OpsOrbital: Real-Time Server Operations from Your Pocket

Yuan Guo
gy2022@sjtu.edu.cn

Junran Zhang
junran_618@sjtu.edu.cn

Abstract

This technical report presents our group project for SJTU CS3338, Computer Network (Honor). In this work, we introduce OPSORBITAL, a novel and practical framework designed for the supervision of CS/AI-related experiments. OPSORBITAL offers a scenario-specific, convenient, and user-friendly solution for remote server management on mobile devices. Built on a server-client architecture, the framework consists of a server script running on the remote server and a client app on an Android device. Through thoughtful front-end and back-end design, OPSORBITAL enables users to easily inspect, control, and receive notifications about their remote server status via mobile devices. Key features include real-time monitoring of system resources, network details, and processes, as well as functionalities like running scripts, killing processes, adjusting hyperparameters, and receiving timely notifications. By simplifying server monitoring and control, OPSORBITAL significantly enhances the efficiency of computer science researchers, enabling them to manage their servers from anywhere. Project resources, including a video demo, code, and slides, are available on the official project website: opsorbital.github.io.

1 Introduction

Contemporary computer science research often involves time-intensive experiments, particularly in the burgeoning fields of AI and LLM research, where extensive model training and benchmark evaluations are routine. Researchers are typically unable to obtain immediate experimental results. As a result, a common practice is to execute experiments as background processes (using tools like tmux or screen) and intermittently monitor their progress. However, this approach often introduces several challenges:

- When researchers are away from their workstations (e.g., needing to check experiment status while out for other tasks), it becomes inconvenient to monitor the progress or control the process (e.g., terminate a specific experiment or initiate another script).
- Researchers may miss timely updates on the experiment status (e.g., completion or failure), leading to unnecessary delays and inefficiencies.

Despite the evident need for a convenient, portable, and user-friendly tool for monitoring and managing experiments, limited efforts have been made to address this gap. Existing solutions include remote desktop applications (ToDesk Inc., 2025; RustDesk Project, 2025) and mobile SSH clients (Feng, 2025), but these are often too generalized and cumbersome for specific use cases. For example, using command-line interfaces and keyboard shortcuts on mobile devices can be overly complex.

Other tools like tf-watcher (Dagli, 2023), developed some years ago, are restricted to monitoring TensorFlow-based models and require modifications to experimental code. Weight and Bias (W&B) (Biases, 2025), a popular framework for training monitoring, focuses on training processes and pays little attention to the server's overall system status. Additionally, both tf-watcher and W&B are primarily monitoring tools with limited support for control operations. As a result, they are not well-suited for real-world scenarios. According to our survey of over 50 Ph.D. students and researchers from institutions such as SJTU, PKU, HKU, and companies like ByteDance, no existing tools have gained wide adoption for addressing these specific needs.

To bridge this gap and provide an effective solution for computer science researchers, we propose an application that offers the following functionalities:

1. Effortlessly check the server's status (e.g., pro-

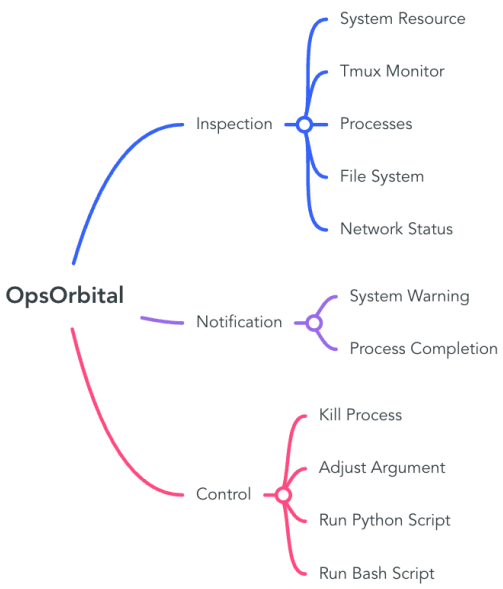

Figure 1: Function Overview of OPSORBITAL

cesses, GPU usage, CPU utilization, and network status) on mobile devices.

2. Conveniently track the progress of specific experiments (e.g., real-time training logs for model training) on mobile devices.

3. Receive timely notifications about experiment statuses (e.g., completion, unexpected termination, or system warnings).

4. Perform essential operations on mobile devices, including terminating processes, running scripts, managing script queues and setting experiment configurations.

To meet these requirements, we developed OPSORBITAL, a novel, lightweight, and practical framework for remote server supervision. The proposed framework comprises two components: a server program and a client application. Through a carefully crafted front-end and back-end design, OPSORBITAL is capable of handling a wide range of server-related tasks, including inspection, notification, and control. A detailed overview of OPSORBITAL's functionalities is provided in Figure 1.

Our project offers several key advantages:

1. **Novel:** To the best of our knowledge, this is the first mobile remote control framework specifically designed for task scheduling in computer science research. Our work incorporates innovative software designs to address challenging functionalities.

2. **User-Friendly:** The framework features an intuitive interface that simplifies remote monitoring and control, overcoming the typical challenges of using mobile devices for server operations.

3. **Lightweight:** OPSORBITAL's scenario-specific focus ensures a lightweight design that enhances accessibility and ease of use.

4. **Extensible:** The framework serves as a robust codebase for lightweight communication and control between mobile devices and remote servers. Developers can easily extend its capabilities. Additionally, with SSH forwarding, the framework is compatible with a diverse range of servers.

2 Related Work

2.1 Remote Device Control

Remote desktop and mobile SSH applications are two primary approaches that allow users to access and control computers from mobile devices. ToDesk (ToDesk Inc., 2025) provides users with full control over a remote device, while RustDesk (RustDesk Project, 2025) offers an open-source remote desktop platform with options for self-hosted servers, ensuring user privacy and control. Additionally, mobile SSH clients, such as Mobile SSH (Secure Shell) (Feng, 2025), enable users to manage remote servers directly through command-line from their smartphones, facilitating on-the-go system administration.

However, while these applications may appear useful, they fall short in addressing the specific needs of CS experiment monitoring. Remote desktop applications are typically designed for general operations, such as those performed on Windows systems, making them overly complex and redundant for the focused requirements of computer science research and experimentation. Similarly, mobile SSH applications lack user-friendly interfaces for mobile devices and do not support proactive notifications—an essential feature for mobile-based server monitoring tools. Additionally, performing operations that rely on keyboard shortcuts or command-line inputs is cumbersome on mobile devices, further limiting their practicality. Consequently, these solutions are not well-suited for the specific and popular scenarios faced by CS researchers.

2.2 AI Training Monitor

Efforts have also been made to develop AI training monitors that provide timely tracking of AI training processes. TF-Watcher (Dagli, 2023) is an open-source project designed for monitoring, visualizing, and sharing machine learning training progress on mobile devices. Other well-known frameworks, such as Tensorboard (Project, 2025b) and Weight-and-Bias (W&B) (Biases, 2025), are widely used for tracking and visualizing the status of AI tasks.

Although these tools can support the tracking of specific AI tasks to some extent, they have significant limitations. Many of these tools are highly dependent on the explicit integration of their packages into training scripts, restricting their applicability to single training process tracking. Moreover, these AI training monitors primarily focus on data analysis and visualization, with minimal support for system-level supervision and control operations. As a result, they fail to provide the comprehensive functionality required for broader server monitoring and control.

3 Design and Implementation of OpsOrbital

3.1 Overall Functions and Framework Design

In the specific scenario of computer science experiments, we developed OpsOrbital, a mobile device-based remote server management framework designed to address the challenges of remote experiment supervision. The framework revolves around three core functional categories: Inspection, Control, and Notification. This design prioritizes seamless interaction and efficiency, providing users with a practical and effective tool for server management.

1. **Inspection:** Users can obtain real-time information about server system resources through the mobile application, including:
   - CPU and GPU utilization
   - Memory usage and allocation
   - Network interface status
   - Running process list and detailed information
   - Real-time Tmux window interface preview
2. **Control:** The framework supports essential operations for computer science experiments, including:
   - Process management: Terminating specific processes
   - File system operations: Editing configuration files
   - GPU resource scheduling: Selecting specific GPUs to execute Python or Bash scripts
3. **Notification:** The application incorporates proactive monitoring and alerting capabilities:
   - Detecting system anomalies
   - Monitoring and detecting termination of Tmux window processes
   - Pushing critical events and notifications directly to the mobile device

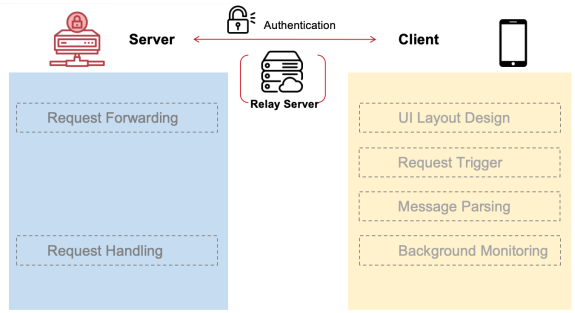

Figure 2: Design structure Overview of OpsOrbital

Architectural Design: The framework adopts a classic client-server architecture (as shown in Figure 2), with distinct client and server components to ensure flexibility and scalability:

- **Client (Mobile Application):**
  - Designed for responsive and intuitive interaction
  - Capable of managing diverse functional modules for inspection, control, and notifications
  - Handles network requests and processes server responses efficiently
- **Server-side:**
  - Handles client requests and provides system resource and process information
  - Implements robust user authentication mechanisms
  - Supports proxy forwarding for servers without public IP access

Implementation Details: The server was developed using Python with the Flask framework to ensure lightweight and efficient communication. Additionally, the framework includes a streamlined PC client for enhanced monitoring of CPU and

GPU usage. The mobile application was implemented for Android devices, adhering to the Model-View-Controller (MVC) architecture to maintain modularity and scalability.

Security remains a cornerstone of the framework, with mechanisms for user authentication, encrypted communication, and fine-grained access control to ensure the safety and privacy of server operations. This design effectively addresses the diverse needs of computer science experiments while delivering a user-friendly and efficient remote server management experience.

3.2 Account Management

For user account management, we implement login and registration functions, along with initial support for user group functionality.

The registration function is a critical component of the user management system. It begins with input parameter validation to ensure that the username and password fields are not empty. Uniqueness checks are performed for both the username and the device MAC address. By querying the database, the system verifies whether the username already exists and, if a MAC address is provided, ensures it is not already associated with another user. For user group management, if no group ID is specified during registration, a new device group is automatically created, named after the username, and the registering user is assigned as the group administrator. Once all validations are passed, relevant data is inserted into the users and devices tables, including user ID, hashed password, salt, email (optional), group ID, and device details such as device ID, name, type, token, MAC address, and online status. The registration function also incorporates robust error handling, specifically addressing database integrity issues and other exceptions, ensuring clear and actionable feedback to the user.

The login function is primarily focused on user verification. It queries the database using the provided username to retrieve the password, salt, and group ID, and then verifies the input password against the stored one using a password verification method. If a MAC address is provided, the system checks whether it is associated with another user. For device and session management, the function generates a unique device token and either inserts or updates the device information in the devices table. Additionally, a session record is created, containing details such as session ID, user ID, device ID, session token, and expiration time. The user's last login time is also updated. Upon successful login, the function returns a dictionary containing essential details such as the session token, user ID, group ID, and device ID, which are crucial for subsequent operations. Comprehensive exception handling ensures that all potential errors are addressed, detailed error information is logged, and a failure gracefully returns 'None', thereby enhancing the robustness and reliability of the authentication process.

3.3 Connection Management

In this section, we outline the connection management design of our project, including the connection protocol, methodology, and request forwarding implementation.

Our implementation adopts a standard client-server architecture and leverages Flask for communication management. When a user performs communication-related actions in the mobile app, such as requesting information or executing operations, these actions trigger corresponding requests to the server. For the mobile application, we utilize the OKHttp library, a robust and efficient HTTP client for Android. OKHttp streamlines network communication by offering features such as connection pooling, transparent GZIP compression, response caching, and asynchronous request handling to prevent blocking the main thread. These capabilities enhance both the efficiency and reliability of network communication, making OKHttp an optimal choice for managing mobile application requests (Project, 2025a).

In scenarios where the server lacks a public IP address or faces restrictions for external connections, we incorporate SSH forwarding with a relay server as a simple yet effective solution. Drawing inspiration from popular remote control applications like ToDesk, we deploy an intermediary server with a public IP address as the relay. This relay server facilitates secure communication between the monitored server and the client device. By establishing an SSH tunnel, the monitored server forwards its data or services through the relay server, effectively bypassing network constraints or the absence of a public IP. This method ensures seamless connectivity, reduces latency, and enhances security through encrypted communication channels. Additionally, it enables real-time monitoring and remote access without the need for complex network reconfigurations or VPN setups.

3.4 Server Information Transmission

With the connection mechanism in place, the next step involves designing various request types to retrieve detailed server information across multiple dimensions. Some app interfaces for server information demonstration is shown in Figure 3.

The back-end logic on the server side has been carefully implemented to facilitate the collection of diverse server information. For CPU and memory details, we utilize the platform library for processor information and psutil for core count, usage, and memory status. These metrics are returned in a structured dictionary format, which includes attributes such as core count, usage percentage, and frequency. To gather network details, psutil.net_if_addrs is used to obtain interface information, such as IP address and netmask. This data is organized into a list of dictionaries for each network interface. To access the file system, we provide an endpoint (/get_file_system) that recursively constructs a tree structure starting from the root directory. This data is transmitted in JSON format and includes hierarchical details about directories and files, with attributes like name, type (directory or file), and child nodes for directories. Permission errors or inaccessible paths are gracefully handled by adding error nodes to the tree structure. This implementation ensures a seamless and user-friendly way to query and represent complex system details in a format easily consumable by client-side applications.

A critical requirement for monitoring computer science experiments is the ability to observe real-time console output, such as training and evaluation logs. To address this, we have implemented this functionality specifically for tmux windows. Tmux, a terminal multiplexer, allows users to manage multiple terminal sessions within a single window. It is particularly advantageous for running and monitoring long-duration processes, parallel tasks, or remote sessions, as tmux sessions persist even after disconnection. Its additional benefits include enhanced productivity, session recovery, and the capability to split terminal screens for multitasking.

To support this functionality, we designed a Tmux table interface that allows users to view all sessions and windows on the remote server. Additionally, a Tmux detail interface synchronizes real-time details of specific tmux windows with the mobile application, enabling users to track experiment progress and monitor real-time logs. This same

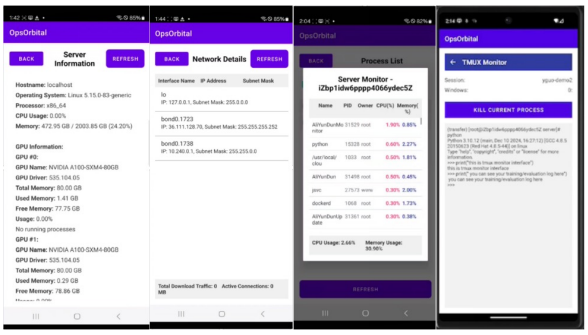

Figure 3: OPSORBITAL interface of system status, processes and tmux details

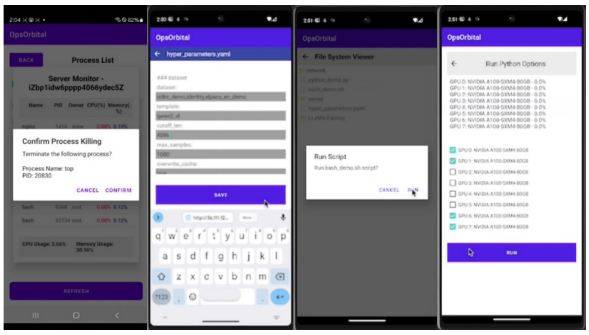

Figure 4: OPSORBITAL interface of terminating process, setting configuration, and running script

information transmission pipeline is also utilized in the background monitoring system for detecting process termination events.

3.5 Common Remote Control

We have implemented several simple yet essential control functionalities in our project, including process termination, execution of Python and Bash scripts with a single click, and adjusting hyperparameter configuration files through the mobile interface. Typical app interface for common remote control is demonstrated in Figure 4

Support for process termination is available in both the "My Process" and "Tmux Monitor" sessions. In the "My Process" session, users can interact with a specific process by selecting its name, which triggers a prompt asking whether to terminate the process. If the termination operation is confirmed, the client sends the corresponding process ID along with a ‘kill_process‘ request to the remote server. The server validates permissions and terminates the process associated with the provided process ID.

File editing support in the current version of OPSORBITAL is tailored to Llama-Factory argument configuration. Llama-Factory (Zheng et al.,

2024) is a widely used open-source framework for training Large Language Models (LLMs). It encompasses all stages of LLM training and supports nearly all commonly used open-source LLMs. Officially recommended by the Qwen group, the framework standardizes training configurations in a ‘.yaml‘ file format, with key-value pairs specifying the arguments. To facilitate this, we have implemented data transmission, parsing, and display logic, enabling users to edit these configurations directly through the mobile interface by adjusting editable fields. When the "save" option is selected, the updated YAML content is sent to the server as a string, where it overwrites the existing configuration file.

Another key control feature is the run-script functionality. In the file system interface, the back-end checks the file’s suffix upon selection. If the suffix matches predefined criteria (e.g., ‘.py‘ for Python scripts or ‘.sh‘ for Bash scripts), corresponding handling logic is invoked. For ‘.sh‘ scripts, the system creates a new tmux session and executes the script within it. For ‘.py‘ scripts, the client retrieves and displays GPU usage information from the server. The user can then select specific GPUs as needed and execute the Python script accordingly.

These control functionalities simplify remote server management, making essential operations more accessible and user-friendly for researchers.

3.6 Background Monitor

One of the key features of OPSORBITAL is its background monitoring functionality. This module allows the system to periodically check the status of the remote server, transitioning the server information retrieval process from a reactive to a proactive approach. By leveraging this functionality, users can receive timely notifications on their mobile devices regarding server anomalies or process termination events.

The background monitoring module comprises three main components: the Permission Checker, Boot Receiver, and Monitoring Service. The Permission Checker operates during the login stage of the OPSORBITAL app, ensuring that the necessary system permissions are granted and prompting users to adjust permission settings if required. The Boot Receiver listens for the ‘ANDROID_BOOT_COMPLETED‘ signal, enabling the monitoring module to initialize automatically upon device boot. The Monitoring Service periodically and automatically retrieves server status using

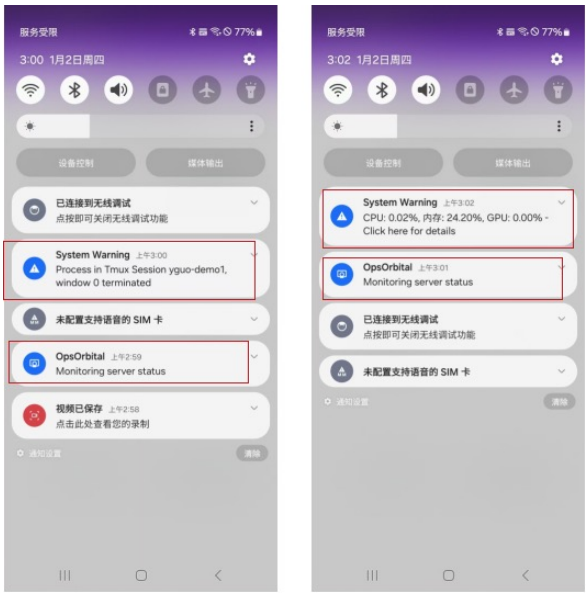

Figure 5: Two types of notification message in OPSORBITAL

the established connection module. When an abnormal server status is detected, such as anomalies or the termination of a process within a Tmux window, the system sends a high-priority notification directly to the user’s mobile device.

This proactive background monitoring functionality enhances the user experience by providing real-time updates and alerts, ensuring that users remain informed about critical server events without requiring constant manual intervention.

3.7 Security Management

Considering our application offers remote control of the remote server like killing certain process, running script and editing configuration file, we have adopted multiple approaches to ensure the security of our system.

3.7.1 Authentication Mechanism

1. **Password Security**
   - Utilizes PBKDF2 (Password-Based Key Derivation Function 2) with SHA-256
   - Applies 100,000 iterations for password hashing
   - Generates a unique cryptographic salt for each user
   - Prevents rainbow table and dictionary attacks
2. **Session Management**
   - Generates UUID-based session tokens
   - Token expiration set to 7 days

- Implements stateless authentication mechanism
- Supports multiple device sessions per user

3. **Request Protection**
   - Rate limiting to prevent brute-force attacks
   - IP-based request tracking
   - Anti-replay protection using request hash tracking
   - Maximum 10 requests per 60 seconds per IP

3.7.2 Security Enhancements

- **Input Validation**
  - Regex-based input sanitization
  - Prevents SQL injection and XSS attacks
  - Enforces strict input length and character restrictions
- **Logging and Auditing**
  - Comprehensive event logging
  - Tracks authentication attempts
  - Captures system and user activity

Key Security Principles:

- One-to-one device and user binding
- Cryptographically secure password storage
- Flexible, role-based group management
- Comprehensive request validation
- Continuous security monitoring

4 Future Work

We acknowledge that not all of our ideas have been fully implemented, primarily due to the constraints of being a two-person team with limited time. Beyond the designs presented above, we have identified several areas for improvement and further development in subsequent versions:

1. **Enhanced UI Design and Visualization:** We aim to improve the interface layout to make it more user-friendly and intuitive. Additionally, we plan to incorporate more advanced visualization features. For instance, GPU usage could be displayed not only as text but also through graphical representations, making it easier for users to interpret the data at a glance.
2. **Support for Additional Operations:** Leveraging the extensible nature of the OPSORBITAL framework, we plan to continue expanding its functionality to include more common operations. For example, in the hyper-parameter adjustment feature, our current implementation supports only '.yaml' files for Llama-Factory configuration. Future updates will extend support to other data formats, such as '.json', for training and evaluation configurations.
3. **Improved Security Measures:** We aim to enhance the security of the framework by refining the authentication mechanism and transitioning to more secure communication protocols. This will ensure better protection of user data and server operations.
4. **iOS App Development:** Currently, the mobile app is available only for Android devices. We plan to develop an iOS version to make this tool more widely accessible to a broader range of users.

5 Conclusion

In this work, we introduced OPSORBITAL, a novel and lightweight framework for real-time remote server supervision, specifically designed to address the unique needs of computer science researchers. By providing a seamless interface for mobile devices, OPSORBITAL bridges the gap between server management and convenience, empowering users to monitor, control, and receive timely notifications about their remote servers regardless of their location. Through its innovative design, the framework achieves a delicate balance between functionality, user-friendliness, and extensibility.

Our implementation not only focuses on system resource inspection, real-time training log monitoring, and proactive notifications but also extends to remote control functionalities such as process termination, script execution, and configuration file editing. Security remains a cornerstone of our framework, ensuring that user data and operations are protected through robust authentication mechanisms, encrypted communication, and fine-grained access control.

With its scenario-specific, lightweight, and extensible design, OPSORBITAL stands as a versatile and valuable tool for researchers. Beyond the functionalities presented, the framework holds significant potential for future enhancements, including

better visualization, support for more operations, improved security, and expansion to additional platforms like iOS. We believe OPSORBITAL not only fills an important gap in existing tools but also opens new avenues for efficient server management in computer science research.

6 Acknowledgment

We extend our heartfelt gratitude to Prof. Linghe Kong for his exceptional teaching, which introduced us to the fascinating world of Computer Networks. His thoughtfully designed course structure encouraged us to freely explore this domain, and his invaluable feedback played a crucial role in refining our project. We also sincerely thank Dr. Jiaming Liu for his dedication and support as our TA throughout this endeavor.

References

Weights & Biases. 2025. W&b: Experiment tracking and model management tool. https://github.com/wandb/wandb. Accessed: 2025-01-02.

Rishit Dagli. 2023. Tf-watcher: A tool for monitoring tensorflow experiments. https://github.com/Rishit-dagli/TF-Watcher. Accessed: 2025-01-02.

Gao Feng. 2025. Mobile ssh (secure shell). https://play.google.com/store/apps/details?id=mobileSSH.feng.gao. Accessed: 2025-01-03.

OkHttp Project. 2025a. Okhttp. https://square.github.io/okhttp/. Accessed: 2025-01-02.

TensorBoard Project. 2025b. Tensorboard: Tensorflow's visualization toolkit. https://www.tensorflow.org/tensorboard. Accessed: 2025-01-02.

RustDesk Project. 2025. Rustdesk: Open-source remote desktop with self-hosted server solutions. https://rustdesk.com. Accessed: 2025-01-03.

ToDesk Inc. 2025. Todesk: Free, secure, and smooth remote desktop software. https://www.todesk.com. Accessed: 2025-01-03.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.