.","AI is changing industries by
automating work and improving decisions."
"Define gravity.","Gravity is the force that attracts two bodies toward each
other, typically noticeable as the force that gives weight to objects on
Earth."
"Write a headline for a story about a new space discovery.","NASA Discovers
Potentially Habitable Planet 40 Light Years Away"
"Give a synonym for 'happy'.","Joyful"
```

🧠 How to Use This

1. **Save the file** as `dataset.csv` in your project folder.

File Structure Example

```
CopyEdit
your-project/
├── dataset.csv
├── train_tinyllama.py
```

🚀 Minimal Working Fine-Tune Script

Create a file `train_tinyllama.py` with the following:

```
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForCausalLM,
TrainingArguments, Trainer, DataCollatorForLanguageModeling
from peft import get_peft_model, LoraConfig, TaskType
import torch
```

```
# Step 1: Load CSV dataset
dataset = load_dataset("csv", data_files="dataset.csv")

# Step 2: Format text prompt-response pairs
def format_prompt(example):
    return {
        "text": f"### Instruction:\n{example['prompt']}\n\n###
Response:\n{example['response']}"
    }

dataset = dataset.map(format_prompt)

# Step 3: Load tokenizer and model
model_name = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    load_in_8bit=True,
    torch_dtype=torch.float16
)

# Step 4: Apply LoRA
lora_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    r=8,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none"
)
model = get_peft_model(model, lora_config)

# Step 5: Tokenize
def tokenize(example):
    return tokenizer(
        example["text"],
        truncation=True,
        padding="max_length",
        max_length=512
    )

tokenized_dataset = dataset.map(tokenize, batched=True)

# Step 6: Training setup
training_args = TrainingArguments(
    output_dir="./tinyllama-finetuned",
    per_device_train_batch_size=4,
    num_train_epochs=3,
    fp16=True,
    save_steps=100,
    logging_steps=10,
    save_total_limit=2,
    report_to="none"
)

trainer = Trainer(
```

```
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    tokenizer=tokenizer,
    data_collator=DataCollatorForLanguageModeling(tokenizer, mlm=False)
)

# Step 7: Train
trainer.train()

# Step 8: Save
trainer.save_model("./tinyllama-finetuned")
tokenizer.save_pretrained("./tinyllama-finetuned")
```

Notes

- You can split your CSV manually if you want train/test separation (e.g. with pandas).
- Make sure your `dataset.csv` is in the same directory or update the path in the script.
- For **low VRAM**, reduce `per_device_train_batch_size`.

Run It

Activate your environment and run:

```
python train_tinyllama.py
```