The Beginner's Guide to SFT (Supervised Fine-Tuning)

Stack: Llama 3.1 8B + Unsloth + Google Colab + Ollama

1. The Philosophy & Model Choice

```
We are using a technique called LoRA (Low-Rank Adaptation). Instead of retraining the whole model, we tr
Selected Model: Llama-3.1-8B-Instruct (4-bit Quantized).
```

2. Step 1: The Data (On your PC)

```
Dataset format: CSV with 3 columns — instruction, input, output.

Example:
instruction,input,output
"Classify this text.","The mitochondria is the powerhouse...","{"label": "Biology"}"
```

3. Step 2: Training (Google Colab)

```
# Install Unsloth
!pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
!pip install --no-deps "xformers<0.0.27" "trl<0.9.0" peft accelerate bitsandbytes

import torch
from unsloth import FastLanguageModel
from datasets import load_dataset
from trl import SFTTrainer
from transformers import TrainingArguments

# Load model
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name="unsloth/Meta-Llama-3.1-8B-Instruct-bnb-4bit",
    max_seq_length=2048,
    dtype=None,
    load_in_4bit=True,
)

# Add LoRA
model = FastLanguageModel.get_peft_model(
    model, r=16, target_modules=["q_proj","k_proj","v_proj","o_proj"],
    lora_alpha=16, lora_dropout=0, bias="none",
    use_gradient_checkpointing="unsloth",
)

# Format (Alpaca prompt)
alpaca_prompt = """### Instruction:
{}
### Input:
{}
### Response:
{}"""
EOS_TOKEN = tokenizer.eos_token

def formatting_prompts_func(examples):
    texts = []
    for instruction, input, output in zip(examples["instruction"], examples["input"], examples["output"]
        texts.append(alpaca_prompt.format(instruction, input, output) + EOS_TOKEN)
    return {"text": texts}

dataset = load_dataset("csv", data_files="dataset.csv", split="train")
```

```
dataset = dataset.map(formatting_prompts_func, batched=True)

# Train
trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    train_dataset=dataset,
    dataset_text_field="text",
    max_seq_length=2048,
    args=TrainingArguments(
        per_device_train_batch_size=2,
        gradient_accumulation_steps=4,
        max_steps=60,
        learning_rate=2e-4,
        fp16=not torch.cuda.is_bf16_supported(),
        bf16=torch.cuda.is_bf16_supported(),
        logging_steps=1,
        output_dir="outputs",
    ),
)

trainer.train()

# Save model
model.save_pretrained_gguf("my_model", tokenizer, quantization_method="q4_k_m")
```

4. Step 3: Local Inference (Ollama)

```
Modelfile content:

FROM ./my_model-unsloth.Q4_K_M.gguf
PARAMETER temperature 0.1
SYSTEM "You are a JSON file classifier."

Commands:
ollama create filesense -f Modelfile
ollama run filesense
```

5. Step 4: Python Integration

```
import requests, json

def classify_local(text):
    url = "http://localhost:11434/api/generate"
    payload = {
        "model": "filesense",
        "prompt": f"### Instruction:\nClassify into JSON.\n\n### Input:\n{text}\n\n### Response:\n",
        "format": "json",
        "stream": False
    }
    response = requests.post(url, json=payload)
    return json.loads(response.json()['response'])
```