Nanochat

——SYSU ChenZhenhuang

一、环境要求

服务器配置：

python3.10-3.11

Torch-2.8.0+cu128

Rust: rustc 1.90.0

Rustbpe

GPU: NVIDIA GeForce RTX 3090 * 2

教程推荐配置：

8 * A100（40GB） or 8 * H100（80GB）

二、简易版

从Github上克隆nanochat项目

```
git clone https://github.com/karpathy/nanochat.git
cd nanochat
```

配置国内镜像

```
# 创建配置脚本
cat > setup_mirrors.sh << 'EOF'
#!/bin/bash

# PyPI 镜像
export PIP_INDEX_URL=https://pypi.tuna.tsinghua.edu.cn/simple
mkdir -p ~/.pip
cat > ~/.pip/pip.conf << 'PIPCONF'
[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
trusted-host = pypi.tuna.tsinghua.edu.cn
PIPCONF

# Rust 镜像
export RUSTUP_DIST_SERVER=https://rsproxy.cn
export RUSTUP_UPDATE_ROOT=https://rsproxy.cn/rustup
cat > ~/.cargo/config << 'CARGOCONF'
[source.crates-io]
replace-with = 'rsproxy-sparse'
[source.rsproxy]
registry = "https://rsproxy.cn/crates.io-index"
[source.rsproxy-sparse]
```

```
registry = "sparse+https://rsproxy.cn/index/"
[registries.rsproxy]
index = "https://rsproxy.cn/crates.io-index"
[net]
git-fetch-with-cli = true
CARGOCONF

# HuggingFace 镜像
export HF_ENDPOINT=https://hf-mirror.com

echo "✅ 镜像配置完成！"
EOF

chmod +x setup_mirrors.sh
source setup_mirrors.sh
```

如果环境没问题，可以一键配置完成进行训练

```
# 方式1：直接运行（前台）
bash speedrun.sh

# 方式2：后台运行（推荐）
nohup bash speedrun.sh > train.log 2>&1 &

# 方式3：使用 tmux
tmux new -s nanochat
bash speedrun.sh
# 按 Ctrl+B 然后 D 来 detach
# 重新连接：tmux attach -t nanochat
```

如果发现直接运行speedrun.sh出现报错，可能有多种原因，可能是网络不稳定、CUDA版本低不支持、显卡内存小等原因。可以接着看第三部分。

三、准备工作（主要针对训练过程中可能出现的问题）

在第一次运行speedrun.sh的时候其实很大一部分的依赖已经下载完毕了，这里主要是检查以及补充。

安装Pytorch

```
pip install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu128
```

更新CUDA

```
首先用nvidia-smi检查cuda版本（参考教程用了cu121或者cu118，但是官方文档中要求cu128，因此建议更新）
sudo apt install nvidia-driver-570
sudo apt-get install -y cuda-toolkit-12-8
```

装uv，配置国内镜像，安装依赖：

```
pip install uv -i https://pypi.tuna.tsinghua.edu.cn/simple
export UV_INDEX_URL=https://pypi.tuna.tsinghua.edu.cn/simple

uv sync --extra gpu
```

创建（激活）虚拟环境

```
cd nanochat
uv venv

source .venv/bin/activate
```

安装 Rust

```
export RUSTUP_DIST_SERVER=https://rsproxy.cn
export RUSTUP_UPDATE_ROOT=https://rsproxy.cn/rustup
curl --proto '=https' --tlsv1.2 -sSf https://rsproxy.cn/rustup-init.sh | sh
```

配置Cargo镜像

```
# 创建 Cargo 配置文件
mkdir -p ~/.cargo
cat > ~/.cargo/config << 'EOF'
[source.crates-io]
replace-with = 'rsproxy-sparse'

[source.rsproxy]
registry = "https://rsproxy.cn/crates.io-index"

[source.rsproxy-sparse]
registry = "sparse+https://rsproxy.cn/index/"

[registries.rsproxy]
index = "https://rsproxy.cn/crates.io-index"

[net]
git-fetch-with-cli = true
EOF
```

编译 rustbpe 分词器

```
cd nanochat

# 安装 maturin
pip install maturin -i https://pypi.tuna.tsinghua.edu.cn/simple

# 编译并安装 rustbpe
uv run maturin develop --release --manifest-path rustbpe/Cargo.toml
```

手动下载数据集

预训练数据集

```
mkdir -p ~/.cache/nanochat/base_data
cd ~/.cache/nanochat/base_data
```

```
（在终端运行的脚本）
for i in {0..239}; do
    wget https://hf-mirror.com/datasets/karpathy/fineweb-edu-100b-shuffle/resolve/main/shard_$(printf "%05d" $i).parquet
done
```

如果不想就等，可以使用nohup在后台下载

```
nohup bash -c 'for i in {0..239}; do
    wget https://hf-mirror.com/datasets/karpathy/fineweb-edu-100b-shuffle/resolve/main/shard_$(printf "%05d" $i).parquet
done' > download.log 2>&1 &
```

下载评估数据

```
curl -L -o eval_bundle.zip https://karpathy-public.s3.us-west-2.amazonaws.com/eval_bundle.zipunzip -q eval_bundle.zipmv eval_bundle ~/.cache/nanochat/rm eval_bundle.zip
```

微调数据集

```
export HF_ENDPOINT=https://hf-mirror.com    设置环境变量

# SmolTalk（对话数据）
huggingface-cli download HuggingFaceTB/smoltalk --repo-type dataset

# GSM8K（数学推理）
huggingface-cli download gsm8k --repo-type dataset

# ARC（科学推理）
huggingface-cli download ai2_arc --repo-type dataset
```

环境检查

```
cat > check_env.sh << 'EOF'
#!/bin/bash

echo "🔍 检查 nanochat 环境..."
echo ""

# Python
echo -n "Python: "
python --version || echo "❌ 未安装"

# PyTorch
echo -n "PyTorch: "
```

```
python -c "import torch; print(torch.__version__)" 2>/dev/null || echo "❌ 未安装"

# CUDA
echo -n "CUDA: "
python -c "import torch; print('✅ 可用' if torch.cuda.is_available() else '❌ 不可用')" 2>/dev/null

# GPU 信息
echo -n "GPU: "
nvidia-smi --query-gpu=name --format=csv,noheader 2>/dev/null || echo "❌ 未检测到"

# Rust
echo -n "Rust: "
rustc --version 2>/dev/null || echo "❌ 未安装"

# rustbpe
echo -n "rustbpe: "
python -c "import rustbpe; print('✅ 已安装')" 2>/dev/null || echo "❌ 未安装"

# 磁盘空间
echo -n "磁盘空间: "
df -h . | tail -1 | awk '{print $4 " 可用"}'

# 网络测试
echo -n "HuggingFace 连接: "
curl -s -o /dev/null -w "%{http_code}" https://hf-mirror.com > /dev/null && echo "✅ 正常" || echo "❌ 失败"

echo ""
echo "检查完成！"
EOF

chmod +x check_env.sh
bash check_env.sh
```

四、训练

前面的准备工作做完后，如果你的显卡配置足够，可以直接使用speedrun.sh进行训练

因为在训练过程中一个训练阶段完成后会保存模型参数，因此可以分步进行训练，防止由于多个进程执行导致显卡内存不够（容易出现torch.OutOfMemoryError: CUDA out of memory的报错）

```
清空报告
python -m nanochat.report reset
```

构建Tokenizer

```
uv run maturin develop --release --manifest-path rustbpe/Cargo.toml
```

（可选）验证数据下载情况

```
python -m nanochat.dataset -n 8
ython -m nanochat.dataset -n 240 &
DATASET_DOWNLOAD_PID=$!
```

训练tokenizer以及评估

```
python -m scripts.tok_train --max_chars=2000000000

python -m scripts.tok_eval
```

模型预训练以及评估（这一步需要最多时间）

```
# pretrain the d20 model
torchrun --standalone --nproc_per_node=2 -m scripts.base_train -- --depth=20 --device_batch_size=32
# 这里如果配置低，可以降低模型的层数，如果显卡内存不够，建议减少batch_size
# 对于3090显卡使用以下
torchrun --standalone --nproc_per_node=2 -m scripts.base_train -- --depth=4 --device_batch_size=8 --num_iterations=500
# evaluate the model on a larger chunk of train/val data and draw some samples
torchrun --standalone --nproc_per_node=2 -m scripts.base_loss
# evaluate the model on CORE tasks
torchrun --standalone --nproc_per_node=2 -m scripts.base_eval
```

模型训练完成则输出如下：

```
Step 00500 | CORE metric: 0.0262
<|bos|>The capital of France is the capital of the country, the capital of the country, the capital of the
<|bos|>The chemical symbol of gold is the symbol of the gold. The symbol of gold is the symbol of the gold
<|bos|>If yesterday was Friday, then tomorrow will be the day of the day.
The day of the day is the day of the
<|bos|>The opposite of hot is the same as the other two. The other two are the same as the other
<|bos|>The planets of the solar system are: the planets of the solar system. The planets of the solar system are called the
<|bos|>My favorite color is the color of the color. The color is the color of the color. The
<|bos|>If 5*x + 3 = 13, then x is 5*x + 5*x + 5*x +
2025-10-30 16:08:05,394 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved model file to: /home/chenzhenhuang/.cache/nanochat/base_checkpo
2025-10-30 16:08:05,569 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved optimizer file to: /home/chenzhenhuang/.cache/nanochat/base_che
2025-10-30 16:08:05,569 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved metadata file to: /home/chenzhenhuang/.cache/nanochat/base_chec
Peak memory usage: 5830.34MiB
Total training time: 21.83m
```

中间训练以及评估

```
# 下载2.3MB的合成身份对话数据，为nanochat赋予个性特征
curl -L -o $NANOCHAT_BASE_DIR/identity_conversations.jsonl https://karpathy-public.s3.us-west-2.amazonaws.com/identity_conversations.jsonl

# run midtraining and eval the model（同样，根据显卡配置来调整batch_size）
torchrun --standalone --nproc_per_node=2 -m scripts.mid_train -- --device_batch_size=8
torchrun --standalone --nproc_per_node=2 -m scripts.chat_eval -- -i mid

```

中间训练完成后则输出如下：

```
step 00760 (98.66%) | loss: 7.225719 | lrm: 0.07 | dt: 1375.28ms | tok/sec: 23,826 | mfu: 2.79 | total time: 30.06m
step 00761 (98.72%) | loss: 7.236032 | lrm: 0.06 | dt: 1382.99ms | tok/sec: 23,693 | mfu: 2.77 | total time: 30.08m
step 00762 (98.86%) | loss: 7.253546 | lrm: 0.06 | dt: 1311.49ms | tok/sec: 24,985 | mfu: 2.92 | total time: 30.10m
step 00763 (99.00%) | loss: 7.241513 | lrm: 0.05 | dt: 1310.00ms | tok/sec: 25,013 | mfu: 2.93 | total time: 30.13m
step 00764 (99.13%) | loss: 7.259986 | lrm: 0.04 | dt: 1407.74ms | tok/sec: 23,277 | mfu: 2.72 | total time: 30.15m
step 00765 (99.25%) | loss: 7.245582 | lrm: 0.04 | dt: 1425.46ms | tok/sec: 22,987 | mfu: 2.69 | total time: 30.17m
step 00766 (99.38%) | loss: 7.234273 | lrm: 0.03 | dt: 1295.78ms | tok/sec: 25,288 | mfu: 2.96 | total time: 30.20m
step 00767 (99.51%) | loss: 7.234347 | lrm: 0.02 | dt: 1425.97ms | tok/sec: 22,979 | mfu: 2.69 | total time: 30.22m
step 00768 (99.64%) | loss: 7.229212 | lrm: 0.02 | dt: 1311.91ms | tok/sec: 24,977 | mfu: 2.92 | total time: 30.24m
step 00769 (99.77%) | loss: 7.235914 | lrm: 0.01 | dt: 1425.77ms | tok/sec: 22,982 | mfu: 2.69 | total time: 30.26m
step 00770 (99.91%) | loss: 7.244533 | lrm: 0.00 | dt: 1281.03ms | tok/sec: 25,579 | mfu: 2.99 | total time: 30.29m
step 00771 (99.99%) | loss: 7.238132 | lrm: 0.00 | dt: 1155.84ms | tok/sec: 28,349 | mfu: 3.32 | total time: 30.31m
Step 00771 | Validation bpb: 1.7614
2025-10-30 16:16:20,512 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved model file to: /home/chenzhenhuang/.cache/nanochat/mid_checkpoi
2025-10-30 16:16:20,601 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved optimizer file to: /home/chenzhenhuang/.cache/nanochat/mid_chec
2025-10-30 16:16:20,601 - nanochat.checkpoint_manager - [32m[1mINFO[0m - Saved metadata file to: /home/chenzhenhuang/.cache/nanochat/mid_check
Peak memory usage: 3234.26MiB
Total training time: 30.31m
```

微调

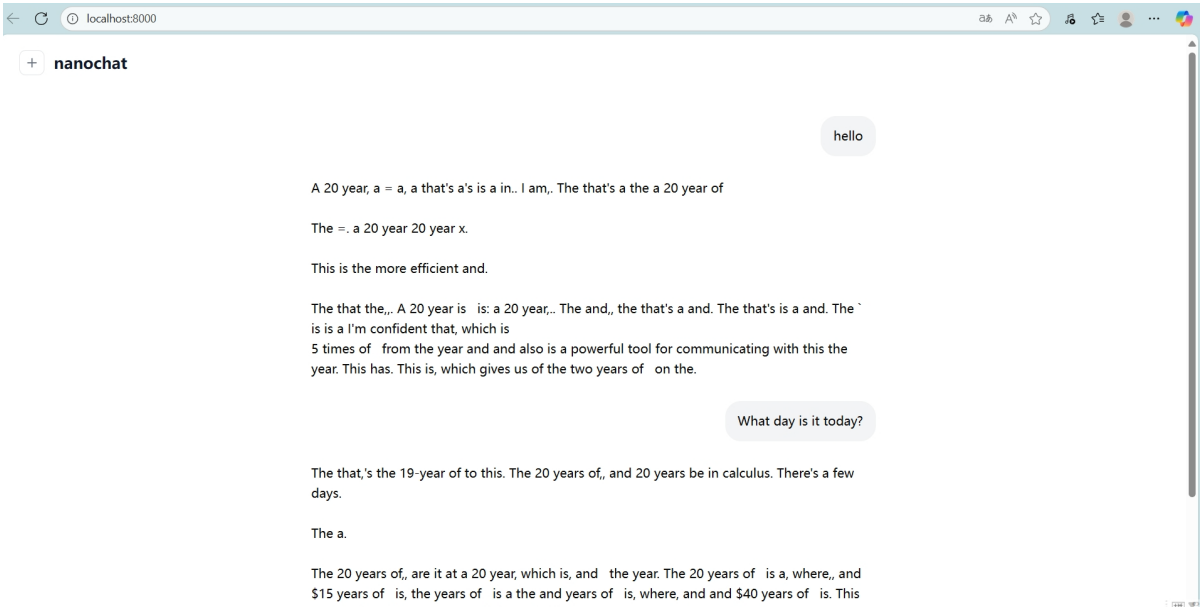
```
torchrun --standalone --nproc_per_node=2 -m scripts.chat_sft -- --device_batch_size=8
torchrun --standalone --nproc_per_node=2 -m scripts.chat_eval -- -i sft
```

微调完成后输出如下：

```
Step 00667/00682 | Training loss: 3.196783| lrm: 0.021994| num_tokens: 10,003
Step 00668/00682 | Training loss: 3.480313| lrm: 0.020528| num_tokens: 11,842
Step 00669/00682 | Training loss: 4.231853| lrm: 0.019062| num_tokens: 11,847
Step 00670/00682 | Training loss: 3.791309| lrm: 0.017595| num_tokens: 10,691
Step 00671/00682 | Training loss: 3.287172| lrm: 0.016129| num_tokens: 11,507
Step 00672/00682 | Training loss: 5.419115| lrm: 0.014663| num_tokens: 10,030
Step 00673/00682 | Training loss: 4.631368| lrm: 0.013196| num_tokens: 10,719
Step 00674/00682 | Training loss: 4.303693| lrm: 0.011730| num_tokens: 12,834
Step 00675/00682 | Training loss: 3.911473| lrm: 0.010264| num_tokens: 8,749
Step 00676/00682 | Training loss: 3.910254| lrm: 0.008798| num_tokens: 10,127
Step 00677/00682 | Training loss: 4.308422| lrm: 0.007331| num_tokens: 10,183
Step 00678/00682 | Training loss: 4.589873| lrm: 0.005865| num_tokens: 13,337
Step 00679/00682 | Training loss: 3.684354| lrm: 0.004399| num_tokens: 6,468
Step 00680/00682 | Training loss: 4.490584| lrm: 0.002933| num_tokens: 12,337
Step 00681 | Validation loss: 4.029108
Final: 254/1024 (24.80%)
Final: 252/1024 (24.61%)
Step 00681 | mmlu_acc: 0.248047, arc_easy_acc: 0.246094
2025-10-30 16:29:52,045 - nanochat.checkpoint_manager - INFO - Saved model file to: /home/chenzhenhuang/.cache/nanochat/chatsft_checkpoints/d4/model_000681.pt
2025-10-30 16:29:52,046 - nanochat.checkpoint_manager - INFO - Saved metadata file to: /home/chenzhenhuang/.cache/nanochat/chatsft_checkpoints/d4/meta_000681.json
✅ Saved model checkpoint to /home/chenzhenhuang/.cache/nanochat/chatsft_checkpoints/d4
[W1030 16:29:53.569351836 AllocatorConfig.cpp:28] Warning: PYTORCH_CUDA_ALLOC_CONF is deprecated, use PYTORCH_ALLOC_CONF instead (function operator())
```

运行

```
chenzhenhuang@pku-biostat:~/code/nanochat$ source .venv/bin/activate
(nanochat) chenzhenhuang@pku-biostat:~/code/nanochat$ python -m scripts.chat_web
Autodetected device type: cuda
/home/chenzhenhuang/code/nanochat/.venv/lib/python3.10/site-packages/torch/__init__.py:1617: UserWarning: Please use the new API settings to control TF32 behavior, such as torch.backends.cudnn.conv.fp32_precision = 'tf32' or torch.backends.cuda.matmul.fp32_precision = 'ieee'. Old settings, e.g, torch.backends.cuda.matmul.allow_tf32 = True, torch.backends.cudnn.allow_tf32 = True, allowTF32CuDNN() and allowTF32CuBLAS() will be deprecated after Pytorch 2.9. Please see https://pytorch.org/docs/main/notes/cuda.html#tensorfloat-32-tf32-on-ampere-and-later-devices (Triggered internally at /pytorch/aten/src/ATen/Context.cpp:80.)
  _C._set_float32_matmul_precision(precision)
2025-10-30 16:31:50,275 - nanochat.common - INFO - Distributed world size: 1
Starting NanoChat Web Server
Temperature: 0.8, Top-k: 50, Max tokens: 512
INFO:     Started server process [1407963]
INFO:     Waiting for application startup.
Loading nanochat models across GPUs...
Initializing worker pool with 1 GPUs...
Loading model on GPU 0...
2025-10-30 16:31:50,300 - nanochat.checkpoint_manager - INFO - No model tag provided, guessing model tag: d4
2025-10-30 16:31:50,300 - nanochat.checkpoint_manager - INFO - Loading model from /home/chenzhenhuang/.cache/nanochat/chatsft_checkpoints/d4 with step 681
2025-10-30 16:31:50,478 - nanochat.checkpoint_manager - INFO - Building model with config: {'sequence_len': 2048, 'vocab_size': 65536, 'n_layer': 4, 'n_head': 2, 'n_kv_head': 2, 'n_embd': 256}
All 1 workers initialized!
Server ready at http://localhost:8000
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:48906 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:48906 - "GET /health HTTP/1.1" 200 OK
INFO:     127.0.0.1:48912 - "GET /logo.svg HTTP/1.1" 200 OK
```



可以进行简单对话（起码d12以上，目前由于层数太少因此训练效果较差），至此nanochat的部署到此完成。

五、Reference

[1] 从分词器构建到强化学习：nanochat开源项目下载与部署全流程教程，教你一步步训练ChatGPT语言模型-CSDN博客

[2] Hoshino-wind/nanochat-cn