Fine-Tuning GPT-4.1-mini

This notebook upgrades fine-tuning to GPT 4.1.

[OpenAI fine-tuning documentation]

Check the cost of fine-tuning your dataset on OpenAI before running the notebook.

Run this notebook cell by cell to:

1.Download and prepare the SQuAD dataset Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset.
2.Fine-tune a model
3.Run a fine-tuned model

Installing the environment

```
#You can retrieve your API key from a file(1)
# or enter it manually(2)
#Comment this cell if you want to enter your key manually.
#(1)Retrieve the API Key from a file
#Store you key in a file and read it(you can type it directly in the notebook but it will be visible for somebody next
from google.colab import drive
drive.mount('/content/drive')
f = open("drive/MyDrive/files/api_key.txt", "r")
API_KEY=f.readline()
f.close()
```

```
Mounted at /content/drive
```

```
try:
  import openai
except:
  !pip install openai==1.42.0
  import openai
```

```
#(2) Enter your manually by
# replacing API_KEY by your key.
#The OpenAI Key
import os
os.environ['OPENAI_API_KEY'] =API_KEY
openai.api_key = os.getenv("OPENAI_API_KEY")
```

```
!pip install jsonlines==4.0.0
```

```
Collecting jsonlines==4.0.0
  Downloading jsonlines-4.0.0-py3-none-any.whl.metadata (1.6 kB)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonlines==4.0.0) (25.3.
Downloading jsonlines-4.0.0-py3-none-any.whl (8.7 kB)
Installing collected packages: jsonlines
Successfully installed jsonlines-4.0.0
```

```
!pip install datasets==2.20.0
```

Show hidden output

Listing the installed packages

```
import subprocess

# Run pip list and capture the output
result = subprocess.run(['pip', 'list'], stdout=subprocess.PIPE, text=True)

# Split the output into lines and count them
```

```
package_list = result.stdout.split('\n')

# Adjust count for headers or empty lines
package_count = len([line for line in package_list if line.strip() != '']) - 2

print(f"Number of installed packages: {package_count}")
```

```
Number of installed packages: 636
```

```
import subprocess

# Run pip list and capture the output
result = subprocess.run(['pip', 'list'], stdout=subprocess.PIPE, text=True)

# Print the output
print(result.stdout)
```

```
google-pasta                     0.2.0
google-resumable-media           2.7.2
googleapis-common-protos         1.70.0
googledrivedownloader            1.1.0
gradio                           5.41.0
gradio_client                    1.11.0
graphviz                         0.21
greenlet                         3.2.3
groovy                           0.1.2
grpc-google-iam-v1               0.14.2
grpc-interceptor                 0.15.4
grpcio                           1.74.0
grpcio-status                    1.71.2
grpclib                          0.4.8
gspread                          6.2.1
gspread-dataframe                4.0.0
gym                              0.25.2
gym-notices                      0.1.0
gymnasium                        1.2.0
h11                              0.16.0
h2                               4.2.0
h5netcdf                         1.6.4
h5py                             3.14.0
hdbscan                          0.8.40
hf_transfer                      0.1.9
hf-xet                           1.1.7
highspy                          1.11.0
holidays                         0.78
holoviews                        1.21.0
hpack                            4.1.0
html5lib                         1.1
httpcore                         1.0.9
httpimport                       1.4.1
httplib2                         0.22.0
httpx                            0.28.1
huggingface-hub                  0.34.3
humanize                         4.12.3
hyperframe                       6.1.0
hyperopt                         0.2.7
ibis-framework                   9.5.0
idna                             3.10
imageio                          2.37.0
imageio-ffmpeg                   0.6.0
imagesize                        1.4.1
imbalanced-learn                 0.13.0
immutabledict                    4.2.1
importlib_metadata               8.7.0
importlib_resources              6.5.2
imutils                          0.5.4
inflect                          7.5.0
iniconfig                        2.1.0
intel-cmplr-lib-ur               2025.2.0
intel-openmp                     2025.2.0
ipyevents                        2.0.2
ipyfilechooser                   0.6.0
ipykernel                        6.17.1
ipyleaflet                       0.20.0
ipyparallel                      8.8.0
ipython                          7.34.0
```

counting the number of packages

1.Preparing the dataset for fine-tuning

1.1.Downloading and displaying the dataset

```
from datasets import load_dataset
import pandas as pd

# Load the SQuAD dataset from HuggingFace
dataset = load_dataset("squad", split="train[:500]")

# Filter the dataset to ensure context and answer are present
filtered_dataset = dataset.filter(lambda x: x["context"] != "" and x["answers"]["text"] != [])

# Extract prompt (context + question) and response (answer)
def extract_prompt_response(example):
    return {
        "prompt": example["context"] + " " + example["question"],
        "response": example["answers"]["text"][0]  # Take the first answer
    }

filtered_dataset = filtered_dataset.map(extract_prompt_response)

# Print the number of examples
print("Number of examples: ", len(filtered_dataset))
```

Show hidden output

```
# Convert the filtered dataset to a pandas DataFrame
df_view = pd.DataFrame(filtered_dataset)

# Display the DataFrame
df_view.head()
```

	id	title	context	question	answers	prompt	response
0	5733be284776f41900661182	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	To whom did the Virgin Mary allegedly appear i...	{'text': ['Saint Bernadette Soubirous'], 'answ...	Architecturally, the school has a Catholic cha...	Saint Bernadette Soubirous
1	5733be284776f4190066117f	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	What is in front of the Notre Dame Main Building?	{'text': ['a copper statue of Christ'], 'answe...	Architecturally, the school has a Catholic cha...	a copper statue of Christ
2	5733be284776f41900661180	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	The Basilica of the Sacred heart at Notre Dame...	{'text': ['the Main Building'], 'answer_start'...	Architecturally, the school has a Catholic cha...	the Main Building
3	5733be284776f41900661181	University_of_Notre_Dame	Architecturally, the school has a Catholic cha...	What is the Grotto at Notre Dame?	{'text': ['a Marian place of prayer and reflec...	Architecturally, the school has a Catholic cha...	a Marian place of prayer and reflection
			Architecturally,	What sits	{'text': ['a	Architecturally,	a golden

1.2A Streaming the output to JSON

```
import json
import pandas as pd
```

1.2. Preparing the dataset for fine-tuning

```
import jsonlines
import pandas as pd
from datasets import load_dataset

# Convert to DataFrame and clean
df = pd.DataFrame(filtered_dataset)
#columns_to_drop = ['title','question','answers']
#df = df.drop(columns=columns_to_drop)

# Prepare the data items for JSON lines file
items = []
for idx, row in df.iterrows():
    detailed_answer = row['response'] + " Explanation: " + row['context']
    items.append({
        "messages": [
            {"role": "system", "content": "Given a SQuAD question built from Wikipedia with crowdworders, provide the
            {"role": "user", "content": row['question']},
            {"role": "assistant", "content": detailed_answer}
        ]
    })

# Write to JSON lines file
with jsonlines.open('/content/QA_prompts_and_completions.json', 'w') as writer:
    writer.write_all(items)
```

Visualizing the JSON file

```
dfile="/content/QA_prompts_and_completions.json"
```

```
import pandas as pd

# Load the data
df = pd.read_json(dfile, lines=True)
df
```

	messages
0	[{'role': 'system', 'content': 'Given a SQuAD ...
1	[{'role': 'system', 'content': 'Given a SQuAD ...
2	[{'role': 'system', 'content': 'Given a SQuAD ...
3	[{'role': 'system', 'content': 'Given a SQuAD ...
4	[{'role': 'system', 'content': 'Given a SQuAD ...
...	...
495	[{'role': 'system', 'content': 'Given a SQuAD ...
496	[{'role': 'system', 'content': 'Given a SQuAD ...
497	[{'role': 'system', 'content': 'Given a SQuAD ...
498	[{'role': 'system', 'content': 'Given a SQuAD ...
499	[{'role': 'system', 'content': 'Given a SQuAD ...

500 rows × 1 columns

2.Fine-tuning the model

```
from openai import OpenAI
import jsonlines
client = OpenAI()
# Uploading the training file

result_file = client.files.create(
```

```
  file=open("QA_prompts_and_completions.json", "rb"),
  purpose="fine-tune"
)

print(result_file)
param_training_file_name = result_file.id
print(param_training_file_name)

# Creating the fine-tuning job
ft_job = client.fine_tuning.jobs.create(
  training_file=param_training_file_name,
  model="gpt-4.1-mini-2025-04-14"
)

# Printing the fine-tuning job
print(ft_job)
```

```
FileObject(id='file-6VgWqp3wmd6Tos6aUihLLZ', bytes=645098, created_at=1755077028, filename='QA_prompts_and_completions
file-6VgWqp3wmd6Tos6aUihLLZ
FineTuningJob(id='ftjob-f9JxcdrNVYiVnO4M3EZkLaLT', created_at=1755077029, error=Error(code=None, message=None, param=N
```

Monitoring the fine-tunes

```
import pandas as pd
from openai import OpenAI
client = OpenAI()
# Assume client is already set up and authenticated
response = client.fine_tuning.jobs.list(limit=3)# increase to see history

# Initialize lists to store the extracted data
job_ids = []
created_ats = []
statuses = []
models = []
training_files = []
error_messages = []
fine_tuned_models = []  # List to store the fine-tuned model names

# Iterate over the jobs in the response
for job in response.data:
    job_ids.append(job.id)
    created_ats.append(job.created_at)
    statuses.append(job.status)
    models.append(job.model)
    training_files.append(job.training_file)
    error_message = job.error.message if job.error else None
    error_messages.append(error_message)

    # Append the fine-tuned model name
    fine_tuned_model = job.fine_tuned_model if hasattr(job, 'fine_tuned_model') else None
    fine_tuned_models.append(fine_tuned_model)

# Create a DataFrame
df = pd.DataFrame({
    'Job ID': job_ids,
    'Created At': created_ats,
    'Status': statuses,
    'Model': models,
    'Training File': training_files,
    'Error Message': error_messages,
    'Fine-Tuned Model': fine_tuned_models  # Include the fine-tuned model names
})

# Convert timestamps to readable format
df['Created At'] = pd.to_datetime(df['Created At'], unit='s')
df = df.sort_values(by='Created At', ascending=False)

# Display the DataFrame
df
```

	Job ID	Created At	Status	Model	Training File	Error Message	Fine-Tuned Model
0	ftjob-f9JxcdrNVYiVnO4M3EZkLaLT	2025-08-13 09:23:49	succeeded	gpt-4.1-mini-2025-04-14	file-6VgWqp3wmd6Tos6aUihLLZ	None	ft:gpt-4.1-mini-2025-04-14:personal::C42Vkm29
1	ftjob-5BnCL6gUsFCTzyDYAN4CfvlZ	2024-12-02 10:04:26	succeeded	gpt-4o-mini-2024-07-18	file-KYWaDkNUDFSqqUcjnSv8hw	None	ft:gpt-4o-mini-2024-07-18:personal::AZy8s878
	ftjob-	2024-12-		gpt-4o-	file-		ft:gpt-4o-mini-2024-07-

Make sure to obtain your fine-tune model here

If your OpenAI notifications are activated you should receive an email.

Otherwise run the "Monitoring the fine-tunes" cell above to check the status of your fine-tune job.

```
import pandas as pd

generation=False  # False until the last model fine-tuned is found. Make sure it used the dataset you trained it on!
# Attempt to find the first non-empty Fine-Tuned Model
non_empty_models = df[df['Fine-Tuned Model'].notna() & (df['Fine-Tuned Model'] != '')]

if not non_empty_models.empty:
    first_non_empty_model = non_empty_models['Fine-Tuned Model'].iloc[0]
    print("The latest fine-tuned model is:", first_non_empty_model)
    generation=True
else:
    first_non_empty_model='None'
    print("No fine-tuned models found.")
```

```
The latest fine-tuned model is: ft:gpt-4.1-mini-2025-04-14:personal::C42Vkm29
```

```
# Fine-tuned model found(True) or not(False)
generation
```

```
True
```

Note: Only continue to Step 3, to use the fine-tuned model when your fine-tuned model is ready. If your OpenAI notifications is activiated, you will receive an email with the status of your fine-tunning job.

3.Using the fine-tuned OpenAI model

Note: The is a fine-tuning. As such, be patient! Rune the `Monitoring the fine-tunes` cell and the `first_non_empty_model` cell from time to time.

If the fine-tunning succeeded and your model is ready, the name of your model will be `first_non_empty_model`

1.Go to the OpenAI Playground to test your model: https://platform.openai.com/playground

2.Check the metrics in the fine-tuning UI: https://platform.openai.com/finetune/

3.Try the fined-tune model out in the cell below.

```
# Define the prompt
prompt="Which prize did Frederick Buechner create?"
```

Note: Only run the following cell if your fine-tune job has succeeded and a fined-tuned model is found in the *Monitoring the fine-tunes"* section of *2.Fine-tuning the model.*

```python
# Assume first_non_empty_model is defined above this snippet
if generation==True:
    response = client.chat.completions.create(
        # fine-tuned model or fallback if the model is not fine-tuned yet
        model=first_non_empty_model or "gpt-4.1-mini-2025-04-14",
        temperature=0.0,  # Adjust as needed for variability
        messages=[
            {"role": "system", "content": "Given a question, reply with a complete explanation for students."},
            {"role": "user", "content": prompt}
        ]
    )
else:
    print("Error: Model is None, cannot proceed with the API request.")
```

```python
if generation==True:
  print(response)
```

```
ChatCompletion(id='chatcmpl-C42hsGT0RiJhxJmjhV0kI7VCERHOX', choices=[Choice(finish_reason='stop', index=0, logprobs=No
```

```python
if (generation==True):
  # Access the response from the first choice
  response_text = response.choices[0].message.content
  # Print the response
  print(response_text)
```

```
Frederick Buechner Prize for Preaching Explanation: The university is affiliated with the Congregation of Holy Cross (
```

```python
import textwrap

if generation==True:
  wrapped_text = textwrap.fill(response_text.strip(), 60)
  print(wrapped_text)
```

```
Frederick Buechner Prize for Preaching Explanation: The
university is affiliated with the Congregation of Holy Cross
(Latin: Congregatio a Sancta Cruce, abbreviated
postnominally as "CSC"). While religious affiliation is not
a criterion for admission, more than 93% of students
identify as Christian, with over 80% of the total being
Catholic. Collectively, Catholic Mass is celebrated over 100
times per week on campus, and a large campus ministry
program provides for the spiritual needs of the community.
There are multitudes of religious statues and artwork around
campus, most prominent of which are the statue of Mary on
the Main Building, the Notre Dame Grotto, and the Word of
Life mural on Hesburgh Library depicting Christ as a
teacher. Additionally, every classroom displays a crucifix.
There are many religious clubs (catholic and non-Catholic)
at the university, including Council #1477 of the Knights of
Columbus (KOC), Baptist Collegiate Ministry (BCM), Jewish
Club, Muslim Student Association, Orthodox Christian
Fellowship, The Mormon Club, and many more. The Notre Dame
KofC are known for being the first collegiate council of
KofC, operating a charitable concession stand during every
home football game and owning their own building on campus
which can be used as a cigar lounge. Fifty-seven chapels are
located throughout the campus.
```

Consult OpenAI fine-tune documentation for more