



aiDAPTIV+ GUI

Pro Suite User guide

Version 1.1

MiPhi Semiconductors Pvt Ltd

Tel: +91 89044 05190 // +91 80421 66207
E-mail: inquire@miphi.in

ALL RIGHTS ARE STRICTLY RESERVED. ANY PORTION OF THIS PAPER SHALL NOT BE REPRODUCED, COPIED, OR TRANSLATED TO ANY OTHER FORMS WITHOUT PERMISSION FROM MIPHI SEMICONDUCTORS PVT LTD.

MiPhi may make changes to specifications and product description at any time without notice. MIPHI and the MiPhi logo are trademarks of MiPhi Semiconductors Pvt Ltd, registered in the United States and other countries. Products and specifications discussed herein are for reference purposes only. Copies of documents which include information of part number or ordering number, or other materials may be obtained by emailing us at inquire@miphi.in .

©2024 MiPhi Semiconductors Pvt Ltd. All Rights Reserved.

Contents

1. ENVIRONMENT PREPARATION	6
2. OVERVIEW.....	6
2.1 FEATURE OVERVIEW	6
3. MIPHI LOGIN PAGE.....	7
3.1 Sign In to MiPhi.....	7
3.2 Reset Your Password	8
3.3 User Menu & Account Settings.....	8
3.4 User Management (Admin Only)	10
4. MIPHI SYSTEM DASHBOARD	12
4.1 Operating System Overview.....	12
4.2 CUDA & Kernel Information	12
4.3 GPU Information	13
4.4 CPU Details.....	13
4.5 DRAM (Memory) Specifications	13
4.6 MiPhi aiDAPTIV+ Cache Information	13
4.7 Storage Utilization Summary	14
5. MIPHI DATASETS PAGE.....	15
5.1 Upload Area	15
5.1.1 Select and Upload Your Dataset.....	16
5.1.2 Review & Upload Dataset	16
5.1.3 Upload Complete	17
5.1.4 File Already Exists	17
5.1.5 Validation Errors.....	18
5.1.6 Upload Multiple Datasets.....	18
5.2 List of Available Datasets.....	19
5.2.1 Previewing a Dataset.....	19
5.2.2 Downloading a Dataset	20
5.2.3 Deleting a Dataset	21
5.2.4 Searching a Dataset	21
6. MIPHI MODELS PAGE.....	22
6.1 Upload Area	22
6.1.1 Select and Upload Your Model	23

6.1.2 File Already Exists	23
6.1.3 Upload Multiple Models	24
6.1.4 Upload Complete	24
6.2 List of Available Models	25
6.2.1 Model Preview	25
6.2.2 Deleting a Model.....	26
6.3.3 Searching a Model	26
7. MIPHI FINE-TUNING PAGE	27
7.1 Model Configuration Section	27
7.1.1 Select Model	27
7.1.2 Select Dataset.....	27
7.1.3 Task type	28
7.1.4 Max sequence length	28
7.1.5 Weight File Format	29
7.1.6 Load Model Structure From Config:.....	29
7.2 Path Configuration Section.....	29
7.2.1 aiDAPTIV+ Cache Mount Point.....	29
7.2.2 Output_dir	29
7.2.3 Log_name	29
7.3 Hardware Configuration Section	30
7.3.1 Select GPUs.....	30
7.3.2 Precision Mode.....	30
7.3.3 Enable Triton	31
7.4 Training Parameters Section.....	32
7.4.1 Train Batch Size	32
7.4.2 Total Batch Size	32
7.4.3 Epoch	32
7.4.4 Max Iteration	32
7.5 Optimizer Settings Section	33
7.5.1 Learning rate	33
7.5.2 LR Scheduler Mode.....	33
7.5.3 Beta1	34
7.5.4 Beta2	34
7.5.5 Epsilon	34
7.5.6 Weight decay.....	34
7.6 LoRA Configuration Section	35

7.6.1 Enable LoRa	35
7.6.2 LoRa rank	35
7.6.3 LoRa alpha	35
7.6.4 LoRa Task Type	35
7.7 Configuration Summary	36
8. MiPhi Monitor Page	37
8.1 Training Statistics	37
8.2 Elapsed Time	38
8.3 Epoch and Micro-batch	38
8.4 NVMe Device Details	38
8.5 GPU Monitoring	39
8.6 Stop Training	39
8.7 Job History	39
8.8 Training Logs	41
8.9 System Metrics	41
8.10 Training Graph	41
8.11 Live GPU Statistics	41
9. MIPHI INFERENCE PAGE	42
9.1 Selecting a Model for Inference	42
9.1.1 Components of Model Table	42
9.1.2 Model Actions	43
9.1.3 Model Configuration	43
9.1.4 Pin the Model	44
9.1.5 Navigate to Chat Bot	45
9.1.6 Create New Chat	46
9.1.7 Start Your Conversation	46
9.2 Model Quantization from GUI	46
9.2.1 Selecting a model for quantization	46
9.2.2 Adjusting Model Parameters for Quantization	47
9.2.3 Monitor Quantization Progress	48
10. MIPHI LOGS OVERVIEW	50
10.1 Log File Selection	50
10.2 Real-Time Analysis	51
10.3 Clear Selection	51
10.4 Download Logs	51

1. ENVIRONMENT PREPARATION

To ensure optimal performance and compatibility, the following system requirements must be met:

- Supported Operating System: Ubuntu 22.04.4 LTS
- Supported NVIDIA Driver Version: 535

2. OVERVIEW

MiPhi aiDAPTIV GUI is an advanced, user-friendly platform designed to streamline the AI development lifecycle. It provides an intuitive graphical interface for managing datasets, models, fine-tuning processes, and inference workflows.

2.1 FEATURE OVERVIEW

The MiPhi aiDAPTIV GUI includes the following core functionalities:

- Login
- System Dashboard
- Datasets
- Models
- Fine-tuning
- Monitor
- Inference
- Logs

3. MIPHI LOGIN PAGE

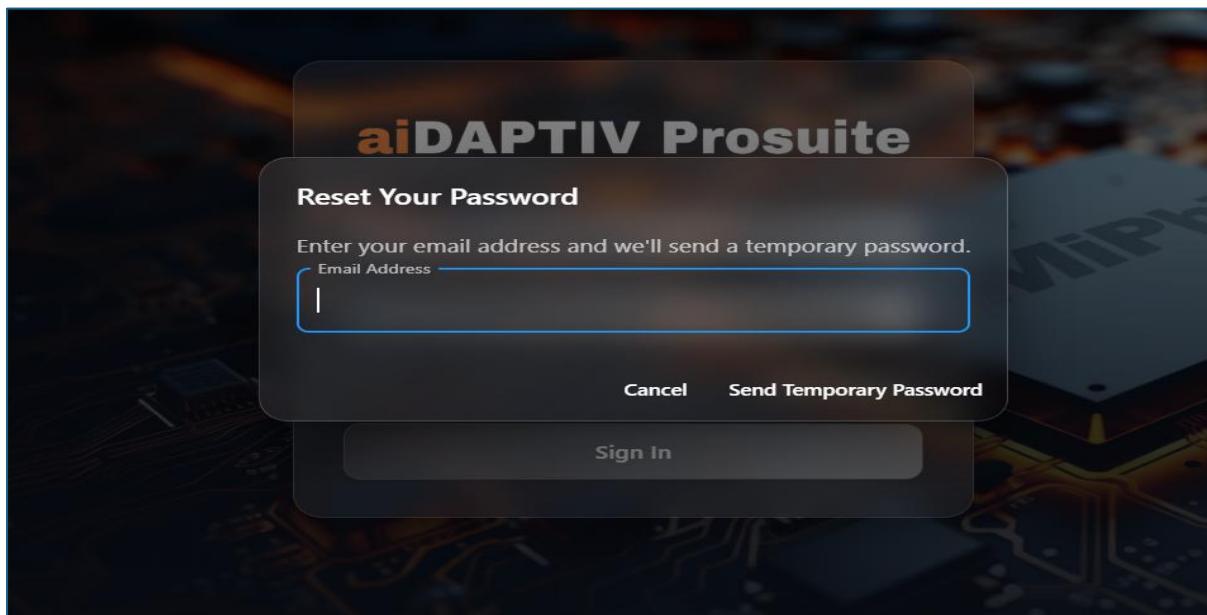
The **MiPhi Login** is the secure gateway to your MiPhi environment. All user accounts—including role-based credentials—are provisioned and managed by a default admin user. Once the admin has created your username and initial password, simply enter those credentials here to access the system. This centralized control ensures that only authorized personnel can log in, while still allowing the admin to delegate additional user-creation or management tasks as needed.

3.1 Sign In to MiPhi

- Enter the Email and Password provided by your admin account to authenticate.
- Use the “

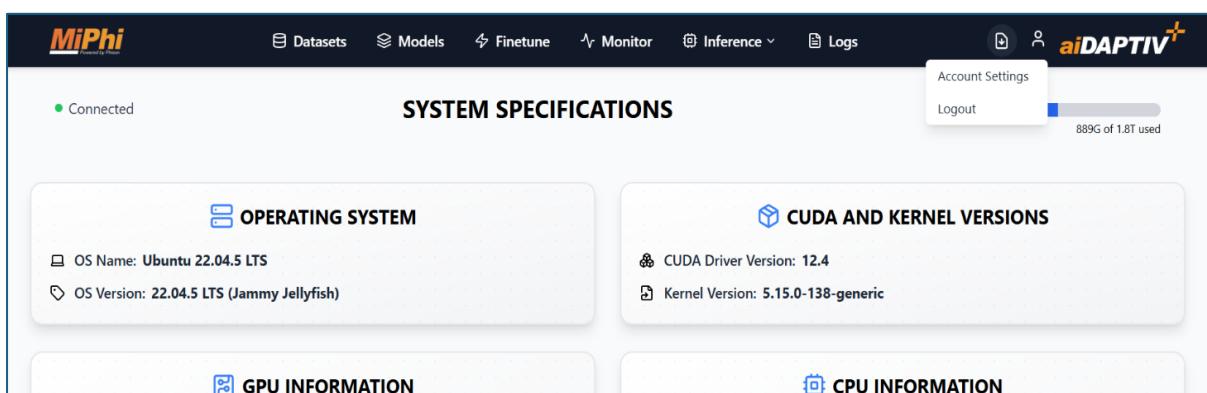
3.2 Reset Your Password

- Enter the Email Address associated with your MiPhi account to request a temporary login credential.
- Click Send Temporary Password to have a one-time password sent to your inbox.
- Use that temporary password to sign in, then update your credentials to something more secure once you're logged in.



3.3 User Menu & Account Settings

- Click the user-icon in the top-right to open the dropdown.
- Select Account Settings to manage your profile or Logout to end your session.
- The Account Settings page has two tabs—General and Security—for profile and password management.



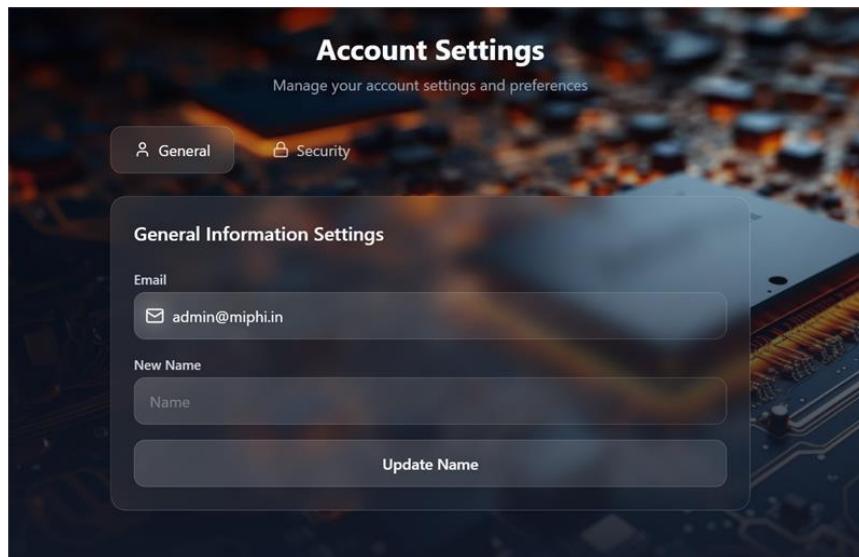
SYSTEM SPECIFICATIONS

OPERATING SYSTEM	CUDA AND KERNEL VERSIONS
OS Name: Ubuntu 22.04.5 LTS	CUDA Driver Version: 12.4
OS Version: 22.04.5 LTS (Jammy Jellyfish)	Kernel Version: 5.15.0-138-generic

GPU INFORMATION	CPU INFORMATION
-----------------	-----------------

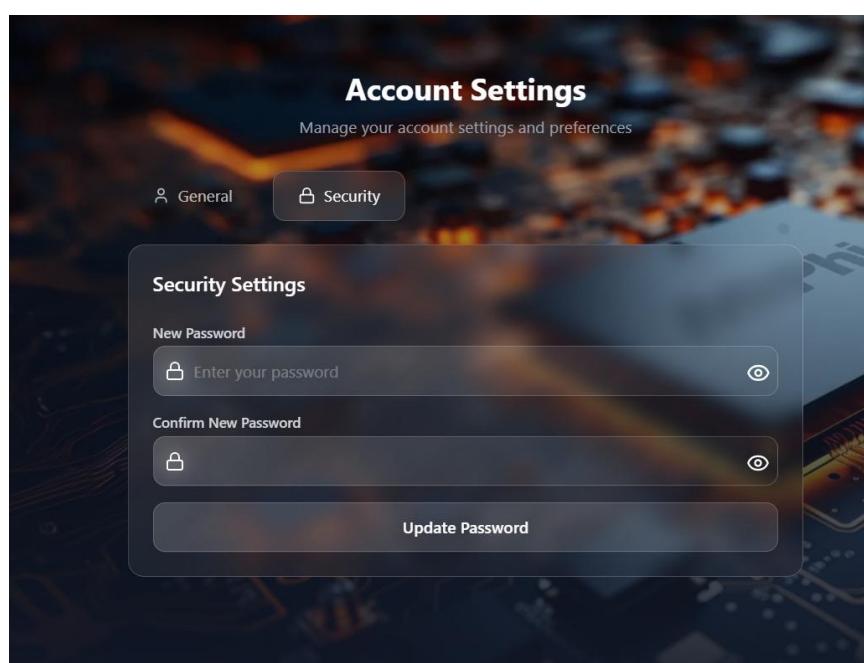
3.3.1 General Settings

- Email (read-only): Displays your registered address.
- New Name field: Enter a new display name.
- Update Name button: Apply and save your name change instantly.



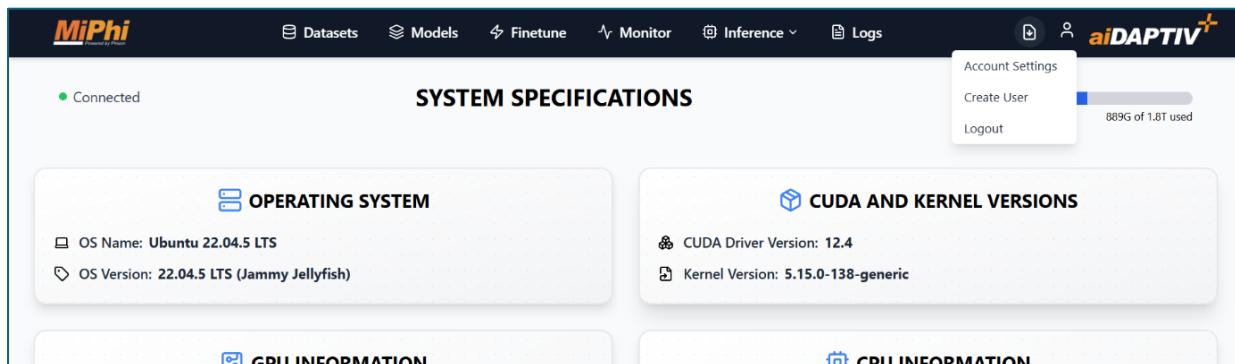
3.3.2 Security Settings

- Switch to the Security tab under Account Settings.
- Enter your New Password, then repeat it in Confirm New Password.
- Use the “” icon to toggle visibility and avoid typos.
- Click Update Password to save your changes.



3.4 User Management (Admin Only)

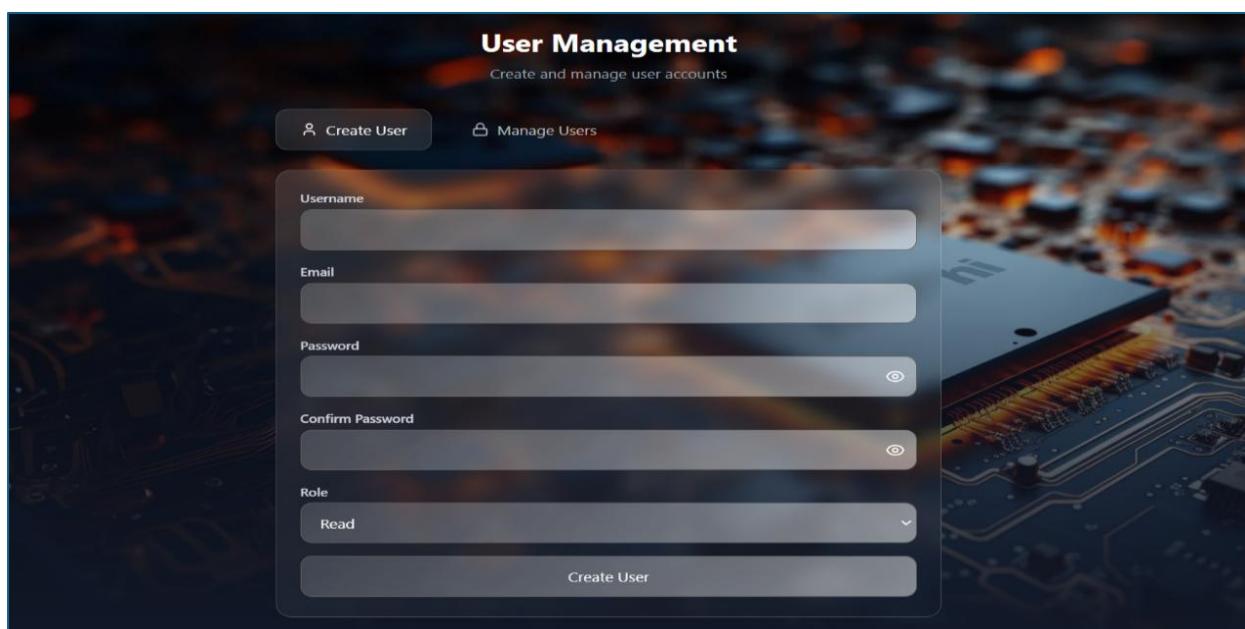
- Accessible via the user-icon menu as Create User and Manage Users.
- Empowers the admin to provision new accounts or review and delete existing ones.
- Ensures fine-grained control over who can log in and what role they receive.



The screenshot shows the 'SYSTEM SPECIFICATIONS' section of the MiPhi interface. It includes four main sections: 'OPERATING SYSTEM' (OS Name: Ubuntu 22.04.5 LTS, OS Version: 22.04.5 LTS (Jammy Jellyfish)), 'CUDA AND KERNEL VERSIONS' (CUDA Driver Version: 12.4, Kernel Version: 5.15.0-138-generic), 'GPU INFORMATION', and 'CPU INFORMATION'. A user menu in the top right corner shows 'Account Settings', 'Create User' (with a progress bar at 889G of 1.8T used), and 'Logout'.

3.4.1 Create User

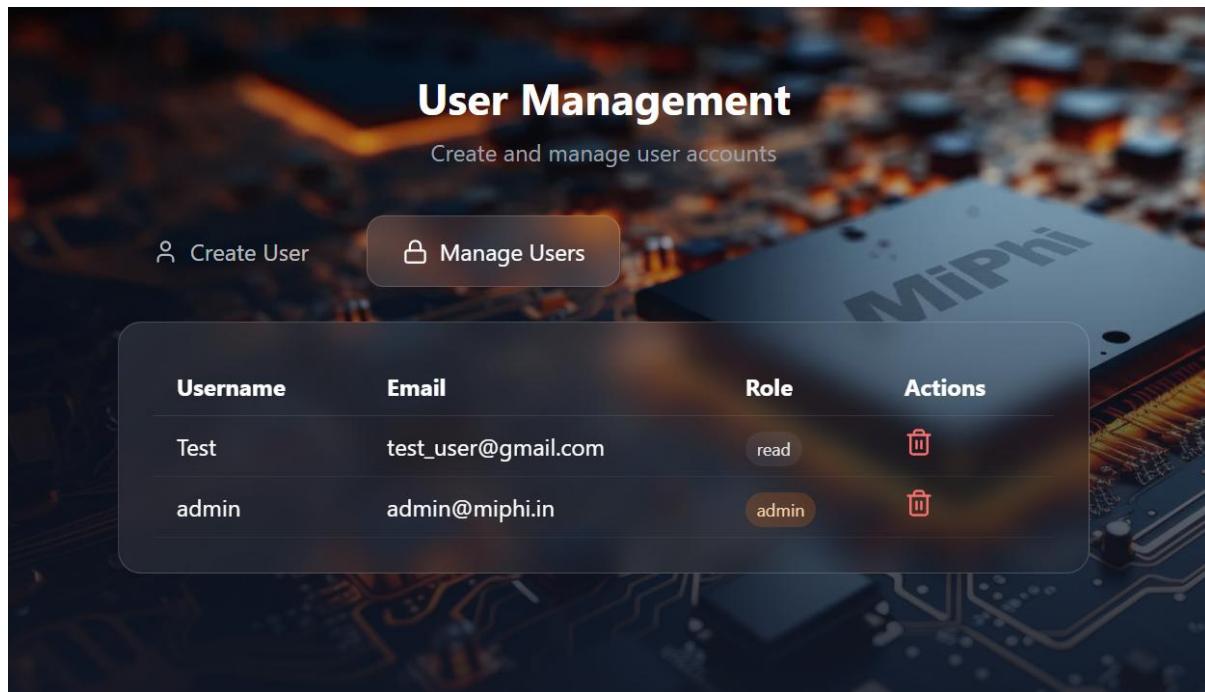
- Username, Email, Password and Confirm Password fields collect the new account details.
- The Role dropdown assigns permissions (e.g. “read”, “write”, “admin”).
- Click Create User to provision the account and send the credentials to the user.



The screenshot shows the 'User Management' form. It has two buttons at the top: 'Create User' (highlighted) and 'Manage Users'. The form itself contains fields for 'Username', 'Email', 'Password', 'Confirm Password', and a 'Role' dropdown set to 'Read'. At the bottom is a large 'Create User' button.

3.4.2 Manage Users

- Displays a table of all current users with Username, Email, and Role.
- The  Delete icon removes an account permanently (with confirmation).
- Use this view to audit active users and revoke access instantly.

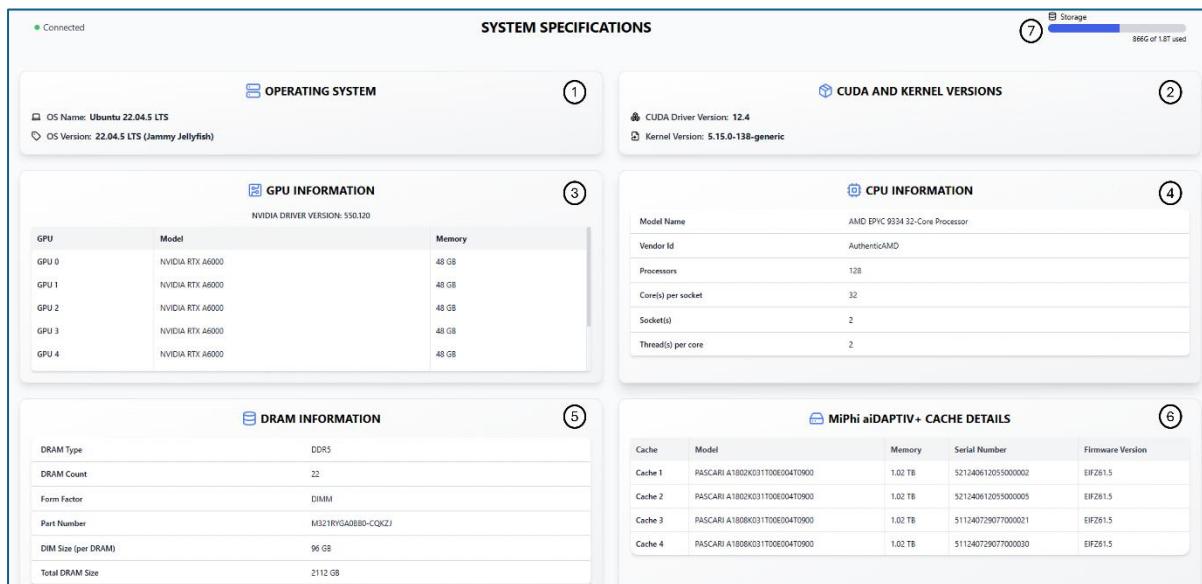


The screenshot shows the MiPhi User Management interface. At the top, there is a title "User Management" and a subtitle "Create and manage user accounts". Below this, there are two buttons: "Create User" and "Manage Users". The "Manage Users" button is highlighted with a rounded rectangle. A table below the buttons displays user information:

Username	Email	Role	Actions
Test	test_user@gmail.com	read	
admin	admin@miphi.in	admin	

4. MIPHI SYSTEM DASHBOARD

The **MiPhi System Dashboard** delivers a unified and detailed view of the system's hardware and software specifications. Designed for users to facilitate real-time monitoring and management of compute nodes within high-performance environments. This dashboard dynamically reflects the current server configuration, ensuring accurate and up-to-date information for performance analysis, troubleshooting, and capacity planning.



The screenshot displays the MiPhi System Dashboard interface with the following sections:

- SYSTEM SPECIFICATIONS** (Top Center): Shows a status indicator "Connected" and a storage usage bar indicating 86G of 1.8T used.
- OPERATING SYSTEM** (①): Shows OS Name: Ubuntu 22.04.5 LTS and OS Version: 22.04.5 LTS (Jammy Jellyfish).
- CUDA AND KERNEL VERSIONS** (②): Shows CUDA Driver Version: 12.4 and Kernel Version: 5.15.0-138-generic.
- GPU INFORMATION** (③): Shows GPU details for five NVIDIA RTX A6000 GPUs, each with 48 GB of memory.
- CPU INFORMATION** (④): Shows CPU details for an AMD EPYC 9334 32-Core Processor.
- DRAM INFORMATION** (⑤): Shows DRAM details for DDR5 memory, including DRAM Type, Count, Form Factor, Part Number, and Total DRAM Size.
- MiPhi aiDAPTIV+ CACHE DETAILS** (⑥): Shows cache details for four PASCARI A180ZK031T00E004T0900 units, each with 1.02 TB of memory.

4.1 Operating System Overview

- Linux Distribution:** Shows the full name of the Linux OS currently running.
Example: Ubuntu 22.04.5 LTS
- Release Version & Codename:** Indicates the release version and associated codename for easy identification.
Example: 22.04.5 LTS (Jammy Jellyfish)

4.2 CUDA & Kernel Information

- CUDA Driver Version:** Displays the installed CUDA driver version used for GPU-accelerated applications.
Example: CUDA Driver Version: 12.4
- Linux Kernel Version:** Shows the current kernel version running on the system.
Example: Kernel Version: 5.15.0-138-generic

4.3 GPU Information

- **Installed GPUs:** Lists each GPU model available in the system.
Example: NVIDIA RTX A6000
- **Memory per GPU:** Shows how much memory each GPU has.
Example: 48 GB per unit
- **NVIDIA Driver Version:** Indicates the active driver version used by all GPUs.
Example: Driver Version: 550.120

4.4 CPU Details

- **Processor Model & Architecture:** Displays the CPU vendor, model, and core architecture.
Example: AMD EPYC 9334 32-Core Processor (AuthenticAMD)
- **Core & Thread Configuration:** Shows how many cores and threads are present and how they're distributed across sockets.
Example: 128 threads, 32 cores/socket × 2 sockets
- **Per-Core Threading:** Provides thread-level visibility useful for tuning performance or virtualization.

4.5 DRAM (Memory) Specifications

- **Memory Type & Total Capacity:** Indicates the type (e.g., DDR5) and total installed memory.
Example: DDR5, Total DRAM Size: 2112 GB
- **Number of DIMMs:** Displays how many memory modules are installed.
Example: 22 modules
- **Module Details:** Lists form factors, part numbers, and sizes.
Example: M321RYGA0B80-CQKZJ – 96 GB each

4.6 MiPhi aiDAPTIV+ Cache Information

- **Cache Unit Identification:** Identifies each aiDAPTIV+ Cache unit by name (Cache 1 – Cache 4).

- **Model & Specifications:** Provides device model numbers, capacity, and firmware version.

Example: PASCARI A1802K031T00E004T09000 – 1.02 TB – Firmware: EIFZ61.5

- **Serial Numbers:** Enables tracking and service reference.

4.7 Storage Utilization Summary

- **Boot Drive Status:** Graphically and numerically shows how much storage is used vs. available.

Example: 866 GB of 1.8 TB used

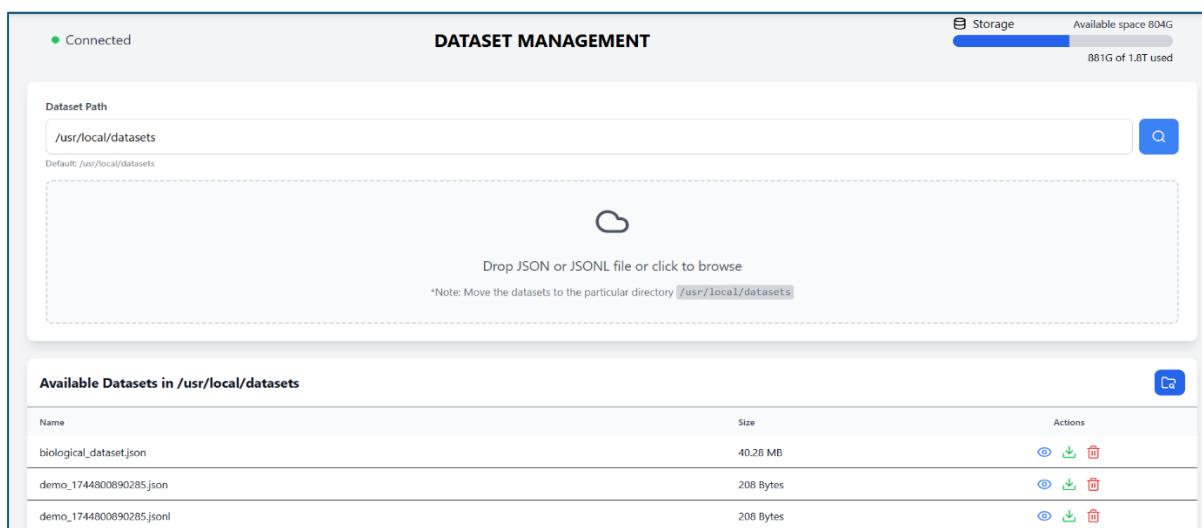
- **Visual Indicators:** Color-coded bar (e.g., blue/red) represents usage health.

● **Blue:** Usage is **below 90%** – healthy range

● **Red:** Usage is **90% or above** – critical threshold; action recommended

5. MIPHI DATASETS PAGE

The **MiPhi Datasets** delivers a streamlined interface for managing your local JSON and JSONL files. Designed around two core areas—a drag-and-drop (or click-to-browse) upload zone and a live directory listing of available datasets—it makes it easy to onboard new data and keep track of existing files. The upload area enforces your chosen dataset path, while the dataset list dynamically reflects every file in that folder, complete with size information and quick-access actions for previewing, downloading, or deleting.

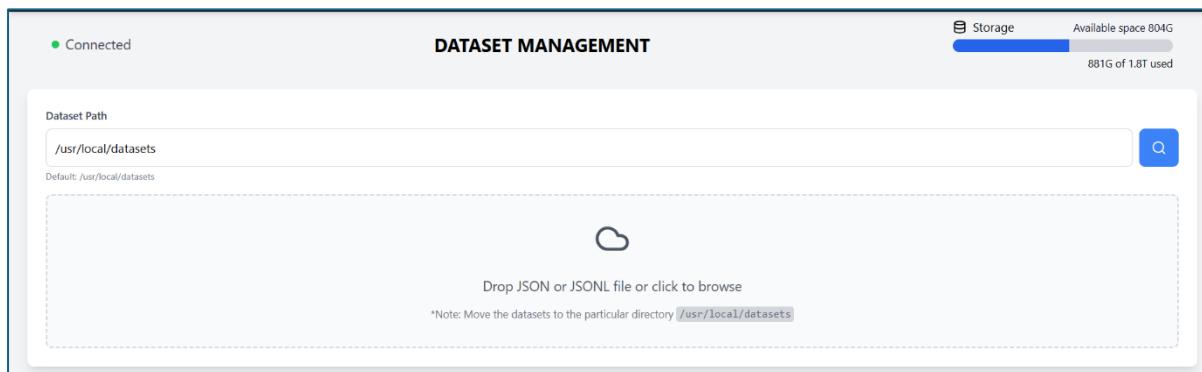


The screenshot shows the 'DATASET MANAGEMENT' page. At the top, there's a status bar with 'Connected' and storage information: 'Storage Available space 804G' and '881G of 1.8T used'. Below this is a 'Dataset Path' input field set to '/usr/local/datasets' with a search icon. A large dashed box labeled 'Drop JSON or JSONL file or click to browse' is centered, with a note below it: '*Note: Move the datasets to the particular directory /usr/local/datasets.' To the right is a table titled 'Available Datasets in /usr/local/datasets' with three rows:

Name	Size	Actions
biological_dataset.json	40.26 MB	
demo_1744800890285.json	208 Bytes	
demo_1744800890285.jsonl	208 Bytes	

5.1 Upload Area

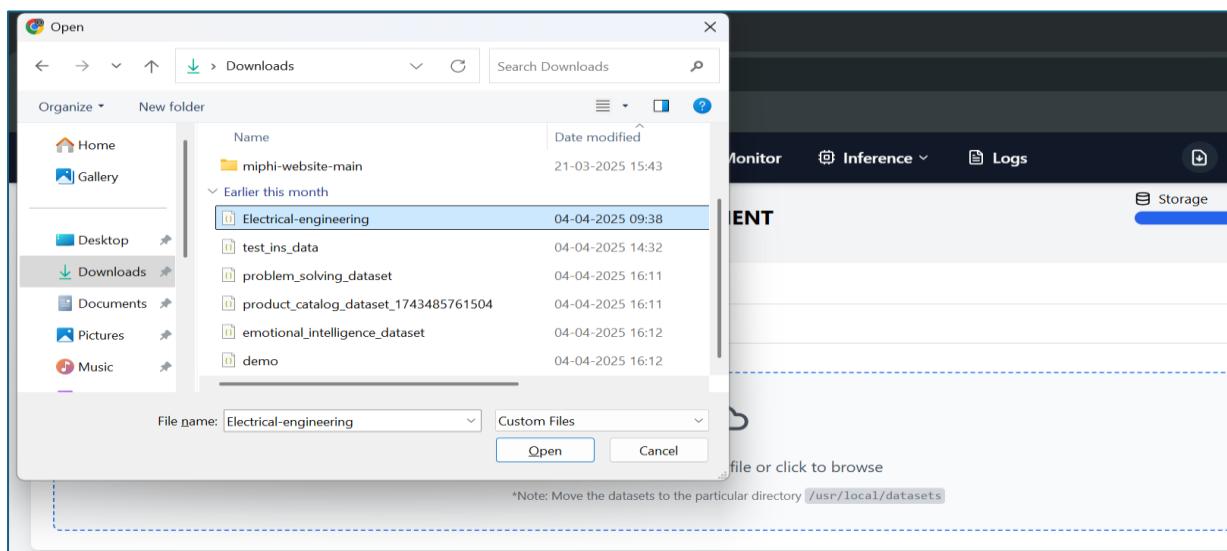
- This section allows you to upload datasets in JSON or JSONL format.
- You can either drag and drop a JSON/JSONL file or click to browse and select one.
- The default dataset storage path is /usr/local/datasets, but you can change it as needed.
- In the top-right corner, you'll see storage usage, showing how much space is occupied and how much remains available.



This screenshot is identical to the one above, showing the 'DATASET MANAGEMENT' interface with the same storage information, dataset path, and available datasets table. The main difference is that the upload area (dashed box) is currently empty, indicating no files have been uploaded.

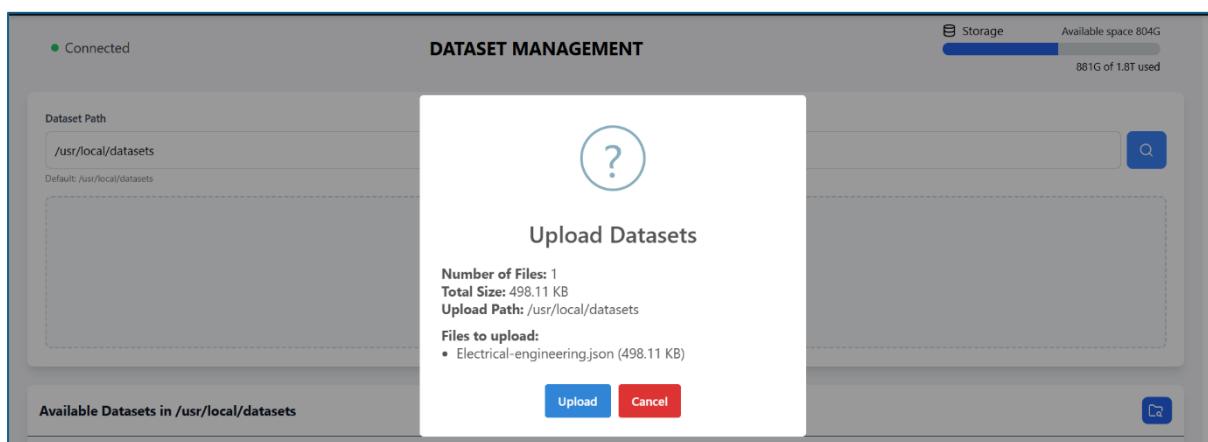
5.1.1 Select and Upload Your Dataset

- Click the Drop JSON/JSONL file or click to browse area to open your system's file picker.
- In the dialog (e.g. browsing Downloads), select your desired .json or .jsonl dataset and hit Open.
- Once chosen, the file name appears in the upload zone and the dataset is copied into /usr/local/datasets for immediate use.



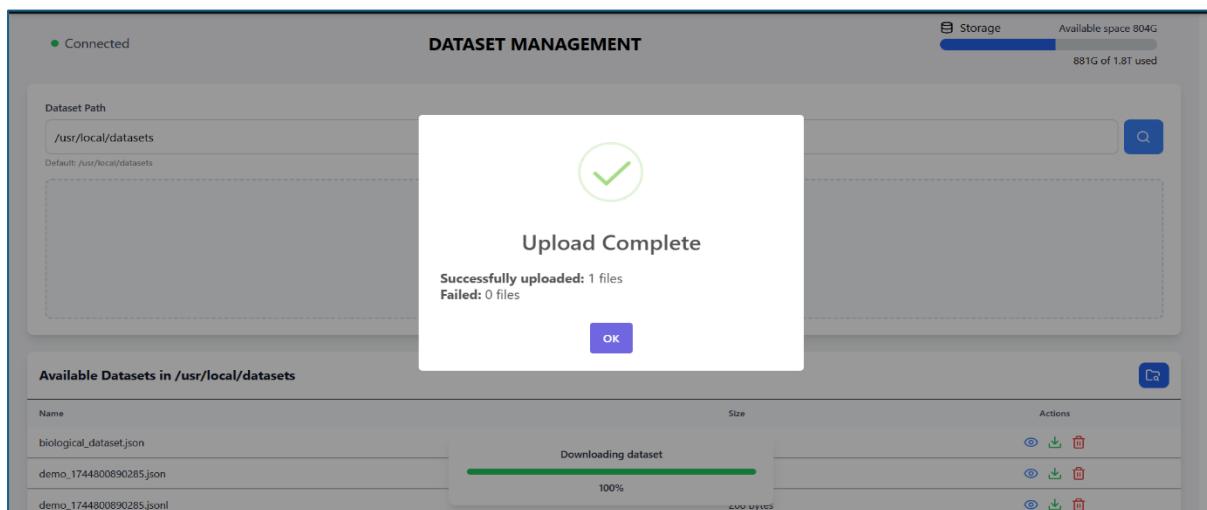
5.1.2 Review & Upload Dataset

- As soon as you pick one or more JSON/JSONL files, a “Upload Datasets” confirmation dialog appears.
- It summarizes your selection: Number of Files, Total Size, and the target Upload Path, followed by a list of the exact filenames and their sizes.
- Click Upload to copy the files into /usr/local/datasets (or Cancel to return and adjust your selection).



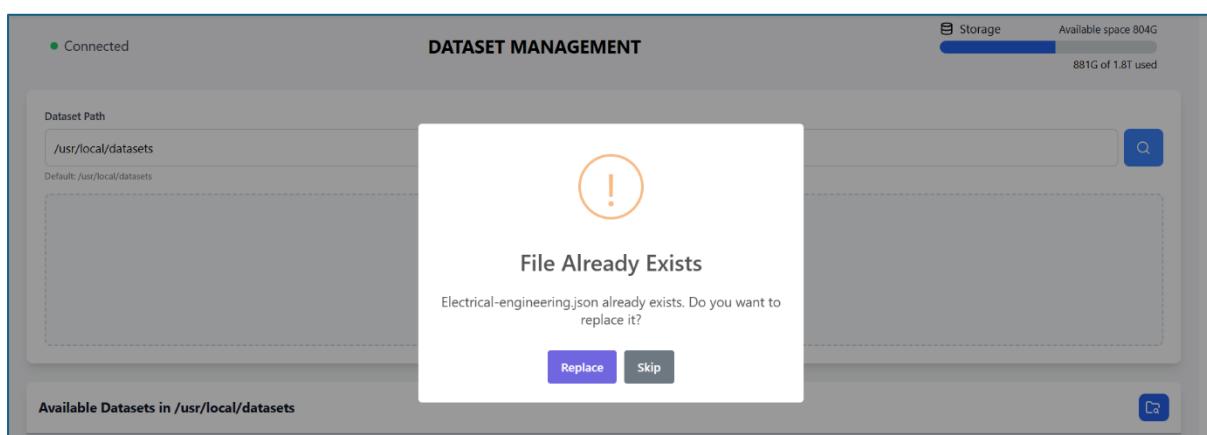
5.1.3 Upload Complete

- The dialog confirms your upload results, showing how many files were successfully uploaded and if any failed.
- Your new datasets have been added to /usr/local/datasets and will appear in the “Available Datasets” list once you click OK.
- Click OK to dismiss and refresh the page, updating storage usage and file listings.



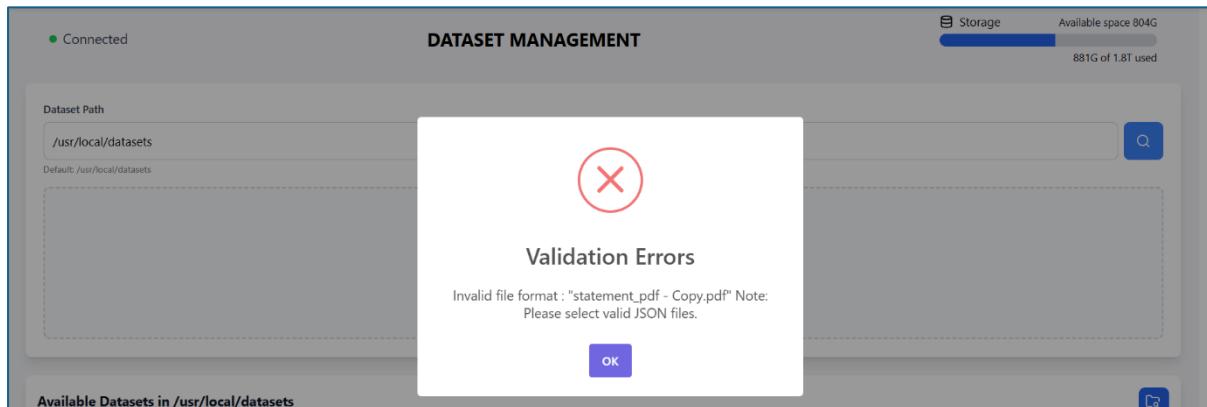
5.1.4 File Already Exists

- The system has detected that Electrical-engineering.json is already present in /usr/local/datasets.
- Click Replace to overwrite the existing file with this new version.
- Click Skip to cancel uploading that file and keep the current copy intact.



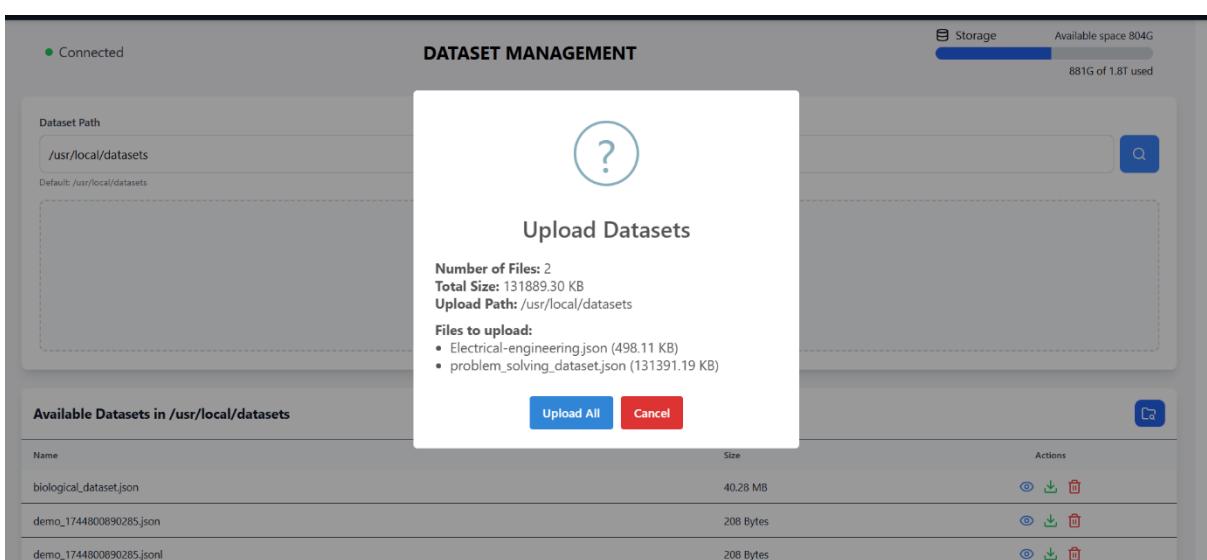
5.1.5 Validation Errors

- The file "statement_pdf - Copy.pdf" is not a valid JSON/JSONL dataset and was rejected.
- Only files with a .json or .jsonl extension and proper JSON structure can be uploaded.
- Click OK, then select a correctly formatted JSON/JSONL file to proceed.



5.1.6 Upload Multiple Datasets

- Click the upload area and, in your file picker, use Ctrl/Cmd + Click or Shift + Click to select several .json or .jsonl files at once.
- All chosen files will be listed in the confirmation dialog along with their individual sizes.
- Hit Upload All to import every dataset in one go, or Cancel to adjust your selection.



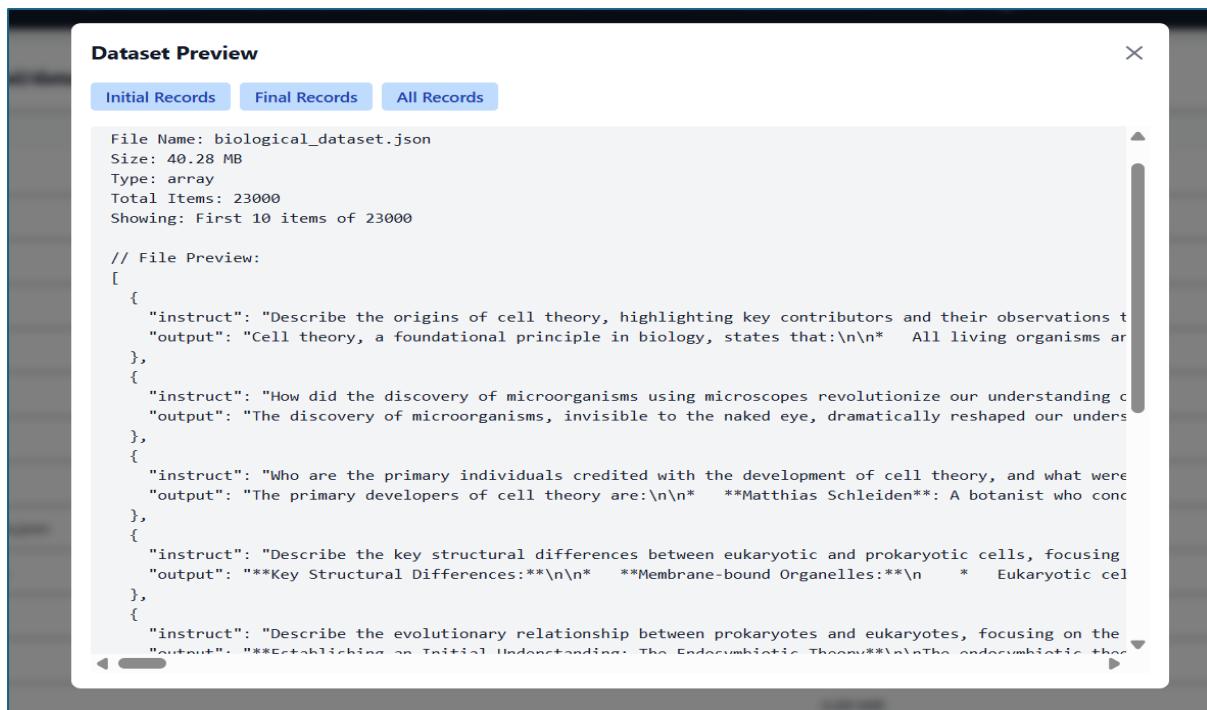
5.2 List of Available Datasets

- Displays every .json and .jsonl file in your selected dataset directory, alongside its size.
- Offers three action icons per row:
 - Preview — view a snippet of the file in a modal
 - Download — save a local copy via your browser
 - Delete — permanently remove the file (with confirmation)
- Automatically updates after any upload or deletion, ensuring you always see the current contents.
- Use the Search icon in the top-right to filter files by name or extension

Available Datasets in /usr/local/datasets		
Name	Size	Actions
biological_dataset.json	40.28 MB	  
demo_1744800890285.json	208 Bytes	  
demo_1744800890285.jsonl	208 Bytes	  
Electrical-engineering (1).json	498.11 KB	  
Electrical-engineering.json	498.11 KB	  
emotional_intelligence_dataset.json	40.4 MB	  
english_to_tamil.json	4.24 MB	  
final_cooking_dataset.json	9.08 MB	  
indian_historical_events_50k_2048_tokens.json	109.23 MB	  
instruction_dataset_1745298843130.json	396.37 KB	  
insuranceQA_dataset.json	14.3 MB	  
large_cs_finetuning_dataset.json	3.16 MB	  
law_india.json	6.89 MB	  

5.2.1 Previewing a Dataset

- Tab controls at the top let you switch between Initial Records, Final Records, or All Records for quick navigation.
- A summary section displays File Name, Size, Type, Total Items, and the current slice (e.g., “Showing: First 10 items of 23000”).
- Below that, a scrollable JSON preview shows the actual array or object entries, allowing in-browser inspection of dataset contents.



Dataset Preview

Initial Records Final Records All Records

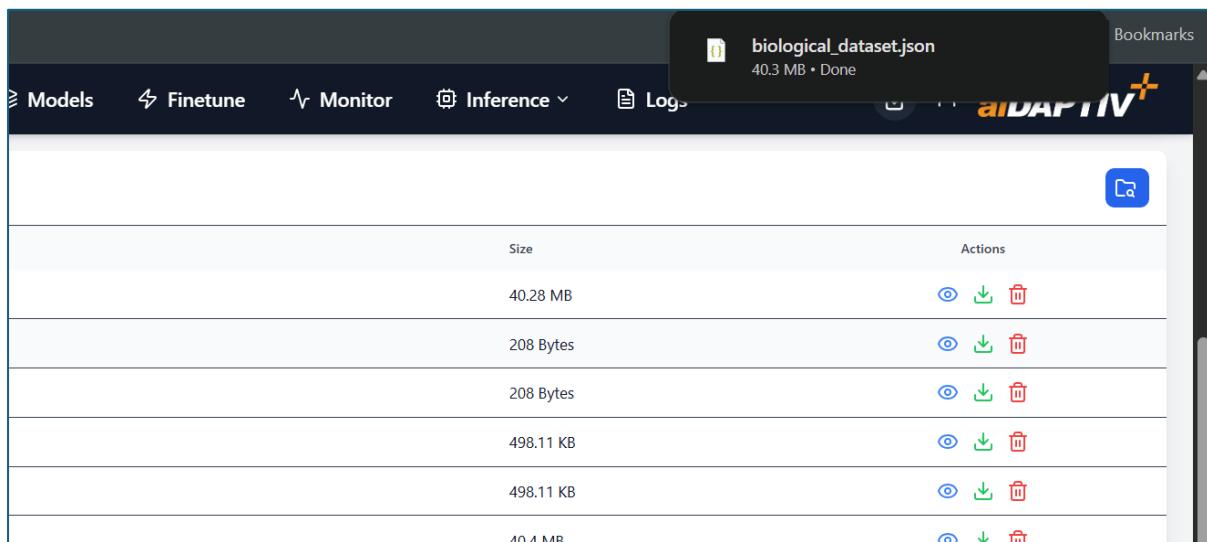
```

File Name: biological_dataset.json
Size: 40.28 MB
Type: array
Total Items: 23000
Showing: First 10 items of 23000

// File Preview:
[
  {
    "instruct": "Describe the origins of cell theory, highlighting key contributors and their observations to support it.",
    "output": "Cell theory, a foundational principle in biology, states that: All living organisms are composed of cells. The theory was developed through the work of Matthias Schleiden, Theodor Schwann, and Rudolf Virchow. Their observations and hypotheses led to the formation of this theory in the early 19th century."
  },
  {
    "instruct": "How did the discovery of microorganisms using microscopes revolutionize our understanding of biology?",
    "output": "The discovery of microorganisms, invisible to the naked eye, dramatically reshaped our understanding of biology. It led to the development of microbiology and provided evidence for the theory of evolution by Charles Darwin, who used the concept of natural selection to explain the diversity of life at the microscopic level."
  },
  {
    "instruct": "Who are the primary individuals credited with the development of cell theory, and what were their contributions to the field?",
    "output": "The primary developers of cell theory are: Matthias Schleiden, Theodor Schwann, and Rudolf Virchow. They contributed to the theory by observing plant and animal cells under microscopes and formulating the basic principles of cell structure and function. Schleiden proposed that all plants are composed of cells, while Schwann and Virchow proposed that all animals are composed of cells. Virchow also proposed the concept of 'cell division' as a mechanism for growth and reproduction in living organisms."
  },
  {
    "instruct": "Describe the key structural differences between eukaryotic and prokaryotic cells, focusing on the nucleus and organelles.",
```

5.2.2 Downloading a Dataset

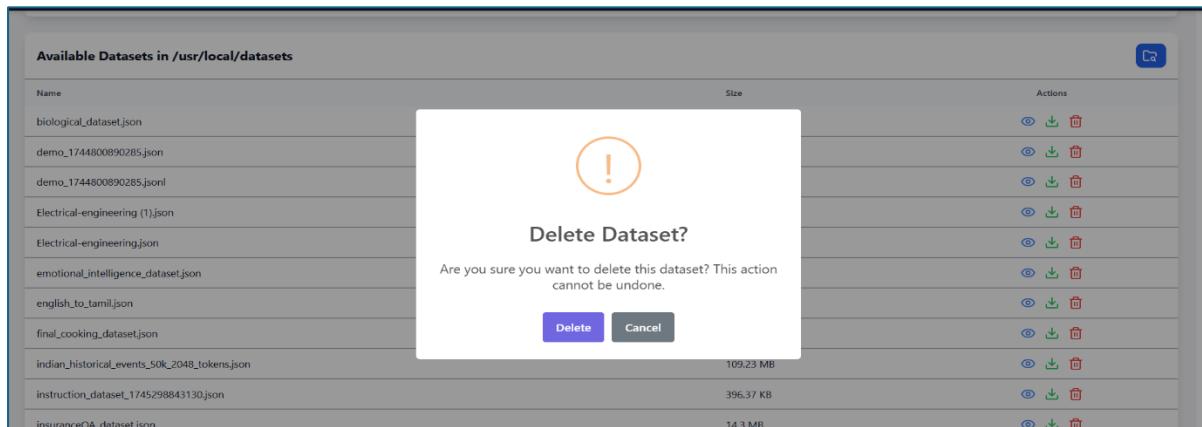
- A browser notification confirms that `biological_dataset.json` (40.3 MB) has finished downloading.
- The file is saved to your local Downloads folder (or your configured default download location).
- Click the notification (or open your Downloads folder) to access and open the dataset immediately.



Size	Actions
40.28 MB	
208 Bytes	
208 Bytes	
498.11 KB	
498.11 KB	
40.4 MB	

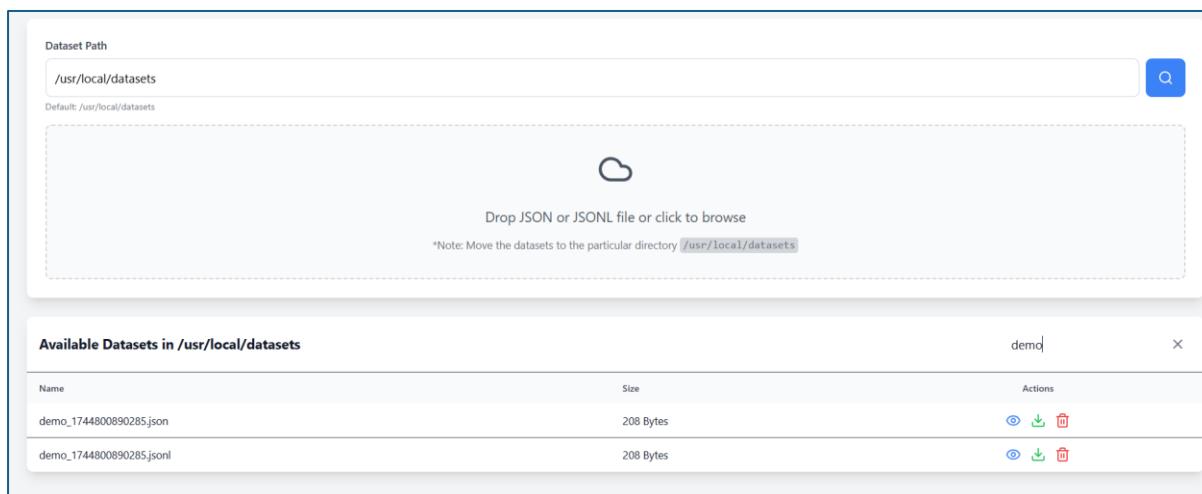
5.2.3 Deleting a Dataset

- You’re about to permanently remove the selected file from /usr/local/datasets; this action cannot be undone.
- Click Delete to confirm and remove the dataset immediately.
- Click Cancel to abort and keep the file safely stored.



5.2.4 Searching a Dataset

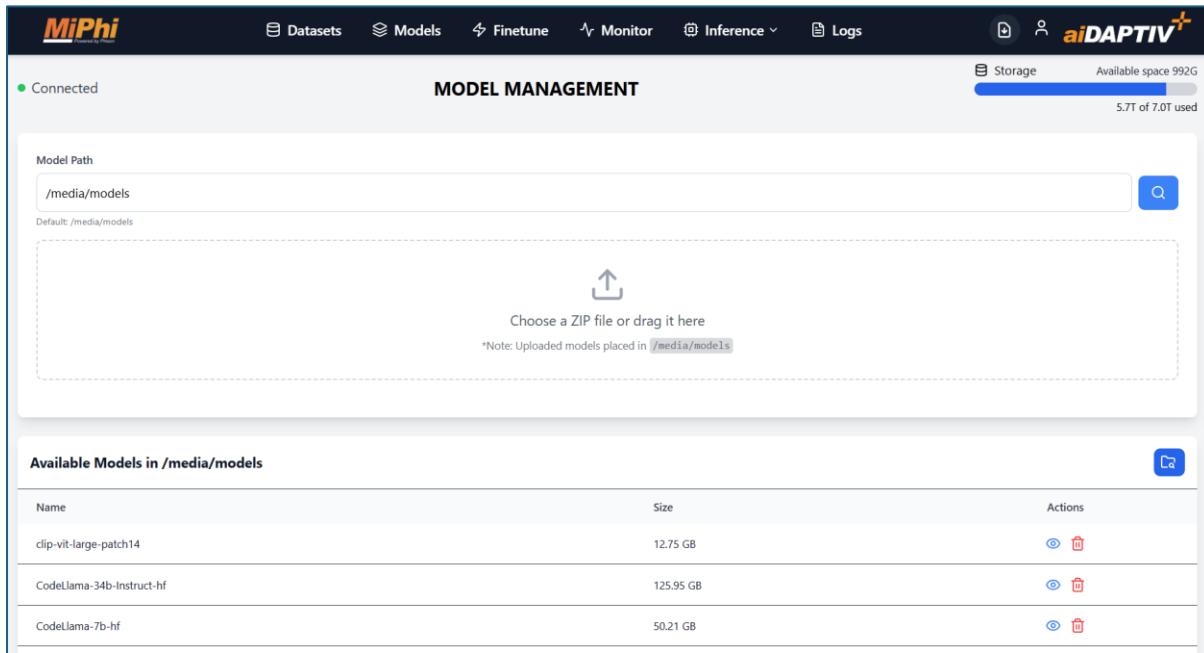
- Use the search box in the top-right of the list to filter files by name or extension—typing “demo” instantly shows only matching entries.
- Results update in real time as you type; hit the × icon to clear your query and restore the full list.
- Supports partial and full filename matches (e.g. entering “.jsonl” will display only JSONL files).



6. MIPHI MODELS PAGE

The **MiPhi Models** is designed to help users manage Large Language Models (LLMs) efficiently through an interactive and organized interface. This page supports operations like loading models from a specific directory, uploading new models, viewing configuration files, and deleting unwanted models.

Model management page consists of two main sections upload area and list area.



The screenshot shows the MiPhi Model Management interface. At the top, there are navigation links: Datasets, Models, Finetune, Monitor, Inference, and Logs. On the right, there's a storage status bar showing "Available space 992G" and "5.7T of 7.0T used". The main area is titled "MODEL MANAGEMENT". It has a "Model Path" input field set to "/media/models" with a search icon. Below it is a dashed box for file upload with an "up arrow" icon and the text "Choose a ZIP file or drag it here". A note below says "*Note: Uploaded models placed in /media/models". The bottom section is titled "Available Models in /media/models" and lists three models:

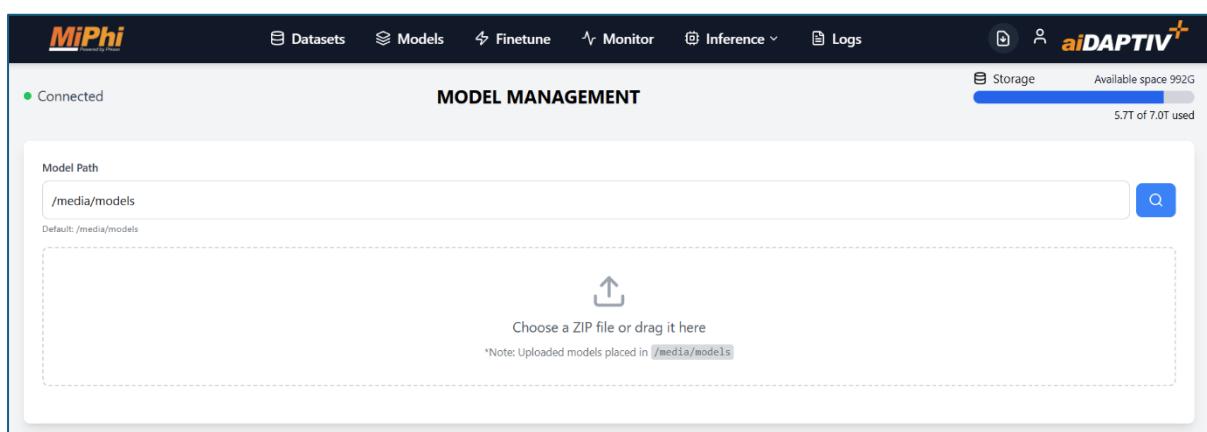
Name	Size	Actions
clip-vit-large-patch14	12.75 GB	
CodeLlama-34b-Instruct-hf	125.95 GB	
CodeLlama-7b-hf	50.21 GB	

6.1 Upload Area

- This section allows you to upload model in **ZIP format only**.
- You can either drag and drop a ZIP file or click to browse and select a file.
- The **default model's storage path** is /media/models, but you can change it as needed.
- If a model with the same name already exists: The system prompts: "**Replace existing model?**"
- Options: Replace | Skip.
- This avoids accidental overwriting and ensures version control.

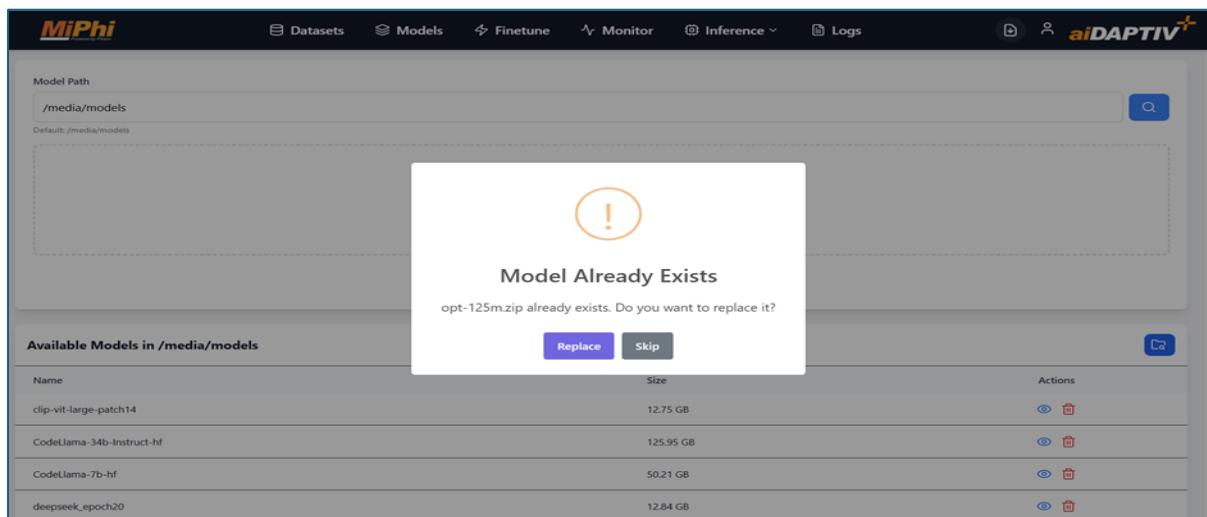
6.1.1 Select and Upload Your Model

- Click the Choose a ZIP file or drag it here file to browse area to open your system's file picker.
- In the dialog (e.g. browsing **Downloads**), select your desired model zip file and hit **Open**.
- Once chosen, the file name appears in the upload zone and the dataset is copied into /media/models for immediate use.



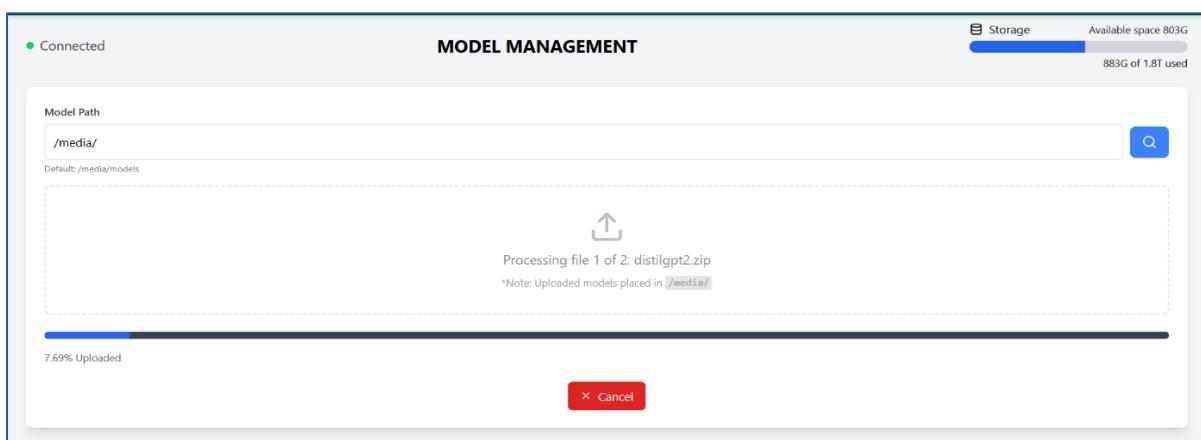
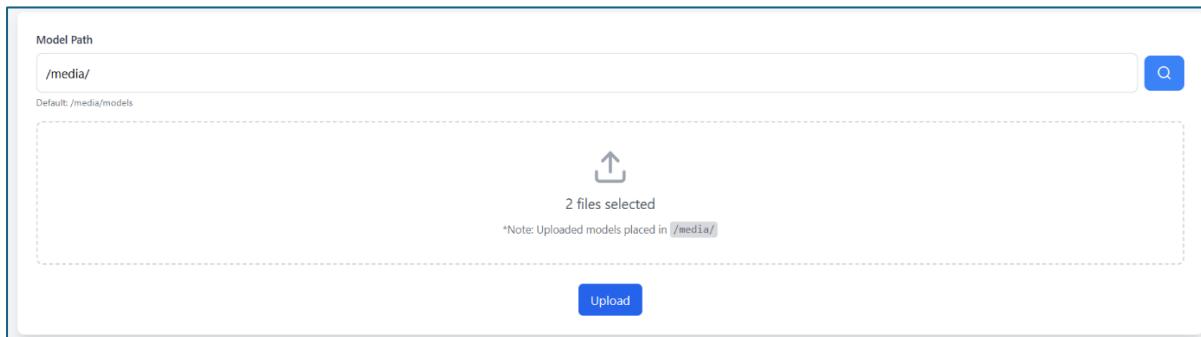
6.1.2 File Already Exists

- The system has detected that **opt-125m Model** is already present in /media/models.
- Click **Replace** to overwrite the existing file with this new version.
- Click **Skip** to cancel uploading that file and keep the current copy intact.



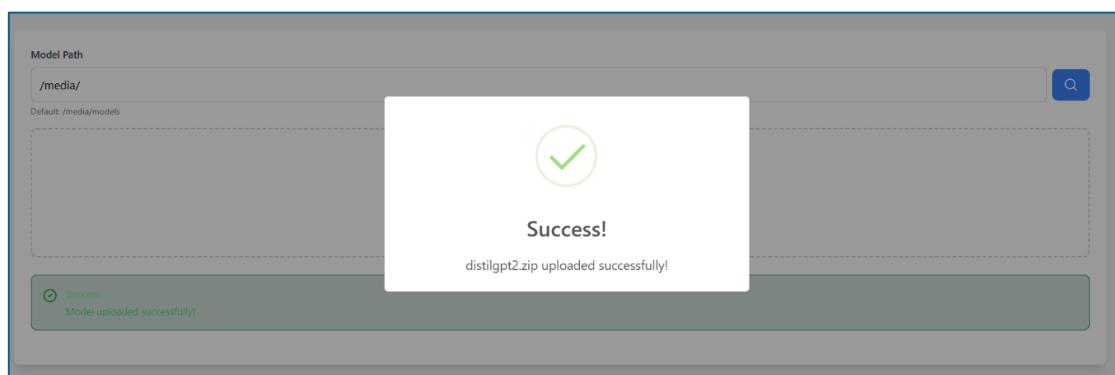
6.1.3 Upload Multiple Models

- Click the upload area and, in your file picker, use Ctrl/Cmd + Click or Shift + Click to select several ZIP files at once.
- All chosen files will be listed in the confirmation dialog along with their individual sizes.
- Hit **Upload** to upload the models one after the other.



6.1.4 Upload Complete

- The dialog confirms your upload results, showing how many files were successfully uploaded and if any failed.
- Your new models have been added to /media/models and will appear in the “Available Models” list once you click OK.
- Click OK to dismiss and refresh the page, updating storage usage and file listings.



6.2 List of Available Models

- Displays all the uploaded models.
- Each dataset entry shows its **name and size**.
- You have the following actions available:

View – To check the model contents.

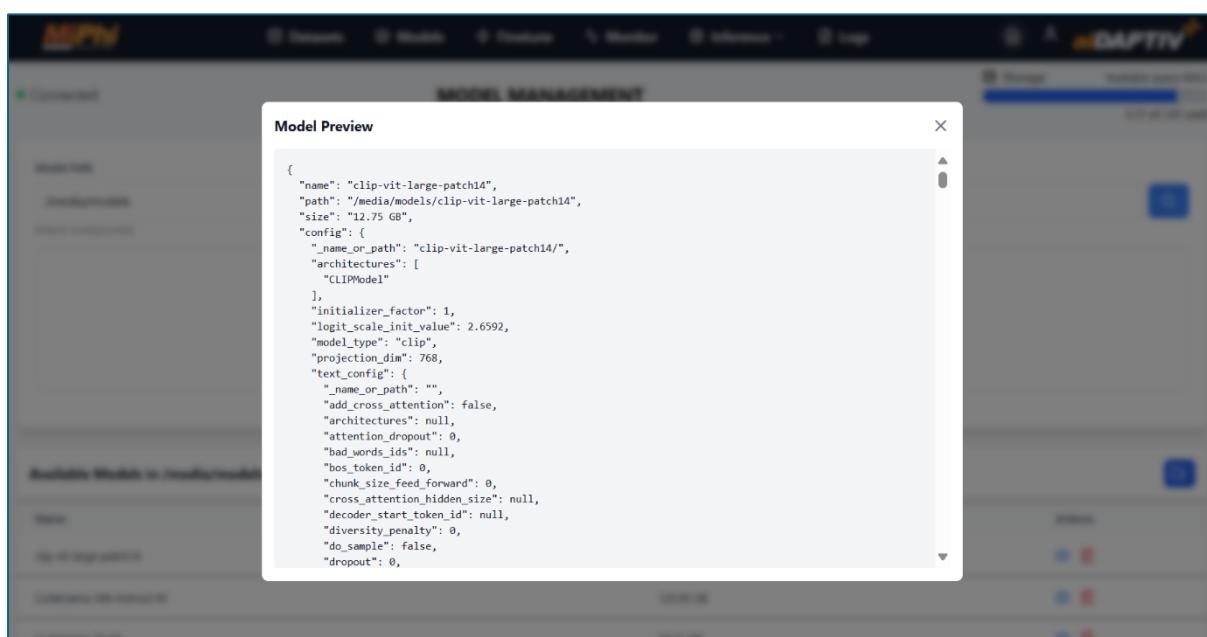
Delete – To remove the model from storage.

- A **search bar** is provided to quickly find specific models.

Available Models in /media/models		
Name	Size	Actions
clip-vit-large-patch14	12.75 GB	 
CodeLlama-34b-Instruct-hf	125.95 GB	 
CodeLlama-7b-hf	50.21 GB	 
deepseek_epoch20	12.84 GB	 
deepseek_llm-67b-chat	125.59 GB	 
deepseek_llm-7b-chat	12.88 GB	 
DeepSeek-R1	641.31 GB	 
DeepSeek-R1-Distill-Llama-8B	29.93 GB	 

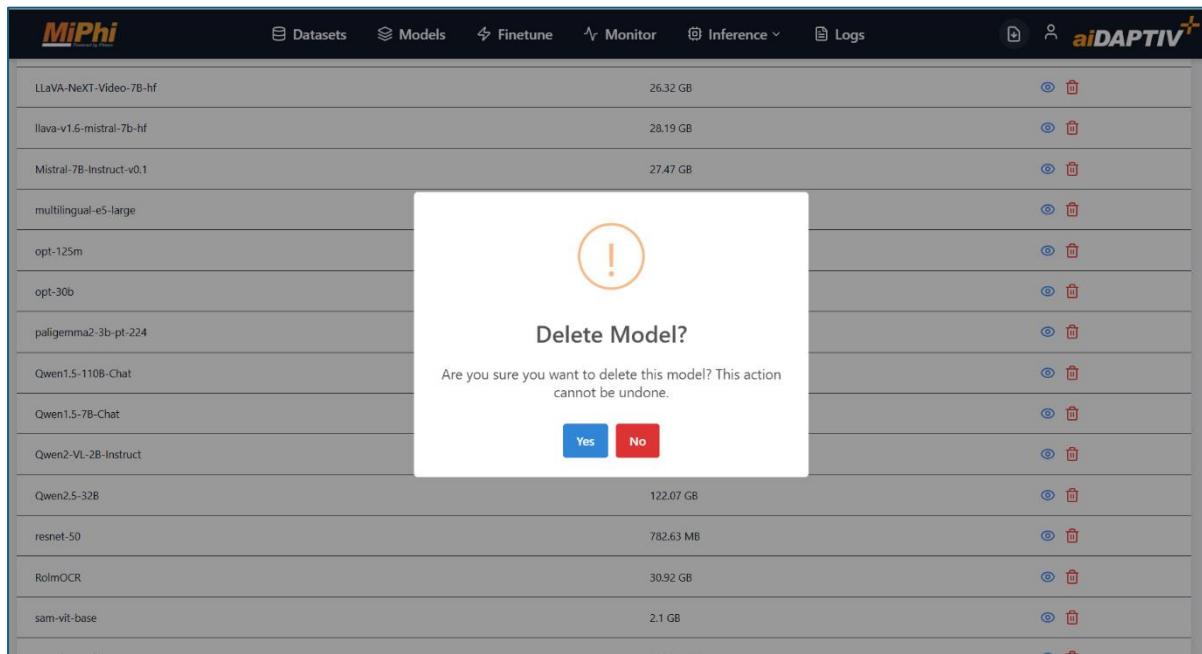
6.2.1 Model Preview

- The configuration details of the model are presented through a scrollable JSON view.
- This allows users to easily inspect the contents of the config.json file for better understanding and verification.



6.2.2 Deleting a Model

- You’re about to permanently remove the selected file from /media/models; this action cannot be undone.
- Click **Delete** to confirm and remove the model immediately.
- Click **Cancel** to abort and keep the file safely stored.



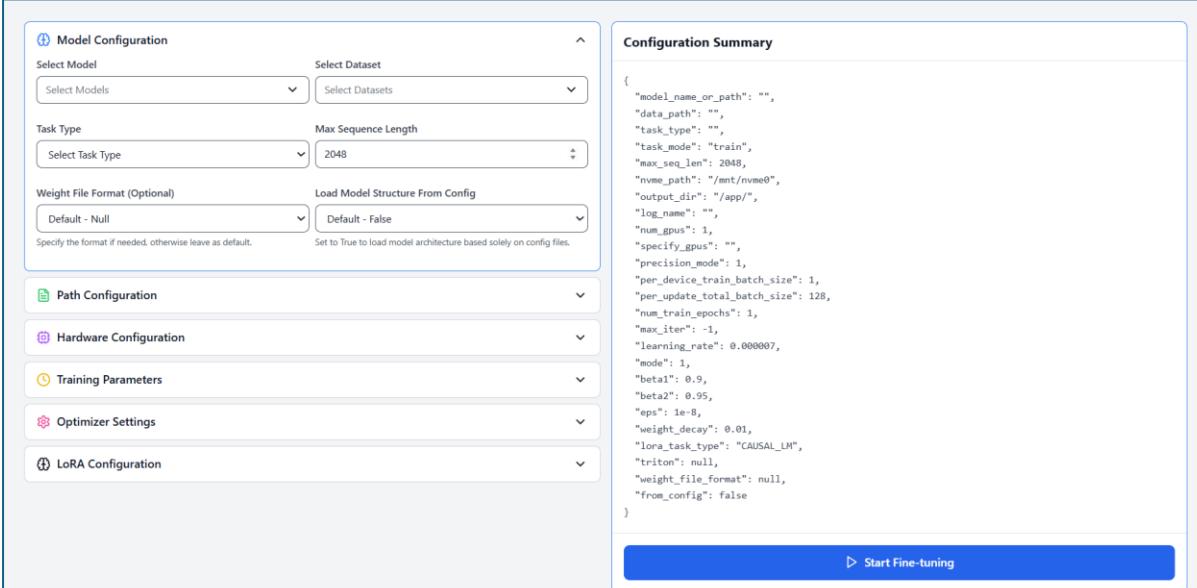
6.3.3 Searching a Model

- Use the search box in the top-right of the list to filter files by name or extension—typing “code” instantly shows only matching entries.
- Results update in real time as you type; hit the × icon to clear your query and restore the full list.
- Supports partial and full filename matches (e.g. entering “code” will display only files having the word “code”)

Available Models in /media/models		
Name	Size	Actions
CodeLlama-34b-Instruct-hf	125.95 GB	
CodeLlama-7b-hf	50.21 GB	

7. MIPHI FINE-TUNING PAGE

The fine-tuning page have two main sections: on left side section assign parameters and on right side section for viewing a configuration summary and then you can start fine-tuning.



The screenshot shows the MiPhi Fine-Tuning interface. On the left, the 'Model Configuration' section contains fields for 'Select Model' (dropdown), 'Select Dataset' (dropdown), 'Task Type' (dropdown), 'Max Sequence Length' (input field set to 2048), 'Weight File Format (Optional)' (dropdown), and 'Load Model Structure From Config' (checkbox). Below these are collapsed sections for 'Path Configuration', 'Hardware Configuration', 'Training Parameters', 'Optimizer Settings', and 'LoRA Configuration'. On the right, the 'Configuration Summary' section displays a JSON configuration object:

```
{
    "model_name_or_path": "",
    "data_path": "",
    "task_type": "",
    "task_mode": "train",
    "max_seq_len": 2048,
    "nvm_path": "/mnt/nvme0",
    "output_dir": "/app/",
    "log_name": "",
    "num_gpus": 1,
    "specify_gpus": "",
    "precision_mode": 1,
    "per_device_train_batch_size": 1,
    "per_update_total_batch_size": 128,
    "num_train_epochs": 1,
    "max_iter": -1,
    "learning_rate": 0.000007,
    "mode": 1,
    "beta1": 0.9,
    "beta2": 0.95,
    "eps": 1e-8,
    "weight_decay": 0.01,
    "lora_task_type": "CAUSAL_LM",
    "triton": null,
    "weight_file_format": null,
    "from_config": false
}
```

At the bottom right is a blue button labeled 'Start Fine-tuning'.

7.1 Model Configuration Section

This section allows users to set up the parameters for fine-tuning the model. It typically includes:

7.1.1 Select Model

- This dropdown lists available models based on the path specified in the Models page.
- When clicked, it displays all models stored in the configured path.
- Users can select the desired model for their task, such as text generation, classification, Fill mask and Speech recognition.

7.1.2 Select Dataset

- This dropdown lists datasets available in the path defined in the Dataset page.
- When clicked, it shows all datasets present in that directory.
- Users can choose the dataset relevant to their selected model and training objectives.
- The system dynamically fetches models and datasets from the **predefined paths**, ensuring users always see the latest available options.

- The **search feature** allows users to quickly find the required model or dataset without scrolling through the entire list.

7.1.3 Task type

In Task type, you can choose different types of tasks based on the model's purpose. Here are some common ones:

Task Types for Fine-Tuning:

- **Text Generation:** The model generates new text based on a given input (e.g., GPT models). Fine-tuning can be done to specialize the model for specific types of text, like poetry or code.
- **Image Classification:** In this task where the model classifies images into predefined categories. Fine-tuning involves adapting a pre-trained model to a specific set of image classes.
- **Fill Mask:** This is a task where the model predicts missing words in a sentence (e.g., BERT models).
- **Automatic Speech Recognition (ASR):** This involves converting spoken language into text.

7.1.4 Max sequence length

The **max sequence length** refers to the maximum number of tokens (words, sub words, or characters) that a model can process in a single input sequence.

- **128 tokens** – Best for short texts like tweets, sentiment analysis, and simple classification tasks.
- **256 tokens** – Suitable for small paragraphs, short emails, and FAQ-based classification.
- **512 tokens** – Standard for most NLP tasks like document classification, named entity recognition (NER), and summarization.
- **1024 tokens** – Ideal for longer inputs such as news articles, research abstracts, and complex question-answering.
- **2048 tokens** – Used for multi-paragraph text generation, document analysis, and long-form conversations.

7.1.5 Weight File Format

This parameter specifies the format of the model's weight file, such as .bin, .pt, or .safetensors, depending on the type of model being used or required.

7.1.6 Load Model Structure From Config:

When enabled, this parameter ensures that the model's architecture is loaded entirely from the config.json file.

Model Configuration

Select Model	Select Dataset
<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="CodeLlama-13b-Instruct-hf"/>	<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="biological_dataset.json"/>
Task Type	Max Sequence Length
<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="Text Generation"/>	<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="2048"/>
Weight File Format (Optional)	Load Model Structure From Config
<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="Default - Null"/>	<input style="border: 1px solid #ccc; padding: 2px; width: 100%;" type="text" value="Default - False"/>
Specify the format if needed, otherwise leave as default.	
Set to True to load model architecture based solely on config files.	

7.2 Path Configuration Section

The Path Configuration Section is crucial for defining where various files and logs will be stored during the fine-tuning process. It typically includes:

7.2.1 aiDAPTIV+ Cache Mount Point

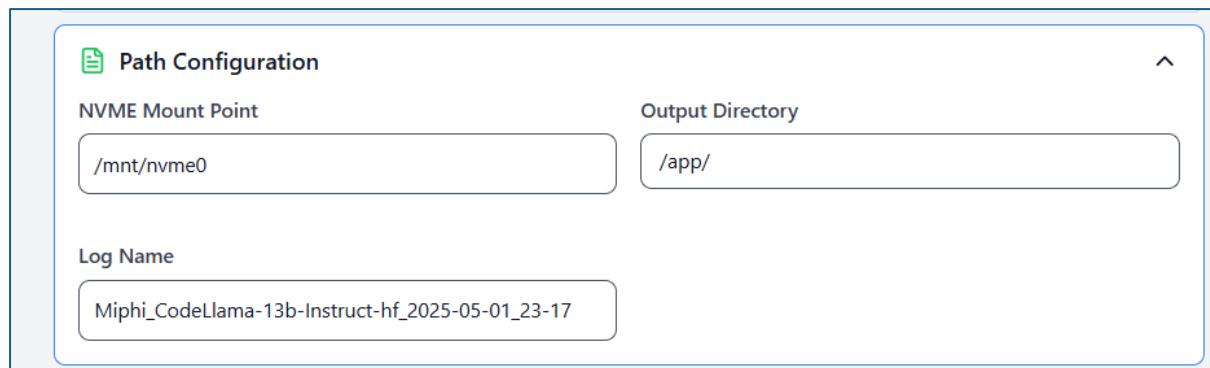
It serves as a high-speed storage location for caching and accessing large amounts of data efficiently. It plays a crucial role in storing and loading model parameters, datasets during training and fine-tuning.

7.2.2 Output_dir

Directory where the fine-tuned model will be saved. (But you can change it as needed.)

7.2.3 Log_name

The log name is the filename used to track the training process. It will be automatically assigned based on the selected model.



7.3 Hardware Configuration Section

This section typically includes options to select GPUs and set the precision mode.

7.3.1 Select GPUs

This option allows you to choose which GPUs will be used for the training process. It also displays the memory usage of each GPU, helping you manage resource allocation efficiently. The available GPU selection may vary based on server availability.

7.3.2 Precision Mode

The Precision Mode setting determines how numerical values are processed during model training. It impacts **accuracy, memory usage, and computational speed**.

- **FP32 (Single Precision)**: Standard 32-bit floating point, offering high accuracy but slower computation.
- **FP16 (Half Precision)**: 16-bit floating point, faster than FP32 but with reduced precision.
- **BF16 (Bfloat16)**: 16-bit floating point with a wider exponent range than FP16, balancing speed and accuracy.

You can select from the following precision options

Precision Mode	Memory Usage	Speed	Accuracy	Best Use Case
FP32	High	Normal	High	High-precision models, sensitive computations
FP16	Low	Fast	Moderate	Deep learning, speed-optimized training
BF16	Moderate	Fast	High	Fine-tuning large models, balanced efficiency

7.3.3 Enable Triton

Select whether to enable Triton Inference Server features (optional). Triton is useful for optimized model serving, supporting features like dynamic batching, concurrent model execution, and multi-framework support, which can significantly improve inference performance during fine-tuning and deployment.

 **Hardware Configuration**

Select GPUs

- GPU 0 (48GB RAM) • 37425MiB used
- GPU 1 (48GB RAM) • 4MiB used**
- GPU 2 (48GB RAM) • 4MiB used**
- GPU 3 (48GB RAM) • 4MiB used
- GPU 4 (48GB RAM) • 4MiB used
- GPU 5 (48GB RAM) • 4MiB used
- GPU 6 (48GB RAM) • 4MiB used
- GPU 7 (48GB RAM) • 4MiB used

Click to toggle GPU selection.

Precision Mode

FP32 - Full Precision

Select Precision Mode

- FP32 - Full Precision**
- FP16 - Mixed Precision
- BF16 - Brain Float 16

Enable Triton

Default - Null

Select whether to enable Triton inference server features (optional).

7.4 Training Parameters Section

The **Training Parameters** section defines key settings that control the training process, including how data is processed in batches, the total number of training steps, and how long the model trains.

7.4.1 Train Batch Size

- The number of training samples processed per GPU in one forward and backward pass.
- Affects memory usage and training speed. Larger batch sizes improve training stability but require more GPU memory.

7.4.2 Total Batch Size

- The total number of samples used for one update of model weights
- Helps control gradient updates when training across multiple GPUs or accumulating gradients over multiple steps to fit larger batch sizes in memory.

Example: If you are running on 4 GPUs with Total batch size =4 and want to update the model every 80 batches, then you should set the total_batch_size to 80.

The machine will run $80/(4 \text{ GPUs} \times 4 \text{ Batch Size per GPU}) = 5$ iterations and update the model once.

Note: If not divisible, round up to the next whole number.

7.4.3 Epoch

- The number of times the model processes the entire dataset during training.
- Controls how long the model learns from the data. Too few epochs might lead to underfitting, while too many can cause overfitting.

Example: If your dataset has **100,000 samples** and you set batch size = 1000, then:

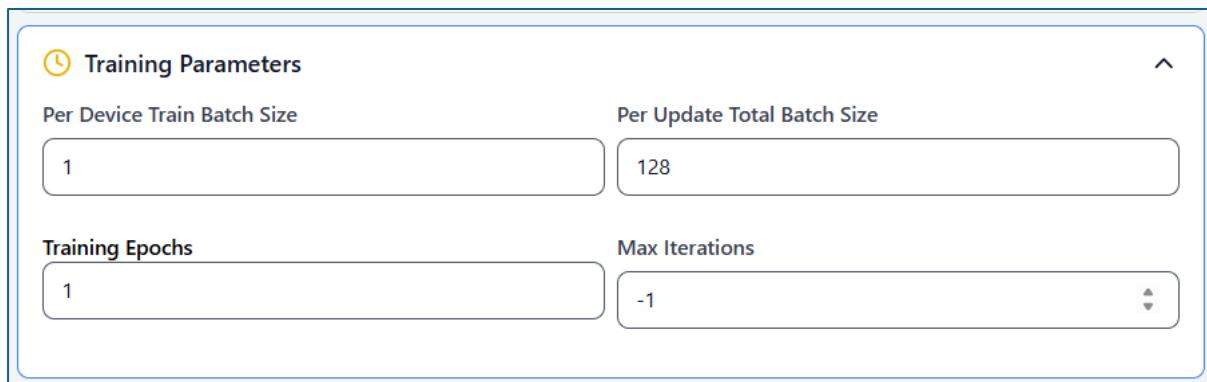
- **1 epoch = 100 iterations**
- If training for **5 epochs**, the model will perform **500 iterations**.

7.4.4 Max Iteration

- The maximum number of training steps (iterations) to perform during training
- Controls when to stop training, independent of epochs.

Example:

- If batch size = 32 and dataset has **320,000 samples**, then **1 epoch = 10,000 iterations**.
- If max_iterations = 20,000, the training will stop after **2 epochs**, even if more epochs are set.



7.5 Optimizer Settings Section

The **Optimizer Settings** section defines how the model updates its parameters during the fine-tuning process.

7.5.1 Learning rate

- It is the step size at which the model's weights are updated during training. It controls how much the model learns from the gradients at each step.
- A well-chosen learning rate ensures effective training—too high can cause instability, and too low can result in slow convergence.

7.5.2 LR Scheduler Mode

- Determines how the learning rate changes over time during training.
- Adjusting the learning rate during training helps prevent overfitting and improves convergence.
- mode: choose a learning rate mode (default: 1).
- mode == -1: LinearLR (optimizer, start_factor=1, total_iters=1)
- mode == 0: LinearLR (optimizer, start_factor=0.5, total_iters=20) mode == 1: CosineAnnealingLR (optimizer, T_max=150)

- mode == 2: ExponentialLR (optimizer, gamma=0.99)
- mode == 3: MultiplicativeLR (optimizer, lr_lambda=lambda epoch: 0.95)
- mode == 4: StepLR (optimizer, step size=30, gamma=0.1)
- mode == 5: MultiStepLR (optimizer, milestones= [30,80], gamma=0.1)

7.5.3 Beta1

Adam Optimizer Hyperparameter(default:0.9)

- The exponential decay rate for the first moment estimate (momentum term) in the Adam optimizer.
- Controls how much past gradients influence the current update, helping smooth training.
- Range: Typically, between 0.8 and 0.99

7.5.4 Beta2

Adam Optimizer Hyperparameter(default:0.95)

- The exponential decay rate for the second moment estimate (variance) in the Adam optimizer.
- Helps stabilize the optimization process by adjusting for variance in gradients.
- Range: Typically, between 0.95 and 0.9999

7.5.5 Epsilon

hyper-parameter for adam optimizer (default:1e-8).

- A small constant added to prevent division by zero in the Adam optimizer.
- Helps maintain numerical stability when computing updates.
- Range: 1e-8 to 1e-6 (Lower values are generally stable.)

7.5.6 Weight decay

Set weight decay coefficient (default:0.01)

- A regularization parameter that applies L2 penalty to model weights to prevent overfitting.

- Helps reduce the complexity of the model by discouraging large weights.

 Optimizer Settings

Learning Rate	LR Scheduler Mode
0.000007	1
Beta1	Beta2
0.9	0.95
Epsilon	Weight Decay
1e-8	0.01

7.6 LoRA Configuration Section

7.6.1 Enable LoRa

trigger lora training procedure (default: disabled).

7.6.2 LoRa rank

dimension of the low-rank matrices for lora (default: 8).

7.6.3 LoRa alpha

LoRA Alpha adjusts how much influence the LoRA layers have on the model's final predictions. (default: 16).

7.6.4 LoRa Task Type

The task type for LoRa adaptation is CAUSAL_LM by default.

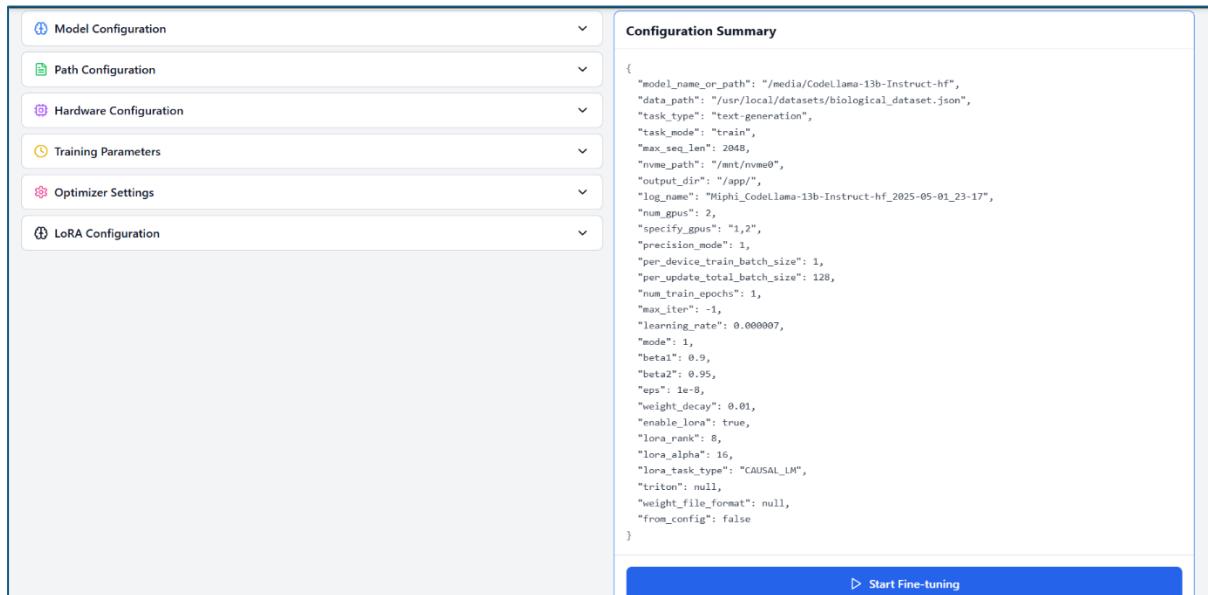
 LoRA Configuration

<input checked="" type="checkbox"/> Enable LoRA	
LoRA Rank	LoRA Alpha
8	16
LoRA Task Type	
CAUSAL_LM	

Specify the task type for LoRA adaptation (e.g., CAUSAL_LM).

7.7 Configuration Summary

- After setting all parameters, they will automatically update in the **Configuration Summary**, where you can review all the values you have provided.
- Once everything is confirmed, click the **Start Fine-Tuning** button to begin the process.
- The system will then redirect you to the **Monitor** page, where you can track the training progress in real time



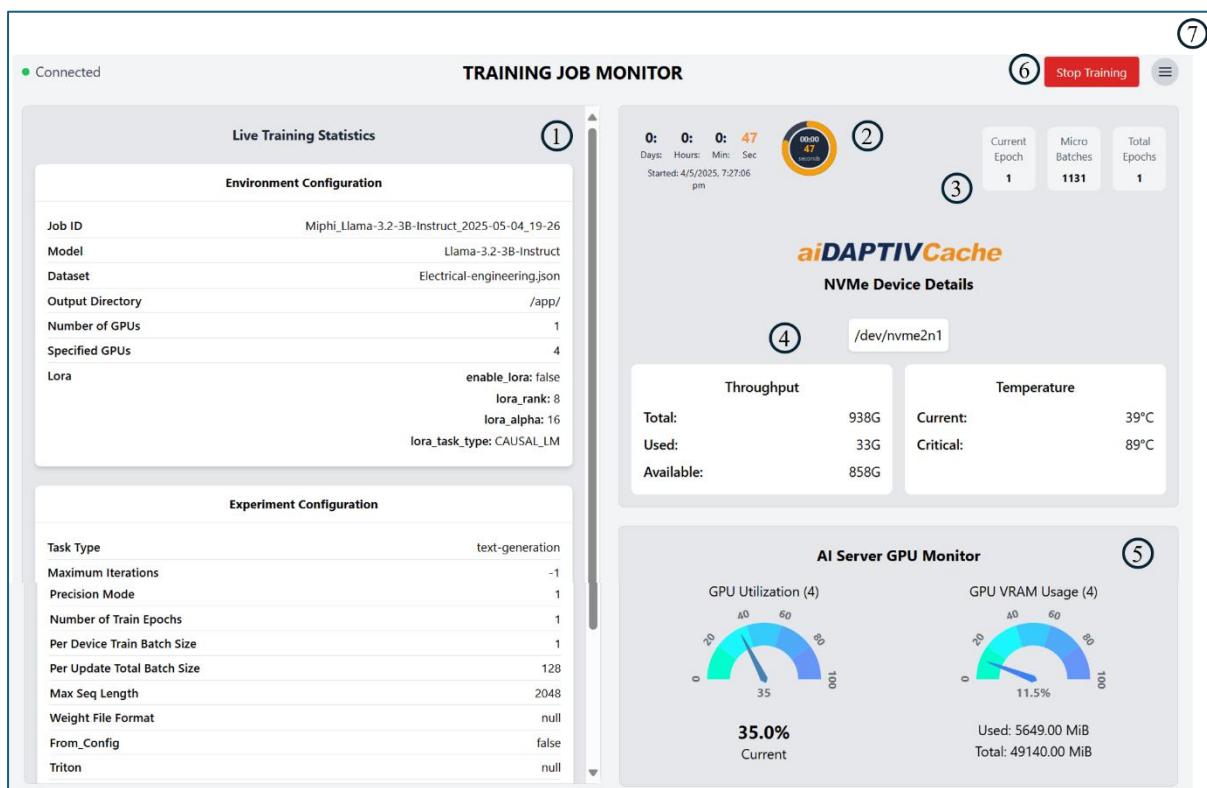
The screenshot shows the MiPhi Configuration Summary interface. On the left, there is a sidebar with expandable sections: Model Configuration, Path Configuration, Hardware Configuration, Training Parameters, Optimizer Settings, and LoRA Configuration. The Training Parameters section is currently expanded. To the right of the sidebar is a large text area titled "Configuration Summary" containing a JSON dump of the configuration settings. At the bottom right of this area is a blue button labeled "▷ Start Fine-tuning".

```
{
  "model_name_or_path": "/media/CodeLlama-13b-Instruct-hf",
  "data_path": "/usr/local/datasets/biological_dataset.json",
  "task_type": "text-generation",
  "task_mode": "train",
  "max_seq_len": 2048,
  "nvme_path": "/mnt/nvme0",
  "output_dir": "/app/",
  "log_name": "Miphic_Codellama-13b-Instruct-hf_2025-05-01_23-17",
  "num_gpus": 2,
  "specify_gpus": "1,2",
  "precision_mode": 1,
  "per_device_train_batch_size": 1,
  "per_update_total_batch_size": 128,
  "num_train_epochs": 1,
  "max_iter": -1,
  "learning_rate": 0.000007,
  "mode": 1,
  "beta1": 0.9,
  "beta2": 0.95,
  "eps": 1e-08,
  "weight_decay": 0.01,
  "enable_lora": true,
  "lora_rank": 8,
  "lora_alpha": 16,
  "lora_task_type": "CAUSAL_LM",
  "triton": null,
  "weight_file_format": null,
  "from_config": false
}
```

Note: Before starting fine-tuning, ensure that all parameters are correctly updated to avoid errors

8. MiPhi Monitor Page

The **MiPhi Monitor** offers a real-time overview of the training process, providing key insights into both system performance and training metrics. It displays training configurations, including NVMe cache details, device usage, elapsed time, current epoch, total epochs, and microbatches. The page tracks GPU utilization, VRAM, CPU, and RAM usage, ensuring efficient resource monitoring. Users can access and download training logs for analysis, view a loss vs. timestamp graph, and examine detailed GPU statistics. The training process can be stopped at any time, and the sidebar allows quick access to the latest five training jobs for efficient session tracking.



The screenshot displays the MiPhi Monitor interface with several sections:

- Top Left:** Status indicator "Connected".
- Top Center:** Title "TRAINING JOB MONITOR".
- Top Right:** A red button labeled "Stop Training" with a circled number "6" above it, and a menu icon with a circled number "7" above it.
- Left Panel (Live Training Statistics):**
 - Environment Configuration:**

Job ID	Miphi_Llama-3.2-3B-Instruct_2025-05-04_19-26
Model	Llama-3.2-3B-Instruct
Dataset	Electrical-engineering.json
Output Directory	/app/
Number of GPUs	1
Specified GPUs	4
Lora	enable_lora: false lora_rank: 8 lora_alpha: 16 lora_task_type: CAUSAL_LM
 - Experiment Configuration:**

Task Type	text-generation
Maximum Iterations	-1
Precision Mode	1
Number of Train Epochs	1
Per Device Train Batch Size	1
Per Update Total Batch Size	128
Max Seq Length	2048
Weight File Format	null
From_Config	false
Triton	null
- Top Right Panel (aiDAPTIV Cache):**
 - Elapsed time: 0: 0: 0: 47 (Days: Hours: Min: Sec)
 - Started: 4/5/2025, 7:27:06 pm
 - Current Epoch: 1
 - Micro Batches: 1131
 - Total Epochs: 1
- Middle Right Panel (NVMe Device Details):**
 - Device: /dev/nvme2n1
 - Throughput:
 - Total: 938G
 - Used: 33G
 - Available: 858G
 - Temperature:
 - Current: 39°C
 - Critical: 89°C
- Bottom Right Panel (AI Server GPU Monitor):**
 - GPU Utilization (4): 35.0%
 - GPU VRAM Usage (4): 11.5% (Used: 5649.00 MiB, Total: 49140.00 MiB)

8.1 Training Statistics

The **Training Statistics** section provides configuration details for training jobs and dynamically updates based on the training status. It is divided into two parts:

- Environment Configuration:** Displays metadata such as Job ID, model name, dataset, output directory, GPU settings, and LoRA parameters.
- Experiment Configuration:** Shows training-specific parameters including task type, number of epochs, batch sizes, sequence length, optimizer settings, learning rate scheduler, and precision mode.

Behavior:

- If **no training** has been run yet, the section shows "**No process running.**"
- If a training job is **in progress**, this section displays **live statistics** of the current job.
- Once a job is **completed**, the section shows details of the **latest completed job** along with its Job ID.

It helps to easily track and reference the most recent training configurations and progress.

8.2 Elapsed Time

The **Elapsed Time** section provides a real-time display of the duration since the current training job began. Time is presented in a structured format—**days, hours, minutes, and seconds**—accompanied by a visual circular progress indicator for enhanced clarity. Additionally, the exact start timestamp of the training session is shown below the timer for precise reference.

This section becomes visible **only when a training process is actively running**, allowing to monitor the session duration as it progresses. When no training is in progress, the section remains hidden.

8.3 Epoch and Micro-batch

This section displays real-time training metrics, including:

- **Current Epoch:** The epoch currently in progress.
- **Total Epochs:** The total number of training epochs defined.
- **Microbatches:** The cumulative number of microbatches processed within the current epoch.

This panel is visible **only while a training session is active**, providing clear visibility into training progress at both the epoch and batch level.

8.4 NVMe Device Details

This section provides detailed monitoring of the NVMe storage device actively used during training. It is designed to offer critical insights into both the storage capacity and thermal performance of the device in real time.

- **Device Path:** Displays the device path used of the active NVMe mount point (e.g., /dev/nvme2n1).

Throughput

- **Total** – The total capacity of the NVMe device.
- **Used** – The amount of storage currently occupied by training processes or other operations.
- **Available** – Remaining free space available on the device.

Temperature

- **Current** – Real-time operating temperature of the NVMe device.
- **Critical** – The maximum temperature threshold before the device may throttle or become unstable.

8.5 GPU Monitoring

The **GPU Monitoring** section displays real-time metrics of the GPU actively used for training. It includes:

- **GPU Utilization**: Shows the current processing load (%) on the training GPU.
- **GPU VRAM Usage**: Displays the used and total memory (MiB), along with the percentage of VRAM consumption.

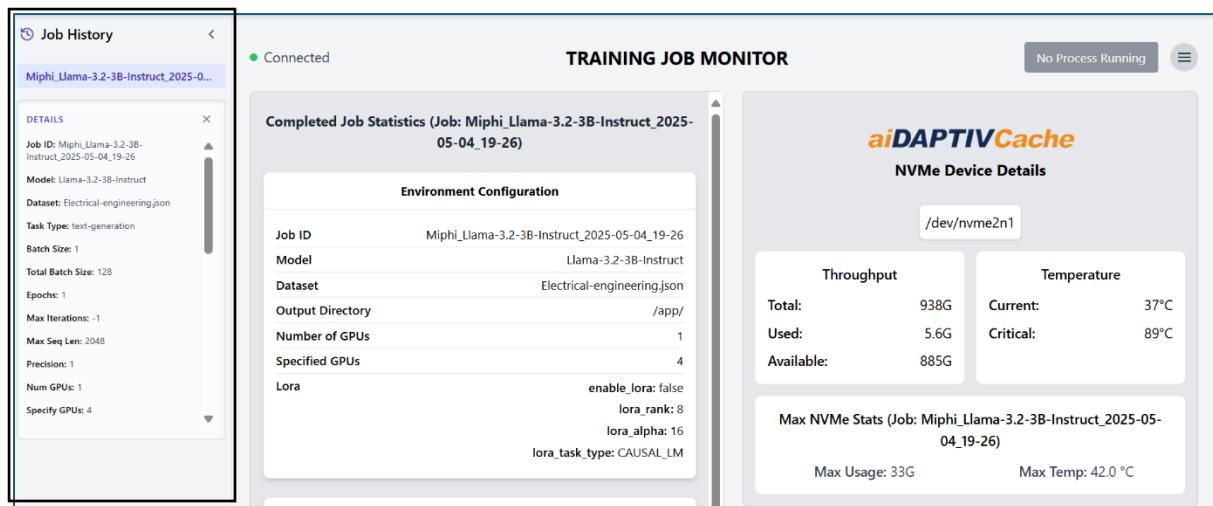
8.6 Stop Training

This section displays a **Stop Training** button when a training job is in progress. Clicking this button will immediately terminate the active training process.

- When **training is active**, the button is visible and functional.
- When **no training is running**, the section displays "**No process running.**"

8.7 Job History

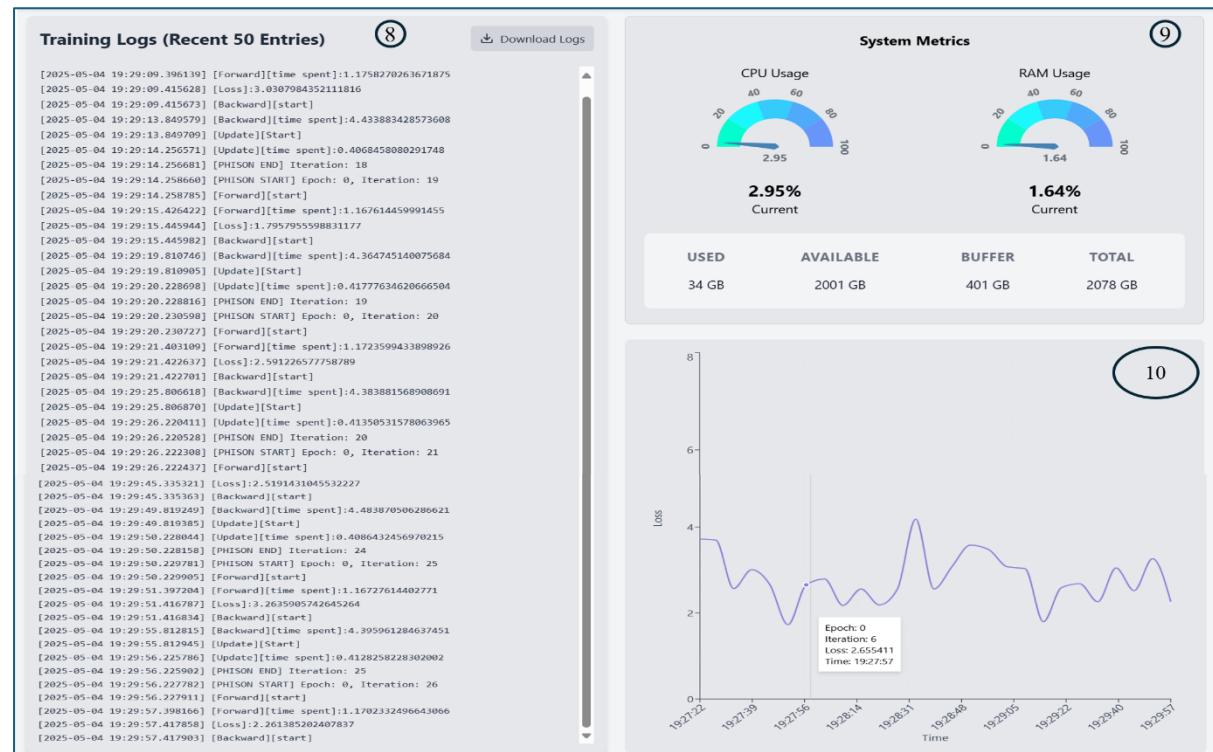
This section provides access to the **five most recent finetuned training jobs**. Clicking the toggle button expands a panel displaying detailed configuration and system usage stats for each job.



The screenshot shows the MiPhi Training Job Monitor interface. On the left, a sidebar titled "Job History" displays detailed training parameters for a job named "Miph_Llama-3.2-3B-Instruct_2025-05-04-19-26". These include Job ID, Model Name (Llama-3.2-3B-Instruct), Dataset (Electrical-engineering.json), Task Type (text-generation), Batch Size (1), Total Batch Size (128), Epochs (1), Max Iterations (-1), Max Seq Len (2048), Precision (1), Num GPUs (1), and Specified GPUs (4). In the center, the "TRAINING JOB MONITOR" section shows "Completed Job Statistics" for the same job. It lists Environment Configuration details such as Job ID, Model, Dataset, Output Directory, Number of GPUs (1), Specified GPUs (4), Lora settings (enable_lora: false, lora_rank: 8, lora_alpha: 16), and lora_task_type (CAUSAL_LM). On the right, the "aiDAPTIV Cache" section provides NVMe Device Details for /dev/nvme2n1, showing Throughput (Total: 938G, Used: 5.6G, Available: 885G) and Temperature (Current: 37°C, Critical: 89°C). It also displays Max NVMe Stats (Max Usage: 33G, Max Temp: 42.0 °C).

Details include:

- Job Metadata:** Job ID, Model Name, Dataset, Task Type, Output Directory
- Training Configuration:** Batch sizes, Epochs, Max Sequence Length, Optimizer, Learning Rate Scheduler, Precision, LoRA settings
- System Utilization:** Max CPU/RAM usage, GPU Utilization %, VRAM (MiB), NVMe usage and temperature
- Timestamps:** When training parameters and system stats were last saved



The screenshot shows the MiPhi interface with three main sections. On the left, the "Training Logs (Recent 50 Entries)" section displays a list of log entries from May 4, 2025, at 19:29:00 to 19:29:57. The logs include forward and backward operations, update steps, and PHISON START/END markers. In the center, the "System Metrics" section features two circular gauges: "CPU Usage" (2.95%) and "RAM Usage" (1.64%). Below the gauges, a table shows memory usage: USED (34 GB), AVAILABLE (2001 GB), BUFFER (401 GB), and TOTAL (2078 GB). On the right, a line graph plots "Loss" over time, showing values fluctuating between 0 and 8 across the period from 19:27:22 to 19:29:57. A callout box highlights a specific data point: Epoch: 0, Iteration: 6, Loss: 2.655411, Time: 19:28:31.

8.8 Training Logs

This section displays the **latest 50 log entries** from the current training session in real time.

- When **training is active**, logs update live to show forward/backward passes, loss values, iteration details, and time spent.
- When **training is inactive**, it displays logs from the most recent completed session.

A **Download Logs** button is available to view the full log file for deeper inspection or record-keeping.

8.9 System Metrics

This section provides real-time insights into system resource usage:

- **CPU Usage:** Displays the current percentage of CPU utilization.
- **RAM Usage:** Shows the current memory consumption as a percentage.
- **Disk Summary:** Indicates used, available, buffer, and total disk capacity.

8.10 Training Graph

This section visualizes the **loss over time**.

- When **training is active**, the graph updates in real-time, reflecting live loss values.
- When **training is inactive**, it displays the most recent complete loss graph from the last fine-tuning session.

8.11 Live GPU Statistics

This section provides **real-time monitoring** of all available GPUs, displaying:

- **Utilization (%)**
- **Total, used, and free memory (MiB)**
- **Temperature (°C)**

If a GPU's temperature exceeds **85°C**, it is highlighted with a **red background** to indicate potential overheating.

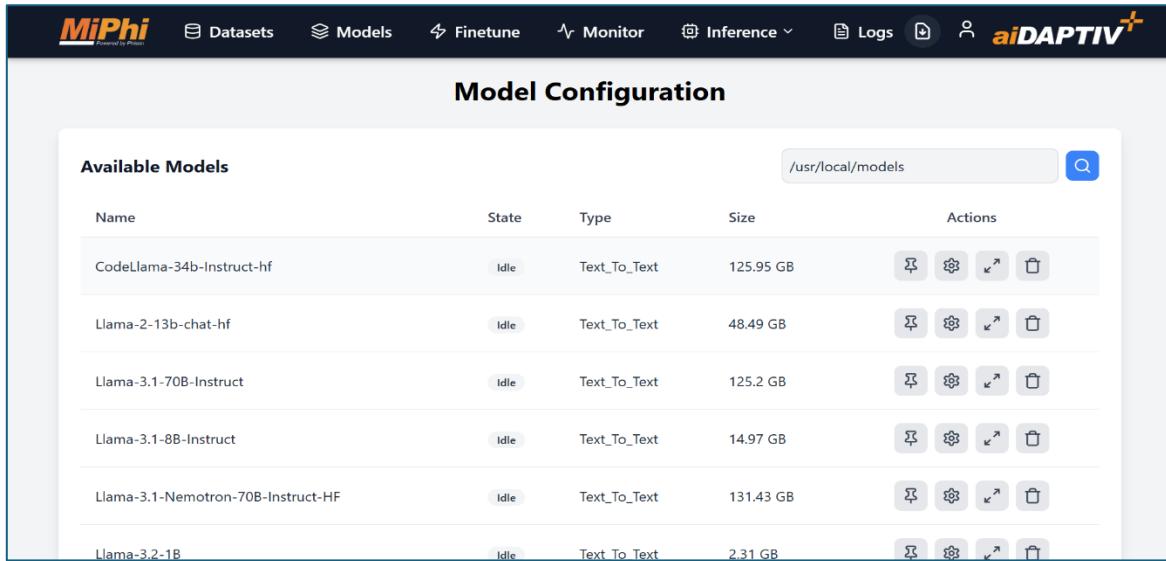
11

Live GPU Statistics						
IDX	NAME	UTIL(%)	MEM TOTAL(MiB)	MEM USED(MiB)	MEM FREE(MiB)	TEMP(°C)
0	NVIDIA RTX A6000	0%	49140	4	48666	37°C
1	NVIDIA RTX A6000	0%	49140	4	48666	40°C
2	NVIDIA RTX A6000	0%	49140	4	48666	42°C
3	NVIDIA RTX A6000	0%	49140	4	48666	42°C
4	NVIDIA RTX A6000	0%	49140	4	48666	53°C
5	NVIDIA RTX A6000	0%	49140	4	48666	45°C
6	NVIDIA RTX A6000	0%	49140	4	48666	43°C
7	NVIDIA RTX A6000	0%	49140	1397	47273	37°C

9. MIPHI INFERENCE PAGE

9.1 Selecting a Model for Inference

- After opening the URL in your browser, you will be directed to the model configuration page.
- On this page, all available text-to-text generation models are listed based on the model path provided during container creation.



The screenshot shows the 'Model Configuration' page of the MiPhi web interface. At the top, there is a navigation bar with links for Datasets, Models, Finetune, Monitor, Inference, Logs, and a user icon. To the right of the navigation bar is the 'aiDAPTIV+' logo. Below the navigation bar, the title 'Model Configuration' is centered. Underneath the title is a section titled 'Available Models'. This section contains a table with the following data:

Name	State	Type	Size	Actions
CodeLlama-34b-Instruct-hf	Idle	Text_To_Text	125.95 GB	
Llama-2-13b-chat-hf	Idle	Text_To_Text	48.49 GB	
Llama-3.1-70B-Instruct	Idle	Text_To_Text	125.2 GB	
Llama-3.1-8B-Instruct	Idle	Text_To_Text	14.97 GB	
Llama-3.1-Nemotron-70B-Instruct-HF	Idle	Text_To_Text	131.43 GB	
Llama-3.2-1B	Idle	Text_To_Text	2.31 GB	

- Here you can give the directory path where the models are located.



9.1.1 Components of Model Table

Column	Description
Name	The name of the model (e.g., Llama-3.1-8B-Instruct, Gemma-2-9B-It)
State	Status of the model (Idle or Running)
Type	Model task type (e.g., Text_To_Text)
Size	Size of the model in gigabytes (GB)
Actions	Buttons for model operations

9.1.2 Model Actions

For each model, the following actions are available (represented by icons in the Actions column):

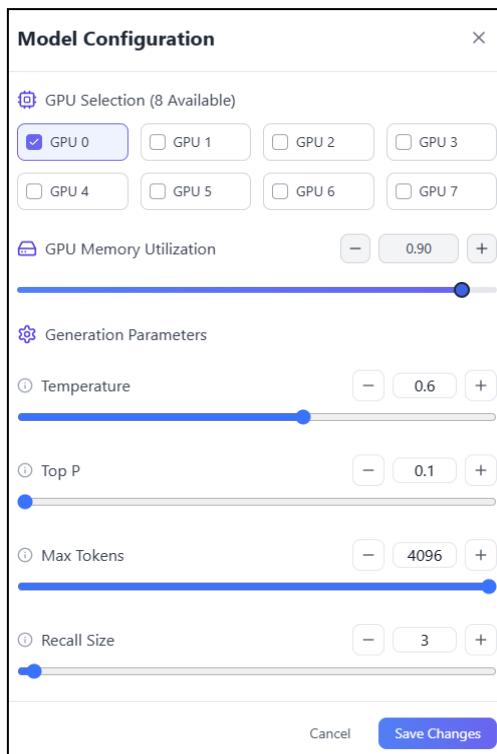
- **Pin Model** : Load and pin a model for use
- **Settings** : Configure model settings
- **Expand** : View detailed model information for Quantization of the Model
- **Delete** : Remove the model from the system

9.1.3 Model Configuration

- Click the settings button to adjust model parameter inference.



- Now the model configuration box will appear. Select the required values to load model and click ‘Save Changes’.



Adjusting Model Parameters

You can modify the model's behavior by adjusting the following parameters:

- GPU Selection: Select the specific GPUs for loading a model (default: [0])
- GPU Memory Utilization: Select the value of GPU VRAM that can be utilized for loading model (default: 0.90)

Generation Parameters

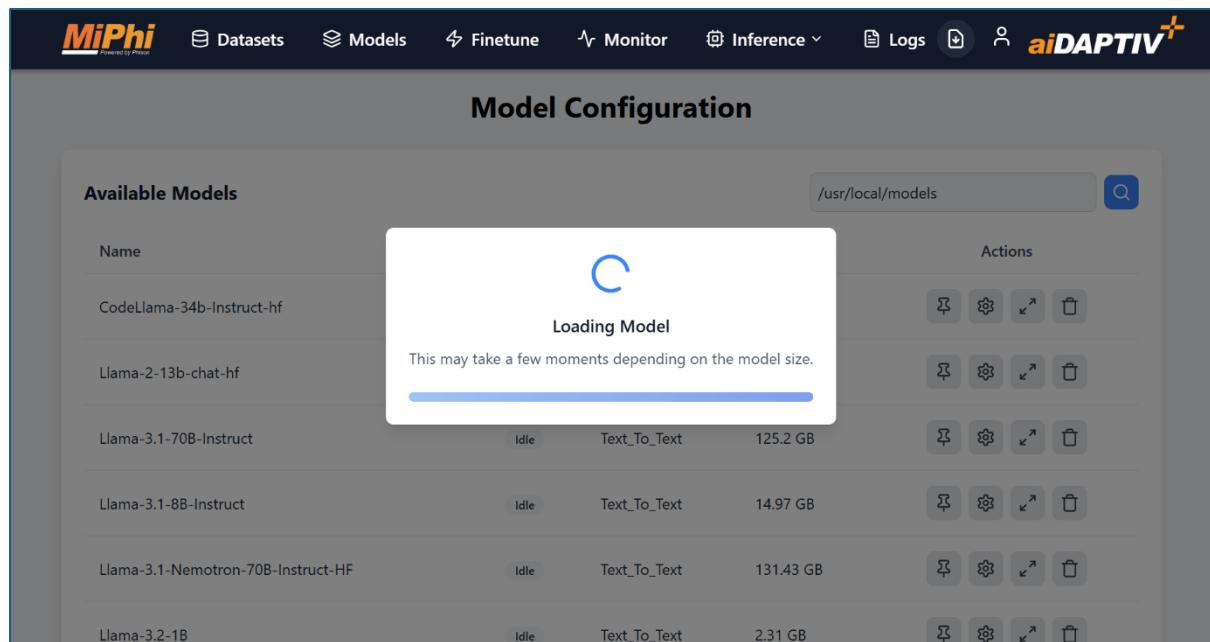
- **Temperature:** Controls randomness/creativity of responses (default: 0.6)
 - Range: 0.0 to 1.0
 - Lower values (closer to 0) make responses more deterministic and focused
 - Higher values (closer to 1) make responses more creative and diverse
- **Top P:** Controls nucleus sampling threshold (default: 0.1)
 - Range: 0.1 to 1.0
 - Works alongside temperature to influence response variety
- **Max tokens:** Controls the maximum length of generated responses (default: 4096)
 - Use slider or +/- buttons to adjust
 - Higher values produce longer responses but take more time
- **Recall Size:** The value for reading chunks in file (default: 3) while using RAG.
 - Use slider or +/- buttons to adjust

9.1.4 Pin the Model

- Click the pin button to load the model and prepare it for inference.



- The model may take a few moments to load.



A screenshot of the MiPhi "Model Configuration" page. The top navigation bar includes the MiPhi logo, Datasets, Models, Finetune, Monitor, Inference, Logs, and the aiDAAPTIV logo. The main section is titled "Model Configuration" and contains a table titled "Available Models". The table lists several models:

- CodeLlama-34b-Instruct-hf
- Llama-2-13b-chat-hf
- Llama-3.1-70B-Instruct (highlighted with a blue border)
- Llama-3.1-8B-Instruct
- Llama-3.1-Nemotron-70B-Instruct-HF
- Llama-3.2-1B

 Each model entry includes its name, status (Idle or Busy), task (Text_To_Text), size (e.g., 125.2 GB, 14.97 GB, 131.43 GB, 2.31 GB), and a set of four action buttons. A modal window is open over the table, centered on the Llama-3.1-70B-Instruct row. The modal has a circular loading icon and the text "Loading Model" above a progress bar. Below the progress bar is the message "This may take a few moments depending on the model size.".

- The selected model will be pinned and appear on the top of the model table. The model is ready for inference.

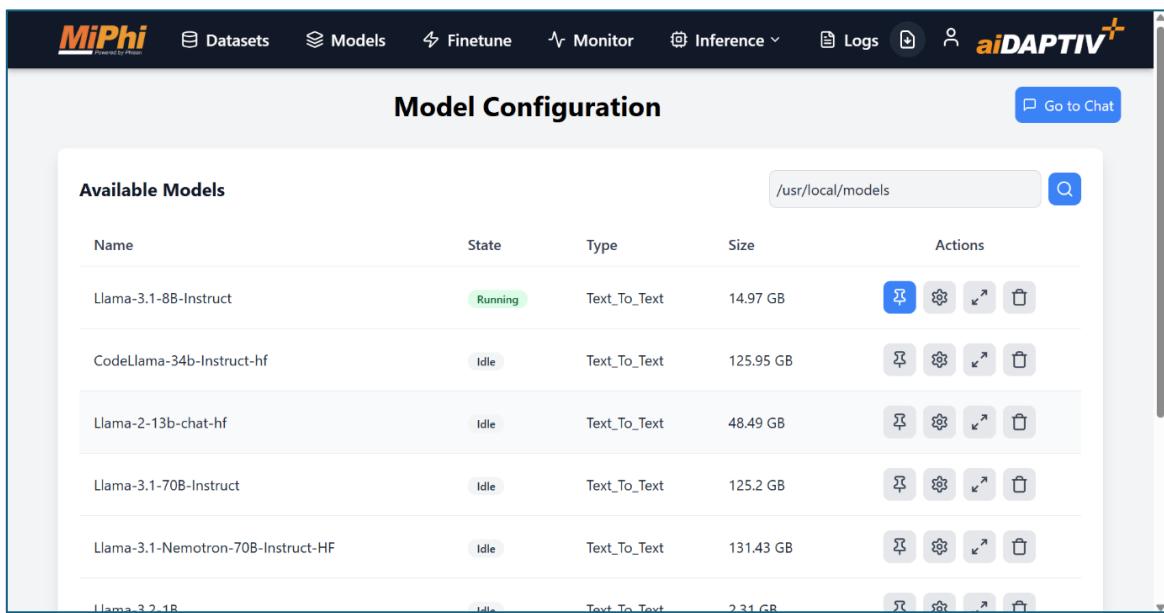
Model Configuration

[Go to Chat](#)

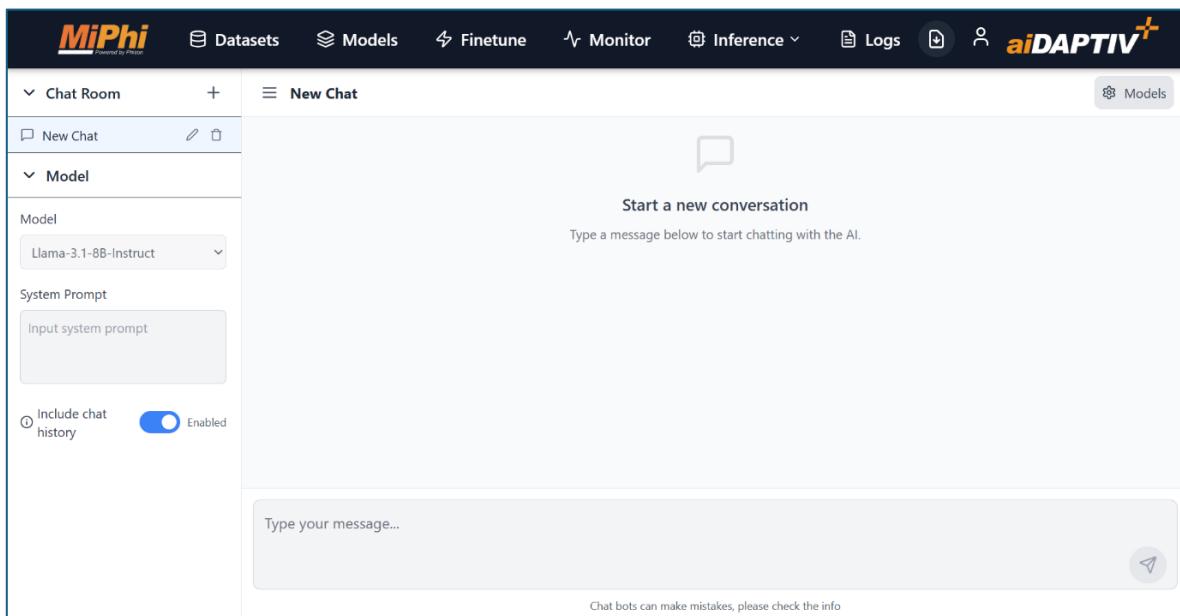
Available Models		/usr/local/models		🔍
Name	State	Type	Size	Actions
Llama-3.1-8B-Instruct	Running	Text_To_Text	14.97 GB	
CodeLlama-34b-Instruct-hf	Idle	Text_To_Text	125.95 GB	
Llama-2-13b-chat-hf	Idle	Text_To_Text	48.49 GB	

9.1.5 Navigate to Chat Bot

- Click 'Go to Chat' or visit the Chat-Room page to start interacting with the model.

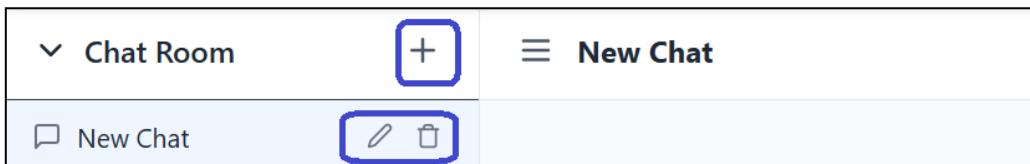


- The chatroom page for inference will appear as below image:



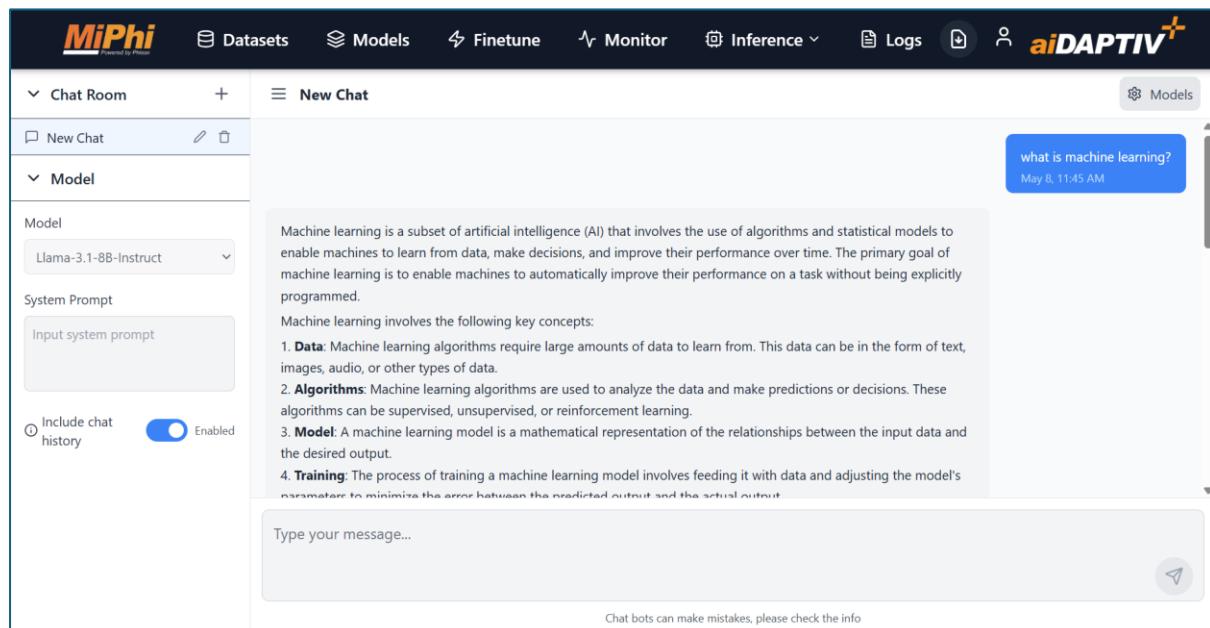
9.1.6 Create New Chat

- You can create separate chat sessions for multiple queries based on requirement for the loaded model by clicking the '+' icon in the Chat Room.
- You can also rename the chat session title or delete the chat session as needed.



9.1.7 Start Your Conversation

All set! You can now start interacting with the model by asking questions or sharing ideas.



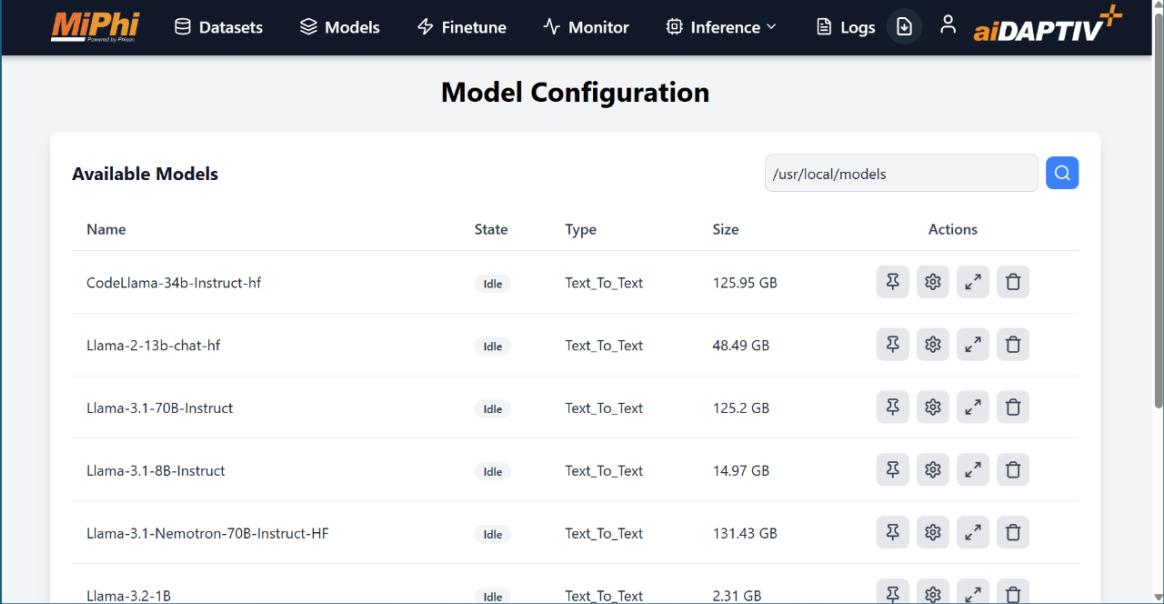
The screenshot shows the MiPhi interface with the 'New Chat' session active. On the left, there's a sidebar with 'Chat Room' and 'Model' sections. The 'Model' section has 'Llama-3.1-8B-Instruct' selected. Below it, there's a 'System Prompt' input field containing 'Input system prompt'. A toggle switch for 'Include chat history' is turned on. The main area shows a message from the user: 'what is machine learning?' followed by the model's response: 'Machine learning is a subset of artificial intelligence (AI) that involves the use of algorithms and statistical models to enable machines to learn from data, make decisions, and improve their performance over time. The primary goal of machine learning is to enable machines to automatically improve their performance on a task without being explicitly programmed.' The response continues with key concepts like Data, Algorithms, Model, and Training. At the bottom, there's a text input field for typing messages and a send button.

9.2 Model Quantization from GUI

Model quantization is a technique used to reduce the precision of the numbers used to represent a model's parameters, such as weights and activations, from higher precision formats like FP32 (32-bit floating point) to lower precision formats like INT8 (8-bit integer) or even INT4.

9.2.1 Selecting a model for quantization

- On the Model Configuration page, all available text-to-text generation models are listed based on the model path provided during container creation.



The screenshot shows the MiPhi Model Configuration interface. At the top, there is a navigation bar with links for Datasets, Models, Finetune, Monitor, Inference, Logs, and a partner logo for aiDAPTIV+. Below the navigation bar is a search bar with the placeholder text "/usr/local/models" and a magnifying glass icon.

The main area is titled "Model Configuration" and contains a table titled "Available Models". The table has columns for Name, State, Type, Size, and Actions. The "Actions" column includes icons for viewing, editing, cloning, and deleting each model. The models listed are:

Name	State	Type	Size	Actions
CodeLlama-34b-Instruct-hf	Idle	Text_To_Text	125.95 GB	
Llama-2-13b-chat-hf	Idle	Text_To_Text	48.49 GB	
Llama-3.1-70B-Instruct	Idle	Text_To_Text	125.2 GB	
Llama-3.1-8B-Instruct	Idle	Text_To_Text	14.97 GB	
Llama-3.1-Nemotron-70B-Instruct-HF	Idle	Text_To_Text	131.43 GB	
Llama-3.2-1B	Idle	Text_To_Text	2.31 GB	

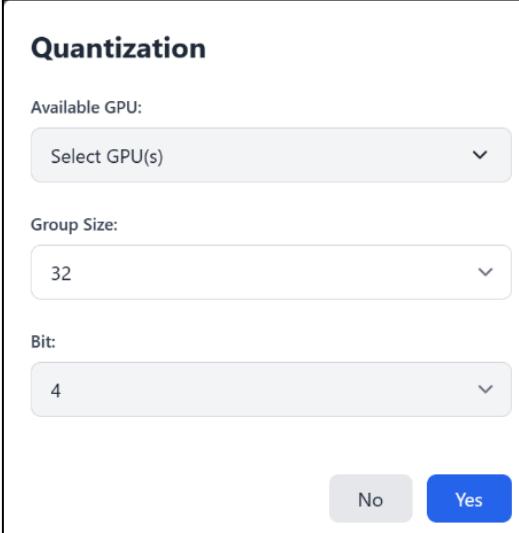
9.2.2 Adjusting Model Parameters for Quantization

- Click the settings button to adjust model parameters.



This screenshot shows the same MiPhi Model Configuration interface as the previous one, but with a specific model selected for configuration. The "Llama-3.1-8B-Instruct" model is highlighted in the list, and its row shows a settings icon with a gear symbol, which is highlighted with a blue square.

- Now the quantization option box will appear.

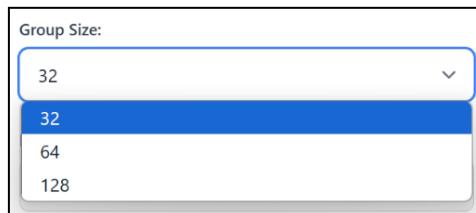


The dialog box is titled "Quantization". It contains three dropdown menus: "Available GPU:", "Group Size:", and "Bit:". The "Available GPU:" dropdown is set to "Select GPU(s)". The "Group Size:" dropdown is set to "32". The "Bit:" dropdown is set to "4". At the bottom right of the dialog are two buttons: "No" and "Yes", with "Yes" being highlighted in blue.

- **Adjusting Model Parameters:** You can modify the model's behavior by adjusting the following parameters:
- **Available GPU:** Select specific GPUs from the available list of GPUs you want to use for quantization from the drop-box. You can select one or multiple GPUs.



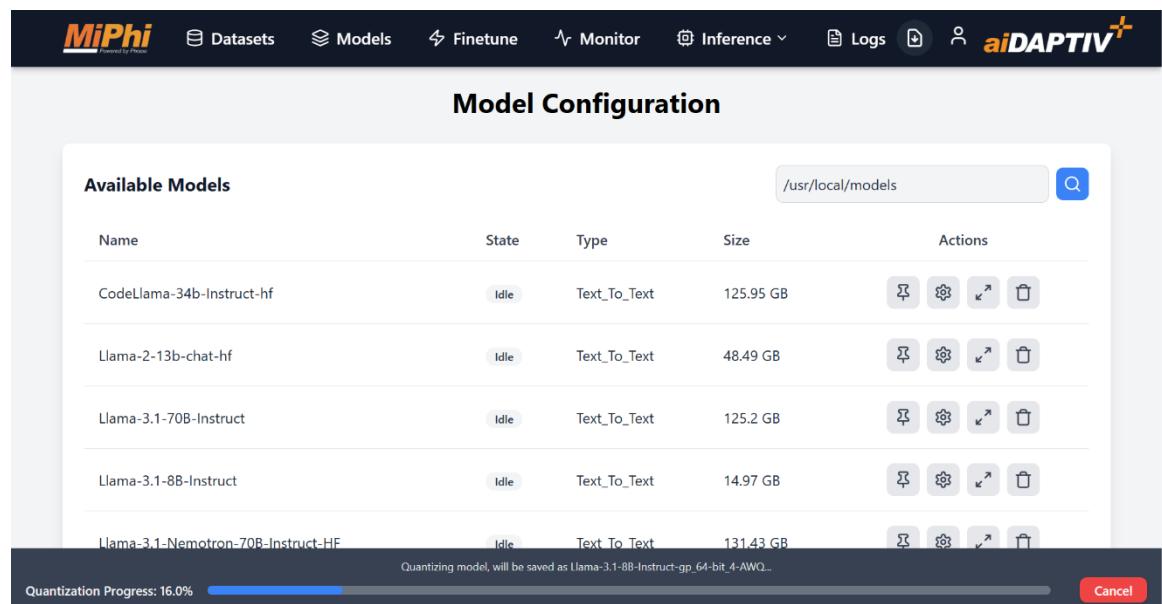
- **Group Size:** It refers to the number of elements that are quantized together as a unit. Select the desired group size for quantization. (default: 32)



- **Bit:** The bit value to quantize the model in lower precision data. (default: 4)
- After selecting all the options click “Yes” to start model quantization.

9.2.3 Monitor Quantization Progress

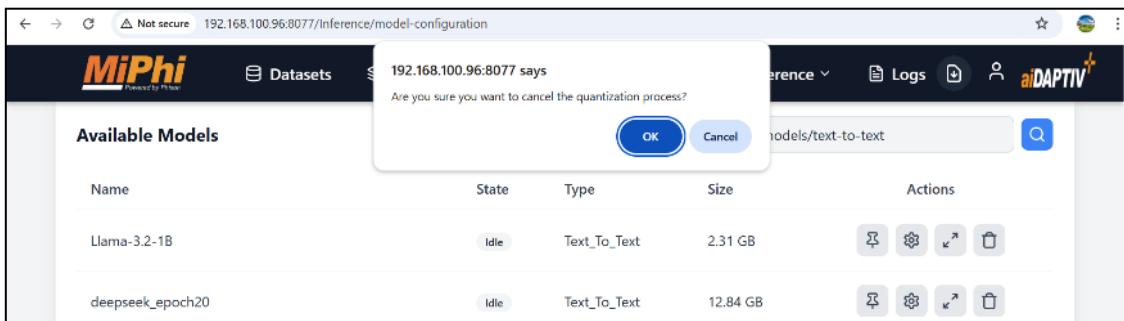
- You will get the below window when the model quantization starts. In the bottom a black progress bar appears.



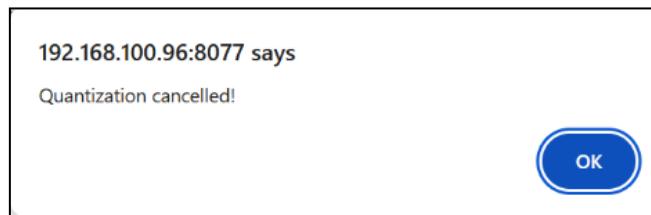
Available Models				
Name	State	Type	Size	Actions
CodeLlama-34b-Instruct-hf	Idle	Text_To_Text	125.95 GB	
Llama-2-13b-chat-hf	Idle	Text_To_Text	48.49 GB	
Llama-3.1-70B-Instruct	Idle	Text_To_Text	125.2 GB	
Llama-3.1-8B-Instruct	Idle	Text_To_Text	14.97 GB	
Llama-3.1-Nemotron-70B-Instruct-HF	Idle	Text_To_Text	131.43 GB	

Quantization Progress: 16.0% Quantizing model, will be saved as Llama-3.1-8B-Instruct-gp_64-bit_4-AWQ...

- The progress bar has the following components:
 - Quantization Progress: Left side is the real time progress percentage of model quantization
 - Quantized model path: Central is the quantizing message along with the name of quantized model
 - Cancellation: Right side is having one “Cancel” button to stop quantization. It will show one warning message after clicking “Cancel” to confirm whether you want to stop quantization.



After clicking “OK”, it will cancel quantization and confirmation message will appear.

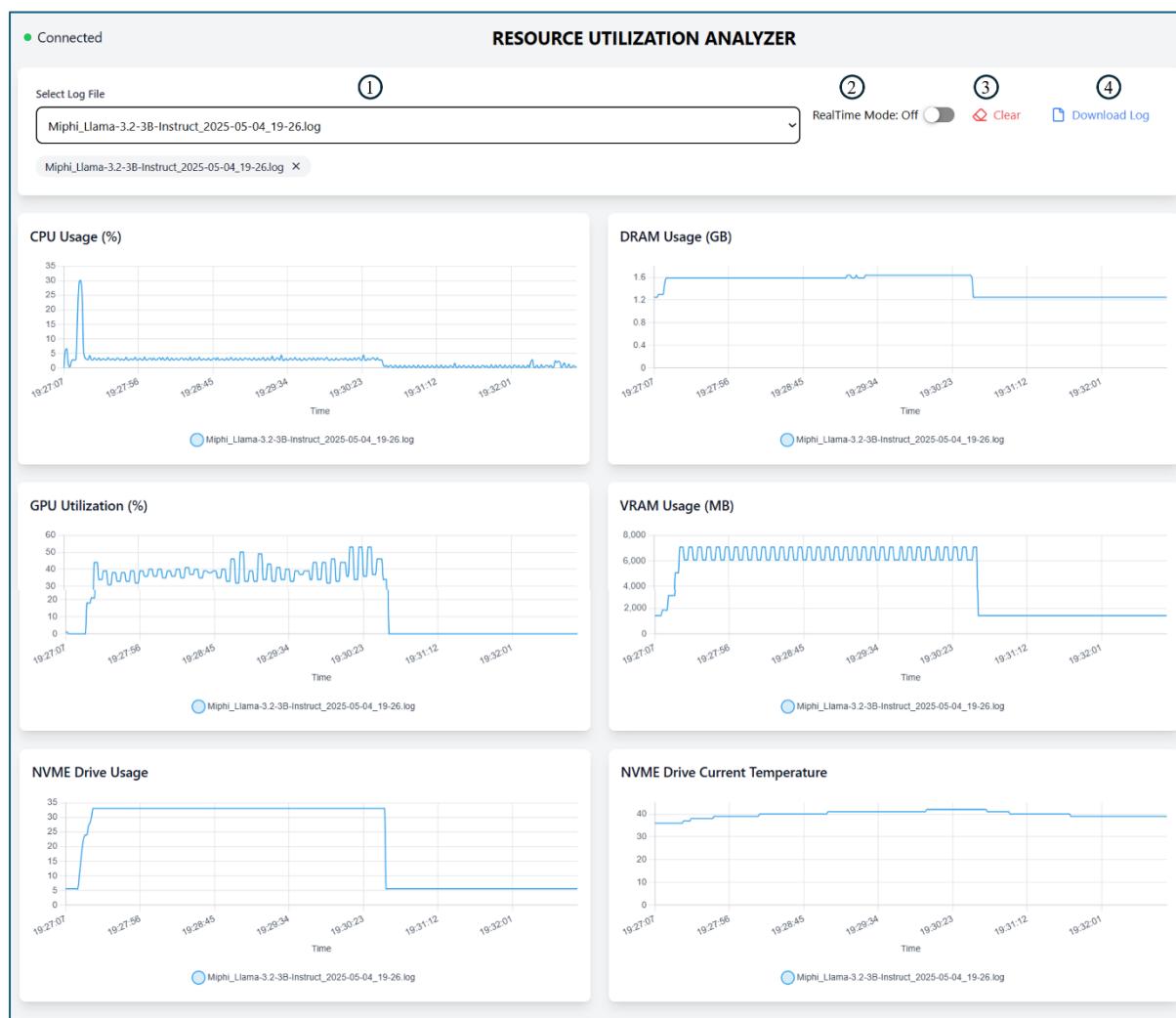


- When the quantization is completed successfully, the quantized model will appear in the model configuration page of the same directory. The quantized model name is followed by the group size and bit value.

Llama-3.1-8B-Instruct-gp_64-bit_4-AWQ	Idle	Text_To_Text	5.48 GB	
---------------------------------------	------	--------------	---------	--

10. MIPHI LOGS OVERVIEW

The **MiPhi Logs** offers a centralized view for monitoring and comparing training performance. Users can select up to six log files to visualize key metrics such as CPU/RAM usage, GPU utilization and VRAM, NVMe usage and temperature, loss vs iteration, iteration time breakdown (forward, backward, update), and overall training efficiency. It includes a comparative summary across all epochs, real-time analysis toggle for active training sessions, and options to download logs or clear selected files—enabling efficient tracking, diagnosis, and comparison of fine-tuning jobs.

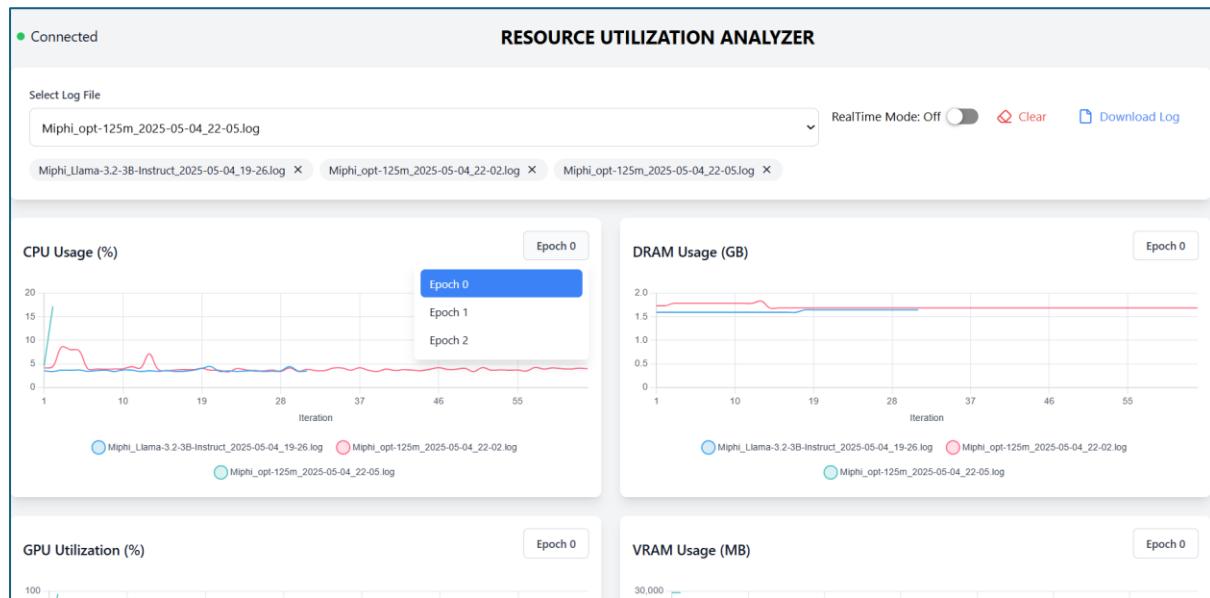


10.1 Log File Selection

This section allows users to select and compare up to six log files simultaneously. Upon selection, the dashboard displays system metrics (CPU usage, RAM usage, GPU utilization, VRAM usage, NVMe usage, and temperature) and training metrics (loss, forward time, backward time, update time, total iteration time, and training efficiency).

- For system metrics, the x-axis represents **time** when viewing a single epoch and **iterations** when comparing across multiple files.
- For training metrics, the x-axis is always **iterations**.

Users can also filter metrics by epoch for focused analysis.



10.2 Real-Time Analysis

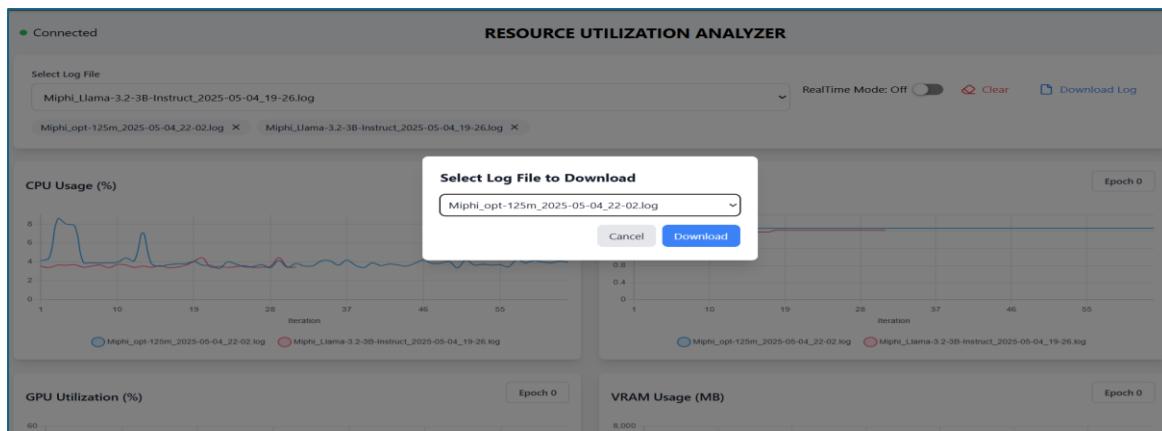
The **Real-Time Analysis** toggle allows users to monitor live system and training metrics during an active fine-tuning process. When enabled, it dynamically fetches and updates graphs in real time, providing up-to-date insights into performance, resource usage, and training behavior. This feature is only active when the selected training process is currently running.

10.3 Clear Selection

The **Clear** button allows users to quickly remove all selected log files from the comparison view. This feature helps reset the interface, enabling users to start a new selection without manually deselecting each file.

10.4 Download Logs

The **Download Logs** button allows users to export detailed log files for further analysis or archival. When a single file is selected, clicking the button downloads a ZIP archive named after the corresponding **Job ID**. This archive includes system metrics logs, dmesg logs, fine-tuning logs, execution logs for each page, and API logs.



If multiple log files are selected, the interface prompts the user to choose a specific file for download. This ensures precise access to the desired logs without downloading unnecessary data.

