

# Fine-Tuning Llama 2 | Generative AI Series

Learn how to fine-tune Llama 2 for specific applications in our comprehensive guide. Discover techniques, tools, and best practices for optimizing this powerful AI model.

# Contents

01	Introduction	3
02	Prerequisites	3
03	Compare RAG With Labelled Datasets	4
04	Install the Required Dependencies	4
05	Implement the AutoTrain Tool	5
06	Prepare the New Model's Dataset	5
07	Conclusion	11

# Introduction

---

Fine-tuning involves updating an AI model with new data to adapt the model to a specific task. You can fine-tune a model using:

1. The retrieval augmented generation (RAG) technique. This method involves tapping real-time data sources like MySQL/PostgreSQL database to generate factual results.
2. The labeled dataset technique: This method uses a smaller labeled task-specific dataset to optimize the model's performance.

Consider the Llama 2 language Models (LLMs) with over 70 billion parameters. The model is trained on vast volumes of data to perform different tasks. However, with time, the model's data can become outdated.

To make the LLama 2 model generate relevant results for queries, you can fine-tune the model with either the RAG or labeled dataset techniques. Each method has its pros and cons, as discussed in this article.

In this guide, you'll auto-tune the LLama 2 model with a new ad generation labeled dataset. You'll then push the new model to your Hugging Face account and use it to perform ad-generation tasks using the Vultr cloud GPU server.

## Prerequisites

---

Before you begin:

- [Deploy a new Ubuntu 22.04 A100 Vultr Cloud GPU Server](#) with at least:
  - 80 GB GPU RAM
  - 12 vCPUs
  - 120 GB Memory
- [Establish an SSH connection to the server.](#)
- [Create a non-root user with `sudo` rights and switch to the account.](#)

- [Create a HuggingFace account](#).
- [Create a Hugging Face user access token with `WRITE` permissions](#).

## Compare RAG With Labelled Datasets

<https://www.youtube.com/embed/X04n0nRxYJ0?si=kh1yEa2UYIf7Cke2>

The RAG technique is more applicable when using structured data while fine-tuning a model using labeled datasets works well with unstructured data. The following table highlights the key differences between the two.

Factor	RAG	Fine-tuning
<b>Capability</b>	Integrates a retriever model that fetches relevant documents, which are then used by a generator model for response generation.	Directly adjusts the model's parameters based on a specific dataset to tailor its responses.
<b>Flexibility</b>	High flexibility in handling diverse topics by leveraging external data sources.	Tailored to specific tasks or domains, with limited flexibility for topics outside the trained dataset.
<b>Complexity</b>	More complex, involving both a retriever and a generator model.	Simpler, involves only the language model itself.
<b>Relevance</b>	Can provide more contextually relevant and current information by accessing up-to-date external sources.	Dependent on the relevance and recency of the training data, which may limit the model's ability to provide the latest information.
<b>Customization</b>	Less customizable in terms of model behavior due to reliance on external data sources.	Highly customizable to specific tasks or domains, allowing for more tailored responses based on training data.
<b>Inference Speed</b>	Slower inference due to the additional retrieval step; requires regular updating of external data sources.	Faster response generation; needs retraining or continuous fine-tuning to stay effective as the domain evolves.

## Install the Required Dependencies

To fine-tune and use the LLama 2 model with Python, follow the steps below to install the required libraries:

1. Install the `huggingface_hub` and `autotrain-advanced` libraries.

CONSOLE

```
$ pip install huggingface_hub autotrain-advanced
```

2. Add the libraries to the system path.

#### CONSOLE

```
$ echo "export PATH=\"`python3 -m site --user-base`/bin:\n$PATH\"" >> ~/.bashrc\nsource ~/.bashrc
```

## Implement the AutoTrain Tool

<https://www.youtube.com/embed/6kNj7Hb7bI8?si=2D5y-as0wAUcvisa>

The Hugging Face AutoTrain tool allows you to train natural language processing (NLP) models like Llama 2 for specific tasks. The tool accepts different custom parameters in a user-friendly format without spending too much time on the technical details.

In this section, you'll feed an ad copy generation dataset to the AutoTrain tool to fine-tune the `meta-llama/Llama-2-7b-chat-hf` model. Then, you'll upload the newly trained model to your Hugging Face account and later run the model using a Docker container to retrieve fine-tuned results.

## Prepare the New Model's Dataset

[https://www.youtube.com/embed/0\\_uc18PXMDA?si=WweBTKBxK30vsD0a](https://www.youtube.com/embed/0_uc18PXMDA?si=WweBTKBxK30vsD0a)

The AutoTrain tool accepts the following specific data format for training the Llama 2 model:

#### XML

```
<s>[INST] <<SYS>>\n  {{system message}}\n<</SYS>>\n{{message}} [/INST] {{answer}} </s>
```

Hugging Face provides a hub for various datasets that adhere to the above format. You can use these datasets to fine-tune your model. For instance, for this guide, you'll use the [ad copy generation](#) labeled dataset to fine-tune your model. Here are sample rows from the dataset.

XML

```
<s>[INST] <<SYS>> Create a text ad given the following product and description. <</SYS>> Product: Wide-leg trousers  
Description: Pants with a loose and flowing fit, creating an elongated and sophisticated silhouette. [/INST] Ad: Introducing our Wide-Leg Trousers: Effortlessly chic and endlessly versatile. Discover a flattering, comfortable fit that creates an elegant, elongated silhouette. Elevate your style today! </s>
```

...

```
<s>[INST] <<SYS>> Create a text ad given the following product and description. <</SYS>> Product: Electric Food Chopper  
Description: Chop ingredients effortlessly with an Electric Food Chopper - designed for quick and efficient meal prep. [/INST]  
Ad: Chop, prep, and conquer your kitchen tasks! ☑☑ Simplify your meal preparation. Perfect for busy cooks and making cooking a breeze with a touch of electric food chopper efficiency! ☑☑☑ </s>
```

## Fine-tune the Llama 2 Model

```
https://www.youtube.com/embed/wHvCQ59hW00?si=-8jBJ0p7Ac4grtI5
```

Follow the steps below to launch and use the AutoTrain tool:

1. Log in to your Hugging Face account.

CONSOLE

```
$ huggingface-cli login
```

2. Enter your Hugging Face access `token` and press Enter to proceed. Then, press N and Enter to skip adding your access `token` as a `git` credential.

## CONSOLE

```
...  
Login successful
```

3. Initialize the following custom environment variables. Replace `YOUR_HUGGING_FACE_ACCESS_TOKEN` and `YOUR_HUGGING_USERNAME` with the correct values. The `smangrul/ad-copy-generation` parameter points to the [ad copy generation](#) labeled dataset discussed earlier.

## CONSOLE

```
$ export model=meta-llama/Llama-2-7b-chat-hf  
export data_path=smangrul/ad-copy-generation  
export text_column=content  
export token=YOUR_HUGGING_FACE_ACCESS_TOKEN  
export project_name=demo-ads  
export repoid=YOUR_HUGGING_FACE_USERNAME/ads-llama2-demo
```

4. Run the following command to train the `Llama-2-7b-chat-hf` model with the new dataset. The following command takes several minutes to complete.

## CONSOLE

```
$ autotrain llm  
  --train  
  --project_name $project_name  
  --model $model  
  --data_path $data_path  
  --text_column $text_column  
  --use_peft  
  --use_int4  
  --learning_rate 2e-4  
  --train_batch_size 2  
  --num_train_epochs 1  
  --trainer sft  
  --block_size 1024  
  --lora_r 16  
  --lora_alpha 32  
  --merge-adapter  
  --lora_dropout 0.045  
  --push-to-hub
```

```
--token $token  
--repo-id $repoid
```



Output:

```
...  
> INFO    Finished training, saving model...  
> INFO    Merging adapter weights...  
> INFO    Loading adapter...  
...
```

5. Visit your Hugging Face profile page to ensure you've trained and pushed the new model to the repository.

```
https://huggingface.co/YOUR_HUGGING_FACE_USERNAME
```

Output:

 **Models** 1 

  /ads-llama2-demo private  
 Text Generation • Updated 30 minutes ago

## Serve the Custom Auto-trained Model

```
https://www.youtube.com/embed/l1wsE4bHK6M?si=uSvM1Sk9HJkkEX7A
```

You've fine-tuned and pushed the model to your Hugging Face account. Follow the steps below to run the model in a Docker container:

1. Initialize the following system variables. Replace `YOUR_HUGGING_FACE_USERNAME` and `YOUR_HUGGING_FACE_TOKEN` with the correct details.

CONSOLE

```
$ export model=YOUR_HUGGING_FACE_USERNAME/ads-llama2-demo
export volume=$PWD/data
export token=YOUR_HUGGING_FACE_TOKEN
```

2. Run the following Docker command to run a container that serves the new model.

#### CONSOLE

```
$ sudo docker run -d
--name hf-tgi
--runtime=nvidia
--gpus all
-e HUGGING_FACE_HUB_TOKEN=$token
-p 8080:80
-v $volume:/data
ghcr.io/huggingface/text-generation-inference:1.1.0
--model-id $model
--max-input-length 2048
--max-total-tokens 4096
```

#### Output:

```
1.1.0: Pulling from huggingface/text-generation-inference
...
Status: Downloaded newer image for ghcr.io/huggingface/text-generation-inference:
1.1.0
```

3. Check the Docker logs to monitor the container as it loads the new model.

#### CONSOLE

```
$ sudo docker logs -f hf-tgi
```

#### Output:

```
...
Warming up model
Setting max batch total tokens to .....
```

```
...Connected
...Invalid hostname, defaulting to 0.0.0.0
```

- Allow port `8890` through the firewall. You need this port to run a Jupyter lab instance.

**CONSOLE**

```
$ sudo ufw allow 8890
$ sudo ufw reload
```

- Run the Jupyter lab instance to retrieve an access `token`.

**CONSOLE**

```
$ jupyter lab --ip 0.0.0.0 --port 8890
```

Output:

```
http://YOUR_SERVER_HOST_NAME:8890/lab?
token=b7ab2bdscb366edsddssfsff0faeb5fa68b6b0cf
```

- Access the Jupyter Notebook on your browser.

```
http://YOUR_SERVER_HOST_NAME:8890/lab?
token=b7ab2bdscb366edsddssfsff0faeb5fa68b6b0cf
```

- Click **Python 3 ipykernel** under **Notebook**. Then, enter the following Python code.

**PYTHON**

```
from huggingface_hub import InferenceClient

URI = 'http://127.0.0.1:8080'
client = InferenceClient(model = URI)

def chat_completion(system_prompt, user_prompt, length =
1000):
    final_prompt = f"""<s>[INST]<<SYS>>
```

```
        {system_prompt}<</SYS>> {user_prompt}
        [/INST]
        """

    return client.text_generation(prompt =
final_prompt,max_new_tokens = length).strip()

system_prompt = """
        Create a text ad given the following product
and description.
        """

user_prompt    = """
        Product: Lightweight Bicycle Description:
Perfect bike for any road conditions. this bike comes with
durable tires, LCD display, and modern disc brakes
        """

print(chat_completion(system_prompt, user_prompt))
```

8. Run the above Python code to generate an ad based on your input.

Output:

```
🚴🚴 Experience the thrill of cycling with a Lightweight Bicycle! 🚴🚴 Perfect for
road adventures, this bike boasts durable tires, LCD display, and modern disc
brakes. Limited stock - ride in style! 🚴🚴
```

## Conclusion

In this guide, you've explored the difference between fine-tuning a model using RAG and labeled datasets to improve the model's accuracy. Then, you've implemented the HuggingFace AutoTrain tool to fine-tune the model with a new dataset. Finally, you've run the fine-tuned model in a Docker container to generate an ad based on a custom product description.



VULTR

