Lab 103: Deep Dive: Train a Mini-GPT on CPU (Optional)

Level: L400 Journey: llm-foundations Time: ~180 min Cost: free

info "Phase 3 · local first" The required route is deterministic and local. The Real route is optional and includes an estimate and cleanup.

What You'll Learn

Implement tokenization, embeddings, attention, and next-token loss Train a tiny model on CPU with fixed seeds Inspect overfitting and sampling Explain why the exercise is educational, not production training

Prerequisites

Python 3.10+ and a terminal No credentials for the required route Read the linked official specification before using a hosted adapter

Visual model

[diagram or code sample]

Scenario

OutdoorGear needs a reviewable workshop implementation with deterministic fixtures, explicit trust boundaries, measurable acceptance criteria, and a reversible optional hosted adapter.

Guided exercise

Step 1: tokenize

1. Run the deterministic fixture for this step. 2. Record the declared input, output, policy decision, and failure behavior. 3. Change one bounded behavior and run the local test again.

Evidence: Save the command, result, and one sentence explaining why the evidence is trustworthy.

Step 2: train

1. Run the deterministic fixture for this step. 2. Record the declared input, output, policy decision, and failure behavior. 3. Change one bounded behavior and run the local test again.

Evidence: Save the command, result, and one sentence explaining why the evidence is trustworthy.

Step 3: sample

1. Run the deterministic fixture for this step. 2. Record the declared input, output, policy decision, and failure behavior. 3. Change one bounded behavior and run the local test again.

Evidence: Save the command, result, and one sentence explaining why the evidence is trustworthy.

Step 4: analyze_limits

1. Run the deterministic fixture for this step. 2. Record the declared input, output, policy decision, and failure behavior. 3. Change one bounded behavior and run the local test again.

Evidence: Save the command, result, and one sentence explaining why the evidence is trustworthy.

Run locally

[diagram or code sample]

Simulation / Real path

Simulation Real path

Use the included deterministic fixtures and local validator. Estimated cost: US$0. Replace only the adapter boundary. Set a spending cap before execution, retain the same

acceptance tests, and record region, retention, identity, telemetry, and pricing assumptions. Delete hosted deployments, endpoints, indexes, traces, and test data after validation.

Supporting kit

The site build generates stable PDF and ZIP downloads. CI also keeps a temporary artifact mirror for 30 days.

README.md: quick start and evidence checklist CHEATSHEET.md: decision and review guide scenario.json: deterministic exercise contract test_scenario.py: local acceptance tests

Download generated PDF Download generated ZIP

Scored knowledge check

question "Why keep a local contract?"

??? success "Answer" It provides a deterministic baseline before adding hosted variability and cost.

question "When is the Real route complete?"

??? success "Answer" Only after validation evidence and cleanup are recorded.

question "How should tool output be treated?"

??? success "Answer" As untrusted data until policy and schema validation succeed.

Exit criteria

[ ] Local tests pass without credentials. [ ] The diagram has a text explanation and does not depend on color. [ ] The scored quiz was reviewed. [ ] The Real path has a cost cap and cleanup plan. [ ] Sources and stability status were checked.

Official public sources

PyTorch language modeling tutorial Attention Is All You Need

Supporting file: CHEATSHEET.md

Deep Dive: Train a Mini-GPT on CPU (Optional) ? Cheat Sheet

Start with the local contract and explicit acceptance criteria. Treat prompts, retrieved content, tool output, and protocol messages as untrusted. Keep identity, authorization, policy, telemetry, cost, and cleanup explicit. Prefer the smallest orchestration or adaptation that satisfies the evidence.

Supporting file: README.md

Deep Dive: Train a Mini-GPT on CPU (Optional)

python -m unittest discover -v

The required route is local and deterministic.

Supporting file: scenario.json

{ "lab": "103", "steps": ["tokenize", "train", "sample", "analyze_limits"], "required_route": "local", "credentials_required": false, "real_path": { "optional": true, "cost_cap_required": true, "cleanup_required": true } }

Supporting file: test_scenario.py

import json import unittest from pathlib import Path

class Phase3ScenarioTests(unittest.TestCase): @classmethod def setUpClass(cls): cls.data = json.loads(Path("scenario.json").read_text(encoding="utf-8"))

def test_identity_and_steps(self): self.assertEqual(self.data["lab"], "103") self.assertEqual(len(self.data["steps"]), 4)

def test_free_first_contract(self): self.assertEqual(self.data["required_route"], "local")

self.assertFalse(self.data["credentials_required"])
def test_real_path_guardrails(self): real = self.data["real_path"]
self.assertTrue(real["optional"]) self.assertTrue(real["cost_cap_required"])
self.assertTrue(real["cleanup_required"])
if __name__ == "__main__": unittest.main()

Supporting file: tiny_gpt_dataset.txt
OutdoorGear workshop fixture. Policy-safe local sample.