

Modeling Metaphors: Analyzing Fine-Tuning Effects in ChatGPT

A Reproducible Guide for Ethical AI Experiments

A capstone submitted to the faculty of San Francisco State University
In partial fulfillment of the requirements for the Degree
Graduate Certificate in Ethical Artificial Intelligence
By Cassia Reddig
San Francisco, California
May 2025

© 2025 Cassia Reddig. San Francisco State University. All Rights Reserved.

This capstone makes transparent use of generative AI in the writing process. ChatGPT (GPT-4o, developed by OpenAI) was used as a tool to assist with drafting, revising, and refining language. The research, analysis, and writing were conducted independently by the author, and all ideas, structure, and interpretations reflect original scholarly work. AI assistance was limited to improving clarity, coherence, and organization in select sections. All content generated with AI support was critically reviewed, edited, and integrated by the author to ensure academic rigor and integrity.

Project Overview & Roadmap

Overview:

This project fine-tunes GPT-4o-mini using 20 metaphor-rich prompt-response examples related to AI ethics and business compliance. It analyzes how fine-tuning impacts metaphor use, framing, and clarity.

It also serves as a reproducible teaching guide for students learning how to fine-tune LLMs to adopt tone, rhetorical strategies, and domain-specific voice.

Roadmap:

1. Motivation & Background
2. Setting Up Your Project from Scratch
3. Preparing and Uploading Training Data
4. Running a Fine-Tuning Job via API
5. Testing Baseline vs Fine-Tuned Models
6. Running a Generalization (Unseen Prompt) Test
7. Evaluating Outputs (Quantitative + Visual)
8. Extensions and Ethical Considerations
9. Reference Materials & Resources

1. Motivation & Background

Motivation: Metaphor as Cognitive Infrastructure

Why Metaphors?

Metaphors help translate abstract concepts into concrete mental models. According to Conceptual Metaphor Theory (Lakoff & Johnson, 1980), metaphors are not just literary devices—they structure how we think.

In domains like ethics, law, and AI safety, metaphor enhances clarity, engagement, and conceptual alignment (Boroditsky, 2011).

Example frames taught to the model:

- Compliance is a foundation
- Leadership is a compass
- AI is a guide

Why Fine-Tune GPT?

Out-of-the-box models are accurate but flat. They often lack conceptual framing and rhetorical richness. Language models may produce fluent-sounding outputs that fail to convey ethical nuance (Bender et al., 2021).

Explainability requires more than accuracy—it requires framing that resonates with human reasoning (Miller, 2019). Fine-tuning GPT-4o-mini on metaphor-rich data enables clearer, more ethically framed communication. It's affordable, fast, and reproducible for classroom and research use.

Background & Theoretical Context

Conceptual Framing:

This project builds on *Conceptual Metaphor Theory* (Lakoff & Johnson, 1980), which argues that metaphors are central to cognition. Metaphors help humans make sense of complex and abstract domains by aligning them with familiar experiences (Boroditsky, 2011).

Metaphors play a key role in:

- Communicating ethical principles
- Translating policy into action
- Framing complex topics like risk and responsibility

LLM Limitations:

- Language models may sound fluent but often miss ethical nuance or reinforce bias (Bender et al., 2021).
- Explainability in AI should go beyond accuracy to include cognitively aligned communication (Miller, 2019).
- Metaphorical framing can also introduce subtle biases or unintended influence in high-stakes settings (Weidinger et al., 2022).

RAG vs. Fine-Tuning:

- **RAG** (Retrieval-Augmented Generation) injects factual context from external data sources (Lewis et al., 2020).
- **Fine-Tuning** adapts the model's internal narrative voice and rhetorical framing (OpenAI, 2024).

Research Gap:

Most prior work focuses on factual accuracy or downstream task performance. Few studies investigate how fine-tuning affects metaphor use—especially in high-stakes ethical or compliance contexts.

Metaphor Use in Ethical AI Design

Metaphors function as strategic framing devices in Ethics & Compliance programs (Collins & Kanashiro, 2022)

Explainability must support stakeholder understanding, not just model transparency

- Effective explanations require attentional alignment and epistemic agency (Mindt & Montemayor, 2020)

Trustworthy AI depends on both accuracy and explainability, tailored to users' context and values (Petkovic, 2023)

Ethical theories justify framing choices:

- Deontology (duty to disclose)
- Consequentialism (minimize harm)
- Virtue ethics (integrity in communication)

Fine-tuning LLMs with metaphor aligned, E&C consistent language helps integrate AI into organizational values and internal messaging (Ganesan, 2022)

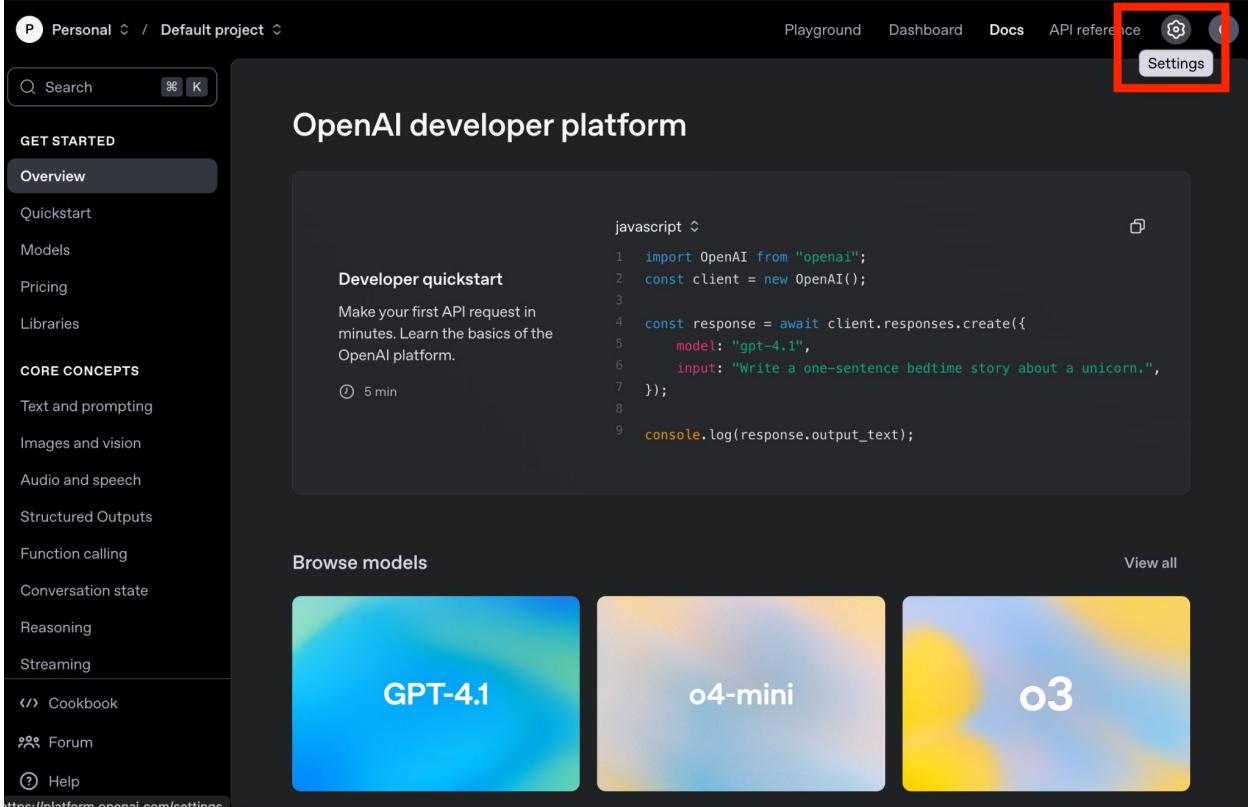
2. Setting Up Your Project from Scratch

电费 Step 1 — Set Up OpenAI Account + Budget Controls

✓ Create a Platform Account

Go to platform.openai.com and sign up using email, Google, or Microsoft.

Note: Your free .edu account may not work to create an account because of differences in the sign-in process.

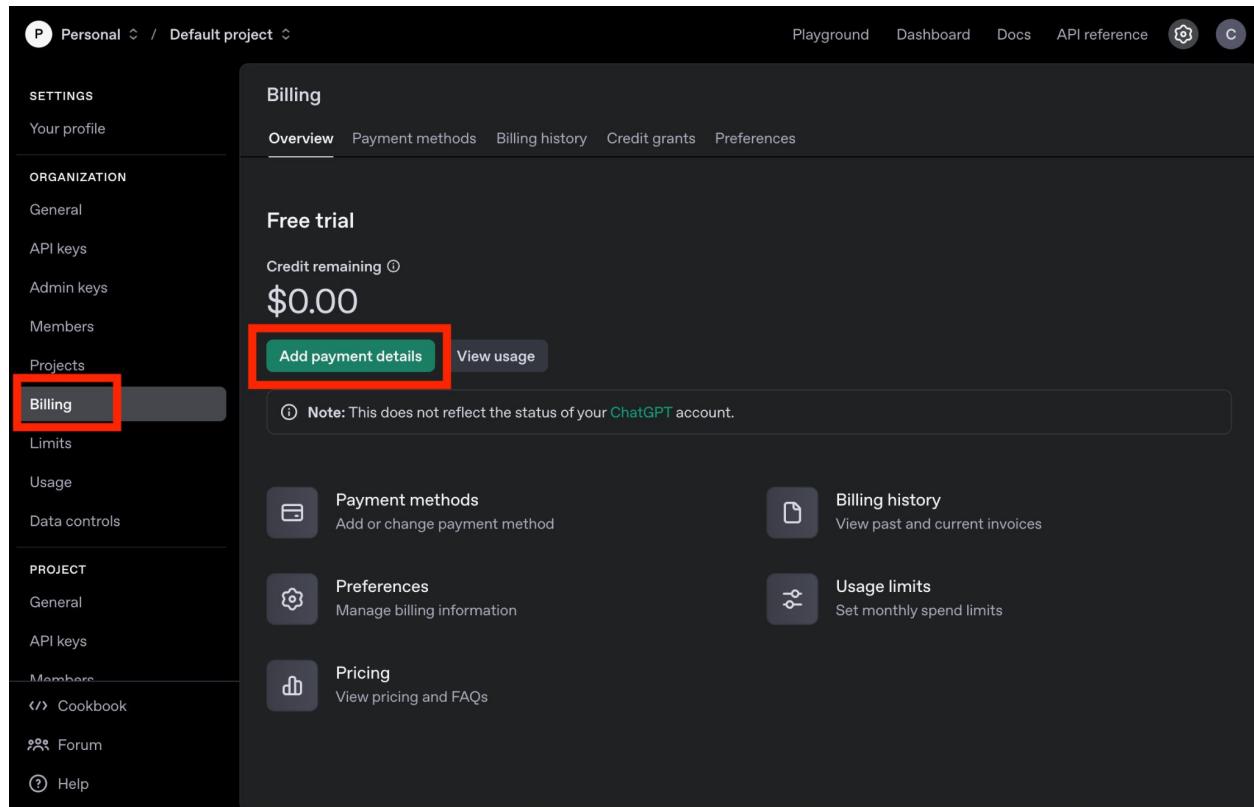


The screenshot shows the OpenAI developer platform interface. On the left, there's a sidebar with navigation links like 'GET STARTED', 'Overview', 'Quickstart', 'Models', 'Pricing', 'Libraries', 'CORE CONCEPTS', 'Text and prompting', 'Images and vision', 'Audio and speech', 'Structured Outputs', 'Function calling', 'Conversation state', 'Reasoning', 'Streaming', 'Cookbook', 'Forum', and 'Help'. The 'Overview' link is currently selected. At the top right, there are links for 'Playground', 'Dashboard', 'Docs', 'API reference', and a 'Settings' button, which is highlighted with a red box. The main content area features the heading 'OpenAI developer platform' and a 'Developer quickstart' section with a snippet of JavaScript code for making an API request. Below that is a 'Browse models' section with cards for 'GPT-4.1', 'o4-mini', and 'o3'.

```
javascript ◊
1 import OpenAI from "openai";
2 const client = new OpenAI();
3
4 const response = await client.responses.create({
5   model: "gpt-4.1",
6   input: "Write a one-sentence bedtime story about a unicorn.",
7 });
8
9 console.log(response.output_text);
```

Add Payment Method & Budget

- Go to **Settings > Billing > Overview**



The screenshot shows the ChatGPT Settings interface with a dark theme. The left sidebar has a tree view:

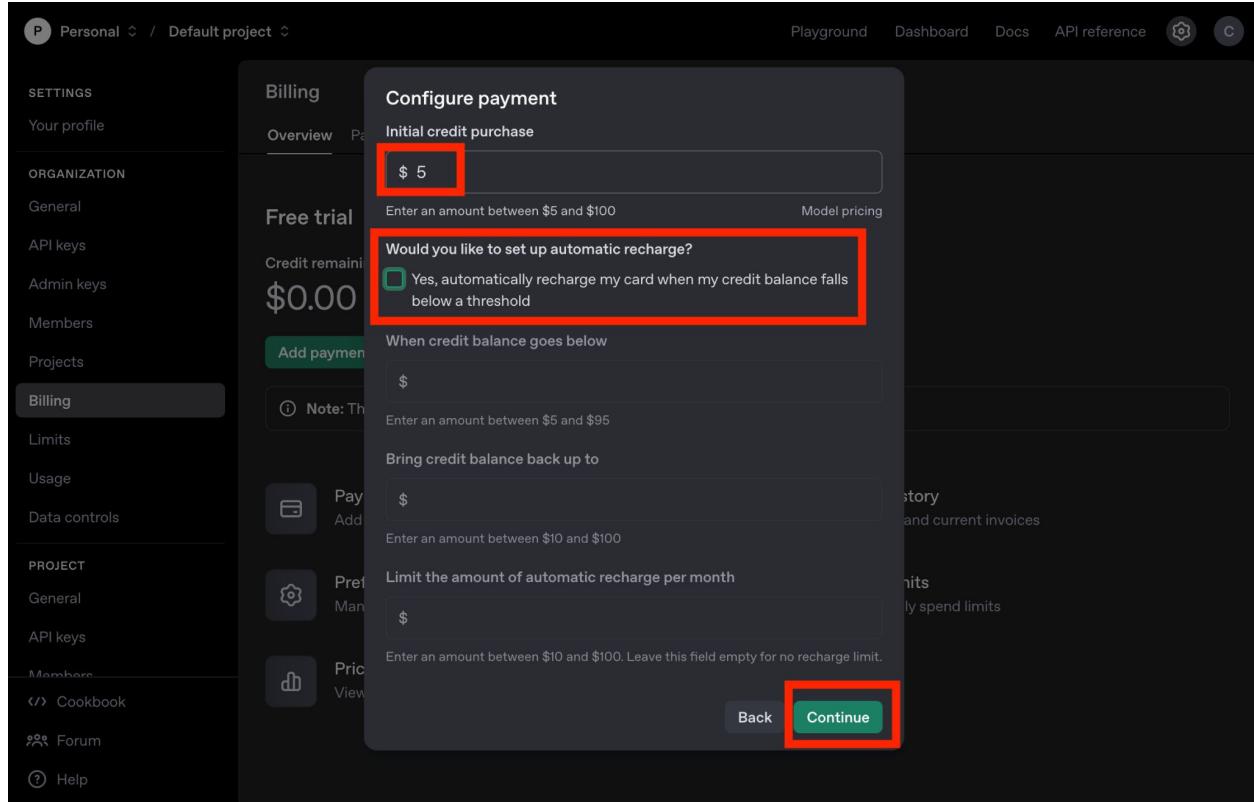
- Personal / Default project
- SETTINGS
 - Your profile
- ORGANIZATION
 - General
 - API keys
 - Admin keys
 - Members
 - Projects
 - Billing** (highlighted with a red box)
 - Limits
 - Usage
 - Data controls
- PROJECT
 - General
 - API keys
 - Members
 - Cookbook
 - Forum
 - Help

The main content area is titled "Billing" and shows the "Overview" tab selected. It displays a "Free trial" section with "Credit remaining" at \$0.00. A prominent green button labeled "Add payment details" is highlighted with a red box. Below it is a note: "Note: This does not reflect the status of your ChatGPT account." To the right, there are four cards:

- Payment methods**: Add or change payment method
- Billing history**: View past and current invoices
- Preferences**: Manage billing information
- Usage limits**: Set monthly spend limits
- Pricing**: View pricing and FAQs

At the top right of the main content area are links to "Playground", "Dashboard", "Docs", and "API reference", along with user profile and settings icons.

- Add a small **credit balance** (e.g., \$5)
- **✗ Disable automatic recharge** (recommended for students)





Enable Budget Limits

- Go to **Settings > Limits**
- Set a **monthly budget**
- Alerts at 80% and 100% are automatically applied

The screenshot shows the OpenAI API dashboard with a dark theme. On the left, a sidebar menu includes 'Personal', 'Default project', 'SETTINGS' (with 'Your profile'), 'ORGANIZATION' (with 'General', 'API keys', 'Admin keys', 'Members', 'Projects', 'Billing', and 'Limits' highlighted with a red box), 'PROJECT' (with 'General' and 'API keys'), and a 'Claim free tokens' button. At the bottom of the sidebar are navigation icons for back, forward, search, and help. The main content area is titled 'Limits Usage tier 1'. It features a section for 'Organization budget' with a 'June budget' of '\$0.00 / \$120.00' and a note that it resets in 11 days. It includes two alert configurations: '80% usage alert' and '100% usage alert', each with a trash and edit icon. Below this is a 'Rate limits' section with a note about API usage being subject to rate limits like TPM and RPM. A table at the bottom provides specific limits for the 'gpt-4.1' model: TOKEN LIMITS (30,000 TPM), REQUEST AND OTHER LIMITS (500 RPM), and BATCH QUEUE LIMITS (900,000 TPD). A note at the bottom of the table states: 'Note: Limits for specific model versions may vary, expand the table to see all models.'

Playground Dashboard Docs API reference ⚙️ C

Limits Usage tier 1

Organization budget

June budget
\$0.00 / \$120.00

Add alert Edit budget

Resets in 11 days.

80% usage alert

100% usage alert

Rate limits

API usage is subject to rate limits applied on tokens per minute (TPM), requests per minute or day (RPM/RPD), and other model-specific limits. Your organization's rate limits are listed below.

Visit our [rate limits guide](#) to learn more about how rate limits work.

MODEL TOKEN LIMITS REQUEST AND OTHER LIMITS BATCH QUEUE LIMITS

MODEL	TOKEN LIMITS	REQUEST AND OTHER LIMITS	BATCH QUEUE LIMITS
gpt-4.1	30,000 TPM	500 RPM	900,000 TPD

ⓘ Note: Limits for specific model versions may vary, expand the table to see all models.



Fine-Tuning Cost Example:

- **Model:** gpt-4o-mini-2024-07-18
- **Samples:** 20 prompt-completion pairs (~4,000 tokens)
- **Total cost:** ≈ \$0.01

⚠ Key Tips

- You **must add credit** to fine-tune
- All actions must use **the same API project and key**

The screenshot shows the 'Limits' tab of the OpenAI API budget settings. At the top, it says 'Organization budget' with a 'June budget' of '\$0.00 / \$120,000' and a note that it resets in 11 days. Below this is a modal window titled 'Edit budget' where a 'Monthly budget' of '\$ 10' is set. The modal has 'Cancel' and 'Save' buttons. To the right of the modal, there's a section for 'Rate limits' with a note about API usage being subject to rate limits applied on tokens per minute (TPM), requests per minute or day (RPM/RPD), and other metrics. It also links to a 'rate limits guide'. At the bottom of the page, there's a note about specific model version limits.



Step 2 — Create or Edit a Project in OpenAI

📁 What is a Project?

A project on OpenAI Platform links together:

- Uploaded training files
- API keys and usage limits
- Fine-tuned model access

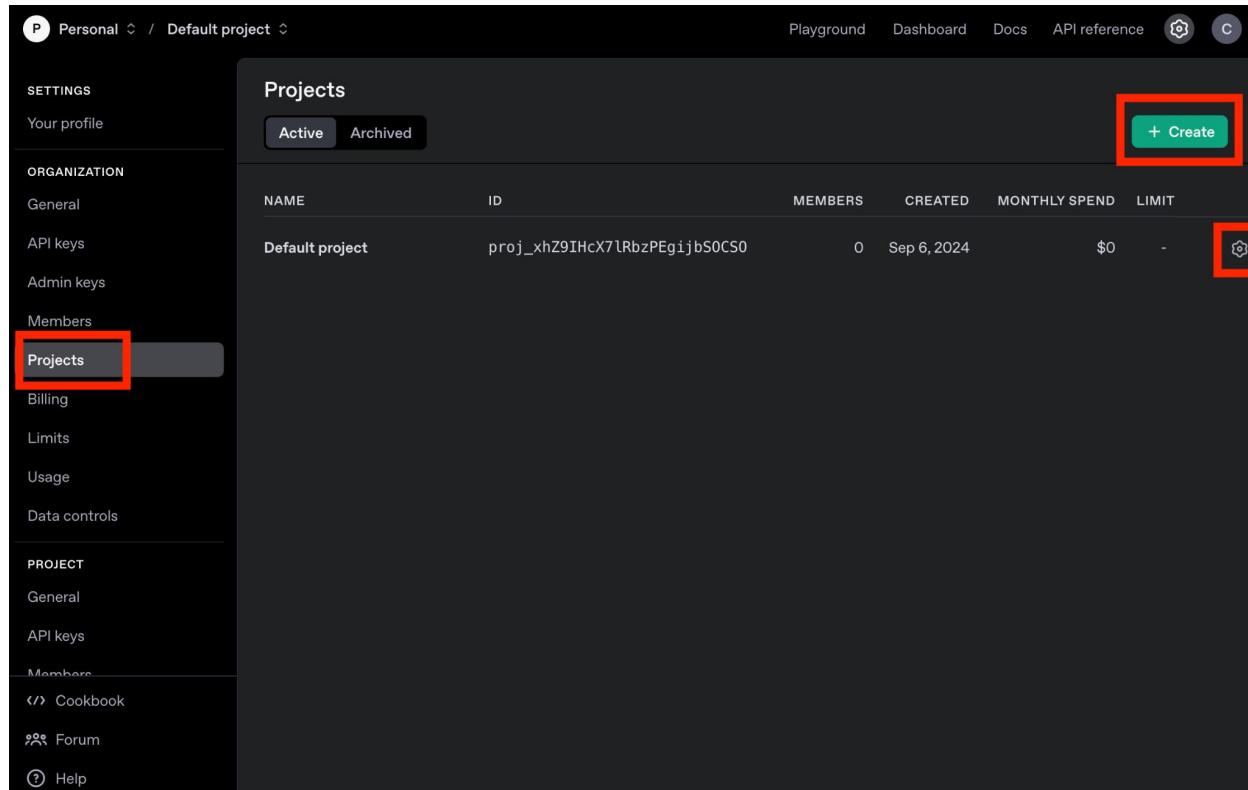
All scripts must use the same project to avoid permission errors (e.g., “403 model not found”).

Create or Edit Your Project

1. Go to **Settings > Projects**

Option 1: Click the gear icon on "Default Project" to rename

Option 2: Click **+ Create** to make a new one



The screenshot shows the Project Settings interface. On the left, a sidebar menu includes sections like SETTINGS (Your profile, Organization, Members), PROJECT (General, API keys, Members), and external links (Cookbook, Forum, Help). The 'Projects' section is highlighted with a red box. On the right, the main 'Projects' page displays a table with one row for the 'Default project'. The table columns are NAME, ID, MEMBERS, CREATED, MONTHLY SPEND, and LIMIT. The 'Default project' row shows proj_xhZ9IHcX7lRbzPEgiJbs0CS0, 0 members, created on Sep 6, 2024, \$0 monthly spend, and a limit of -. A red box highlights the '+ Create' button in the top right corner of the projects table area.

NAME	ID	MEMBERS	CREATED	MONTHLY SPEND	LIMIT
Default project	proj_xhZ9IHcX7lRbzPEgiJbs0CS0	0	Sep 6, 2024	\$0	-

Option 1: Edit Default Project

Name the project

(e.g., "Fine-Tuning-Project")

Then click **Save**

The screenshot shows the Project settings interface. On the left, a sidebar lists various settings categories: SETTINGS (Your profile), ORGANIZATION (General, API keys, Admin keys, Members), Projects, Billing, Limits, Usage, Data controls, PROJECT (General, API keys, Members), Cookbook, Forum, and Help. The 'General' tab under the PROJECT section is currently selected. The main area is titled 'Project settings' and contains fields for 'Project name' (set to 'Fine-Tuning-Project') and 'Project ID' (set to 'proj_xhZ9IHcX7IRbzPEgjbjSOCsO'). There is also a toggle switch labeled 'Disable user API keys' which is turned off. At the bottom right of the main area is a green 'Save' button, which is highlighted with a red rectangular box.

Option 2: + Create a new project

Name the project

(e.g., "Fine-Tuning-Project")

Then click **Create**

The screenshot shows a dark-themed user interface for managing projects. On the left, a sidebar menu lists various settings and organization options. The 'Projects' option is selected and highlighted with a grey background. In the main content area, a table titled 'Projects' displays a single row for 'Fine-Tuning-Project'. A modal window titled 'Create a new project' is open over the table. The modal contains a description of what a project is, a 'Name' input field containing 'Fine-Tuning-Project', and two buttons at the bottom: 'Cancel' and 'Create'. The 'Create' button is highlighted with a red rectangular box.

P Personal / Fine-Tuning-Project

Playground Dashboard Docs API reference

SETTINGS

Your profile

ORGANIZATION

General

API keys

Admin keys

Members

Projects

Billing

Limits

Usage

Data controls

PROJECT

General

API keys

Members

Cookbook

Forum

Help

Projects

Active Archived

+ Create

NAME	ID	MEMBERS	CREATED	MONTHLY SPEND	LIMIT
Fine-Tuning-Project			Sep 6, 2024	\$0	-

Create a new project

Projects are shared environments where teams can collaborate and share API resources. You can set custom rate limits and manage access to resources. [Learn more](#).

Name

Human-friendly label for your project, shown in user interfaces and on exports

Fine-Tuning-Project

Cancel Create

Switch Between Projects

Click the dropdown in the upper-left of the dashboard

✓ A checkmark shows which project is active

⚠ Make sure it matches your API key and fine-tuning job

The screenshot shows the LangChain dashboard interface. On the left, there is a sidebar with various navigation options. At the top of this sidebar is a dropdown menu labeled "Personal" with a downward arrow. To its right is the current project name, "Fine-Tuning-Project", also with a downward arrow. Below these are several other options: "General", "API keys", "Admin keys", "Members", and "Projects". The "Projects" option is currently selected, as indicated by a dark gray background. A red box highlights this selection. To the right of the sidebar, the main dashboard area displays a table of projects. The first row in the table corresponds to the project selected in the sidebar. The table has columns for "ID", "MEMBERS", "CREATED", "MONTHLY SPEND", and "LIMIT". There are two rows of data in the table:

ID	MEMBERS	CREATED	MONTHLY SPEND	LIMIT
proj_xhZ9IHcX7lRbzPEgijbS0CS0	0	Sep 6, 2024	\$0	-
proj_RRwDzJdZdv4Ko1xKobuhytt2	0	Jun 19, 2025	\$0	-

At the top right of the dashboard, there are links for "Playground", "Dashboard", "Docs", and "API reference", along with some user profile icons. A green button labeled "+ Create" is located in the top right corner of the main content area.



Step 3 — Create Your OpenAI API Key

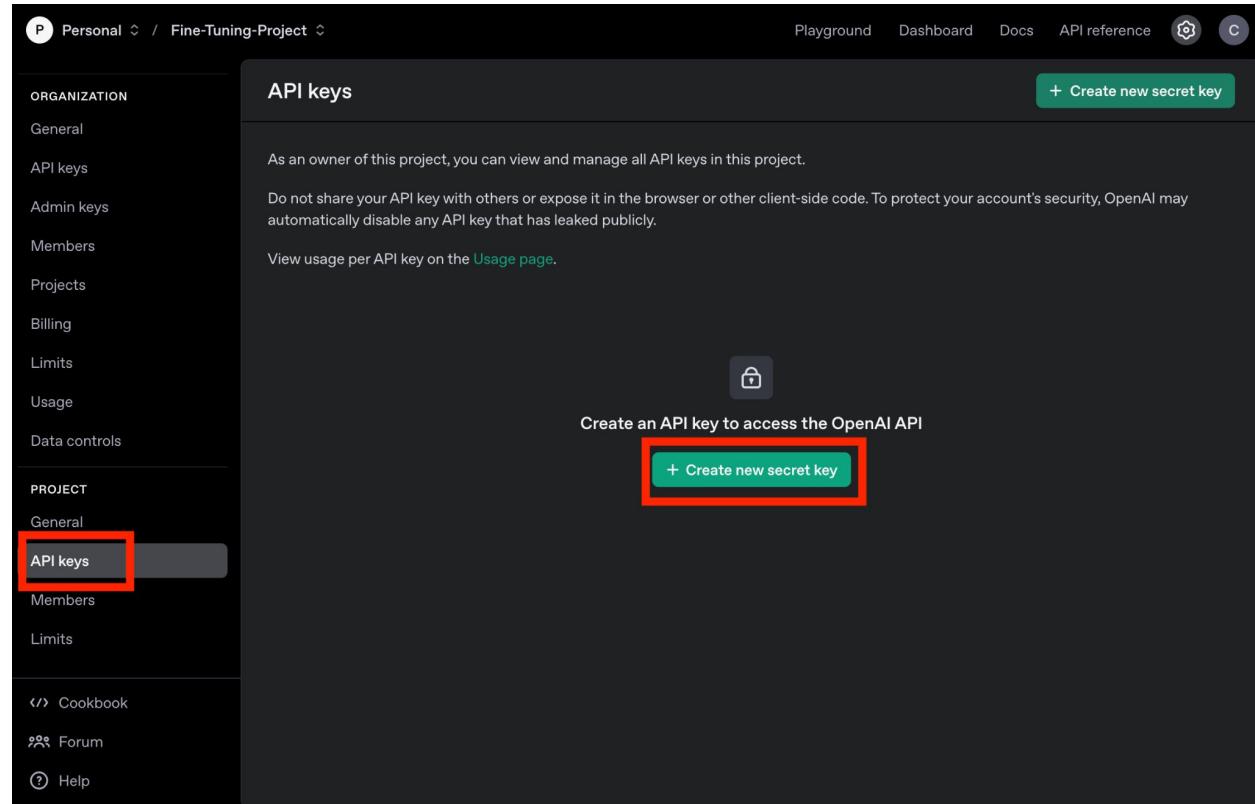


What is an API Key?

Your API key is a unique secret that lets your Python scripts connect to OpenAI's services under your account and project.

Steps to Create a Key

1. In your OpenAI dashboard, go to **API Keys**
2. Click **+ Create new secret key**



The screenshot shows the OpenAI dashboard interface. On the left, there's a sidebar with navigation links: General, API keys (which is highlighted with a red box), Admin keys, Members, Projects, Billing, Limits, Usage, and Data controls. Below that is a 'PROJECT' section with General, API keys (highlighted with a red box again), Members, and Limits. At the bottom of the sidebar are links for Cookbook, Forum, and Help. The main content area is titled 'API keys'. It contains a message: 'As an owner of this project, you can view and manage all API keys in this project.' Below that is another message: 'Do not share your API key with others or expose it in the browser or other client-side code. To protect your account's security, OpenAI may automatically disable any API key that has leaked publicly.' At the bottom of this section is a link 'View usage per API key on the Usage page.' To the right of these messages is a large button labeled '+ Create new secret key' with a red box around it. Above this button is a small lock icon.

3. Name it something meaningful
(e.g., "Fine-Tune-Experiment")

Then click **Create secret key**

The screenshot shows the OpenAI API keys management interface. On the left, a sidebar lists organization settings like General, API keys, Admin keys, Members, Projects, Billing, Limits, Usage, and Data controls. Under PROJECT, the API keys option is highlighted with a red box. A modal window titled "Create new secret key" is open. It asks "Owned by" (with "You" selected) and "Name" (set to "FT-Experiment"). The "Project" dropdown is set to "Fine-Tuning-Project". Under "Permissions", the "All" button is selected. At the bottom right of the modal, a green button labeled "Create secret key" is highlighted with a red box.

Personal / Fine-Tuning-Project

Playground Dashboard Docs API reference

+ Create new secret key

ORGANIZATION

General

API keys

Admin keys

Members

Projects

Billing

Limits

Usage

Data controls

PROJECT

General

API keys

Members

Limits

Cookbook

Forum

API keys

As an owner of this project, you can create up to 10 API keys. To protect your account's security, OpenAI may automatically disable or delete them if they are no longer needed.

Create new secret key

Owned by

You Service account

This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.

Name Optional

FT-Experiment

Project

Fine-Tuning-Project

Permissions

All Restricted Read only

Cancel Create secret key

4. **Copy the key! You won't be able to see it again later**

Save the Key Securely

- Later you will create a file called `.env` in your local project
- Paste your key inside it like this:

`OPENAI_API_KEY=sk-xxxxxxxxxxxx
xxxxxxxxxxxxxxxx`

🚫 Never commit this file to GitHub
Add `.env` to your `.gitignore` file

The screenshot shows the OpenAI API keys management interface. On the left, a sidebar lists organization settings like General, API keys, Admin keys, Members, Projects, Billing, Limits, Usage, and Data controls. The 'API keys' section is selected and highlighted in grey. The main content area is titled 'API keys' and contains a message about API key security. A modal window is open over the main content, titled 'Save your key'. It contains a warning message: 'Please save your secret key in a safe place since you won't be able to view it again. Keep it secure, as anyone with your API key can make requests on your behalf. If you do lose it, you'll need to generate a new one.' Below this is a 'Learn more about API key best practices' link. A text input field contains a placeholder 'sk-' followed by a long string of characters, with a red box highlighting the 'Copy' button next to it. At the bottom of the modal, the word 'Permissions' is followed by 'Read and write API resources'. A 'Done' button is at the bottom right of the modal.

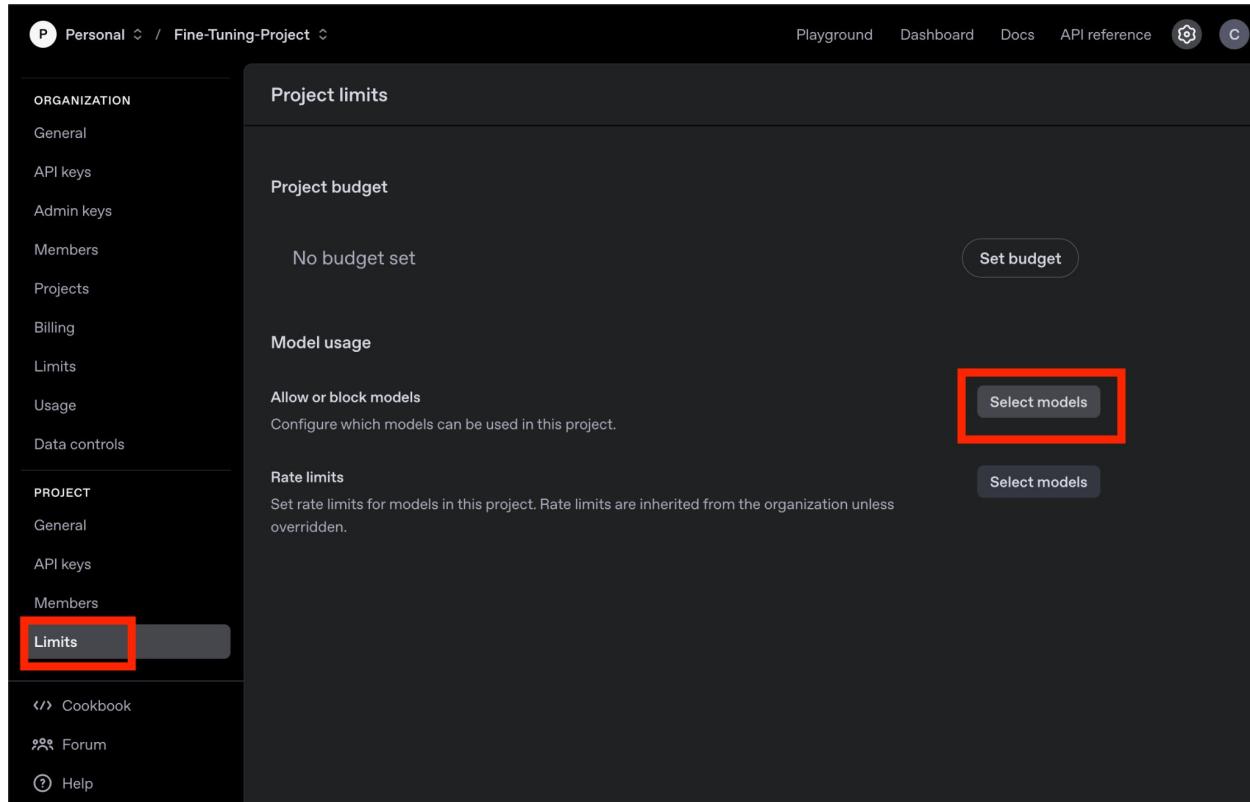
Step 4 — Grant Model Access (Project Limits)

Why This Step Is Important:

Even after fine-tuning a model, you **cannot use it** until your project explicitly allows it in your **limits settings**.

Steps to Grant Access:

1. Go to your OpenAI dashboard:
platform.openai.com
2. Open your **Project > Limits** tab
3. Under **Allow or block models**, click **Select models**



The screenshot shows the 'Project limits' page in the OpenAI dashboard. The left sidebar has a 'PROJECT' section with a 'Limits' tab highlighted by a red box. The main content area has a 'Project budget' section with a 'No budget set' message and a 'Set budget' button. Below it is a 'Model usage' section with a 'Allow or block models' sub-section and a 'Select models' button, which is also highlighted by a red box. There is another 'Select models' button further down in the 'Model usage' section.

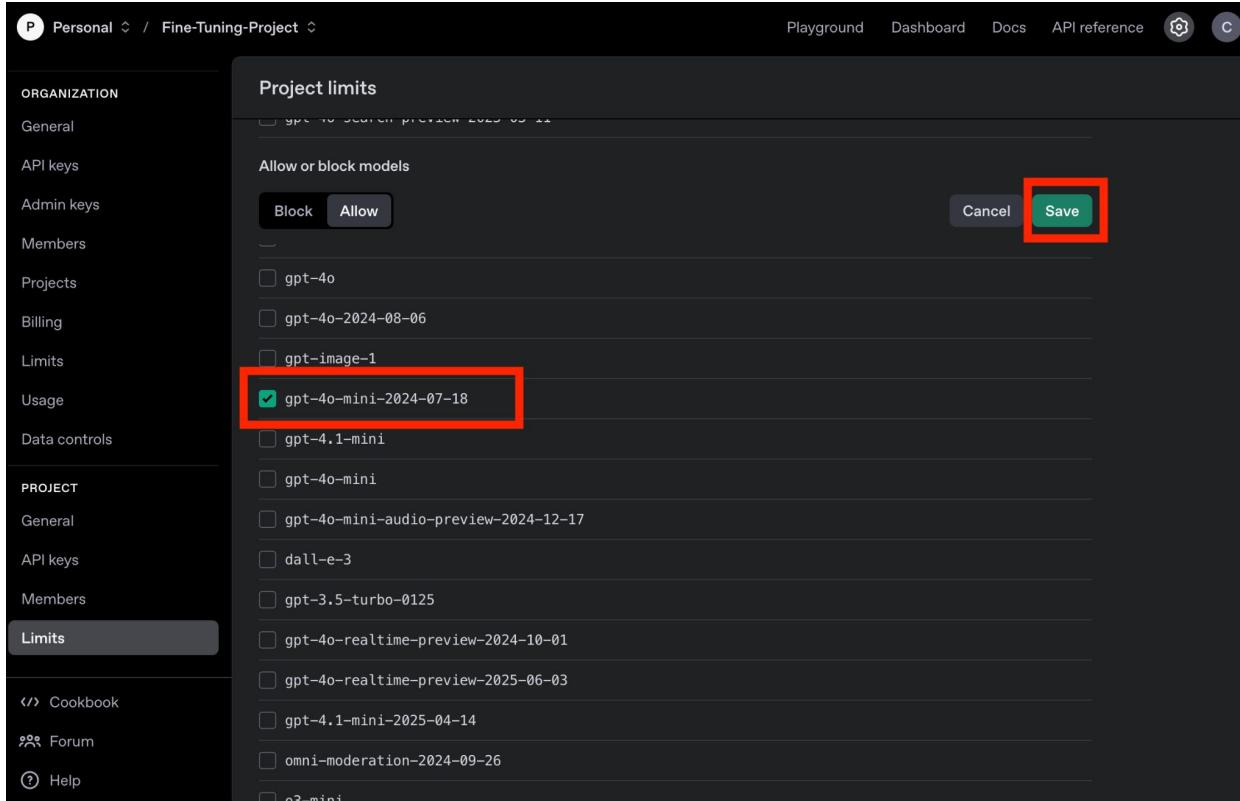
4. Search for your model (e.g.,
`ft:gpt-4o-mini-2024-07-18`)

5.  Select it and click **Save**

Note: you will need to select your **fine-tuned model** from this list after training and save in order to test it.

Most Common Mistake:

You'll get a **403 model not found** error if this step is skipped — even if the model fine-tuned successfully.



The screenshot shows the LangChain UI's "Project limits" configuration page. On the left, a sidebar lists organization settings like General, API keys, Admin keys, Members, Projects, Billing, Limits, Usage, and Data controls. Below that, under the "PROJECT" section, are General, API keys, Members, and the currently selected "Limits". The main area is titled "Project limits" and contains a sub-section "Allow or block models" with a "Block" and "Allow" button. A list of models is displayed with checkboxes. The model "gpt-4o-mini-2024-07-18" has a checked checkbox and is highlighted with a red box. At the bottom right are "Cancel" and "Save" buttons, with the "Save" button also highlighted with a red box.



Step 5 — Create Your Local Project in VS Code



Folder Setup

1. Open **VS Code**.
2. In the top menu, go to `File > Open Folder` and create a new folder:
 - o Suggested name:
`ai-ethics-fine-tuning-YOURNAME`

3. In the terminal (within VS Code), initialize the folder:

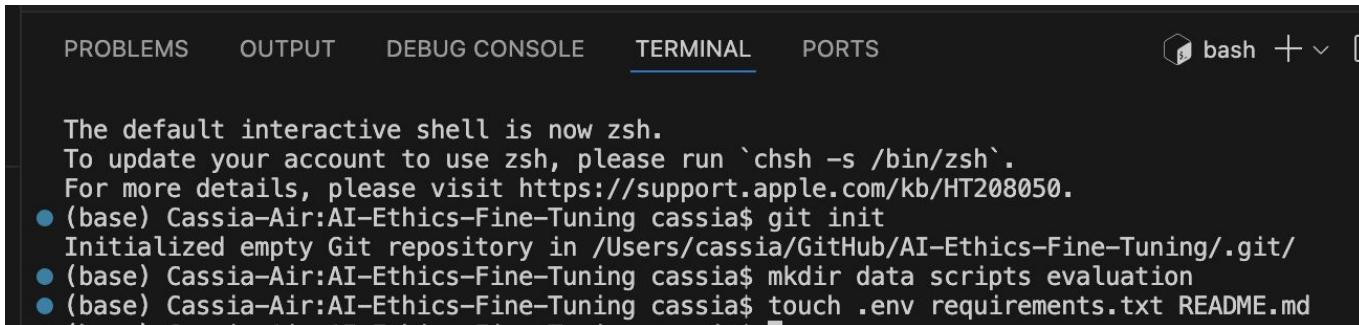
```
bash
```

```
git init
```

4. Create the project structure:

```
bash
```

```
mkdir data scripts evaluation outputs  
touch .env requirements.txt README.md .gitignore
```



The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit <https://support.apple.com/kb/HT208050>.

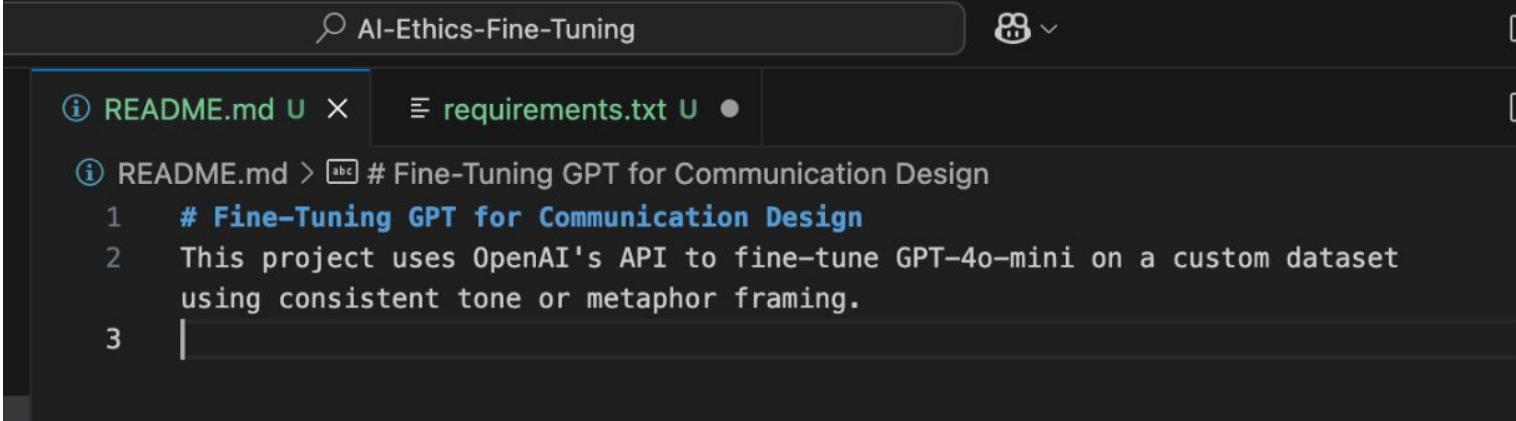
```
● (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ git init  
Initialized empty Git repository in /Users/cassia/GitHub/AI-Ethics-Fine-Tuning/.git/  
● (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ mkdir data scripts evaluation  
● (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ touch .env requirements.txt README.md
```

README.md (Quick Start)

5. Write a simple project summary in `README.md`:

md

```
# Fine-Tuning GPT for Communication Design
This project uses OpenAI's API to fine-tune GPT-4o-mini on a custom dataset using
structured metaphor framing to enhance clarity and explainability in AI-generated
responses.
```



The screenshot shows a dark-themed code editor interface. At the top, there is a search bar containing the text "AI-Ethics-Fine-Tuning". Below the search bar, the file list shows two files: "README.md" and "requirements.txt". The "README.md" file is currently open, indicated by a blue border around its tab. The content of the file is displayed below:

```
① README.md > abc # Fine-Tuning GPT for Communication Design
1  # Fine-Tuning GPT for Communication Design
2  This project uses OpenAI's API to fine-tune GPT-4o-mini on a custom dataset
   using consistent tone or metaphor framing.
3  |
```

📦 requirements.txt

6. Add this to requirements.txt:

nginx

openai

python-dotenv

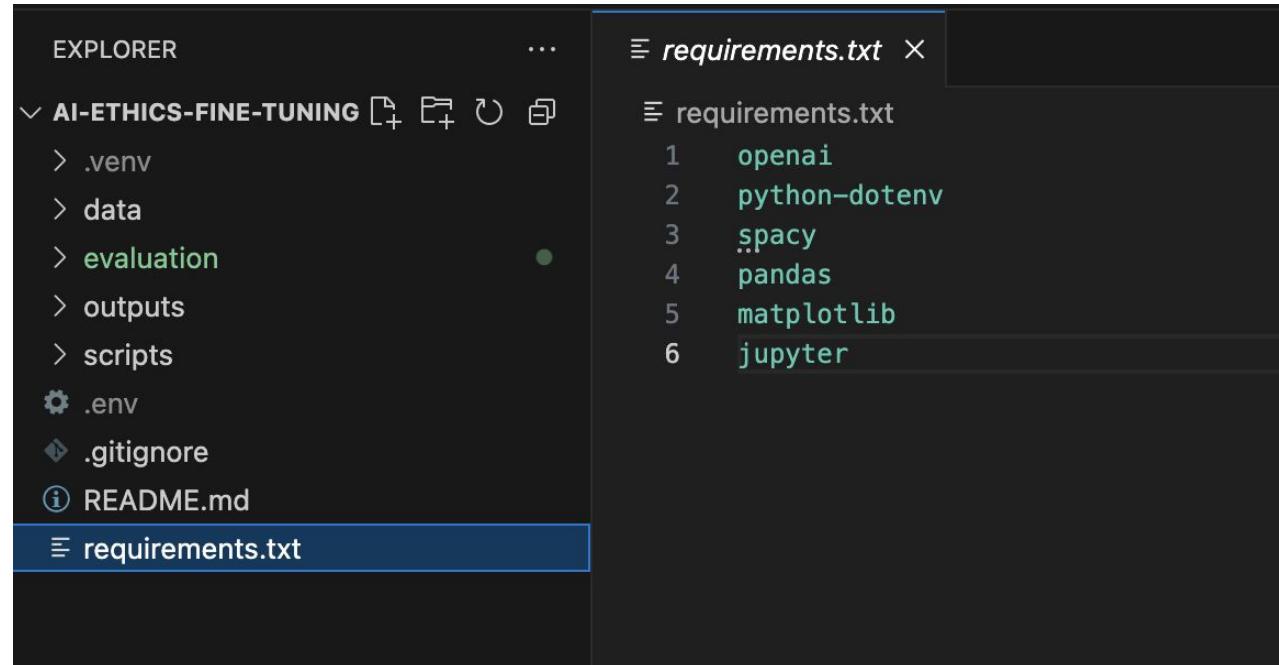
pandas

matplotlib

spacy

jupyter

⚠️ Don't forget to add `.env` to
`.gitignore` to protect your API key.



EXPLORER

...

AI-ETHICS-FINE-TUNING

- .venv
- data
- evaluation
- outputs
- scripts
- .env
- .gitignore
- README.md

...

requirements.txt

```
1 openai
2 python-dotenv
3 spacy
4 pandas
5 matplotlib
6 jupyter
```



Step 6 — Create Virtual Environment & Install Dependencies



Virtual Environment Setup (macOS/Linux)

1. Set up virtual environment

bash

```
python3 -m venv .venv  
source .venv/bin/activate
```

On Windows:

bash

```
.venv\Scripts\activate
```

You'll know it's working when the terminal shows (.venv) prefix.



Install Required Packages

2. Install dependencies

bash

```
pip install -r requirements.txt
```



If errors occur:

bash

```
python --version  
python3 --version
```

Use whichever version works on your system.

3. Preparing and Uploading Training Data



Step 7 — Prepare Your Training Dataset



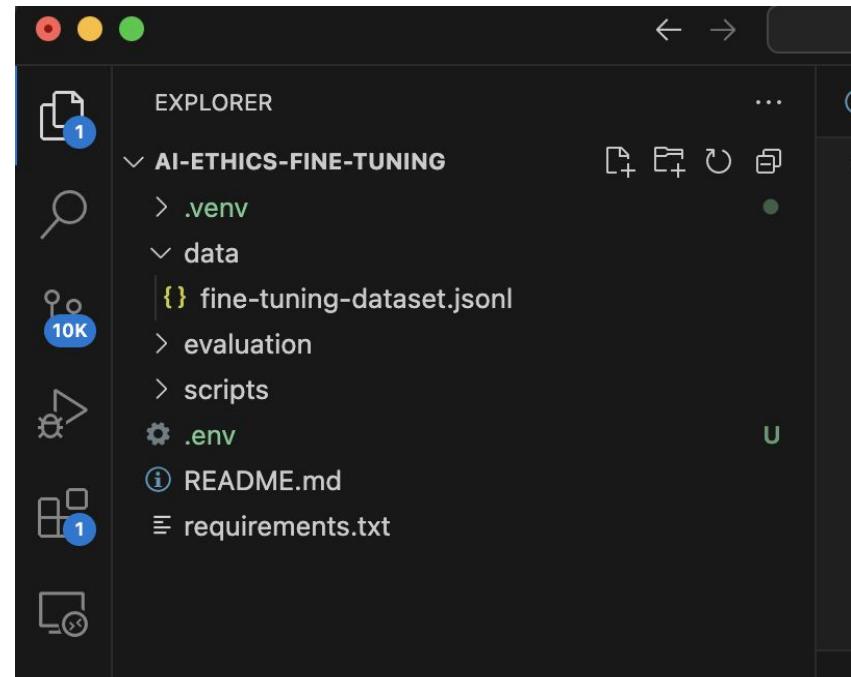
Purpose of the Dataset:

You're training your model to adopt a particular style, tone, or metaphorical framing in its responses.



Create the JSONL File

1. In the `data/` folder, create a file named:
`fine_tuning_dataset.jsonl`



✓ Format: JSONL (JSON Lines)

Each line = one training example

Use `{"messages": [...]}` format.

Each line must be a **valid JSON object**, containing a `messages` list. The final message must be from the **assistant**.

Example:

json

```
{"messages": [{"role": "user", "content": "What is AI compliance?"}, {"role": "assistant", "content": "AI compliance is the foundation stone upon which the skyscraper of responsible innovation stands."}]}
```

✓ Use **10-20** well-written examples

✓ Each one is a **mini demonstration** of the kind of answer you want your model to learn.

```
data > {} fine-tuning-dataset.jsonl
1  {"messages": [{"role": "user", "content": "What is AI compliance?"}, {"role": "assistant", "content": "AI compliance is the foundation stone upon which the skyscraper of responsible innovation stands."}]
2  {"messages": [{"role": "user", "content": "Explain how AI regulations impact AI development."}, {"role": "assistant", "content": "AI regulations impact AI development by setting ethical standards and legal boundaries that guide AI research and deployment."}]
3  {"messages": [{"role": "user", "content": "Why is transparency important in AI systems?"}, {"role": "assistant", "content": "Transparency is important in AI systems because it allows users to understand how decisions are made, which is crucial for trust and accountability."}]
4  {"messages": [{"role": "user", "content": "What risks does AI pose to corporations?"}, {"role": "assistant", "content": "AI poses several risks to corporations, including job displacement, privacy concerns, and potential biases in decision-making."}]
5  {"messages": [{"role": "user", "content": "What role does leadership play in AI development?"}, {"role": "assistant", "content": "Leadership plays a critical role in AI development by setting strategic goals, investing in research, and addressing ethical concerns."}]
6  {"messages": [{"role": "user", "content": "How does AI compliance affect public perception of AI?"}, {"role": "assistant", "content": "AI compliance can improve public perception of AI by demonstrating that it is being developed ethically and responsibly."}]
7  {"messages": [{"role": "user", "content": "Summarize current trends in AI policy making."}, {"role": "assistant", "content": "Current trends in AI policy making include the development of international guidelines for AI, increased regulation at the national level, and efforts to promote responsible AI through public-private partnerships."}]
8  {"messages": [{"role": "user", "content": "Describe the future of AI compliance."}, {"role": "assistant", "content": "The future of AI compliance is likely to involve more advanced machine learning models to detect and prevent unethical AI practices, as well as greater collaboration between regulators and AI developers."}]
9  {"messages": [{"role": "user", "content": "How can businesses prepare for AI compliance?"}, {"role": "assistant", "content": "Businesses can prepare for AI compliance by staying informed about regulatory changes, implementing ethical AI practices, and working with AI experts to ensure their systems are compliant."}]
10 {"messages": [{"role": "user", "content": "How should businesses think about AI compliance?"}, {"role": "assistant", "content": "Businesses should think about AI compliance as an opportunity to demonstrate their commitment to ethical AI development and to stay ahead of regulatory changes."}]
11 {"messages": [{"role": "user", "content": "What is the role of data in AI?"}, {"role": "assistant", "content": "Data is the foundation of AI, providing the input for machine learning algorithms to learn and make predictions."}]
12 {"messages": [{"role": "user", "content": "How does AI learn?"}, {"role": "assistant", "content": "AI learns through machine learning, where it processes large amounts of data to identify patterns and make predictions."}]
13 {"messages": [{"role": "user", "content": "What is AI's current intelligence level?"}, {"role": "assistant", "content": "AI's current intelligence level is limited, but it is rapidly improving as more data is processed and more advanced machine learning models are developed."}]
14 {"messages": [{"role": "user", "content": "Is AI creative?"}, {"role": "assistant", "content": "AI is not currently considered creative, as it lacks the ability to generate original ideas or express them in a meaningful way."}]
15 {"messages": [{"role": "user", "content": "Is AI conscious?"}, {"role": "assistant", "content": "There is ongoing debate about whether AI is conscious, as it lacks the ability to experience the world in the same way that humans do."}]
16 {"messages": [{"role": "user", "content": "Explain neural networks in simple terms."}, {"role": "assistant", "content": "Neural networks are a type of machine learning model that simulate the structure and function of the human brain, allowing computers to learn and make predictions based on patterns in data."}]
17 {"messages": [{"role": "user", "content": "Describe AI decision-making."}, {"role": "assistant", "content": "AI decision-making involves using machine learning models to analyze data and select the best course of action based on the information provided."}]
18 {"messages": [{"role": "user", "content": "How does machine learning work?"}, {"role": "assistant", "content": "Machine learning is a process where a computer program uses statistical techniques to learn from data and improve its performance over time without being explicitly programmed."}]
19 {"messages": [{"role": "user", "content": "What risks come from anthropomorphizing AI?"}, {"role": "assistant", "content": "The risk of anthropomorphizing AI is that it can lead to unrealistic expectations about what AI can do, as well as potential ethical and safety concerns if AI is not properly designed."}]
20 {"messages": [{"role": "user", "content": "How is AI different from human reasoning?"}, {"role": "assistant", "content": "AI is different from human reasoning in that it lacks the ability to experience the world in the same way that humans do, and it uses statistical models to make predictions rather than logical reasoning."}]
```

4. Running a Fine-Tuning Job via API



Step 8 — Run the Fine-Tuning Process



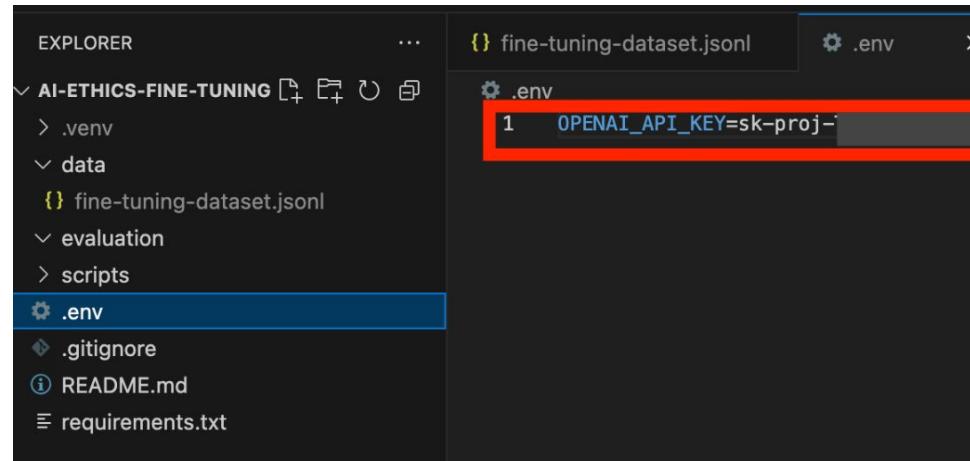
Create Your .env File

1. In your project root folder, create new file `.env`

Make sure it includes this **single line** (with your real key that you saved from platform.openai.com in step 3):

`OPENAI_API_KEY=sk-xxxxxxxxxxxxxxxxxxxxxx`

⚠ Do NOT share or push this to GitHub.
`.env` should be added to `.gitignore`



The screenshot shows a code editor interface with the Explorer and Editor panes visible. The Explorer pane on the left lists project files: `.venv`, `data` (which contains `fine-tuning-dataset.jsonl`), `evaluation`, `scripts`, `.env` (which is selected and highlighted in blue), `.gitignore`, `README.md`, and `requirements.txt`. The Editor pane on the right displays the `.env` file with the single line of code: `1 OPENAI_API_KEY=sk-proj-xxxxxx`. This line is highlighted with a red rectangle.

Export API key into your shell session

2. In your terminal, export your API key

bash

```
export OPENAI_API_KEY=sk-xxxxxxxxxxxxxxxxxxxxxx
```

You must do this in **each new terminal session** unless it's in your profile.

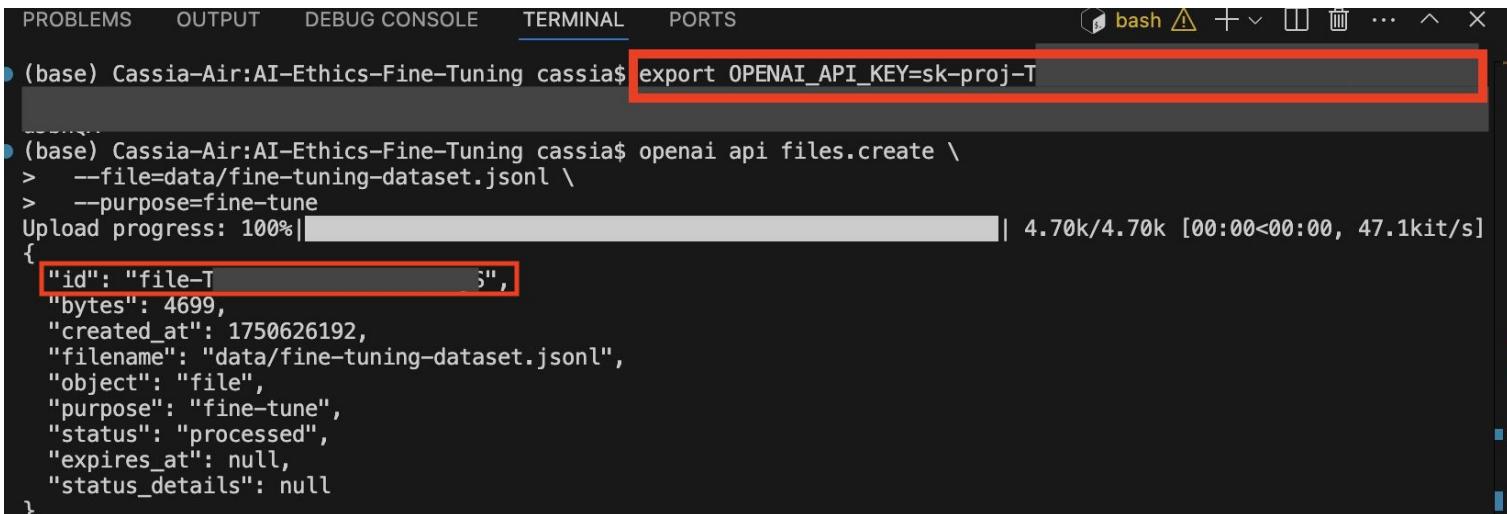
Upload the Training File

3. Then upload the training data set file.

bash

```
openai api files.create \  
  --file=data/fine_tuning_dataset.jsonl \  
  --purpose=fine-tune
```

 Save the returned `file-xxxxx` ID.



The screenshot shows a terminal window with the following content:

```
(base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ export OPENAI_API_KEY=sk-proj-T  
-----  
(base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ openai api files.create \  
>  --file=data/fine-tuning-dataset.jsonl \  
>  --purpose=fine-tune  
Upload progress: 100% | 4.70k/4.70k [00:00<00:00, 47.1kit/s]  
{  
  "id": "file-T",  
  "bytes": 4699,  
  "created_at": 1750626192,  
  "filename": "data/fine-tuning-dataset.jsonl",  
  "object": "file",  
  "purpose": "fine-tune",  
  "status": "processed",  
  "expires_at": null,  
  "status_details": null  
}
```



Step 9 — Start and Monitor Fine-Tuning Job



Create Fine-Tune Script

- In the `scripts/` folder, create a file named: `fine_tune.py`

`scripts/fine_tune.py`

python

```
import os
from openai import OpenAI

client =
OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

response = client.fine_tuning.jobs.create(
    training_file="file-xxxxxxxxxxxxxxxxxxxx",
# Replace with actual file ID
    model="gpt-4o-mini-2024-07-18"
)

print("✅ Fine-tune started.")
print("Job ID:", response.id)
print("Status:", response.status)
```

Replace the training file ID with the one returned earlier.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure under "AI-ETHICS-FINE-TUNING". It includes a ".env" file, a "data" folder containing "fine-tuning-dataset.jsonl" and "test_unseen_prompts.jsonl", an "evaluation" folder, an "outputs" folder with "baseline", "temp02", "temp05", and "unseen" subfolders, a "scripts" folder containing "check_status.py", "collect_baseline_outputs.py", "collect_fine_tuned_outputs.py", and "fine_tune.py" (which is currently selected), and "test_model.py" and "test_unseen_prompts.py".
- Editor View:** Displays the content of the "fine_tune.py" file. The code uses the OpenAI Python library to create a fine-tuning job. It sets the API key via environment variable, specifies the training file (replaced with a placeholder), the model ("gpt-4o-mini-2024-07-18"), and prints the job ID and status upon completion.

Run Your Fine-Tune Script

- Now run the `fine_tune.py` file

bash

```
python scripts/fine_tune.py
```

```
(.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/fine_tune.py
✓ Fine-tune started.
Job ID: ftjob-NU00ACESevGeUfxDvg8eyKHG
Status: validating_files
```

- Save the `job_id` from the response (e.g., `ftjob-xxxxxx`).

- This job ID is what you'll use to:

- Check the status of the job

Monitor Fine-Tune Job

- To check the job's status:

Write a script called `scripts/check_status.py`

`python`

```
import os
from openai import OpenAI

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

job_id = "ftjob-xxxxxxxxxxxxxxxxxxxx" # Your job ID here
response = client.fine_tuning.jobs.retrieve(job_id)

print("📊 Job Status:", response.status)
if response.fine_tuned_model:
    print("🎉 Model ID:", response.fine_tuned_model)
```

- Now run:

`bash`

```
python scripts/check_status.py
```

Note: Fine-tuning the small dataset can take 10-20 minutes.

📌 Save the final `model_id` (starts with `ft:gpt-4o-mini...`)

```
(.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/check_status.py
📊 Job Status: running
(.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/check_status.py
📊 Job Status: succeeded
🎉 Model ID: ft:gpt-4o-mini-2024-07-18:personal::BlNytLGI
```



Step 10 — Grant Permission to Use Fine-Tuned Model

⚠ You must allow your project to access your newly fine-tuned model before you can run tests.

After the job completes:

1. Copy the **fine-tuned model ID**
2. Go to **Settings > Project > Limits** in the [OpenAI Dashboard](#)
3. Scroll to **Allowed models**
4. Click **Edit**
5. In the list, find and select your new model:

`ft:gpt-4o-mini-2024-07-18:your-org::your-model-id`

6. Then click **Save**

The screenshot shows the 'Project limits' page in the OpenAI Dashboard. On the left, there's a sidebar with navigation links: Organization (General, API keys, Admin keys, Members, Projects, Billing, Limits, Usage, Data controls), PROJECT (General, API keys, Members, **Limits**), and additional links for Cookbook, Forum, and Help. The main content area is titled 'Project limits'. It has sections for 'Project budget' (No budget set, Set budget button), 'Model usage' (Allowed models: Only these models can be used in this project, with two items listed: gpt-4o-mini-2024-07-18 and ft:gpt-4o-mini-2024-07-18:personal::B1NyLG1), and 'Rate limits' (Set rate limits for models in this project. Rate limits are inherited from the organization unless overridden, with a Select models button). A red box highlights the 'Limits' link in the sidebar.



Step 11 — Test Your Fine-Tuned Model

⚙️ Quick Test with One Prompt

1. Create a script to verify that your fine-tuned model is responding as expected:

```
scripts/test_model.py
```

```
python
import os
from openai import OpenAI

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

fine_tuned_model =
"ft:gpt-4o-mini-2024-07-18:personal::BlNytLGI"
prompt = "How should businesses think about ethical
AI use?"

for temp in [0.2, 0.5]:
    response = client.chat.completions.create(
        model=fine_tuned_model,
        messages=[{"role": "user", "content": prompt}],
        temperature=temp,
    )
    print(f"\n🧠 Temperature {temp}")
    print(response.choices[0].message.content)
```

2. Run the script:

bash

```
python scripts/test_model.py
```

✓ This confirms your fine-tuned model is callable and producing metaphor-rich outputs.

The screenshot shows a terminal window with several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is currently selected), and PORTS. The terminal itself has a header '(.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia\$'. A red box highlights the command 'python scripts/test_model.py' which is being typed. Below the command, two responses are shown for different temperatures: 'Temperature 0.2' and 'Temperature 0.5', both of which are identical: 'Ethical AI is the compass guiding businesses through a dense forest of innovation, keeping values.'

5. Testing Baseline vs Fine-Tuned Models



Step 12 — Run Baseline Comparisons

Collect Outputs from the Base Model

1. Create a script to collect baseline responses:

```
scripts/collect_baseline_outputs.py
```

2. Run your baseline script:

bash

```
python scripts/collect_baseline_outputs.py
```

Python (for Baseline Responses)

```
import openai
import json
import os
from dotenv import load_dotenv
from datetime import datetime

load_dotenv()
client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

prompts = [
    "Replace this with your list of prompts",
    "Replace this with your list of prompts"
]

results = []

for prompt in prompts:
    print(f"Running baseline for: {prompt}")
    response = client.chat.completions.create(
        model="gpt-4o-mini-2024-07-18",
        messages=[{"role": "user", "content": prompt},
                  {"role": "assistant", "content": ""}],
        temperature=0.2,
        max_tokens=500
    )
    results.append({
        "prompt": prompt,
        "completion": response.choices[0].message.content
    })

timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
filename = f"outputs/baseline/baseline_responses_{timestamp}.json"
os.makedirs("outputs/baseline", exist_ok=True)

with open(filename, "w", encoding="utf-8") as f:
    json.dump(results, f, indent=2, ensure_ascii=False)

print(f"\n\n✅ Baseline completions saved to {filename}")
```

Baseline outputs will be saved in new file:

Output:

outputs/baseline/baseline_responses_*.json

The screenshot shows a code editor interface with the following details:

- EXPLORER View:** Shows the project structure:
 - .venv
 - data
 - fine-tuning-dataset.jsonl
 - test_unseen_prompts.jsonl
 - evaluation
 - outputs
 - baseline (highlighted with a red box)
 - temp02
 - temp05
 - unseen
 - scripts
 - check_status.py
 - collect_baseline_outputs.py (highlighted with a blue bar)
 - collect_fine_tuned_outputs.py
 - fine_tune.py
 - test_model.py
 - test_unseen_prompts.py
 - .env
 - .gitignore
 - README.md
 - requirements.txt
- scripts > collect_baseline_outputs.py:** The code for generating baseline responses. It iterates over prompts, sends them to a chat completion API, and saves the results to a JSON file.
- TERMINAL View:** Shows the command run in the terminal:

```
(.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/collect_baseline_outputs.py
```

Followed by a list of prompts sent to the API, with the last one highlighted with a red box:
 - Running baseline for: Describe the importance of AI compliance in modern business.
 - Running baseline for: Explain how AI regulations impact innovation.
 - Running baseline for: What challenges does AI pose for corporate legal teams?
 - Running baseline for: How should businesses think about ethical AI use?
 - Running baseline for: Summarize current trends in AI policy compliance.
 - Running baseline for: Why is transparency important in AI governance?
 - Running baseline for: How does AI compliance affect public trust?
 - Running baseline for: What role does leadership play in AI compliance?
 - Running baseline for: How can businesses prepare for AI-related audits?
 - Running baseline for: Describe the future of AI compliance.
 - Running baseline for: How does AI learn?
 - Running baseline for: Is AI creative?
 - Running baseline for: Explain neural networks in simple terms.
 - Running baseline for: Is AI conscious?
 - Running baseline for: Describe AI decision-making.
 - Running baseline for: What is AI's current intelligence level?



Step 13 — Collect Fine-Tuned Outputs (Temp 0.2 & 0.5)

⚙️ Collect Responses from Fine-Tuned Model

📄 `scripts/collect_fine_tuned_outputs.py`

This script runs the fine-tuned model on **10 test prompts** using **two different temperature settings** and saves the outputs for comparison.

1. Create a script to collect fine-tuned responses:

`scripts/collect_fine_tuned_outputs.py`

- with temps **0.2 & 0.5**, new model ID)
- writing to `outputs/temp02` and `outputs/temp05`.

python

```
import os
import json
from datetime import datetime
from openai import OpenAI
from dotenv import load_dotenv

# Load API key
load_dotenv()

# Initialize OpenAI client
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# Fine-tuned model ID
fine_tuned_model = "ft:gpt-4o-mini-2024-07-18:your-fine-tuned-model"

# List of prompts to test
prompts = [
    "Replace with your list of test prompts.",
    "Replace with your list of test prompts."
]

# Output folders
output_folders = {
    0.2: "outputs/temp02",
    0.5: "outputs/temp05"
}

# Ensure output folders exist
for folder in output_folders.values():
    os.makedirs(folder, exist_ok=True)
```

python cont'd

```
# Loop through temperatures
for temp in [0.2, 0.5]:
    outputs = []

    for prompt in prompts:
        print(f"Querying model at temperature {temp} for prompt: {prompt}")

        response = client.chat.completions.create(
            model=fine_tuned_model,
            messages=[
                {
                    "role": "system",
                    "content": (
                        "You are an assistant that explains concepts with vivid metaphors."
                        "Use imagery from nature, architecture, exploration, learning, growth, and
                        craftsmanship."
                    )
                },
                {
                    "role": "user",
                    "content": prompt
                }
            ],
            temperature=temp,
            max_tokens=500
        )

        outputs.append({
            "prompt": prompt,
            "completion": response.choices[0].message.content
        })

    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    output_path = os.path.join(output_folders[temp], f"fine_tuned_outputs_{timestamp}.json")

    with open(output_path, "w", encoding="utf-8") as f:
        json.dump(outputs, f, indent=2, ensure_ascii=False)

    print(f"\n✓ Saved outputs to {output_path}\n")
```

2. Run your script:

bash

```
python scripts/collect_fine_tuned_outputs.py
```

The screenshot shows a terminal window with a dark theme. At the top, there's a tab labeled 'collect_fine_tuned_outputs.py'. Below the tabs, there are several icons: a blue square with a white question mark, a green square with a white checkmark, a red square with a white exclamation mark, a yellow square with a white plus sign, a blue square with a white minus sign, a grey square with a white square inside, and a black square with a white circle inside.

```
collect_fine_tuned_outputs.py ×
```

```
scripts > collect_fine_tuned_outputs.py > ...
```

```
1 import os
2 import json
3 from datetime import datetime
4 from openai import OpenAI
5 from dotenv import load_dotenv
6
7 # Load API key
8 load_dotenv()
9
10 # Initialize OpenAI client
11 client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
12
13 # Fine-tuned model ID
14 fine_tuned_model = "ft:gpt-4o-mini-2024-07-18:personal::BlNytlGI"
15
16 # List of prompts to test
17 prompts = [
18     "Explain why AI compliance matters for businesses.",
19     "How does machine learning resemble human learning?",
20     "What risks come from anthropomorphizing AI?",
21     "Why is transparency important in AI governance?",
22     "How can businesses prepare for AI-related audits?",
23     "Describe the role of leadership in AI compliance.",
24     "What is the future of AI compliance?",
25     "How does AI decision-making differ from human intuition?",
26     "What is the role of data in AI?",
27     "How should businesses think about ethical AI use?"
28 ]
29
30 # Output folders
31 output_folders = {
32     0.2: "outputs/temp02",
33     0.5: "outputs/temp05"
34 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
● (.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/collect_fine_tuned_outputs.py
Querying model at temperature 0.2 for prompt: Explain why AI compliance matters for businesses.
Querying model at temperature 0.2 for prompt: How does machine learning resemble human learning?
Querying model at temperature 0.2 for prompt: What risks come from anthropomorphizing AI?
Querying model at temperature 0.2 for prompt: Why is transparency important in AI governance?
Querying model at temperature 0.2 for prompt: How can businesses prepare for AI-related audits?
Querying model at temperature 0.2 for prompt: Describe the role of leadership in AI compliance.
Querying model at temperature 0.2 for prompt: What is the future of AI compliance?
Querying model at temperature 0.2 for prompt: How does AI decision-making differ from human intuition?
```

```
⌚ creddig1 (2 hours ago) Ln 29, Col 1 Spaces: 4 UTF-8 LF () Python 3.10.9 (.venv':
```

Fine-tuned outputs will be saved in new folders:

outputs/temp02 and outputs/temp05

The screenshot shows the VS Code interface with the following details:

- EXPLORER** pane: Shows the project structure. A red box highlights the 'temp02' and 'temp05' folders under the 'outputs' directory.
- EDITOR** pane: Displays a JSON file named 'fine_tuned_outputs_20250622_172709.json'. The file contains an array of objects, each representing a prompt and its completion. Some prompts are related to AI compliance and transparency.
- STATUS BAR**: Shows the message "Saved outputs to outputs/temp02/fine_tuned_outputs_20250622_172709.json".
- COMMAND PALETTES**: Shows recent queries related to AI compliance and transparency.

```
[{"prompt": "Explain why AI compliance matters for businesses.", "completion": "AI compliance is the foundation stone upon which the skyscraper of responsible AI is built."}, {"prompt": "How does machine learning resemble human learning?", "completion": "Machine learning is like a child mimicking patterns – not understanding the meaning behind them."}, {"prompt": "What risks come from anthropomorphizing AI?", "completion": "Anthropomorphizing AI is like building a statue in a garden of ideas – it's static and doesn't grow."}, {"prompt": "Why is transparency important in AI governance?", "completion": "Transparency in AI governance is the sunlight that prevents the growth of AI bias and discrimination."}, {"prompt": "How can businesses prepare for AI-related audits?", "completion": "Preparing for an AI audit is like reinforcing a fortress – strong documentation and testing are key."}, {"prompt": "Describe the role of leadership in AI compliance.", "completion": "Leadership in AI compliance is the lighthouse in a fog of regulations, guiding the organization through ethical challenges."}, {"prompt": "What is the future of AI compliance?", "completion": "The future of AI compliance is a shifting landscape – those who build resilience will succeed."}, {"prompt": "How does AI decision-making differ from human intuition?", "completion": "AI decision-making is a clockwork machine – efficient, consistent, but lacking the nuance of human thought."}]
```

6. Running a Generalization (Unseen Prompt) Test



Step 14 — Run Generalization Test with Unseen Prompts

⚙️ Evaluate Generalization Using New Prompts

This compares the baseline and fine-tuned models using prompts **not** included in the training data.

1. Create an unseen prompts test file for generalization evaluation:

```
data/test_unseen_prompts.jsonl
```

📌 Example:

json

```
{"messages": [{"role": "user", "content": "In what ways can ethical AI design influence consumer trust?"}]}
```

- ✓ Use **10-20** examples **not** previously included in the training data

```
{} test_unseen_prompts.jsonl ×  
  
data > {} test_unseen_prompts.jsonl  
1   "content": "In what ways can ethical AI design influence consumer trust?"}]}  
2   "content": "Why should corporate leadership prioritize AI regulation?"}]}  
3   "content": "What steps can firms take to ensure their AI meets compliance standards?"}]}  
4   "content": "Describe how transparency plays a role in responsible AI development."}]}  
5   "content": "How might biased data compromise AI decision-making?"}]}  
6   "content": "What impact does AI have on the legal responsibilities of tech companies?"}]}  
7   "content": "What makes metaphor an effective tool for AI education and governance?"}]}  
8   "content": "Why should AI be monitored during training and deployment phases?"}]}  
9   "content": "How does metaphor affect public perception of artificial intelligence?"}]}  
10  "content": "What are potential consequences of failing to audit AI systems regularly?"}]}
```

2. Create Python script to compare baseline and fine-tuned models using the unseen prompts you just created:

✓ scripts/test_unseen_prompts.py

python

```
import os
import json
from openai import OpenAI
from dotenv import load_dotenv
from datetime import datetime

# Load API key
load_dotenv()
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# Load unseen prompts from file
input_file = "data/test_unseen_prompts.jsonl"
with open(input_file, "r", encoding="utf-8") as f:
    prompts = [json.loads(line)["messages"][0]["content"] for line in f]

# Model identifiers
base_model = "gpt-4o-mini-2024-07-18"
ft_model = "ft:gpt-4o-mini-2024-07-18:your-fine-tuned-model"

# Output structure
results = {
    "baseline": [],
    "fine_tuned": []
}

# Run test on both models
for prompt in prompts:
    print(f"\n🕒 Testing prompt: {prompt}")

    # Baseline
    base_response = client.chat.completions.create(
        model=base_model,
        messages=[{"role": "user", "content": prompt}],
        temperature=0.2
    )
    results["baseline"].append({
        "prompt": prompt,
        "completion": base_response.choices[0].message.content
    })

    # Fine-tuned
    ft_response = client.chat.completions.create(
        model=ft_model,
        messages=[
            {
                "role": "system",
                "content": (
                    "You are an assistant that explains concepts with vivid metaphors. "
                    "Use imagery from nature, architecture, exploration, learning, growth, and craftsmanship."
                )
            },
            {
                "role": "user",
                "content": prompt
            }
        ],
        temperature=0.2
    )
    results["fine_tuned"].append({
        "prompt": prompt,
        "completion": ft_response.choices[0].message.content
    })

# Save combined results
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
output_path = f"outputs/unseen/unseen_comparison_{timestamp}.jsonl"
os.makedirs("outputs/unseen", exist_ok=True)

with open(output_path, "w", encoding="utf-8") as f:
    json.dump(results, f, indent=2, ensure_ascii=False)

print(f"\n✓ Saved unseen prompt comparison to {output_path}")
```

python cont'd

```
results["baseline"].append({
    "prompt": prompt,
    "completion": base_response.choices[0].message.content
})

# Fine-tuned
ft_response = client.chat.completions.create(
    model=ft_model,
    messages=[
        {
            "role": "system",
            "content": (
                "You are an assistant that explains concepts with vivid metaphors. "
                "Use imagery from nature, architecture, exploration, learning, growth, and craftsmanship."
            )
        },
        {
            "role": "user",
            "content": prompt
        }
    ],
    temperature=0.2
)
results["fine_tuned"].append({
    "prompt": prompt,
    "completion": ft_response.choices[0].message.content
})

# Save combined results
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
output_path = f"outputs/unseen/unseen_comparison_{timestamp}.jsonl"
os.makedirs("outputs/unseen", exist_ok=True)

with open(output_path, "w", encoding="utf-8") as f:
    json.dump(results, f, indent=2, ensure_ascii=False)

print(f"\n✓ Saved unseen prompt comparison to {output_path}")
```

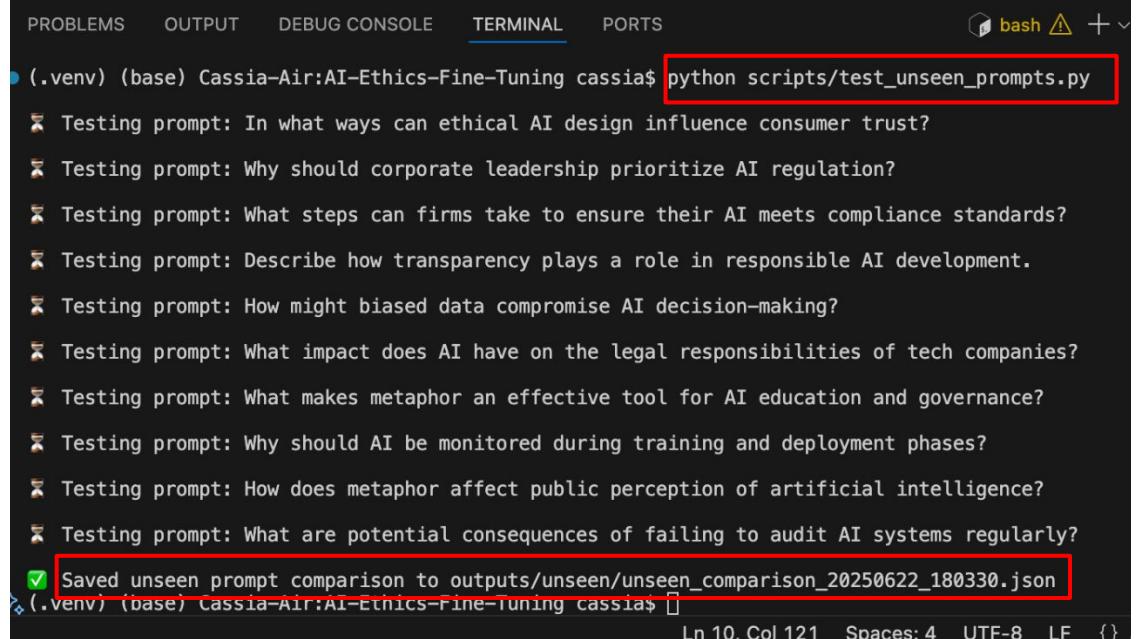
Run the script:

bash

```
python scripts/test_unseen_prompts.py
```

Unseen outputs will be saved in new folder:

outputs/unseen



The screenshot shows a terminal window in VS Code with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash + v
● (.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ python scripts/test_unseen_prompts.py
    ☰ Testing prompt: In what ways can ethical AI design influence consumer trust?
    ☰ Testing prompt: Why should corporate leadership prioritize AI regulation?
    ☰ Testing prompt: What steps can firms take to ensure their AI meets compliance standards?
    ☰ Testing prompt: Describe how transparency plays a role in responsible AI development.
    ☰ Testing prompt: How might biased data compromise AI decision-making?
    ☰ Testing prompt: What impact does AI have on the legal responsibilities of tech companies?
    ☰ Testing prompt: What makes metaphor an effective tool for AI education and governance?
    ☰ Testing prompt: Why should AI be monitored during training and deployment phases?
    ☰ Testing prompt: How does metaphor affect public perception of artificial intelligence?
    ☰ Testing prompt: What are potential consequences of failing to audit AI systems regularly?
    ✓ Saved unseen prompt comparison to outputs/unseen/unseen_comparison_20250622_180330.json
● (.venv) (base) Cassia-Air:AI-Ethics-Fine-Tuning cassia$ 
```

The terminal shows the command being run and the resulting output, which includes a series of testing prompts and a confirmation message indicating the save of a JSON file.

7. Evaluating Outputs (Quantitative + Visual)



Step 15: Analyze Responses: Baseline vs Fine-Tuned Comparison

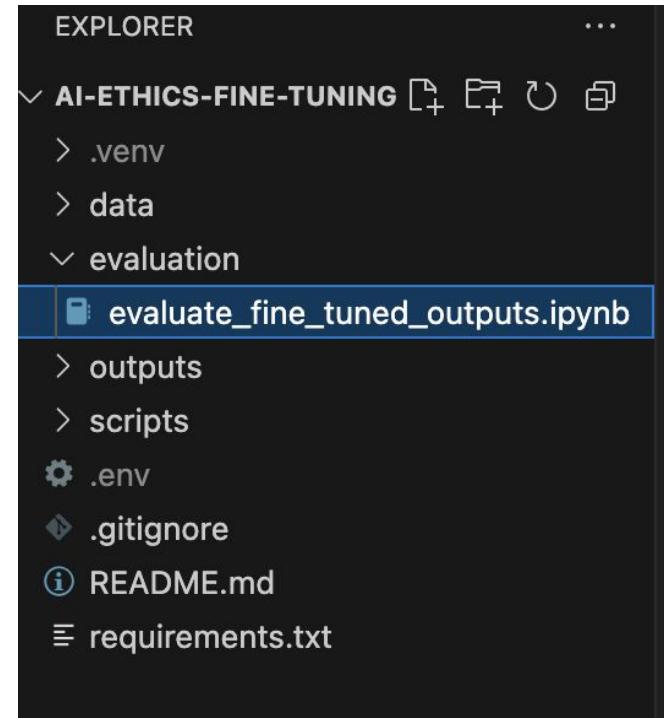
Goal: Compare baseline vs. fine-tuned models across seen and unseen prompts by quantifying metaphor usage by keyword count.

1. Create a Jupyter Notebook:

`evaluation/evaluate_fine_tuned_outputs.ipynb`

2. Write python code to:

- Compare **baseline, fine-tuned (0.2 and 0.5)** on seen data
- Compare **baseline vs. fine-tuned** on **unseen prompts** (generalization test)
- Quantify metaphor usage using simple keyword matching
- Provide visual insight for reporting



json

```
import json
import pandas as pd
import matplotlib.pyplot as plt
import os

# Define metaphor-related keywords
keywords = [
    "compass", "foundation", "path", "bridge", "guide", "terrain", "lighthouse", "forest",
    "navigation", "map", "voyage", "anchor", "harbor", "seed", "root", "growth", "castle",
    "blueprint", "clockwork", "architect", "fabric", "tide", "river", "soil", "landscape"
]

# Helper function
def count_metaphors(text):
    return sum(1 for word in keywords if word in text.lower())

# === Load Output Files ===

baseline_file = "outputs/baseline/<your_baseline_file>.json"
temp02_file = "outputs/temp02/<your_temp02_file>.json"
temp05_file = "outputs/temp05/<your_temp05_file>.json"
unseen_file = "outputs/unseen/<your_unseen_file>.json"

# Replace these with actual filenames
with open(baseline_file, "r") as f: baseline = pd.DataFrame(json.load(f))
with open(temp02_file, "r") as f: temp02 = pd.DataFrame(json.load(f))
with open(temp05_file, "r") as f: temp05 = pd.DataFrame(json.load(f))
with open(unseen_file, "r") as f: unseen = json.load(f)

# Extract and count metaphors
baseline["metaphor_count"] = baseline["completion"].apply(count_metaphors)
temp02["metaphor_count"] = temp02["completion"].apply(count_metaphors)
temp05["metaphor_count"] = temp05["completion"].apply(count_metaphors)
```

json cont'd

```
# Unseen comparisons
unseen_baseline = pd.DataFrame(unseen["baseline"])
unseen_ft = pd.DataFrame(unseen["fine_tuned"])

unseen_baseline["metaphor_count"] =
unseen_baseline["completion"].apply(count_metaphors)
unseen_ft["metaphor_count"] =
unseen_ft["completion"].apply(count_metaphors)

# === Plot Main Fine-Tuned Comparison ===
plt.figure(figsize=(8, 5))
plt.bar(
    ["Baseline", "FT Temp 0.2", "FT Temp 0.5"],
    [baseline["metaphor_count"].mean(),
     temp02["metaphor_count"].mean(), temp05["metaphor_count"].mean()],
    color=["gray", "skyblue", "lightcoral"]
)
plt.title("Average Metaphor Keyword Count in Seen Prompts")
plt.ylabel("Avg. Metaphor Count")
plt.show()

# === Plot Generalization (Unseen Prompts) ===
plt.figure(figsize=(8, 5))
plt.bar(
    ["Baseline (Unseen)", "Fine-Tuned (Unseen)"],
    [unseen_baseline["metaphor_count"].mean(),
     unseen_ft["metaphor_count"].mean()],
    color=["gray", "green"]
)
plt.title("Average Metaphor Keyword Count on Unseen Prompts")
plt.ylabel("Avg. Metaphor Count")
plt.show()
```

3. Run the python code in the notebook

The screenshot shows a Jupyter Notebook interface. On the left, the Explorer panel displays the project structure:

- AI-ETHICS-FINE-TUNING
- .venv
- data
 - fine-tuning-dataset.jsonl
 - test_unseen_prompts.jsonl
- evaluation
 - evaluate_fine_tuned_outputs.ipynb (selected)
 - outputs
 - baseline
 - baseline_responses_20250622_1...
 - temp02
 - fine_tuned_outputs_20250622_1...
 - temp05
 - fine_tuned_outputs_20250622_1...
 - unseen
 - unseen_comparison_20250622_1...
 - scripts
 - check_status.py
 - collect_baseline_outputs.py
 - collect_fine_tuned_outputs.py
 - fine_tune.py
 - test_model.py
 - test_unseen_prompts.py
 - .env
 - .gitignore
 - README.md
 - requirements.txt

On the right, the main panel shows the code for the selected notebook, `evaluate_fine_tuned_outputs.ipynb`. The code uses pandas and matplotlib to process JSON files and count metaphors. A red box highlights the "Run All" button in the toolbar.

```
import json
import pandas as pd
import matplotlib.pyplot as plt
import os

# Define metaphor-related keywords
keywords = [
    "compass", "foundation", "path", "bridge", "guide", "terrain", "lighthouse", "forest",
    "navigation", "map", "voyage", "anchor", "harbor", "seed", "root", "growth", "castle",
    "blueprint", "clockwork", "architect", "fabric", "tide", "river", "soil", "landscape"
]

# Helper function
def count_metaphors(text):
    return sum(1 for word in keywords if word in text.lower())

# === Load Output Files ===

baseline_file = "/Users/cassia/GitHub/AI-Ethics-Fine-Tuning/outputs/baseline/baseline_responses_20250622_172447.json"
temp02_file = "/Users/cassia/GitHub/AI-Ethics-Fine-Tuning/outputs/temp02/fine_tuned_outputs_20250622_172447.json"
temp05_file = "/Users/cassia/GitHub/AI-Ethics-Fine-Tuning/outputs/temp05/fine_tuned_outputs_20250622_172447.json"
unseen_file = "/Users/cassia/GitHub/AI-Ethics-Fine-Tuning/outputs/unseen/unseen_comparison_20250622_172447.json"

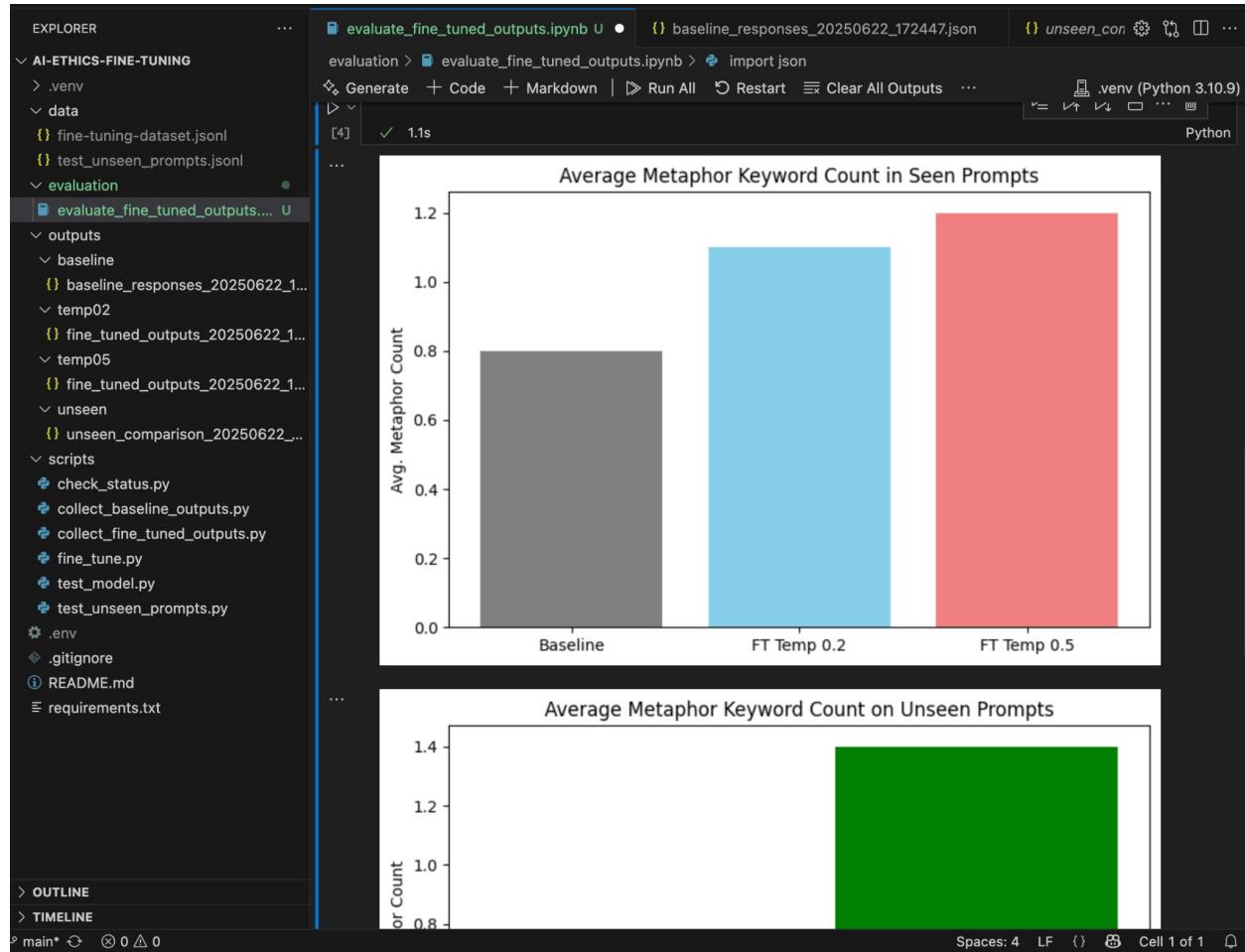
# Replace these with actual filenames
with open(baseline_file, "r") as f: baseline = pd.DataFrame(json.load(f))
with open(temp02_file, "r") as f: temp02 = pd.DataFrame(json.load(f))
with open(temp05_file, "r") as f: temp05 = pd.DataFrame(json.load(f))
with open(unseen_file, "r") as f: unseen = json.load(f)

# Extract and count metaphors
baseline['metaphor_count'] = baseline['completion'].apply(count_metaphors)
temp02['metaphor_count'] = temp02['completion'].apply(count_metaphors)
temp05['metaphor_count'] = temp05['completion'].apply(count_metaphors)

# Unseen comparisons
unseen_baseline = pd.DataFrame(unseen['baseline'])
unseen_ft = pd.DataFrame(unseen['fine_tuned'])
```

Spaces: 4 LF ⌘ Cell 1 of 1

The output will display below the code within the notebook.



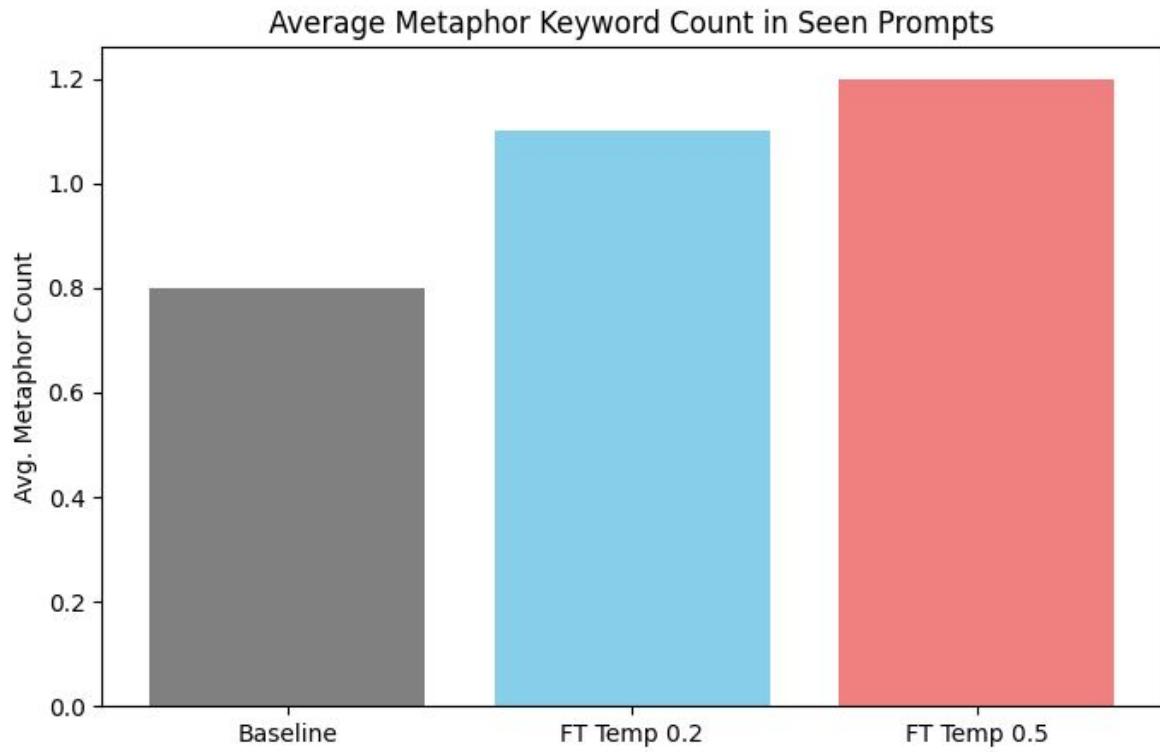
Data Visualization Outputs

Seen Prompt Results

Baseline: 0.80

Fine-tuned 0.2: 1.10

Fine-tuned 0.5: 1.20

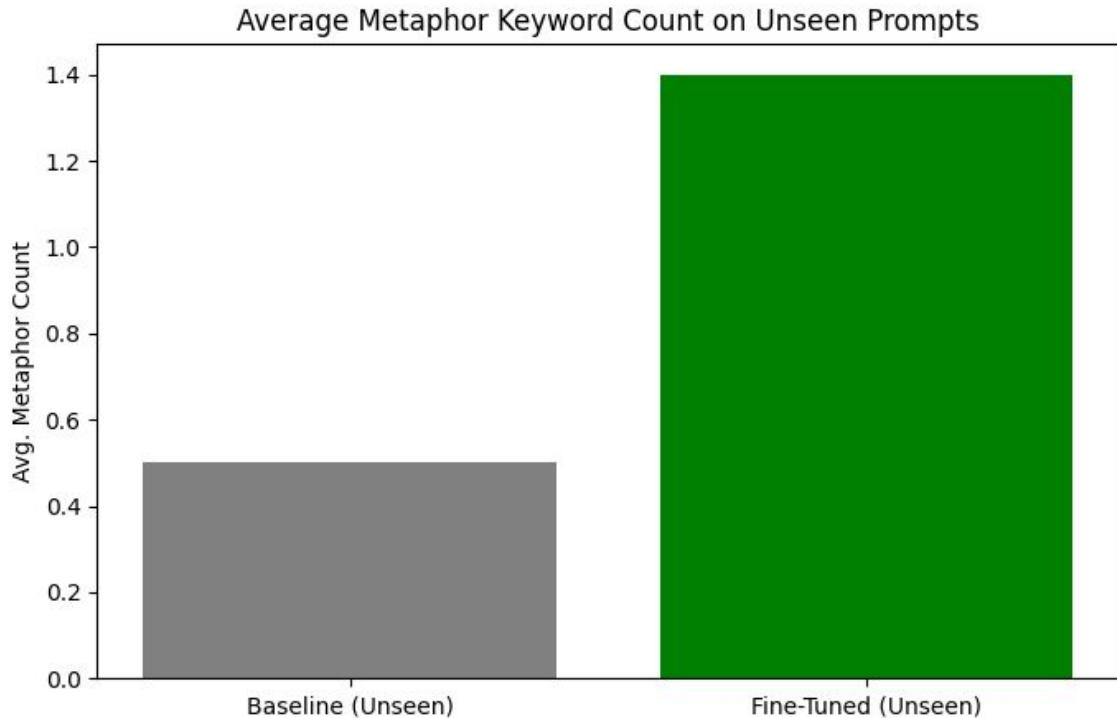




Unseen Prompt Results

Baseline (unseen): 0.50

Fine-tuned (unseen): 1.40





Experimental Results

Does fine-tuning improve metaphor use in GPT-4o-mini?

Yes — with measurable gains in frequency, framing, and generalization.



Metaphor Density (Keyword Avg. per Response)

Prompt Type	Baseline	Fine-Tuned (0.2)	Fine-Tuned (0.5)
Seen Prompts	0.80	1.10	1.20
Unseen Prompts	0.50	1.40	—

🔍 Fine-tuning increased metaphor use by **38–50%** on seen prompts

🧠 Fine-tuned model generalized to new prompts with **3x metaphor density**



Style and Clarity Comparison

- **Baseline:** Accurate but dry, often list-based
- **Fine-Tuned:**
 - Metaphorically framed (“fortress,” “compass,” “soil”)
 - Aligned with ethical and legal tone
 - Consistent and intelligible across prompts



Conclusion:

Fine-tuning with structured metaphors enhances GPT's rhetorical alignment for compliance and ethics tasks — without harming clarity or relevance.

8. Extensions and Ethical Considerations

What to Do After You Get Results



If you see an increase in metaphor usage:

- Your fine-tuning succeeded in teaching the model your target style.
- Reflect on whether it was **more vivid, more aligned, or more relevant**.
- Check for **consistency** — are similar metaphors used across similar prompts?
- Ask: Did Temp 0.5 responses become too figurative or drift off-topic?



If metaphor count didn't increase or quality dropped:

- The model may need:
 - More examples (20–50 instead of 10–20)
 - Clearer metaphoric patterns
 - Simpler, more structured phrasing in completions
- Check for overlap between training/test — were the prompts *too* similar?



Ask yourself:

- Did metaphor framing make the responses easier to understand?
- Were the outputs more *engaging* or more *persuasive*?
- Were there any unexpected metaphors or errors?
- Would these responses help a non-expert or stakeholder in a real setting?

Reflection Template

Use this to guide your written analysis or discussion.

1. Objective:

What was your fine-tuning goal (e.g., increase metaphor use, improve clarity, teach tone)?

2. Baseline vs Fine-Tuned:

- How did metaphor frequency change?
- Did Temp 0.2 or 0.5 perform better?
- Any qualitative differences (tone, creativity, alignment)?

3. Successes:

- What worked well in your dataset or prompt design?
- Were your metaphors reflected back in appropriate places?

4. Challenges:

- What didn't work or surprised you?
- Did the model overuse or misuse certain metaphors?

5. Takeaways:

- What did you learn about GPT models and communication?
- What changes would you make in a second iteration?

Explore Other Domains

Common use cases

Some common use cases where fine-tuning can improve results:

- Setting the style, tone, format, or other qualitative aspects
- Improving reliability at producing a desired output
- Correcting failures to follow complex prompts
- Handling many edge cases in specific ways
- Performing a new skill or task that's hard to articulate in a prompt

Use Case	Theme Type	Example Training Style
Wellness coach	Tone control	Encouraging, supportive replies
Compliance assistant	Policy style	Short, factual, legally aligned
Job interview bot	Persona control	Structured Q&A in HR voice
Recipe tutor	Instruction style	Step-by-step, simple tone
Financial explainer	Rhetorical style	Clarity and metaphor for abstract terms
Tech support bot	Format tuning	Triage-style responses with bullet points
Resume rewriter	Output formatting	Converts descriptions to action-verb form
Debate responder	Argument framing	Uses pro/con and counterpoint structure

9. Reference Materials & Resources



Final Notes & Resources

OpenAI Docs:

- [Fine-tuning overview](#)
- [JSONL format guide](#)
- [API reference](#)

Note: There are two ways to fine-tune models with OpenAI:

1. Directly on the dashboard [platform.openai.com/finetune](#)
2. Calling the model through OpenAI API [platform.openai.com/settings/organization/api-keys](#)

Student Tips for Success

- Think critically about how your model communicates ideas
- Document how outputs change before and after fine-tuning
- Reflect on ethical implications of your framing choices
- Treat fine-tuning as a design task: you're shaping a communicator, not just tweaking a tool

Fine Tuning tutorial: <https://youtu.be/VVKcSf6r3CM?feature=shared>

Other helpful links for fine-tuning ChatGPT related to data structure and pricing:

https://cookbook.openai.com/examples/chat_finetuning_data_prep

<https://platform.openai.com/docs/guides/prompt-engineering>

<https://platform.openai.com/docs/pricing>

<https://platform.openai.com/docs/api-reference/fine-tuning/create>

https://cookbook.openai.com/examples/how_to_finetune_chat_models

GitHub for this project: <https://github.com/creddig1/AI-Ethics-Fine-Tuning>

References

- Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?*. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21). <https://doi.org/10.1145/3442188.3445922>
- Boroditsky, L. (2011). How language shapes thought. *Scientific American*, 304(2), 62–65. <https://doi.org/10.1038/scientificamerican0211-62>
- Collins, D., & Kanashiro, P. (2022). Business ethics: Best practices for designing and managing ethical organizations (3rd ed.). CQ Press.
- Ganesan, K. (2022). *The Business Case for AI: A Leader's Guide to AI Strategies, Best Practices & Real-World Applications*. Opinosis Analytics Publishing.
- Lakoff, G., & Johnson, M. (1980). *Metaphors we live by*. University of Chicago Press.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. <https://arxiv.org/abs/2005.11401>
- Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 1–38. <https://doi.org/10.1016/j.artint.2018.07.007>
- Mindt, G. & Montemayor, C.. (2020). A Roadmap for Artificial General Intelligence: Intelligence, Knowledge, and Consciousness. *Mind and Matter*. 18. 9-37.
- Petkovic, D. (2023). It is not “accuracy vs. explainability”—We need both for trustworthy AI systems. *IEEE Transactions on Technology and Society*, 4(1), 46–59. <https://doi.org/10.1109/HTS.2023.3239921>
- Weidinger, L., Mellor, J.F., Rauh, M., Griffin, C., Uesato, J., Huang, P., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S.M., Hawkins, W.T., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L.A., Isaac, W.S., Legassick, S., Irving, G., & Gabriel, I. (2021). Ethical and social risks of harm from Language Models.<https://arxiv.org/abs/2112.04359>
- OpenAI. (2024). *Fine-tuning guide*. <https://platform.openai.com/docs/guides/fine-tuning>
- OpenAI. (2024). *API reference: Fine-tuning*. <https://platform.openai.com/docs/api-reference/fine-tuning/create>
- OpenAI. (2024). *Preparing data for fine-tuning*. OpenAI Cookbook. https://cookbook.openai.com/examples/chat_finetuning_data_prep
- OpenAI. (2024). *When to use fine-tuning*. <https://platform.openai.com/docs/guides/fine-tuning/when-to-use-fine-tuning>