```
!pip install datasets
!pip install bitsandbytes
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 91.5 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 58.3 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.5 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 4.7 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 26.9 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 16.1 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 4.1 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 61.2 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nv
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed bitsandbytes-0.45.4 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.12
```

```
import pickle
import torch
import numpy as np
from transformers import (
    AutoTokenizer,
    AutoModelForCausalLM,
    BitsAndBytesConfig,
    TrainingArguments,
    Trainer,
    DataCollatorForSeq2Seq,
    TrainerCallback,
    DataCollatorForLanguageModeling
)
from datasets import Dataset
# from peft import PeftModel, LoraConfig, get_peft_model
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training
from google.colab import userdata
userdata.get('HF_TOKEN')
```

```
'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'
```

```
# Load training data
# with open('/content/drive/MyDrive/train_data_postprocessed.pkl', 'rb') as f:
#     train_data = pickle.load(f)

#Load validation data
with open('/content/drive/MyDrive/valid_data_postprocessed.pkl', 'rb') as f:
    valid_data = pickle.load(f)

# Load test data
with open('/content/drive/MyDrive/test_data_postprocessed.pkl', 'rb') as f:
    test_data = pickle.load(f)
```

```
print(valid_data[0])
```

```
{'before_merge': 'def hough_line_peaks(hspace, angles, dists, min_distance=9, min_angle=10,\n                     threshold=None, num_pe
```

```
def create_prompt(record):
    # Start with the buggy code
    prompt = f"### Buggy Code AST:\n{record['old_ast_json']}\n\n"

    # Include traceback information if available
    if record.get('traceback_type') or record.get('full_traceback'):
        prompt += f"### Traceback:\n{record.get('traceback_type', '')}: {record.get('full_traceback', '')}\n\n"

    # Instruction for the model to output the fix
    prompt += "### Provide the corrected code AST below:\n"
    return prompt


def prepare_record(record):
    return {
        "input": create_prompt(record),  # the prompt that includes the context
        "output": record["new_ast_json"]    # the target corrected code
    }

# prepared_train = [prepare_record(r) for r in train_data]
prepared_valid = [prepare_record(r) for r in valid_data]
prepared_test  = [prepare_record(r) for r in test_data]
```

```
#load tokenizer and model
model_name = "meta-llama/CodeLlama-7b-hf"
hf_token = "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"
tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token
```

```
tokenizer_config.json: 100%          749/749 [00:00<00:00, 83.8kB/s]
tokenizer.model: 100%                500k/500k [00:00<00:00, 1.18MB/s]
tokenizer.json: 100%                 1.84M/1.84M [00:00<00:00, 24.5MB/s]
special_tokens_map.json: 100%        411/411 [00:00<00:00, 55.4kB/s]
```

```
def tokenize_record(record, max_length=512):
    tokenized_input = tokenizer(
        record["input"], truncation=True, padding="max_length", max_length=max_length, return_tensors="pt"
    )
    tokenized_output = tokenizer(
        record["output"], truncation=True, padding="max_length", max_length=max_length, return_tensors="pt"
    )
    return {
        "input_ids": tokenized_input["input_ids"].squeeze(),
        "attention_mask": tokenized_input["attention_mask"].squeeze(),
        "labels": tokenized_output["input_ids"].squeeze()
    }
```

```
# tokenized_train = [tokenize_record(r) for r in prepared_train]
tokenized_valid = [tokenize_record(r) for r in prepared_valid]
tokenized_test  = [tokenize_record(r) for r in prepared_test]


print("Sample tokenized training record:")
print(tokenized_valid[0])
```

```
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1]), 'labels': tensor([    1,   518,    13,  1678,   376,  7355, 29898,  2587, 11759,  6678,
         3206, 29898,   978,  2433, 29882,   820, 29918,  1220, 29918,   412,
        10327,   742,  6389, 29922, 25699, 29898,  1066,  6194,  5085, 11759,
         1402,  6389, 11759,  1191, 29898,  1191,  2433, 14158,  5477,  1852,
        29898,  1191,  2433, 19536,  5477,  1852, 29898,  1191,  2433, 29881,
         2879,  5477,  1852, 29898,  1191,  2433,  1195, 29918, 19244,  5477,
         1852, 29898,  1191,  2433,  1195, 29918,  2521,  5477,  1852, 29898,
         1191,  2433,   386, 12268,  5477,  1852, 29898,  1191,  2433,  1949,
        29918,   412, 10327,  1495,  1402,  9049,  6194,  5085, 11759,  1402,
         9049, 29918,  4381, 29879, 11759,  1402, 21274, 11759, 12075,   424,
        29898,  1767, 29922, 29929,   511, 28601, 29898,  1767, 29922, 29896,
        29900,   511, 28601, 29898,  1767, 29922,  8516,   511, 23833, 29898,
         1767, 29922,  1170, 29898,   333,  2433,  9302,   742, 12893, 29922,
         5896, 25739, 12421,  2433,  7192,   742, 12893, 29922,  5896,  3101,
        11724,  3573, 11759, 21176, 29898,  1767, 29922, 12075,   424, 29898,
         1767,  2433, 11609,  1236, 10327,   297,   263,  7812,  1196,   379,
          820,  4327, 29889,  1966, 29876,  1966, 29876,  1678, 13355, 11057,
         1556, 19555,  3454, 13055,   491,   263,  3058, 10696,   322,  5418,
         1966, 29876,  1678,   297,   263,   379,   820,  4327, 29889, 10050,
        29899, 27525,   398,  1462, 23881,   411,  1422, 15786,   338,  1966,
        29876,  1678,  7436, 16949,   297,   278,   937,   313,  5721,  2925,
        29897,   322,  1473,   313, 19536, 29897,  9927,  1966, 29876,  1678,
          310,   278,   379,   820,  2913,   304, 12439,  1236, 10327, 29889,
         1966, 29876,  1966, 29876,  1678, 12662,  2699,  1966, 29876,  1678,
          448,  1378, 29899,  1966, 29876,  1678,   298,  3493,   584,   313,
        29940, 29892,   341, 29897,  1409,  1966, 29876,  4706,   379,   820,
         2913,  4133,   491,   278,   421, 29882,   820, 29918,  1220, 29952,
          740, 29889,  1966, 29876,  1678, 23619,   584,   313, 29924, 29892,
        29897,  1409,  1966, 29876,  4706,  3218,   793,  4133,   491,   278,
          421, 29882,   820, 29918,  1220, 29952,   740, 29889,  4007, 21571,
          304,   367,  9126, 29889,  1966, 29876,  4706,  6695, 19536, 14352,
        29896, 29962,   448, 23619, 29961, 29900, 29962,  1275,   349, 29902,
        12913,  1966, 29876,  1678,  1320, 29879,   584,   313, 29940, 29892,
         1723,  1409,  1966, 29876,  4706,  6652,  2925,  4133,   491,   278,
          421, 29882,   820, 29918,  1220, 29952,   740, 29889,  1966, 29876,
         1678,  1375, 29918, 19244,   584,   938, 29892, 13136,  1966, 29876,
         4706,  3080, 12539,  5418,  2903,  1218,  3454,   313, 27525,   398,
         4175,  2159,   363,   937,  1966, 29876,  4706,  9927,   310,   298,
          820,  2913,   467,  1966, 29876,  1678,  1375, 29918,  2521,   584,
          938, 29892, 13136,  1966, 29876,  4706,  3080, 12539, 10696,  2903,
         1218,  3454,   313, 27525,   398,  4175,  2159,   363,  1473,  1966,
        29876,  4706,  9927,   310,   298,   820,  2913,   467,  1966, 29876,
         1678, 16897,   584,  5785, 29892, 13136,  1966, 29876,  4706,  3080,
        12539, 26171,   310,  1236, 10327, 29889, 13109,   338,   421, 29900,
        29889, 29945,   334,  4236, 29898, 14158, 14466,  1966, 29876,  1678,
          954, 29918,   412, 10327,   584,   938, 29892, 13136,  1966, 29876,
         4706,  5918, 12539,  1353,   310,  1236, 10327, 29889,  1932,   278,
         1353,   310,  1236, 10327, 13461, 29879,   421,  1949, 29918,   412,
        10327,  1673,  1966, 29876,  4706,   736,   421,  1949, 29918,   412,
        10327, 29952, 10350,  2729,   373, 19224, 26171, 29889,  1966, 29876,
         1966, 29876,  1678, 16969,  1966, 29876,  1678,   448, 22158,  1966,
        29876,  1678])}
```

```
print(tokenizer.decode(tokenized_valid[0]["input_ids"]))
print(tokenizer.decode(tokenized_valid[0]["labels"]))
```

```
<s> ### Buggy Code AST:
[
    "Module(body=[FunctionDef(name='hough_line_peaks', args=arguments(posonlyargs=[], args=[arg(arg='hspace'), arg(arg='angles'), arg(ar
<s> [
    "Module(body=[FunctionDef(name='hough_line_peaks', args=arguments(posonlyargs=[], args=[arg(arg='hspace'), arg(arg='angles'), arg(ar
```

```
# Convert to a Hugging Face Dataset.
# train_dataset = Dataset.from_dict({
#     "input_ids": [x["input_ids"].tolist() for x in tokenized_train],
#     "attention_mask": [x["attention_mask"].tolist() for x in tokenized_train],
#     "labels": [x["labels"].tolist() for x in tokenized_train],
# })



# print("A tokenized training sample:")
# print(tokenizer.decode(tokenized_train[0]["input_ids"]))
```

```
A tokenized training sample:
<s> ### Buggy Code AST:
[
    "Module(body=[FunctionDef(name='plot', args=arguments(posonlyargs=[], args=[arg(arg='result_dict_file'), arg(arg='show'), arg(arg='p
    "FunctionDef(name='plot', args=arguments(posonlyargs=[], args=[arg(arg='result_dict_file'), arg(arg='show'), arg(arg='plot_save_file
```

```
#Create a BitsAndBytesConfig object to replace deprecated arguments
quantization_config = BitsAndBytesConfig(
    load_in_4bit=True,                # Enable 4-bit quantization.
    bnb_4bit_compute_dtype="float16", # Use float16 for computations.
    bnb_4bit_quant_type="nf4",        # Common quantization type.
    bnb_4bit_use_double_quant=True    # Improves accuracy.
)


# import bitsandbytes
# print(bitsandbytes.__version__)



#Load the model
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    token=hf_token,  # Corrected parameter name
    quantization_config=quantization_config,
    device_map="auto"  # Automatically assign layers to available devices.
)
```

```
config.json: 100%                                   637/637 [00:00<00:00, 72.8kB/s]
model.safetensors.index.json: 100%                  25.1k/25.1k [00:00<00:00, 2.69MB/s]
Fetching 2 files: 100%                              2/2 [00:43<00:00, 43.74s/it]
model-00002-of-00002.safetensors: 100%              3.50G/3.50G [00:19<00:00, 312MB/s]
model-00001-of-00002.safetensors: 100%              9.98G/9.98G [00:43<00:00, 396MB/s]
Loading checkpoint shards: 100%                     2/2 [00:07<00:00, 3.46s/it]
generation_config.json: 100%                        116/116 [00:00<00:00, 12.9kB/s]
```

```
#Wrap the model with LoRA adapters using the PEFT library

# Prepare the model for k-bit training (this step adapts the model for low-rank adaptation).
model = prepare_model_for_kbit_training(model)

# Define your LoRA configuration.
lora_config = LoraConfig(
    r=8,                    # Rank of the LoRA adapters.
    lora_alpha=32,          # Scaling factor.
    target_modules=["q_proj", "v_proj"],  # Adjust these to match your model's architecture.
    lora_dropout=0.1,       # Dropout for LoRA layers.
    bias="none"
)

# Wrap the model with the PEFT LoRA modules.
model = get_peft_model(model, lora_config)
print("Model is now ready for QLoRA fine-tuning!")
```

```
Model is now ready for QLoRA fine-tuning!
```

```
# Define a custom callback to print a message (basepoint) at the end of each epoch
class PrintEpochCallback(TrainerCallback):
    def on_epoch_end(self, args, state, control, **kwargs):
        # Print epoch number and last logged loss (if available)
        if state.log_history:
            last_log = state.log_history[-1]
            loss_str = f", Loss: {last_log.get('loss', 'N/A')}" if 'loss' in last_log else ""
            print(f"Epoch {state.epoch} completed{loss_str}.")


#Hugging Face’s Trainer API to set up training parameters

# Training arguments
training_args = TrainingArguments(
    output_dir="/content/drive/MyDrive/fine-tuned/codellama-finetuned-ast",
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,
    gradient_checkpointing=True,
    num_train_epochs=3,
    logging_dir="./logs",
    logging_steps=10,
    save_strategy="epoch",
    label_names=["labels"],
    learning_rate=1e-4,
    fp16=True,  # Mixed-precision training if using CUDA
    push_to_hub=False
)

# Data collator for padding/truncating sequences
# data_collator = DataCollatorForSeq2Seq(
#     tokenizer,
#     model=model,
#     padding=True,
#     max_length=512,
#     return_tensors="pt"
# )

data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False  # For causal language modeling
)


# trainer = Trainer(
#     model=model,
#     args=training_args,
#     train_dataset=train_dataset,
#     tokenizer=tokenizer,
#     data_collator = data_collator,
#     callbacks=[PrintEpochCallback()]
# )

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    data_collator=data_collator,
    callbacks=[PrintEpochCallback()]
)



trainer.train()
```

```
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please spe
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ··········
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: bhavyasree2002 (bhavyasree2002-university-of-illinois-chicago) to https://api.wandb.ai. Use `wandb login
```

Tracking run with wandb version 0.19.8
Run data is saved locally in `/content/wandb/run-20250403_163418-5tvpqpdw`
Syncing run **/content/drive/MyDrive/fine-tuned/codellama-finetuned-ast** to Weights & Biases (docs)
View project at https://wandb.ai/bhavyasree2002-university-of-illinois-chicago/huggingface
View run at https://wandb.ai/bhavyasree2002-university-of-illinois-chicago/huggingface/runs/5tvpqpdw

```
`use_cache=True` is incompatible with gradient checkpointing. Setting `use_cache=False`.
```

[1323/1323 2:38:36, Epoch 2/3]

Step	Training Loss
10	0.625800
20	0.503600
30	0.453800
40	0.429900
50	0.410200
60	0.405000
70	0.372800
80	0.399900
90	0.403600
100	0.367000
110	0.378400
120	0.366400
130	0.374500
140	0.357600
150	0.358700
160	0.373900
170	0.368600
180	0.373100
190	0.381300
200	0.358100
210	0.359600
220	0.355100
230	0.343300
240	0.346000
250	0.360700
260	0.371100
270	0.352400
280	0.331600
290	0.349000
300	0.368600
310	0.344800
320	0.344800
330	0.344800
340	0.385700
350	0.344600
360	0.348500

370	0.329800
380	0.329100
390	0.350400
400	0.340400
410	0.332700
420	0.336600
430	0.344400
440	0.347200
450	0.348300
460	0.310600
470	0.352900
480	0.336300
490	0.338500
500	0.337900
510	0.315700
520	0.338100
530	0.336000
540	0.325700
550	0.336600
560	0.329600
570	0.338300
580	0.336400
590	0.323700
600	0.330500
610	0.332400
620	0.308500
630	0.332400
640	0.333400
650	0.324600
660	0.324200
670	0.337800
680	0.337900
690	0.313900
700	0.324700
710	0.339700
720	0.334300
730	0.327500
740	0.323800
750	0.319700
760	0.318600
770	0.314500
780	0.339100
790	0.334700
800	0.341200
810	0.342200
820	0.315200
830	0.326000

840	0.316700
850	0.343200
860	0.315900
870	0.315900
880	0.310100
890	0.312400
900	0.315300
910	0.319100
920	0.331100
930	0.314600
940	0.320500
950	0.317000
960	0.323400
970	0.328900
980	0.331800
990	0.319800
1000	0.327500
1010	0.312300
1020	0.317800
1030	0.323300
1040	0.325400
1050	0.330200
1060	0.321100
1070	0.320600
1080	0.327400
1090	0.314000
1100	0.323300
1110	0.313700
1120	0.316600
1130	0.333800
1140	0.309100
1150	0.317900
1160	0.331300
1170	0.313300
1180	0.315300
1190	0.333100
1200	0.307000
1210	0.336500
1220	0.315200
1230	0.318100
1240	0.296200
1250	0.324400
1260	0.321600
1270	0.317300
1280	0.324400
1290	0.323400
1300	0.293600

```
1310    0.311700
1320    0.321000

Epoch 1.0 completed, Loss: 0.3472.
Epoch 2.0 completed, Loss: 0.3101.
Epoch 2.994900849858357 completed, Loss: 0.321.
TrainOutput(global_step=1323, training_loss=0.3407199298083016, metrics={'train_runtime': 9546.4914, 'train_samples_per_second': 4.437,
'train_steps_per_second': 0.139, 'total_flos': 8.588238045189243e+17, 'train_loss': 0.3407199298083016, 'epoch': 2.994900849858357})
```

Start coding or generate with AI.