

GenAIoT Lab Session 5.3

LLM-Based Text-Generation Inference and Fine-tuning on Jetson AGX Orin

Overview:

Large Language Models require enormous computing power to train, as such it requires the use of a powerful GPU as well as takes a long time (depending on the GPU and model, with some models taking days). Thanks to platforms such as Hugging Face that provide several pre-trained LLM models for several tasks like text-generation, chat etc. In this lab, students will be taught how to download an LLM model from Hugging Face and run the inference on the Jetson AGX Orin developer kit. Furthermore, we will learn how to fine-tune the model with an additional dataset for some specific tasks. The Tables below show the size of LLM models based on the parameters and the memory (Giga-Byte).

Model Name	Parameters	Type
Gemma 2B	2 billion	Small
Gemma 7B	7 billion	Small
Mistral 7B	7 billion	Small
Llama-2	Up to 70 billion	Medium
Orca	13 billion	Medium
GPT-4	~1.76 trillion	Large
Gemini Pro	Hundreds of billions	Large

The size of a Model in GB is calculated as:

$$Model\ size(GB) = \frac{Number\ of\ Parameters \times Size\ of\ Each\ Parameter\ in\ bytes}{2^{30}}$$

Model Name	Parameters	Approximate Size (32-bit)	Approximate Size (16-bit)
Llama-2	70 billion	~280 GB	~140 GB
GPT-4	1.7 trillion	~6800 GB	~3400 GB
Smaller Models	7 billion	~28 GB	~14 GB
Example Model	1 billion	~4 GB	~2 GB

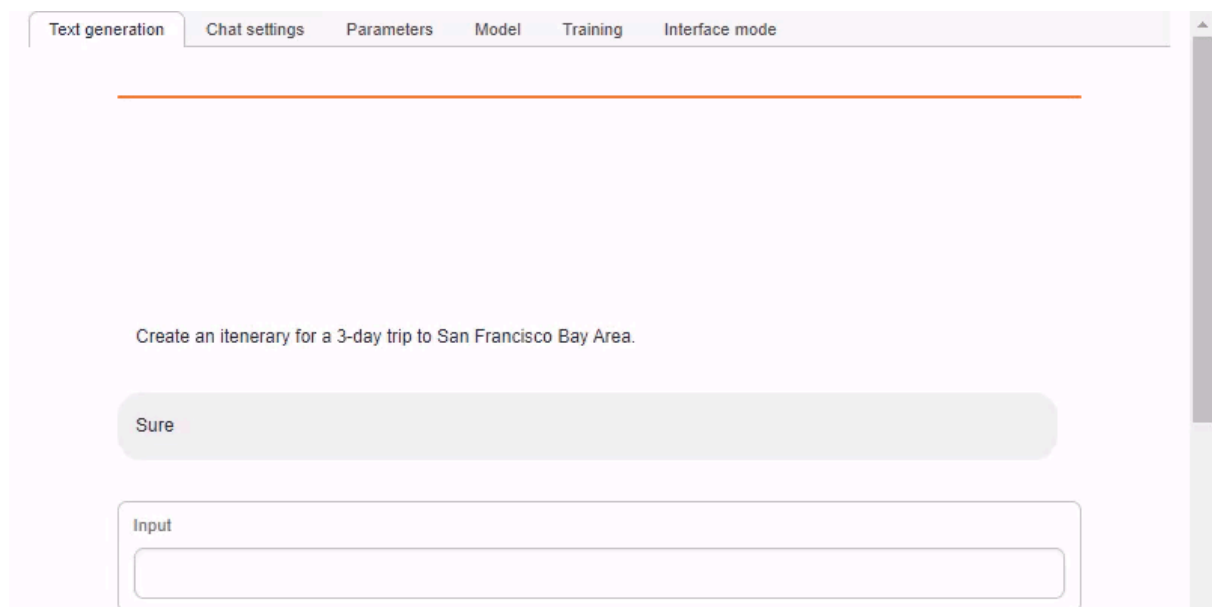
Required:

1. Jetson AGX Orin developer kit
2. Internet connection
3. Assumed you have flashed the board with Jetpack 6

Objectives:

- Part 1: Expanding the memory for the Jetson Board
- Part 2: LLM Text Generation Inference
- Part 3: Fine-tuning the LLM

Expected Text-Generation Web-UI Result:



The screenshot displays a web-based user interface for text generation. At the top, there is a horizontal navigation bar with six tabs: 'Text generation' (which is the active tab), 'Chat settings', 'Parameters', 'Model', 'Training', and 'Interface mode'. Below the navigation bar, a thin orange horizontal line separates the header from the main content area. The main content area has a light pink background. In the center, the text 'Create an itenerary for a 3-day trip to San Francisco Bay Area.' is displayed. Below this text is a light green button with the word 'Sure'. At the bottom of the interface, there is a text input field with a light purple border and a light pink background. The word 'Input' is written in small text above the input field.

Part 1: Expanding the Memory of the Jetson Board **(Already done, students are not required to do this part, please go to part 2)**

The Jetson AGX Orin developer kit only comes with a 64Gb internal storage space, however the size of LLMs is large (2GB and 6GB for small models). Hence to run different models, it is imperative to expand the storage spaces by mounting another SSD memory drive to the Jetson board. This part 1 shows the procedures for mounting the SSD on a Linux system, and also the procedures for changing the docker install directory to move the docker images to the new SSD location.

Procedure: (Reference: https://www.jetson-ai-lab.com/tips_ssd-docker.html)

1. Physical installation

1. Unplug power and any peripherals from the Jetson developer kit.
2. Physically install an NVMe SSD card on the carrier board of your Jetson developer kit, making sure to properly seat the connector and secure it with the screw.
3. Reconnect any peripherals, and then reconnect the power supply to turn on the Jetson developer kit.
4. Once the system is up, verify that your Jetson identifies a new memory controller on PCI bus

>> lspci

The output should look like the following:

0007:01:00.0 Non-Volatile memory controller: Marvell Technology Group Ltd. Device

B. Format and set up auto-mount

1. check if you can find the the installed SSD device name (nvme0n1)

>> lsblk

```

mmcblk0 179:0 0 59.3G 0 disk
--mmcblk0p1 179:1 0 57.8G 0 part /
--mmcblk0p2 179:2 0 128M 0 part
--mmcblk0p3 179:3 0 768K 0 part
--mmcblk0p4 179:4 0 31.6M 0 part
--mmcblk0p5 179:5 0 128M 0 part
--mmcblk0p6 179:6 0 768K 0 part
--mmcblk0p7 179:7 0 31.6M 0 part
--mmcblk0p8 179:8 0 80M 0 part
--mmcblk0p9 179:9 0 512K 0 part
--mmcblk0p10 179:10 0 64M 0 part
--mmcblk0p11 179:11 0 80M 0 part
--mmcblk0p12 179:12 0 512K 0 part
--mmcblk0p13 179:13 0 64M 0 part
--mmcblk0p14 179:14 0 400M 0 part
--mmcblk0p15 179:15 0 479.5M 0 part
mmcblk0boot0 179:32 0 31.5M 1 disk
mmcblk0boot1 179:64 0 31.5M 1 disk
zram0 252:0 0 1.9G 0 disk [SWAP]
zram1 252:1 0 1.9G 0 disk [SWAP]
zram2 252:2 0 1.9G 0 disk [SWAP]
zram3 252:3 0 1.9G 0 disk [SWAP]
zram4 252:4 0 1.9G 0 disk [SWAP]
zram5 252:5 0 1.9G 0 disk [SWAP]
zram6 252:6 0 1.9G 0 disk [SWAP]
zram7 252:7 0 1.9G 0 disk [SWAP]
nvme0n1 259:0 0 238.5G 0 disk

```

2. Format the SSD, create a mount point, and mount it to the filesystem

- format the drive to ext4 file system

>> `sudo mkfs.ext4 /dev/nvme0n1`

```

eeuser@ubuntu:~$ sudo mkfs.ext4 /dev/nvme0n1
[sudo] password for eeuser:
mke2fs 1.46.5 (30-Dec-2021)
Found a dos partition table in /dev/nvme0n1
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 62514774 4k blocks and 15630336 inodes
Filesystem UUID: fc41e7c9-315a-4323-9fa1-77b607349c2f
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done

```

- Create a new mount directory /mnt/nvme_ssd for the SSD

>> `sudo mkdir -p /mnt/nvme_ssd`

- Temporarily mount the drive

>> `sudo mount /dev/nvme0n1 /mnt/nvme_ssd`

- To permanently mount the SSD, check the uuid of the ssd

>> `lsblk -f`

```

zram7 1.0G 0 1.0G 0 disk [SWAP]
nvme0n1 ext4 1.0 fc41e7c9-315a-4323-9fa1-77b607349c2f

```

- Open the /etc/fstab file and add the UUID by adding the below line

```
>> sudo vim /etc/fstab
```

```
# add the following line to the /etc/fstab file to mount the drive permanently
"UUID=*****_****_****_***** /mnt/nvme_ssd ext4 defaults 0 2"
```

```
# /etc/fstab: static file system information.
#
# These are the filesystems that are always mounted on boot, you can
# override any of these by copying the appropriate line from this file into
# /etc/fstab and tweaking it as you see fit. See fstab(5).
#
# <file system> <mount point>          <type>          <options>
p> <pass>
/dev/root          /                  ext4            defaults
0 1
UUID=6B5A-F409 /boot/efi vfat defaults 0 1
UUID=dccc9f82-6fef-4741-ace8-b6fa1eed01a5 /mnt/nvme_ssd ext4 defaults 0 2
~
```

- Add the user to the ownership of the SSD

```
>> sudo chown eeuser:eeuser /mnt/nvme_ssd
```

3. Change docker directory to SSD

- Update and install docker if not installed

```
>> sudo apt update
```

```
>> sudo apt install -y nvidia-container #ensure the docker is installed
```

- Stop the docker service

```
>> sudo systemctl stop docker
```

- Copy the old docker file to the new directory in the new SSD

```
>> sudo du -csh /var/lib/docker/ && \
```

```
sudo mkdir /mnt/nvme_ssd/docker && \
```

```
sudo rsync -axPS /var/lib/docker/ /mnt/nvme_ssd/docker && \
```

```
sudo du -csh /mnt/nvme_ssd/docker
```

```
overlay2/l/ZFAIYWKFVXEY6P35M6QK7Y73A -> ../7f211dc12c18488fe592377459484a84660726f53e7f3788dfa67ab15a
c80a21/diff
plugins/
plugins/storage/
plugins/storage/ingest/
plugins/tmp/
runtimes/
swarm/
tmp/
volumes/
volumes/backingFsBlockDev
volumes/metadata.db
32,768 100% 285.71kB/s 0:00:00 (xfr#101463, to-chk=0/126245)
17G /mnt/nvme_ssd/docker
17G total
```

- Edit /etc/docker/daemon.json

```
>> sudo vi /etc/docker/daemon.json
```

#change the content to the following

```
{
  "runtimes": {
    "nvidia": {
      "path": "nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "default-runtime": "nvidia",
  "data-root": "/mnt/nvme_ssd/docker"
}
```

```
{
  "runtimes": {
    "nvidia": {
      "path": "nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "default-runtime": "nvidia",
  "data-root": "/mnt/nvme_ssd/docker"
}
```

- Rename the old docker directory before final deletion

```
>> sudo mv /var/lib/docker /var/lib/docker.old
```

- Restart the docker, press Ctrl+C to stop the display when done

```
>> sudo systemctl daemon-reload && \
sudo systemctl restart docker && \
sudo journalctl -u docker
```

```
eeuser@ubuntu:/mnt/nvme_ssd$ sudo systemctl daemon-reload && \
sudo systemctl restart docker && \
sudo journalctl -u docker
Oct 04 15:43:32 ubuntu systemd[1]: Starting Docker Application Container Engine...
Oct 04 15:43:32 ubuntu dockerd[1651]: time="2024-10-04T15:43:32.648228200+08:00" level=info msg="Star>
```

```
Oct 31 15:40:10 ubuntu dockerd[3878]: time="2024-10-31T15:40:10.730674723+08:00" level=info msg="Load>
Oct 31 15:40:10 ubuntu dockerd[3878]: time="2024-10-31T15:40:10.753803584+08:00" level=info msg="Dock>
Oct 31 15:40:10 ubuntu dockerd[3878]: time="2024-10-31T15:40:10.754390706+08:00" level=info msg="Daem>
Oct 31 15:40:10 ubuntu dockerd[3878]: time="2024-10-31T15:40:10.805044849+08:00" level=info msg="API >
Oct 31 15:40:10 ubuntu systemd[1]: Started Docker Application Container Engine.
[lines 1-32/32 (END)]
```

- Check if the docker root dir has been changed or not

```
>> sudo docker info
```

```
Total Memory: 29.98GiB
Name: ubuntu
ID: eafafbd5-1476-42a9-96f5-f9c8ceed1961
Docker Root Dir: /mnt/nvme_ssd/docker
Debug Mode: false
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
```

- Check remaining storage

>> df -h

```
eeuser@ubuntu:/mnt/nvme_ssd$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p1  54G   45G   6.7G  87% /
tmpfs           15G   136K   15G   1% /dev/shm
tmpfs           6.0G   27M   6.0G   1% /run
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           3.0G   96K   3.0G   1% /run/user/128
tmpfs           3.0G   80K   3.0G   1% /run/user/1000
/dev/nvme0n1    234G   17G  205G   8% /mnt/nvme_ssd
```

Part 2: LLM Inference with Text-Generation-Webui and Jetson container

The jetson-containers project provides pre-built Docker images for text-generation-webui along with all of the loader API's built with CUDA enabled (llama.cpp, ExLlama, AutoGPTQ, Transformers, ect). You can clone the repo to use its utilities that will automatically pull/start the correct container for you, or you can do it manually. In this part, we are going to use the text-generation-webui.

Procedure:

1. Clone and install jetson-containers(**students do not need to do this, it is already done**)

>> cd /mnt/nvme_ssd

>> git clone <https://github.com/dusty-nv/jetson-containers>

```
eeuser@ubuntu:/mnt/nvme_ssd$ git clone https://github.com/dusty-nv/jetson-containers
Cloning into 'jetson-containers'...
remote: Enumerating objects: 23448, done.
remote: Counting objects: 100% (1441/1441), done.
remote: Compressing objects: 100% (606/606), done.
remote: Total 23448 (delta 878), reused 1349 (delta 814), pack-reused 22007 (from 1)
Receiving objects: 100% (23448/23448), 219.12 MiB | 1.80 MiB/s, done.
Resolving deltas: 100% (15707/15707), done.
```

>> bash jetson-containers/install.sh

```
eeuser@ubuntu:/mnt/nvme_ssd$ bash jetson-containers/install.sh
+++ readlink -f jetson-containers/install.sh
++ dirname /mnt/nvme_ssd/jetson-containers/install.sh
+ ROOT=/mnt/nvme_ssd/jetson-containers
+ INSTALL_PREFIX=/usr/local/bin
+ pip3 --version
```

```

Building wheel for DockerHub-API (setup.py) ... done
Created wheel for DockerHub-API: filename=DockerHub_API-0.5-py3-none-any.whl size=6780 sha256=9771d4
5150beb6f62b1a2af208c8dbe98aa05c0986826df1ebca01aca7d6cfee
Stored in directory: /tmp/pip-ephem-wheel-cache-zcluhmvd/wheels/fa/70/84/1892d686b0e514f657c43d3efd2
f4a7e6b2c1e6c8ca5ae5286
Successfully built wget DockerHub-API
Installing collected packages: wget, pyyaml, orderedmultidict, furl, DockerHub-API
Successfully installed DockerHub-API-0.5 furl-2.1.3 orderedmultidict-1.0.1 pyyaml-6.0.2 wget-3.2
+ sudo ln -sf /mnt/nvme_ssd/jetson-containers/autotag /usr/local/bin/autotag
+ sudo ln -sf /mnt/nvme_ssd/jetson-containers/jetson-containers /usr/local/bin/jetson-containers

```

2. Run the text-generation-webui (Jetson container will pull the image if not found). Use *jetson-containers run* and *autotag* tools to automatically pull or build a compatible container image. The container has a default run command (*CMD*) that will automatically start the webserver.

>> *jetson-containers run \$(autotag text-generation-webui)*

```

Found compatible container dustynv/text-generation-webui:r36.2.0 (2024-02-03, 8.3GB) - would you like
to pull it? [Y/n] |

```

```

Status: Downloaded newer image for dustynv/text-generation-webui:r36.2.0
/usr/local/lib/python3.10/dist-packages/transformers/utils/hub.py:124: FutureWarning: Using `TRANSFORM
ERS_CACHE` is deprecated and will be removed in v5 of Transformers. Use `HF_HOME` instead.
  warnings.warn(
15:54:37-662021 INFO Starting Text generation web UI
15:54:37-668315 WARNING You are potentially exposing the web UI to the entire internet without any
access password.
You can create one with the "--gradio-auth" flag like this:

--gradio-auth username:password

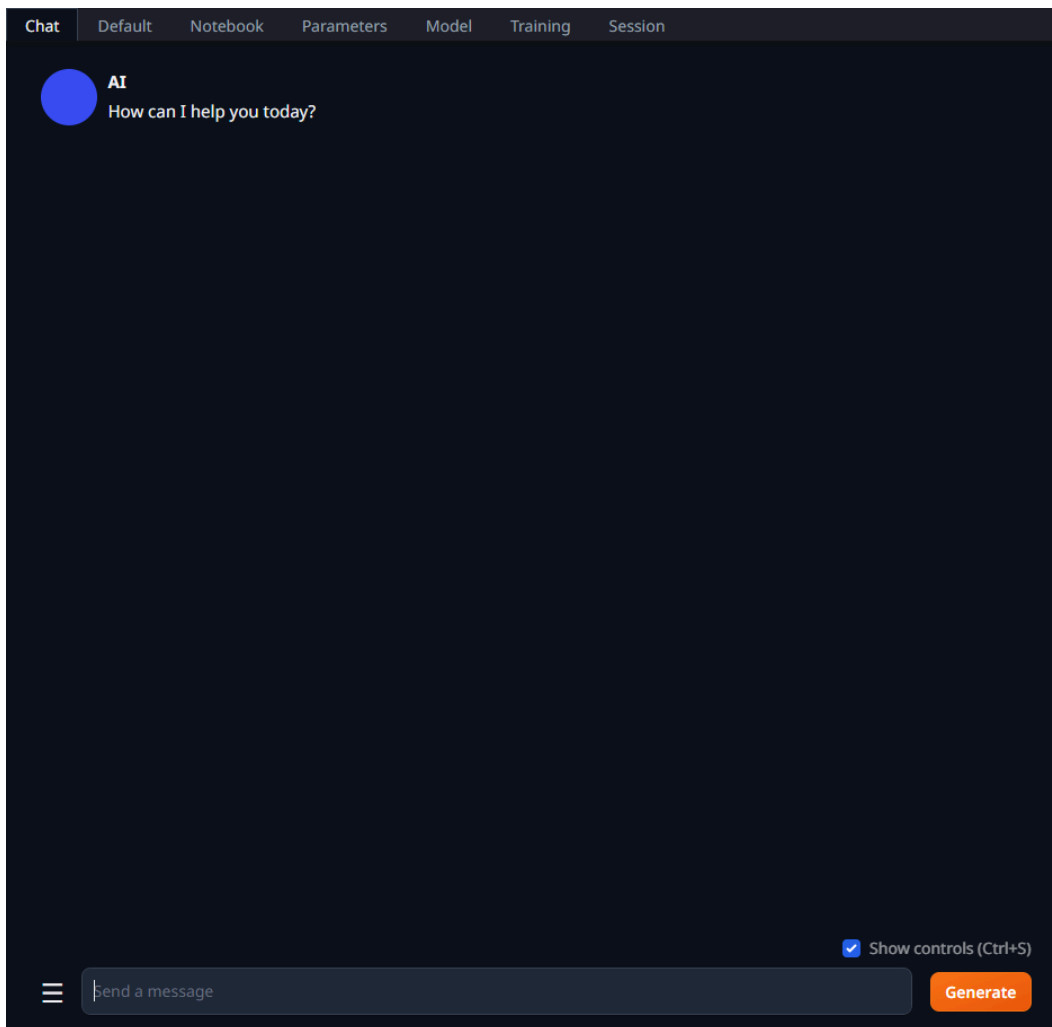
Make sure to replace username:password with your own.
15:54:37-671226 INFO Loading settings from settings.json
15:54:37-675007 INFO Loading the extension "gallery"
Running on local URL: http://0.0.0.0:7860

```

3. Open the Webui from a browser using port:7860
 - Open your browser and access `http://<IP_ADDRESS>:7860`
 - 192.168.50.XX:7860 (XX is your assigned board number e.g 60)

192.168.50.60:7860

- Next we download and load the no model for the system to function.



4. Download the model (Already done !, students do not need to do it)

- Click the Model tab from the menu
- Enter the Huggingface model path in the “**Download model or LoRA**” option i.e., TheBloke/Llama-2-13B-chat-GGUF
- Enter the model version i.e., llama-2-13b-chat.Q4_K_M.gguf
- Click Download

Download model or LoRA

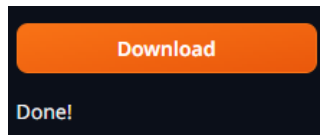
Enter the Hugging Face username/model path, for instance: facebook/galactica-125m. To specify a branch, add it at the end after a ":" character like this: facebook/galactica-125m:main. To download a single file, enter its name in the second box.



Download

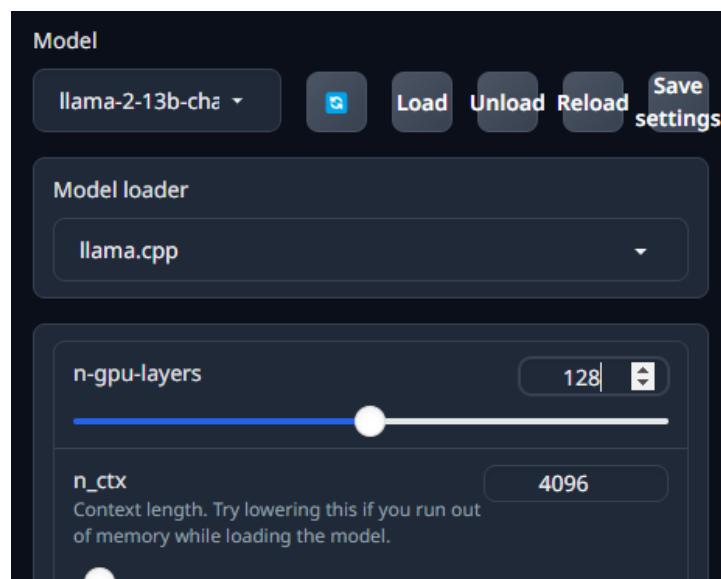
Get file list

To create a public link, set `share=True` in `launch()`.
 Downloading the model to /data/models/text-generation-webui
 12%|

| 958M /7.87G 41.2MiB/s



5. Select and load the downloaded model to the Webui. After you have downloaded a model, click the  button to refresh your model list, and select the model you want to use.
- refresh the model list  and select the new model
 - For a GGUF model, remember to
 1. Set n-gpu-layers to 128
 2. Set n_gqa to 8 if you using Llama-2-70B (on Jetson AGX Orin 64GB)
 - Change n-gpu-layers to 128
 - Then click the Load button.



```
16:03:14-114605 INFO LOADER: llama.cpp
16:03:14-116780 INFO TRUNCATION LENGTH: 4096
16:03:14-118172 INFO INSTRUCTION TEMPLATE: Alpaca
16:03:14-119432 INFO Loaded the model in 12.00 seconds.
```

Successfully loaded llama-2-13b-chat.Q4_K_M.gguf.

It seems to be an instruction-following model with template "Alpaca". In the chat tab, instruct or chat-instruct modes should be used.

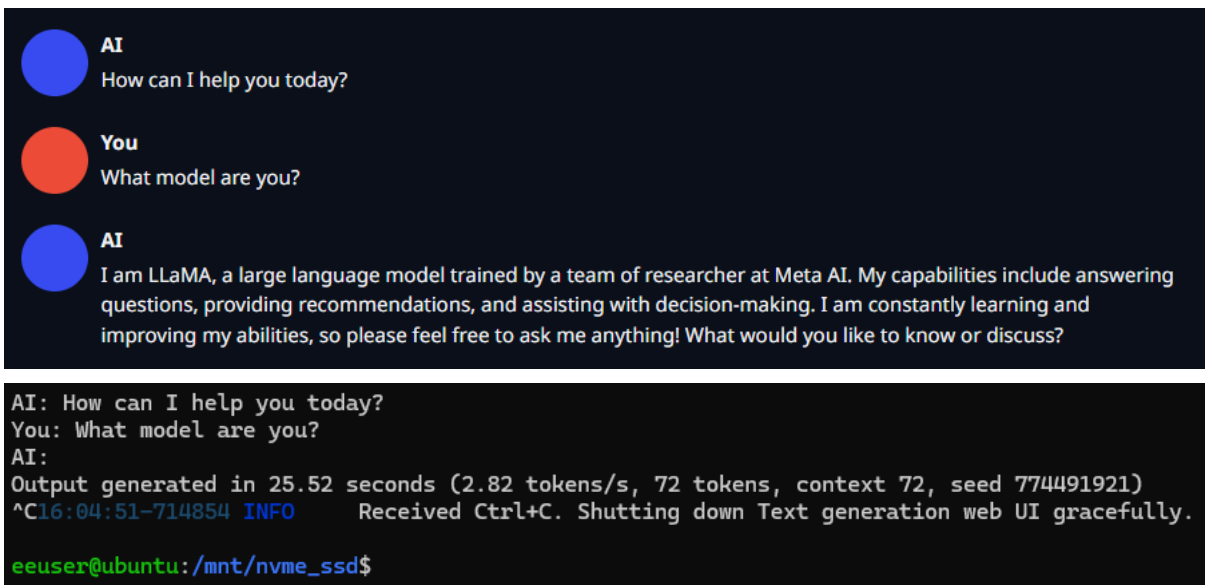
6. Now you can properly chat with the loaded model

If you're using a Llama model fine-tuned for chat, like the models listed above (except for LLaMA-30b), you need to use the oobabooga Instruct mode and set the template. On the **Parameters** tab, go to the **Instruction Template** sub-tab, then select Llama-v2 from the **Instruction Template** drop-down (or Vicuna, Guanaco, ect if you are using one of those models)

For the base text completion models (like LLaMA-30b), use the Default or Notebook tab.

Selecting the right chat template will make sure the model is being [prompted correctly](#) - you can also change the system prompt in the **Context** box to alter the agent's personality and behavior. There are a lot of other settings under the **Generation** tab, like the maximum length it should output per reply, and token sampling parameters like [temperature and top_p](#) for controlling randomness.

Then change back to the **Chat** tab, and under the mode section, make sure **Instruct** is selected (confusingly, not chat mode). Then you can start chatting with the LLM!



- Press Ctrl + c to stop
- To check the installation files are saved in the SSD

>> df -h

```
eeuser@ubuntu:/mnt/nvme_ssd$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p1  54G   45G   6.7G  87% /
tmpfs           15G   136K   15G   1% /dev/shm
tmpfs           6.0G   27M   6.0G   1% /run
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           3.0G   96K   3.0G   1% /run/user/128
tmpfs           3.0G   80K   3.0G   1% /run/user/1000
/dev/nvme0n1    234G   41G  182G  19% /mnt/nvme_ssd
```

Further studies: https://www.jetson-ai-lab.com/tutorial_text-generation.html

Part 3: Fine-Tuning the LLM with llama-factory and Jetson Container

Llama-factory provides a convenient Webui for downloading LLM models and dataset from Hugging Face as well as for fine-tuning the LLM models. The jetson series are mainly designed for running inference, but it can also perform training or fine-tuning models. In this lab, we would use the llama-factory to download the models and dataset, then fine-tune the model. Because fine-tuning LLM can take a lot of time, we would limit the maximum samples to 100 to reduce the training time.

Procedure:

1. Pull and run the Llama-factory. Jetson-container will automatically pull the llama-factory container and start it.

> jetson-containers run \$(autotag llama-factory)

2. Access the Webui from your browser using port:7860 (same as before)
 - Open your browser and access `http://<IP_ADDRESS>:7860`
 - 192.168.50.XX:7860 (XX is your assigned board number e.g 60)

192.168.50.60:7860

3. Choose OpenChat3.5-7B-Chat (This model was already downloaded to the Jetson board to save time)

Lang en	Model name OpenChat3.5-7B-Chat	Model path Path to pretrained model or model identifier from Hugging Face. openchat/openchat-3.5-0106
------------	-----------------------------------	---

4. Choose wikiqa dataset (This dataset was also already downloaded to save time)

Train Evaluate & Predict Chat Export

Stage The stage to perform in training. <div>Supervised Fine-T</div>	Data dir Path to the data directory. <div>/opt/LLaMA-Factory/data</div>	Dataset <div>wikiqa ×</div> <div>× ▾</div>	<div>Preview dataset</div>
---	--	--	----------------------------

5. Set Max samples to 100

Learning rate Initial learning rate for AdamW. <div>5e-5</div>	Epochs Total number of training epochs to perform. <div>3.0</div>	Maximum gradient norm Norm for gradient clipping. <div>1.0</div>	Max samples Maximum samples per dataset. <div>100</div>
Compute type Whether to use mixed precision training. <div>bf16 ▾</div>			

6. Change the name of **Output dir** and **Config path** if you want to, and press start, and wait

Preview command Save arguments Load arguments

Start Abort

Output dir Directory for saving results. <div>Open_wiki ▾</div>	Config path Path to config saving arguments. <div>Open_wiki.yaml ▾</div>	<div>Loss</div>
Device count Number of devices available. <div>1</div>	DeepSpeed stage DeepSpeed stage for distributed training. <div>none ▾</div>	
Enable DeepSpeed offload (slow down training). <input type="checkbox"/> Enable offload		

Device count

Number of devices available

1

Enable DeepSpeed offload

☐ Enable offload

Finished.

7. Choose your model from checkpoint path and press load model

Lang

en

Model name

OpenChat3.5-7B-Chat

Model path

Path to pretrained model or model identifier from Hugging Face.

openchat/openchat-3.5-0106

Finetuning method

lora

Checkpoint path

train_2024-11-11-13-37-51

Advanced configurations

Train

Evaluate & Predict

Chat

Export

Inference engine

huggingface

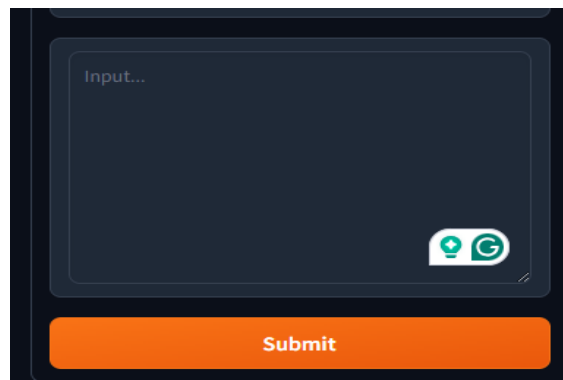
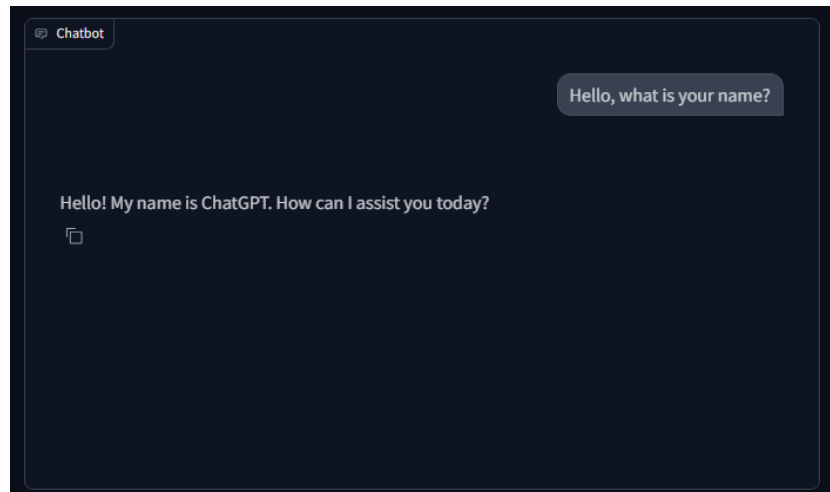
Inference data type

auto

Load model

Unload model

Model unloaded, please load a model first.



Extra Exercise:

If you have completed your tasks and have some time remaining, you can carry out the below task. The task teaches you how to use the text-generation LLM as an API in your program.

Task: Open and perform the tutorial in the below link

https://www.jetson-ai-lab.com/tutorial_api-examples.html