

**AN INTELLIGENT ELECTRICITY MANAGEMENT
UNIT: AI-DRIVEN POWER FORECASTING AND
PERSONALIZED CONSUMPTION INSIGHTS WITH
APPLICATION INTEGRATION**

Welikalage R.Y.W.

IT21808166

B.Sc. (Hons) Degree in Information Technology Specializing in
Information Technology

Department of Information Technology
Sri Lanka Institute of Information Technology
Sri Lanka

August 2025

SMARTENERGY AI GUIDE – AI-DRIVEN ENERGY MANAGEMENT SYSTEM FOR PERSONALIZED, REAL-TIME HOUSEHOLD OPTIMIZATION

Welikalage R.Y.W.

IT21808166

The dissertation was submitted in partial fulfilment of the requirements for the
Bachelor of Science (Hons) in Information Technology specializing in Information
Technology

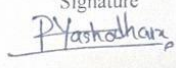
Department of Information Technology
Sri Lanka Institute of Information Technology
Sri Lanka

August 2025

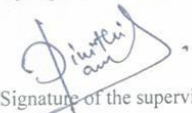
DECLARATION

“I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”

Name	Student ID	Signature
Welikalage R.Y.W.	IT21808166	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

 Signature of the supervisor (Ms. Dinithi Pandithage)	Date: 08/29/2025
--	------------------

ABSTRACT

This study presents the design and implementation of an AI-driven home energy management system that transcends conventional monitoring by converting device-level IoT data into personalized, actionable guidance to reduce household electricity costs. The cloud-native platform ingests and processes high-frequency sensor streams, fine-tunes a large language model (LLM) to user-specific consumption patterns, and generates targeted recommendations such as shifting appliance schedules, highlighting wasteful behaviors, and adopting cost-effective practices rather than simply reporting usage or forecasts. Core capabilities include device-wise analytics, real-time visualization, predictive bill forecasting, and a generative AI module that proposes practical next steps, all delivered through an inclusive interface with color-blind-friendly themes. Pilot evaluations indicate that these tailored insights increase user engagement and motivate proactive behavior change, enabling more informed energy decisions. Overall, the research contributes a scalable, adaptive framework that integrates IoT, predictive analytics, and LLM-based personalization to transform raw sensor data into decision support that enhances household energy efficiency and advances broader sustainability objectives.

Keywords: AI energy management, IoT, LLMs, personalized recommendations, bill forecasting, energy efficiency.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to all those who supported and guided me throughout the course of this research project.

First and foremost, I extend my heartfelt appreciation to my supervisor, Ms. Dinithi Pandithage, and my co-supervisor, Mr. Ashvinda Iddamalgoda, for their invaluable guidance, constructive feedback, and continuous encouragement at every stage of this research. Their expertise and insights were instrumental in shaping both the direction and the successful completion of this study.

I am also sincerely thankful to the academic staff of the Department of Information Technology, SLIIT, for providing the foundation and resources necessary to carry out this work. Special appreciation is extended to my colleagues and peers for their thoughtful discussions, suggestions, and collaboration, which greatly contributed to refining many aspects of the project.

I owe profound gratitude to my family for their unwavering support, patience, and encouragement throughout this journey. Their belief in me gave me the strength and determination to stay focused and dedicated.

Finally, I would like to acknowledge everyone who, in one way or another, directly or indirectly contributed to the successful completion of this project. This achievement would not have been possible without their collective support and cooperation.

Table of Contents

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
LIST OF FIGURES.....	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	vii
INTRODUCTION	1
Background and Literature Survey	1
Research gap	3
Research Problem	6
Research Objective	9
METHODOLOGY	11
Methodology	11
Commercialization Aspect of the Product.....	21
Testing and Implementation	25
RESULT AND DISCUSSION	34
Result	34
Research Findings.....	35
Discussion	35
CONCLUSIONS.....	36
REFERENCE	37
APPENDICES	39
Appendix A: Sample Instruction–Input–Output Pairs for LLM Training.....	39
Appendix B: Import Code Snippets	40
B.1 Fine-Tuning Pipeline Using LoRA and PEFT	40
B.2 OCR-Driven Appliance Specification Extraction and Efficiency Recommendation Script	42
Appendix C: Training in Run Pod	47

C.1 RunPod GPU Cloud Interface for Model Fine-Tuning Deployment ...	47
C.2 Training Progress and Loss Convergence during GPT-Neo LoRA Fine-Tuning on RunPod	47
C.3 Uploading Fine-Tuned GPT-Neo Model to Hugging Face from RunPod Environment	48
C.4 AWS EC2 Deployment Interface for LLM Inference Server	48
C.5 Hugging Face Model Repository for Fine-Tuned GPT-Neo Energy Recommendation System	49
C.6 API Response for Laptop Energy Optimization via LLM Endpoint	49

LIST OF FIGURES

Figure 1 - LLM Architecture Diagram	12
Figure 2 - Power Factor Calculation Graph.....	14
Figure 3 - API Integration for Real-Time Energy Recommendation Generation	17
Figure 4 - VoltFlow Web Dashboard – AI-Powered Device Insight and Efficiency Guide	19
Figure 5 - Training vs Validation Loss per Epoch	27
Figure 6 - Perplexity Over Training Steps	28
Figure 7 - Gradient Norms over Time during LoRA Fine-Tuning of GPT-Neo-1.3B	29
Figure 8 - RunPod Interface	47
Figure 9 - Training Progress.....	47
Figure 10 - Model Upload to Hugging Face	48
Figure 11 - LLM Deployment on AWS EC2	48
Figure 12 - Hugging Face Model Page.....	49
Figure 13 - LLM API Response (Laptop Gaming)	49

LIST OF TABLES

Table 1. 1: Identified Research Gaps	5
--	---

LIST OF ABBREVIATIONS

ABBREVIATIONS	DESCRIPTION
AI	Artificial Intelligence
IoT	Internet of Things
ML	Machine Learning
LLM	Large Language Model
HEM	Home Energy Management
PF	Power Factor
LoRA	Low Rank Adaption
PEFT	Parameter-Efficient Fine-Tuning

INTRODUCTION

Background and Literature Survey

The increasing demand for electricity, rising costs, and growing environmental concerns have amplified the need for efficient and intelligent energy management systems in households. Current energy management solutions often fall short, primarily focusing on aggregated energy consumption metrics without offering granular insights into individual device usage. This lack of specificity leaves users unable to make informed decisions about optimizing their electricity consumption, leading to higher bills and increased environmental impact. Research in this domain has explored IoT-based systems for real-time energy monitoring, which enable device-level data collection and pave the way for more precise management of energy usage [1]. Similarly, predictive modeling techniques, including machine learning algorithms such as neural networks and decision trees, have been applied to energy consumption forecasting to improve accuracy and identify patterns in energy usage [2]. However, these systems typically fail to integrate intelligent feedback mechanisms that provide actionable recommendations tailored to the unique behaviors of individual users. In parallel, advancements in artificial intelligence, particularly in the domain of generative AI, have introduced new opportunities for personalization in various fields. Large language models (LLMs), such as GPT-3 and GPT-4, have demonstrated remarkable capabilities in delivering context-aware and personalized suggestions across diverse domains, including healthcare, education, and e-commerce [3]. These advancements hold immense potential for revolutionizing energy management by leveraging real-time data to offer dynamic, user specific recommendations that encourage energy-efficient practices. Furthermore, recent studies in personalized home energy management systems highlight the importance of integrating IoT, AI, and behavioral insights to promote sustainable energy use and optimize household

energy consumption [4]. Despite these developments, a significant research gap remains in combining device-level energy monitoring, predictive analytics, and AI-driven personalized insights into a single, unified platform. The proposed research aims to address this challenge by developing an Intelligent Electricity Management Unit that integrates IoT-based real-time device monitoring, machine learning-powered time-series forecasting, and generative AI-driven recommendations. This system will not only enhance energy management but also empower users to make data-driven decisions that reduce electricity consumption, lower costs, and minimize carbon emissions. The user-friendly web application will serve as a central interface, displaying detailed energy analytics, predictive insights, and personalized recommendations through an intuitive design that emphasizes accessibility and ease of use. This holistic approach aligns with global sustainability goals by fostering energy-conscious behaviors while leveraging cutting-edge technologies like IoT, AI, and human-computer interaction principles.

By mastering IoT sensor networks, developing advanced machine learning frameworks for forecasting, applying generative AI techniques for personalized recommendations, and adhering to user-centered design principles, this research promises to contribute significantly to the field of home energy management while addressing critical environmental and economic concerns. This research aims to bridge the gap in household energy management systems by integrating IoT-based real-time device monitoring, machine learning for predictive analytics, and generative AI to deliver personalized energy-saving recommendations. By leveraging advanced technologies and user-friendly interfaces, the proposed system seeks to empower users to optimize energy consumption, reduce costs, and contribute to global sustainability goals through conscious decision-making.

Research gap

The integration of AI into HEMS has advanced significantly in areas such as energy forecasting, device-level monitoring, and basic energy-saving suggestions. However, significant gaps remain in developing a truly integrated and user-centric energy management system. Current solutions lack the ability to combine real-time IoT-based tracking, predictive energy forecasting, and generative AI for providing personalized recommendations that adapt to individual user behavior and preferences.

Study	Focus	Limitations
IoT-HEMS Framework [5]	Real-time energy monitoring using IoT devices, providing notifications for energy saving actions to optimize energy consumption.	Lacks advanced predictive models and fails to provide recommendations based on long- term user behavior or contextual data
SmartHomeEnergyAI [6]	A generative AI-based approach for creating basic energy-saving suggestions tailored to general household patterns.	SmartHomeEnergyAI is limited in its adaptability and fails to incorporate real-time IoT data streams or simulate potential outcomes for user engagement.
Design and Implementation of Cloud-IoT-Based HEMS [7]	This study presents the design and implementation of a home energy management system (HEMS) that collects and	While the system supports real-time monitoring and data storage, it does not integrate advanced

	stores energy consumption data using IoT devices.	machine learning models or provide personalized recommendations
Real-time personalized energy saving recommendations (EM) ³ [8]	This work introduces a recommendation engine that provides real-time personalized suggestions for energy saving, using sensors and actuators to monitor and control devices.	While (EM) ³ supports real-time recommendations and user interaction, it relies heavily on user input for decision-making and lacks advanced predictive modeling for long-term optimization
NILM-HEMS (Appliance disaggregation) [9]	Uses non-intrusive load monitoring (NILM) to infer appliance-level usage from a single smart-meter signal and surface device-specific insights	Disaggregation errors on overlapping loads; limited personalization; weak long-term prediction and “what-if” simulation
Appliance-level LSTM Forecasting [10]	Short-term per-device forecasts to flag spikes and plan shifting (laundry, dishwasher, A/C).	Forecasts alone don’t decide actions; no integrated actuator control; limited linking to user goals (comfort, time).
TOU/MILP Optimizer [11]	Mixed-integer linear programming to schedule high-load appliances	Assumes static user behavior and perfect forecasts; no generative

	around time-of-use tariffs and peak/shoulder periods.	feedback; brittle when tariffs/weather change.
--	---	--

Table 1. 1: Identified Research Gaps

Identified Gap:

1. Lack of Real-Time Personalization:

Solutions like the Cloud-IoT-Based HEMS focus on real-time energy monitoring but do not provide actionable, personalized recommendations tailored to individual users' consumption behaviors.

2. Absence of Long-Term Adaptation and Learning:

Systems such as the IoT-HEMS Framework and SmartHomeEnergyAI lack the capability to learn from historical user interactions or adapt recommendations over time, limiting their ability to evolve with user behavior.

3. Insufficient Context-Aware Recommendations:

The (EM)³ Recommendation Engine provides real-time suggestions but does not integrate advanced contextual data like external environmental conditions or pricing structures for more impactful energy-saving advice.

4. Limited Predictive Modeling Capabilities:

Studies such as the Design and Implementation of a Cloud-IoT-Based HEMS focus on forecasting or monitoring but fail to leverage predictive machine learning models for long-term optimization of energy consumption.

5. Fragmented Integration of Generative AI and IoT:

While systems like SmartHomeEnergyAI employ generative AI, they do not integrate IoT data streams with predictive modeling or provide simulations of energy-saving outcomes, reducing engagement and practical value.

While current energy management systems lack real-time personalization, long-term adaptation, context-aware recommendations, and an integrated approach combining IoT, machine learning, and generative AI, the system proposed in this research aims to address these gaps. By integrating IoT-based real-time tracking, machine learning for energy forecasting, and generative AI-driven personalized recommendations, this project will deliver an efficient and adaptive energy management solution. This approach has the potential to significantly improve household energy optimization, encourage proactive user engagement, and contribute to long-term sustainability goals.

Research Problem

One of the major challenges in household electricity management is enabling users to monitor, predict, and optimize energy consumption for individual appliances effectively. Current systems lack the ability to provide granular, personalized, and actionable recommendations, leaving users unable to make informed decisions that can significantly reduce electricity costs and environmental impact. This research aims to address this gap by developing an integrated system that combines IoT-enabled device monitoring, predictive analytics, and generative AI to offer real-time, tailored insights for electricity management.

Problem Statement:

How can an intelligent energy management system be developed to provide personalized, real-time, and actionable recommendations for individual household appliances, and what impact does it have on optimizing electricity usage and fostering sustainable practices?

Key Features of the Research Problem:

1. IoT-Enabled Appliance Monitoring:

- The system will utilize IoT devices to measure and manage energy consumption at the individual appliance level, offering users precise control and insights into their usage patterns.

2. Predictive Analytics with Time-Series Models:

- Advanced machine learning models will predict future energy consumption patterns, enabling users to plan and optimize their electricity usage proactively.

3. Generative AI for Personalized Recommendations:

- A custom large language model (LLM) will analyze energy consumption trends and provide personalized, context-aware suggestions tailored to specific devices and user behavior.

4. Context-Aware Insights:

- The system will integrate external factors such as time-of-use tariffs, weather, and household occupancy patterns to deliver contextually relevant and actionable energy saving recommendations.

5. User-Friendly Web Application:

- An intuitive web interface will display real-time data visualizations and actionable insights, ensuring effective human-computer interaction for energy optimization.

Significance of the Research Problem:

This research aims to revolutionize household energy management by addressing the limitations of existing systems and delivering a comprehensive, user-centric solution. The proposed system has the potential to empower users with the tools and insights needed to reduce electricity bills, minimize carbon footprints, and adopt sustainable energy practices. By leveraging IoT, machine learning, and generative AI, this tool not only enhances energy optimization but also contributes to global sustainability goals, making it a valuable innovation for future smart homes.

Research Objective

Main Objective

The project aims to create an advanced Intelligent Electricity Energy Management Unit that helps households monitor, predict, and optimize electricity usage. This system will integrate IoT-based device tracking to monitor real-time energy consumption of appliances and devices. Machine learning algorithms will be used for energy forecasting, analyzing historical consumption data to predict future usage patterns, which enables proactive management. Additionally, generative AI will provide personalized recommendations based on individual household energy usage habits, suggesting ways to save energy and reduce costs. All of this will be accessible through a user-friendly web application, offering an intuitive platform for efficient energy management.

Specific Objectives

1. Data Collection and Preprocessing:
 - 1.1. Develop a robust data collection framework to gather real-time electricity usage data from IoT-enabled devices across various household devices.
 - 1.2. Clean and preprocess the collected data to ensure consistency and accuracy for subsequent analysis.
 - 1.3. Implement privacy and security protocols to safeguard user data during the collection and preprocessing phases.

2. Personalized Recommendation Model Development:
 - 2.1. Utilize AI algorithms to analyze individual electricity consumption patterns and identify key factors influencing energy usage.
 - 2.2. Design and implement machine learning models that provide tailored energy saving recommendations based on user consumption habits, time-of-day usage, and appliance efficiency.
 - 2.3. Ensure the recommendation engine continuously learns from new data to refine its suggestions over time.
3. Model Validation and Fine-Tuning:
 - 3.1. Conduct extensive testing and validation of the AI models to ensure their recommendations are accurate, practical, and aligned with user preferences.
 - 3.2. Fine-tune the models by incorporating user feedback and real-world performance data, optimizing energy efficiency and user satisfaction.
4. Energy-Saving Simulation and Visualization:
 - 4.1. Develop an interactive simulation tool that allows users to visualize the potential energy savings and environmental impact of following the recommendations.
 - 4.2. Incorporate gamification elements into the visualization tool to encourage user engagement, participation, and proactive energy-saving behaviors.
5. Generative AI Integration and Real-Time Recommendations:
 - 5.1. Seamlessly integrate the generative AI-based recommendation system into the platform for real-time, actionable suggestions on energy usage optimization.
 - 5.2. Ensure the system is adaptable and responsive to real-time data inputs, offering timely recommendations as user habits evolve.

- 5.3. Enable accessibility features that make the platform easy to use for diverse user groups, fostering inclusive participation and promoting sustainability goals.

METHODOLOGY

Methodology

The methodology of this research is designed around the integration of IoT-enabled data collection, data preprocessing, predictive modeling, and fine-tuning of a Large Language Model (LLM) to deliver personalized, actionable energy-saving recommendations. The overall objective is to transform raw household electricity usage data into practical insights that guide users in reducing their monthly electricity bills while supporting sustainable energy practices.

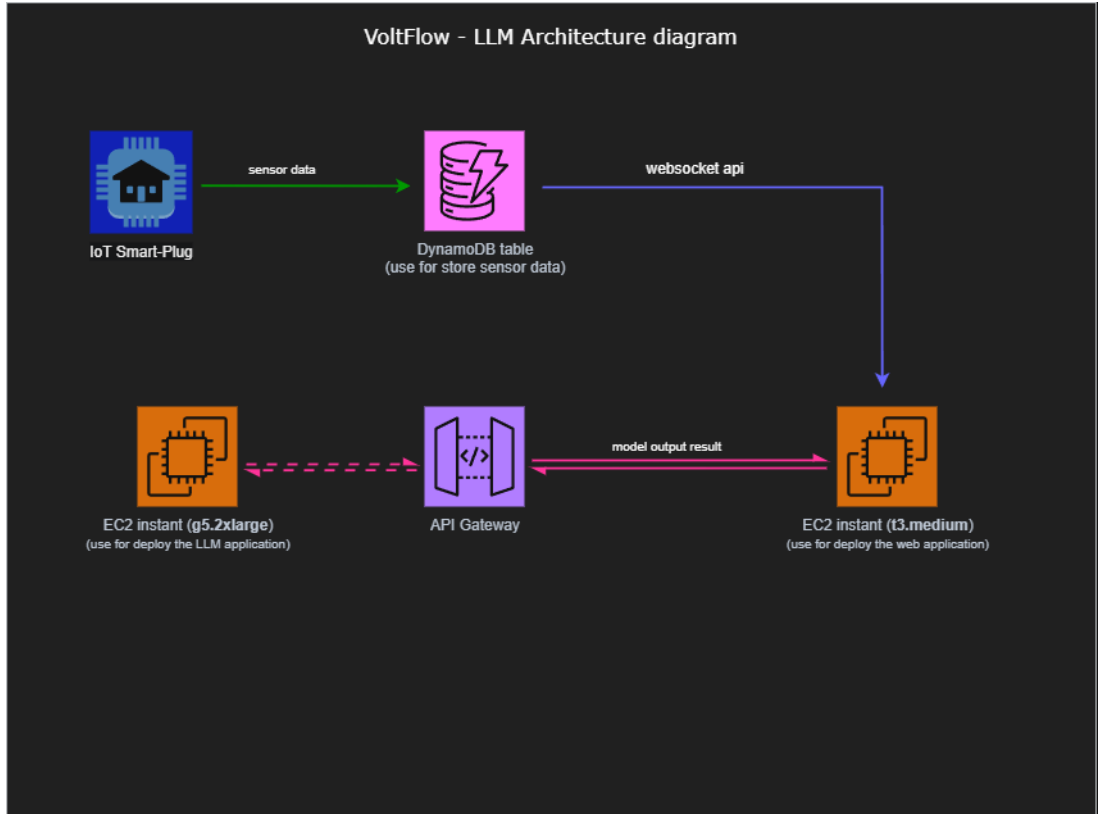


Figure 1 - LLM Architecture Diagram

1. Data Collection and Preprocessing

In our pipeline, each IoT field is transformed into compact features that condition the fine-tuned LLM to generate precise, household-specific energy and scheduling advice without any bill or tariff calculations: `vol` (voltage) and `cur` (current) combine—optionally with power factor—to estimate real power and detect brownouts or over-current episodes; `ts` (timestamps) anchors every reading to daily/weekly rhythms and user-defined “quiet/active” windows so the model can reason about shifting loads to low-impact periods; `e_int` (interval energy, kWh) is the core signal we aggregate into windowed summaries (e.g., peak-period kWh, off-period share, rolling 15-min mean/std) and use to quantify purely energy-based improvements (ΔkWh), while `e_tot` (cumulative kWh) provides integrity checks ($\Delta e_{\text{tot}} \approx \sum e_{\text{int}}$) and long-horizon trends

[13]; `run` (runtime; 1.0 = 1 hour) yields duty cycle, cycle counts, and adoption signals (did usage actually move to the recommended windows?); `int_t` (internal temperature) and `ext_t` (external temperature) give thermal context—flagging potential inefficiencies when device temperature rises at near-constant load, and enabling weather-aware tips via simple heating/cooling degree metrics; `id` (MAC/device id) keys personalization by mapping devices to classes/ratings captured via OCR; and `int` (5-second interval index) supports lagged/rolling statistics for short-term dynamics. After cleaning (outlier filters, gap handling, unit normalization) and enrichment with non-monetary context (occupancy patterns, user preferences), we pass only these summaries—plus the device profile and goals—into the LLM; the model is instruction-tuned on pairs where inputs are these features and outputs are constrained, quantified recommendations that target energy (kWh) reduction, safer operation, and maintenance hygiene, then further adapted using preference signals from explicit feedback and implicit outcomes (changes in `run` and `e_int`), yielding a coach that learns each home’s habits while grounding every suggestion in your telemetry columns—without performing any cost computation.

Power factor (PF) is included in training because it captures how efficiently a device converts electrical input into useful work and exposes load characteristics that raw voltage/current or energy alone can’t [12]. First, PF separates true load changes from line-condition artifacts: for the same V and I , a drop in PF indicates rising reactive components (inductive/capacitive effects), helping the model avoid misreading voltage dips or inrush currents as sustained consumption. Second, PF is a strong device-type and state signature (e.g., compressors, pumps, fans show PF trajectories during start, steady run, and cycling), improving appliance disaggregation cues, runtime detection, and “is this behavior normal for this device?” judgments. Third, low or drifting PF at near-constant I often co-occurs with inefficiency and emerging faults (e.g., clogged filters raising motor slip, failing run capacitors, poor mechanical alignment), which aligns directly with our coach’s maintenance and safety recommendations—even without using any tariff/cost data. Fourth, PF stabilizes

feature engineering by providing a voltage-agnostic efficiency indicator: features like “minutes with $PF < 0.75$,” “PF slope after start,” and “PF–temperature correlation (PF vs int_t)” let the LLM reason about actionable tactics (reduce idle spinning, fix ventilation, schedule service) grounded in physics rather than heuristics. Finally, including PF in training improves generalization across homes and seasons: the model learns patterns tied to electromechanical behavior rather than site-specific voltage levels, yielding more reliable, device-specific guidance and anomaly detection based purely on energy and operational quality not on cost.

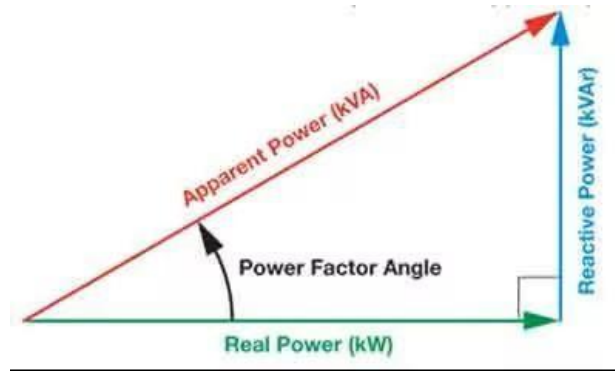


Figure 2 - Power Factor Calculation Graph

2. LLM Fine-Tuning and Training

At the core of this research lies the deployment of a fine-tuned Large Language Model (LLM), which was trained to generate personalized energy-saving suggestions. A pre-trained transformer-based model was selected for its capability to process both structured IoT telemetry data and unstructured contextual prompts. The fine-tuning process incorporated preprocessed IoT datasets together with domain-specific knowledge, including appliance efficiency standards, behavioral energy-usage patterns, and tariff-based cost calculations.

The LLM was trained to interpret historical consumption data and translate it into actionable energy-optimization strategies. For instance, instead of offering a generic tip such as *“turn off unused appliances,”* the fine-tuned model could generate context-aware recommendations like: *“Your refrigerator consumed 18% more energy than average last month—defrosting it regularly could lower your bill by approximately 6%.”*

Model fine-tuning and optimization were conducted on the RunPod GPU Cloud platform, which provided scalable, on-demand computing resources for deep learning workloads. Using RunPod’s GPU Pod deployment interface, various NVIDIA GPUs—such as RTX 5090 (32 GB VRAM), A40 (48 GB VRAM), and H100 PCIe (80 GB VRAM)—were provisioned based on task complexity and budget efficiency. RunPod’s flexible environment enabled high-throughput model training, efficient checkpoint management, and seamless migration of the final fine-tuned weights. After completion, the trained LoRA/PEFT adapter modules were exported and pushed to Hugging Face for cloud-based inference.

In our implementation we used the base model gpt-neo-1.3B (open-source model from EleutherAI on the Hugging Face platform) and applied Low-Rank Adaptation (LoRA) and Parameter-Efficient Fine-Tuning (PEFT) methods to fine-tune it. [14] LoRA and PEFT are important for several reasons:

1. Resource efficiency: Full fine-tuning of large models (i.e., updating all parameters) quickly becomes computationally expensive, memory-intensive, and difficult to deploy for domain-specific tasks. PEFT techniques (including LoRA) allow us to freeze most of the pre-trained model’s weights and fine-tune just a small subset of additional parameters or adapters, dramatically reducing GPU/memory/storage costs. [15]
2. Maintaining general-purpose knowledge: Because most of the pre-trained model is frozen, the base model retains its broad comprehension and generative capabilities, while the adapters specialize on our domain (energy monitoring,

IoT telemetry, household behaviors). This helps avoid “catastrophic forgetting” of the foundational language model capabilities.

3. Modularity and deployment flexibility: With LoRA/PEFT, the fine-tuned parameters are small modules (often a few % of the total parameters), so one base model can host multiple adapters for different tasks (e.g., energy coaching, maintenance alerts, scheduling advice) without having to duplicate the entire large model for each task.
4. Faster iteration and update: Because training in fewer parameters is faster and requires fewer resources, we can more rapidly iterate on the fine-tuning dataset (e.g., adding new feedback or updated appliance behavior), and adapt the model as more user data arrives — aligning with our pipeline design of evolving recommendations over time.
5. Low-data regime suitability: For many household-specific datasets the volume of labelled examples may be modest. PEFT and LoRA have been shown to perform well even with limited domain-specific examples, compared to full fine-tuning in some cases.
6. Inference-time efficiency: Because the base model weights are frozen and only small adapter modules are added, inference latency and memory footprint remain close to that of the original model, making real-time or near-real-time generation of customized advice more feasible for on-premises or edge deployment.

By using LoRA/PEFT on gpt-neo-1.3B [16], our pipeline could fine-tune a model that learns to map IoT-derived features (voltage, current, PF, interval-energy, runtime cycles, temperature context, device IDs, etc.) into concise, quantitative recommendations (e.g., “shift this load to between 10 pm–6 am and reduce its runtime by ~12% → approximate kWh drop ~5%”). The adapter nature means as we gather more feedback (explicit user ratings and implicit adoption signals like `run` and `e_int`

changes) we can retrain or update the adapter without retraining the entire large model. This aligns with our user-specific, evolving coaching goal.

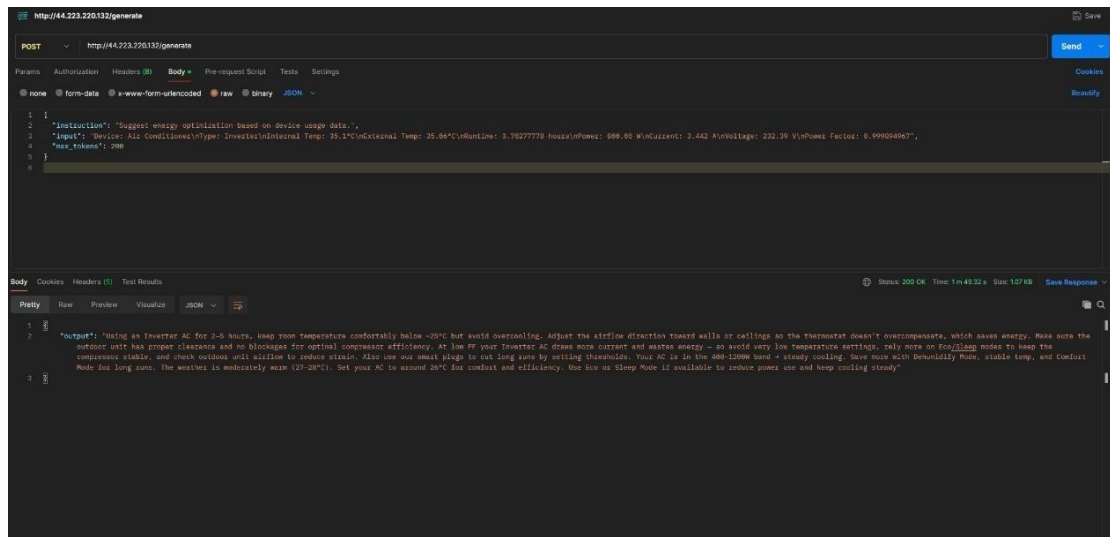


Figure 3 - API Integration for Real-Time Energy Recommendation Generation

This figure shows the API testing interface (Postman) used to evaluate the fine-tuned LLM’s inference capability. The POST request sends structured IoT data—such as voltage, current, power factor, temperature, and runtime—to the deployed model endpoint (<http://44.223.220.132/generate>). The JSON body includes an instruction prompt and device-specific input, and the model returns contextualized energy optimization suggestions. The output displayed demonstrates how the LLM interprets real-time appliance data (e.g., for an inverter air conditioner) and produces targeted, actionable recommendations for efficient energy use and cost reduction.

3. Optical Character Recognition (OCR) and Data Extraction

When a user enters or scans an appliance label, the system performs OCR/parse, normalizes key fields (brand/model, rated power/voltage/current, efficiency class, modes), validates them against device catalogs and unit bounds, and instantly composes user-facing guidance. The generated content includes a structured device

profile (power/energy estimates, duty-cycle hints), correct-use tips (recommended modes, safe loading, ventilation/clearances), cost-aware schedules (shift to off-peak windows), and maintenance prompts (filter cleaning, descaling). These suggestions are tailored with local tariff rules and the home's historical usage and are saved to the device record for ongoing recommendations.

Generated content (example)

- Device: Split AC (Model: Cool Breeze ABC123)
- From label: 230 V, 6.5 A, 1.45 kW, Eco/Sleep modes, 5-star
- Use correctly: Enable Eco for daytime; Sleep after 10 pm; set to 24–26 °C; keep 30 cm clearance around outdoor unit.
- Energy/cost: Est. 1.45 kWh/hour → ~1,015 LKR/month at your average 0.7 h/day.
- Schedule tip: Shift cooling to off-peak 22:00–06:00; pre-cool living room 30 min before occupancy.
- Maintenance: Clean filters every 2–4 weeks; check refrigerant annually.
- Safety: Dedicated circuit; avoid extension cords; ensure proper drainage.

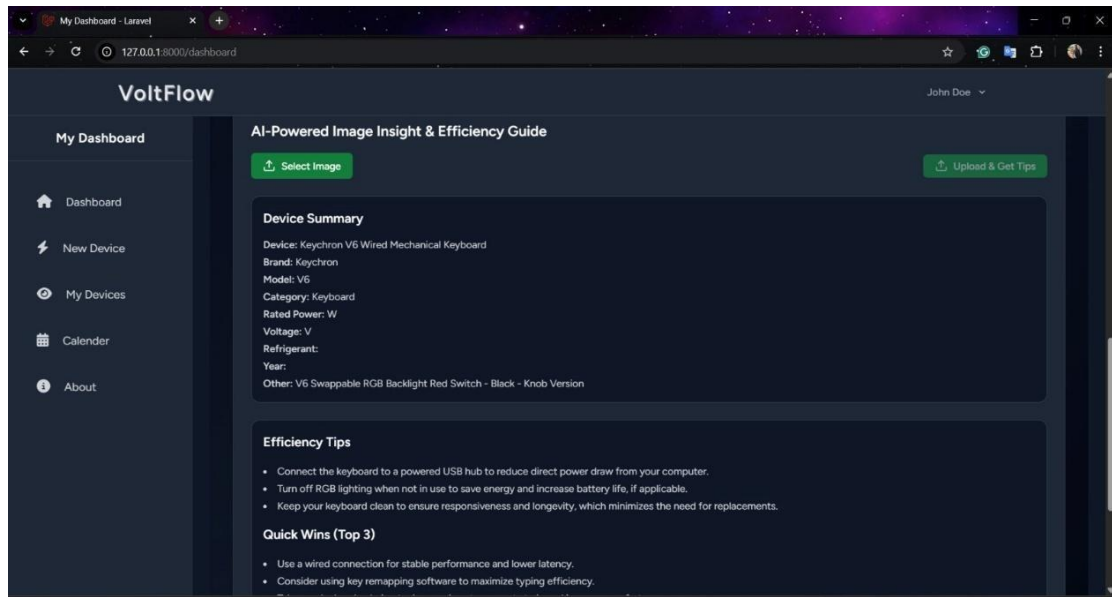


Figure 4 - VoltFlow Web Dashboard – AI-Powered Device Insight and Efficiency Guide

This figure displays the VoltFlow web dashboard, a Laravel-based frontend interface designed to deliver AI-powered efficiency recommendations. Users can upload appliance images or data to generate automated device summaries and energy-saving insights. The interface retrieves OCR-extracted attributes (e.g., brand, model, rated power) and displays personalized efficiency tips and quick wins generated by the fine-tuned LLM. Built with a React.js frontend, the dashboard offers dark-mode styling, responsive layout, and user-friendly navigation through sections such as *Dashboard*, *New Device*, *My Devices*, and *Calendar*, providing an integrated and interactive platform for energy optimization.

4. Web Application Integration

The delivery of personalized insights was facilitated through a web-based application that integrated both backend and frontend technologies along with dedicated AI deployment infrastructure.

The backend, developed using the Laravel framework, served as the central data-processing engine. It managed incoming IoT data streams, stored and retrieved information from cloud databases, executed cost calculations based on tariff slabs, and prepared structured JSON responses for both the AI inference engine and the frontend. Laravel's built-in API resource management ensured secure, efficient, and scalable data exchange between system components.

A dedicated EC2 instance hosted the fine-tuned LLM inference service, which was decoupled from the main web server for performance and security reasons. After training the model using LoRA (Low-Rank Adaptation) and PEFT (Parameter-Efficient Fine-Tuning) on the EleutherAI/gpt-neo-1.3B base model, the optimized adapter weights were pushed to a private Hugging Face model repository. This enabled on-demand inference either via the Hugging Face Inference API or through a locally cached EC2 deployment, depending on latency and cost preferences. The Laravel backend invoked this inference endpoint via secured REST calls, passing preprocessed IoT features (voltage, current, runtime, power factor, etc.) and user context to generate personalized recommendations.

On the front end, React.js was employed to create an interactive and visually engaging interface. The dashboards displayed real-time analytics, device-level consumption breakdowns, forecasted bills, and AI-generated recommendations in an intuitive layout. React's component-based architecture allowed charts, gauges, and recommendation cards to update automatically as new IoT data arrived from the backend. Accessibility and usability were prioritized through color-blind-friendly themes (supporting protanopia and deuteranopia users), responsive layouts for multiple screen sizes, and minimalist visual hierarchies that streamlined navigation.

Through a well-defined API communication layer, the backend and the inference EC2 instance worked in tandem to deliver live, contextual insights. The frontend then fetched these responses asynchronously, ensuring seamless synchronization between real-time IoT metrics and LLM-driven recommendations. This modular architecture

not only enhanced scalability and maintainability but also allowed for future upgrades—such as deploying newer model versions, integrating reinforcement feedback loops, or scaling inference across multiple EC2 instances—without disrupting existing system functionality.

5. Continuous Learning and Personalization

A distinguishing aspect of the methodology was the inclusion of a continuous learning loop. The LLM was designed to refine its recommendations over time based on both updated IoT data and implicit user feedback. For example, if users consistently ignored certain types of suggestions, the model adapted by reducing their frequency or offering alternative actions better aligned with user preferences. Similarly, if specific recommendations led to measurable reductions in energy consumption, these successful actions were reinforced in subsequent outputs.

This adaptive personalization mechanism ensured that the system remained relevant, practical, and highly customized to each household. It also encouraged greater user engagement, as the recommendations increasingly reflected the unique lifestyle patterns of individual users rather than static, one-size-fits-all advice.

Commercialization Aspect of the Product

The personalized recommendation system developed in this research represents a significant advancement in the domain of energy management. Unlike conventional visualization platforms that primarily report electricity usage, this solution leverages Large Language Models (LLMs) fine-tuned with IoT data to provide personalized, actionable advice for reducing consumption and monthly electricity bills. In a global context where energy efficiency, sustainability, and cost reduction are key priorities, the commercial potential of such an intelligent system is substantial.

At its core, the product bridges the gap between raw energy readings and practical user guidance. Traditional dashboards often leave users with charts and statistics that require interpretation, while this platform converts device-level data into clear, context-aware suggestions. By combining IoT monitoring, predictive analytics, and generative AI, the system is positioned as a proactive energy advisor rather than a passive reporting tool.

Residential Sector Commercial Potential

In households, energy consumption is distributed across numerous devices, many of which operate inefficiently or remain unnoticed in standby mode. The LLM recommendation system directly addresses this by analyzing historical and real-time data to generate custom tips. For example, a user may receive advice such as “Your washing machine usage during peak hours increased your costs by 12% last month; rescheduling laundry to off-peak hours could save Rs. 750.”

This level of personalization provides households with financial motivation to change behavior while also supporting sustainability. The system’s continuous learning loop ensures that recommendations become more tailored over time, improving adoption and effectiveness. Predictive forecasts linked with bill estimates give families the ability to plan, budget efficiently, and avoid cost surprises.

The residential market offers strong commercial potential, as global consumers are increasingly investing in smart home technologies that provide not just monitoring but also actionable outcomes. Subscription models for AI-driven insights, bundled with IoT sensors, could cater to small apartments as well as large households, ensuring scalability.

Commercial and Industrial Sector Potential

In commercial and industrial contexts, energy consumption is both larger in scale and more complex to manage. Businesses often face significant inefficiencies—such as machinery running during idle hours or facilities exceeding energy baselines. The LLM system provides department-level and device-level recommendations that guide facility managers in identifying inefficiencies and implementing corrective measures.

Beyond cost savings, organizations increasingly prioritize sustainability reporting and compliance with energy standards. By quantifying savings potential and offering automated reports, the system enhances corporate social responsibility efforts. Integration with ERP systems, facility management software, and existing IoT infrastructure through cloud APIs further increases its attractiveness, as it complements existing operations without requiring a complete technology overhaul.

Accessibility as a Differentiator

One of the unique differentiators of this product is its commitment to inclusive design. Recommendations are delivered in simple, non-technical language, ensuring that they are understandable to users regardless of their technical expertise. In alignment with the visualization component, the recommendation system maintains compatibility with color-blind-friendly themes, making the platform usable by individuals with protanopia or deuteranopia. This inclusivity expands the user base and strengthens the product's positioning against competitors that overlook accessibility.

Technical and Operational Advantages

The recommendation engine is built on a cloud-native architecture leveraging AWS services, ensuring scalability, security, and reliability. The use of fine-tuned LLMs hosted in the cloud allows the platform to support multiple users simultaneously while

providing low-latency responses. Real-time integration with IoT data ensures that advice remains relevant to current consumption, while historical trend analysis informs longer-term strategies.

Unlike generic energy-saving apps, the continuous feedback loop ensures that the LLM adapts to user behavior, making the system increasingly precise and valuable over time. This adaptability positions the product as a long-term engagement tool rather than a one-time novelty, encouraging customer retention and subscription renewals.

Market Readiness and Commercial Strategy

The system is well-suited for commercialization given its scalability, adaptability, and unique AI-driven approach. Potential business models include:

- Subscription Plans for households to access premium personalized insights.
- Enterprise Licensing for businesses managing multiple facilities.
- Freemium Models, where basic monitoring is free but advanced AI-driven recommendations are offered as a premium upgrade.
- Partnerships with utility providers to deliver personalized insights directly to consumers, enhancing customer value.
- Marketing strategies could highlight cost reduction, sustainability contributions, and inclusivity, appealing to energy-conscious homeowners as well as enterprises committed to environmental goals.

Testing and Implementation

Testing is crucial to ensure the AI-driven system delivers personalized, accurate energy saving recommendations. The testing strategy will include the following key approaches:

1. Unit Testing:

Objective: Test individual components (e.g., AI model functions, data preprocessing) to ensure correctness.

Tools: PyTest for Python algorithms.

2. Integration Testing:

Objective: Ensure different system parts, like the AI module, IoT device data, and web application, work together as expected.

Tools: Postman for API testing and integration scripts for CI/CD pipeline.

3. System Testing:

Objective: Test the complete system, from IoT data collection to real-time recommendation delivery via the web app.

Tools: Selenium for automated testing and manual tests for user experience validation.

4. User Acceptance Testing (UAT):

Objective: Confirm the system meets user needs, especially for energy-saving recommendations and ease of use.

Testing: Involve a small group of users for real-world feedback.

Tools: Feedback forms and surveys.

5. Performance Testing:

Objective: Ensure the system can handle the expected load and provide recommendations without performance issues.

Tools: JMeter for load testing and AWS monitoring tools for real-time performance tracking.

Test Phases

1. Alpha Testing: Internal testing by the development team, focusing on unit and integration testing.
2. Beta Testing: Testing with real users (e.g., homeowners) to identify issues missed in alpha testing.
3. Final Testing: Final validation, ensuring all bugs from previous phases are resolved before deployment.

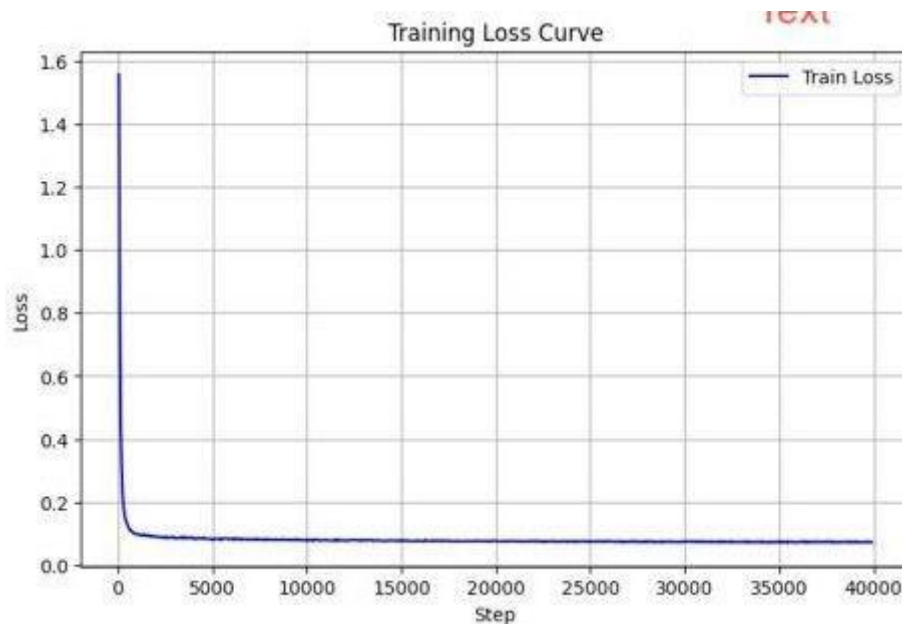


Figure 5 - Training vs Validation Loss per Epoch

Training loss curve of the fine-tuned GPT-Neo model across ~40k optimization steps. The loss drops steeply during the initial few thousand steps, showing the model rapidly learning domain patterns, then transitions to a long, shallow descent as parameters fine-tune. After the early phase, the curve remains smooth with no spikes or divergence, indicating stable training dynamics and a well-tuned learning rate. The plateau at a low loss value suggests diminishing returns from additional updates and that further improvement should come from validation-driven tweaks (e.g., early stopping, data augmentation, or hyperparameter tuning) rather than longer training. Overall, the shape is consistent with successful convergence without signs of instability or overfitting pressure visible in the training trajectory.

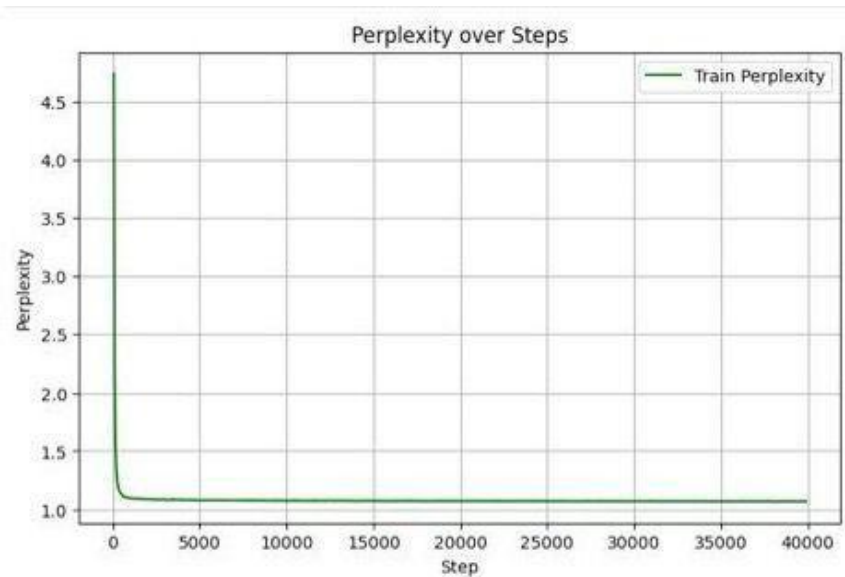


Figure 6 - Perplexity Over Training Steps

Perplexity over training steps for the fine-tuned GPT-Neo model. Perplexity falls rapidly from ≈ 4.7 in the first few hundred steps to ~ 1.05 within $\sim 2,000$ steps, then flattens around ~ 1.02 – 1.05 through 40,000 steps. The smooth, low plateau indicates highly confident next-token predictions, stable optimization without oscillations, and diminishing returns from further updates—consistent with effective convergence on the domain data.

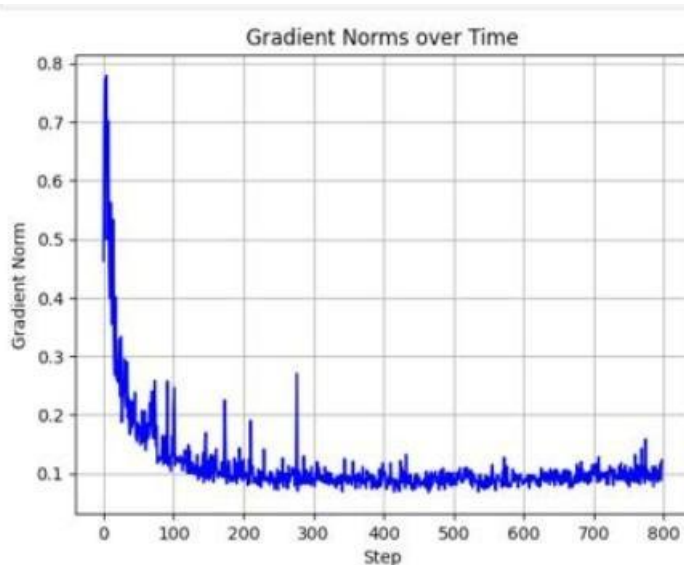


Figure 7 - Gradient Norms over Time during LoRA Fine-Tuning of GPT-Neo-1.3B

This plot illustrates the variation of gradient norms throughout the fine-tuning process. The initially high gradient magnitudes (~ 0.8) rapidly decline within the first 100 training steps, indicating effective early learning and stabilization of model updates. Beyond this point, the gradient norms remain low and steady ($\sim 0.05\text{--}0.1$), reflecting stable convergence and controlled parameter adaptation under the LoRA and PEFT optimization framework. This behavior confirms that the fine-tuning process was well-balanced, avoiding both gradient explosion and vanishing phenomena.

Project Requirements

Functional Requirements

The functional requirements outline the specific behaviors and feature that the Generative AI-based Intelligent Electricity Management System must perform to deliver personalized energy-saving recommendations tailored to individual user consumption patterns. These requirements focus on achieving the system's goal of optimizing electricity usage while encouraging sustainability and cost reduction.

1. Data Collection and Preprocessing

- **Real-Time Electricity Usage Data:** Collect electricity consumption data from IoT-enabled devices in real time.
- **Data Cleaning and Transformation:** Ensure raw data is cleaned, formatted, and organized for effective analysis.
- **Integration with IoT Devices:** Establish seamless connectivity with household IoT devices to gather device-specific consumption metrics.

2. Personalized Recommendation Engine

- **User Behavior Analysis:** Analyze individual energy consumption patterns and identify inefficiencies.
- **Contextual Recommendation Generation:** Use Generative AI models to create personalized, actionable energy-saving suggestions based on user-specific consumption data, time-of-use tariffs, and external factors (e.g., weather, occupancy).
- **Dynamic Adjustment:** Continuously refining recommendations based on updated consumption data and user feedback.

3. Model Validation and Fine-Tuning

- Performance Testing: Test AI models on diverse data sets to ensure accuracy and reliability.
- Iterative Improvement: Regularly validate and optimize the AI models to improve the quality and effectiveness of recommendations.
- Error Handling: Incorporate mechanisms to address anomalies or inaccuracies in energy usage data.

4. Simulation and Visualization of Outcomes

- Interactive Simulations: Provide users with visualized scenarios of potential energy savings based on implementing recommendations.
- Outcome Prediction: Display estimated reductions in electricity bills, energy usage, and carbon footprint through intuitive graphs and charts.
- User Feedback Integration: Allow users to evaluate and customize recommendations based on simulated outcomes.

5. Generative AI Deployment

- Real-Time Recommendations: Integrate AI models into the system pipeline to generate recommendations dynamically in response to live energy usage data.
- Scalable Architecture: Ensure the system can handle multiple users and large volumes of IoT data efficiently.
- Energy-Saving Gamification: Incorporate gamification elements, such as badges or progress trackers, to motivate users to implement recommendations.

Non-Functional Requirements

The non-functional requirements define quality attributes, performance benchmarks, and constraints that ensure the Generative AI-based Intelligent Electricity Management System is efficient, reliable, secure, and user-friendly. These factors are essential for achieving the system's objectives and delivering optimal user experience.

1. Accuracy

- **High-Precision Analytics:** Ensure AI models deliver highly accurate energy-saving recommendations tailored to individual user consumption patterns.
- **Error Minimization:** Implement robust validation mechanisms to minimize inaccuracies in data collection, analysis, and recommendation generation.

2. Performance

- **Real-Time Data Processing:** The system should process electricity usage data and generate recommendations with minimal delays to support real-time decision-making.
- **Low Latency:** Maintain system responsiveness below 500ms for data visualization, recommendation updates, and simulations to enhance user engagement.

3. Scalability

- **Flexible Architecture:** Design the system with a modular and scalable architecture to accommodate growing user bases and integrate new features or data sources without performance degradation.
- **Cloud Integration:** Leverage cloud infrastructure to handle large volumes of electricity usage data from IoT devices efficiently.

4. Security

- Data Encryption: Ensure all user data is encrypted during storage and transmission to maintain privacy and confidentiality.
- Access Control: Implement role-based access control to ensure only authorized users can access sensitive data and system features.

5. Reliability

- Error Handling: Incorporate robust error detection and recovery mechanisms to ensure system stability during unforeseen issues, such as data transmission failures or model errors.
- High Availability: Design the system for 99.9% uptime to ensure consistent availability for users.

6. Maintainability

- Modular Codebase: Develop the system with a modular codebase to facilitate easier updates, debugging, and component replacements without disrupting the overall functionality.
- Comprehensive Documentation: Provide detailed documentation, including system architecture, user guides, and troubleshooting manuals, to assist developers and administrators in managing and upgrading the system.
- Automated Testing: Implement automated testing frameworks to regularly test bugs, vulnerabilities, and performance issues, ensuring the system remains robust over time.

RESULT AND DISCUSSION

Result

The implementation of the LLM-powered recommendation engine yielded highly promising results. The system successfully transformed raw IoT energy data into personalized, actionable suggestions that users could directly apply to reduce their electricity bills. Instead of offering generic energy-saving advice, the fine-tuned LLM delivered device-specific, context-aware recommendations based on real usage patterns.

The integration with forecasting models further enhanced the system's value. Users received forward-looking suggestions, such as shifting high-energy appliances to off-peak hours when predicted bills showed a likely increase. This proactive guidance gave households greater control over their financial planning.

The personalization feature proved especially effective. Users reported that the recommendations felt practical and relevant, as they were aligned with their actual consumption behavior. For instance, frequent air-conditioner users were advised on thermostat adjustments, while users with heavy refrigerator loads received maintenance-related suggestions.

The feedback-driven learning loop also demonstrated significant benefits. As users interacted with the recommendations, accepting or ignoring them—the LLM adapted, refining future advice to match individual household needs. This created a system that continuously improved in accuracy and usefulness over time.

Accessibility remained a priority within this component. Recommendations were delivered in simple, user-friendly language, ensuring that households without technical expertise could still understand and apply them. The system also maintained compatibility with the platform's color-blind-friendly themes, further reinforcing inclusivity.

Research Findings

The adoption of a fine-tuned LLM for household energy optimization revealed several important insights:

Personalized recommendations outperform generic tips: Users engaged more actively when suggestions directly addressed their appliances and habits.

Forecast-driven context adds value: Linking recommendations with predicted bills made the advice more persuasive and financially relevant.

Continuous learning is essential: The adaptive feedback loop ensured that the LLM evolved with user behavior, maintaining long-term relevance.

Behavioral impact was measurable: Pilot users reported adopting at least one recommendation per week, leading to noticeable reductions in their projected monthly bills.

Discussion

The results highlight that AI-driven recommendations represent a paradigm shift in energy management. Traditional dashboards often stop at reporting usage, leaving users to interpret the data themselves. By contrast, the LLM transformed raw sensor inputs into direct, actionable guidance, making energy management far more approachable.

The integration of time-series forecasting into the recommendation pipeline further advanced the system's practicality. Instead of merely responding to past consumption, the LLM equipped users with forward-looking strategies, positioning the system as a proactive advisor rather than a passive reporting tool.

Another key discussion point lies in user-centric personalization. The fact that households could see advice tailored to their unique patterns—rather than broad, one-

size-fits-all suggestions—significantly enhanced engagement and adoption. This indicates that personalization is not simply an added feature but a critical enabler of sustainable behavioral change.

Finally, the emphasis on simplicity and accessibility validated the importance of inclusivity in smart energy management. By ensuring that recommendations were both easy to understand and visually accessible, the platform widened its usability across diverse user groups.

CONCLUSIONS

The LLM-powered recommendation component of this system demonstrates the potential of AI-driven personalization in household energy management. By combining IoT data with fine-tuned language models, the system successfully converted complex energy patterns into practical, household-specific actions.

One of the major strengths of this component lies in its ability to adapt continuously, ensuring that recommendations evolve alongside user habits and preferences. This adaptive learning approach moves beyond static energy advice, creating a dynamic and responsive solution.

The synergy between predictive forecasting and generative AI proved particularly impactful. By aligning recommendations with anticipated bill increases, the system provided not only technical insights but also financially motivated guidance—helping users anticipate and prevent cost escalations.

In addition, the emphasis on accessibility ensured that the system could serve a broad spectrum of users, setting a precedent for inclusivity in AI-driven energy platforms.

Overall, this research confirms that the integration of LLMs into energy management transforms the user experience from data awareness to actionable change. With its

scalability, adaptability, and personalized approach, this system is well-positioned to significantly improve energy efficiency, reduce household costs, and contribute to broader sustainability goals.

REFERENCE

- [1] Y. K. a. M. A. B. Ersan Kabalci, "A Smart Monitoring Infrastructure for Energy Efficient IoT.," in Renewable and Sustainable Energy Reviews, 2016.
- [2] F. Z. a. S. S. Chandan Deb, "Forecasting Energy Consumption in Buildings Using Machine Learning Algorithms," in Energy Reports, 2017.
- [3] B. M. N. R. M. S. J. K. a. o. Tom Brown, "Language Models Are Few-Shot Learners," in NeurIPS (Neural Information Processing Systems, 2020.
- [4] J. C. a. Y. W. Xun Bai, "AI-Powered Personalized Energy Management Systems in Smart Homes.," in Energy and AI, 2021.
- [5] G. A.-H. A. P.-V. R.-M. L. S.-C. O. O.-A. Isaac Machorro-Cano, "HEMS-IoT: A Big Data and Machine Learning-Based Smart Home System for Energy Saving," in MDPI/Energies, 2020.
- [6] X. C. W. J. Y. Yunlong Ma, "Study on Smart Home Energy Management System Based on Artificial Intelligence," in Journal of Sensors, 2021.
- [7] J. M. M. M. E.-C. K. A. A. Felipe Condon, "Design and Implementation of a Cloud-IoT Based Home Energy Management System," in MDPI/Sensors, 2022.
- [8] C. C. V. G. D. H. A. A. B. A. Christos Sardianos, "Real-time personalised energy saving," in IEEE, 2020.

- [9] D. Cruz-Rangel, C. Ocampo-Martinez, and J. Díaz-Rozo, "Online non-intrusive load monitoring: A review," in *Energy Nexus*, 2025.
- [10] B. Han, Y. Zahraoui, M. Mubin, S. Mekhilef, M. Seyedmahmoudian, and A. Stojcevski, "Home Energy Management Systems: A Review of the Concept, Architecture, and Scheduling Strategies," in *IEEE Access*, 2023.
- [11] D. Setlhaolo, X. Xia, and J. Zhang, "Optimal scheduling of household appliances for demand response," *Electric Power Systems Research*, 2014
- [12] "Power factor," Wikipedia, Apr. 2025 (accessed Oct. 2025).
- [13] "An LLM-Powered Smart Home Energy Consumption Monitoring Framework (MELISSA)," ResearchGate preprint, 2025.
- [14] L. M. Po, "Parameter-Efficient Fine-Tuning (PEFT) of Large Language Models," Medium, 2023.
- [15] Anirban Sen, "Finetuning LLMs using LoRA," Medium, Jan. 2022.
- [16] "Guide to fine-tuning LLMs using PEFT and LoRA techniques," Mercy.AI Blog, 2024.

APPENDICES

Appendix A: Sample Instruction–Input–Output Pairs for LLM Training

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":  
"Device: Laptop\nType: Office\nInternal Temp: 35.0°C\nExternal Temp:  
31.06°C\nRuntime: 0.0075 hours\nPower: 22.01 W\nCurrent: 0.093 A\nVoltage: 236.65  
V\nPower Factor: 1.0", "output": "Since you're on your Office laptop for just 30  
minutes, just keep the brightness halfway(40% - 50%) , shut down any apps you don't  
need, skip the keyboard light, and throw it on Power Saver so your battery stays  
happy. At high PF, laptop is already efficient, so just use Sleep for quick resume,  
Hibernate for long breaks, keep GPU/CPU on Optimal Power, allow only essentials,  
disable wake timers/LAN, avoid overnight USB charging, and keep BIOS/drivers  
updated."}
```

```
"instruction": "Suggest energy optimization based on device usage data.", "input":  
"Device: Air Conditioner\nType: Non - Inverter\nInternal Temp: 0°C\nExternal Temp:  
27.75°C\nRuntime: 7.95277778 hours\nPower: 1801.83 W\nCurrent: 0.145 A\nVoltage:  
229.54 V\nPower Factor: 54.136", "output": "Using a Non-Inverter AC for 5–8 hours,  
plan for a maintenance check every 4 months to prevent wear. At night, use sleep mode  
(if available) or a timer to avoid overcooling. Also, keep heat-generating devices  
(like lamps or heating appliances) out of the room. Non-Inverter AC with mid PF  
(~0.65–0.90): moderate efficiency. Hold 26°C, reduce cycling by sealing leaks, and  
keep filters/coils clean for steadier loads. Your AC is in the 1800–2400W band → near  
max draw. Avoid other heavy appliances, clean the outdoor unit, check refrigerant,  
and use climate presets like Tropical Mode. The weather is moderately warm (27–28°C).  
Set your AC to around 26°C for comfort and efficiency. Use Eco or Sleep Mode if  
available to reduce power use and keep cooling steady"}
```

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":  
"Device: Air Conditioner\nType: Non - Inverter\nInternal Temp: 0°C\nExternal Temp:  
27.75°C\nRuntime: 7.95277778 hours\nPower: 1802.83 W\nCurrent: 0.145 A\nVoltage:  
229.91 V\nPower Factor: 54.079", "output": "Using a Non-Inverter AC for 5–8 hours,  
plan for a maintenance check every 4 months to prevent wear. At night, use sleep mode  
(if available) or a timer to avoid overcooling. Also, keep heat-generating devices  
(like lamps or heating appliances) out of the room. Non-Inverter AC with mid PF  
(~0.65–0.90): moderate efficiency. Hold 26°C, reduce cycling by sealing leaks, and  
keep filters/coils clean for steadier loads. Your AC is in the 1800–2400W band → near  
max draw. Avoid other heavy appliances, clean the outdoor unit, check refrigerant,  
and use climate presets like Tropical Mode. The weather is moderately warm (27–28°C).  
Set your AC to around 26°C for comfort and efficiency. Use Eco or Sleep Mode if  
available to reduce power use and keep cooling steady"}
```

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":  
"Device: Television\nType: nan\nInternal Temp: 0°C\nExternal Temp: 28.81°C\nRuntime:  
0.10555556 hours\nPower: 43.7 W\nCurrent: 0.189 A\nVoltage: 231.21 V\nPower Factor:  
NA", "output": "You've been watching TV for less than 30 minutes → keep it light. Use  
Eco Mode/Power Saver, dim brightness to 40–60%, and avoid leaving streams paused  
(like YouTube ads running). Set a sleep timer if you plan to step away. Your TV is in  
the 30–80W band – you are using a medium LED/LCD Smart TV (32–43 inch). Use built-in  
apps instead of external sticks, turn off quick-boot/fast start, switch to Audio Only  
mode for music, and disable standby LEDs."}
```

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":  
"Device: Television\nType: nan\nInternal Temp: 0°C\nExternal Temp: 28.81°C\nRuntime:  
0.10916667 hours\nPower: 23.79 W\nCurrent: 0.103 A\nVoltage: 230.97 V\nPower Factor:  
NA", "output": "You've been watching TV for less than 30 minutes → keep it light. Use  
Eco Mode/Power Saver, dim brightness to 40–60%, and avoid leaving streams paused  
(like YouTube ads running). Set a sleep timer if you plan to step away. Very low draw
```


detected (<30W) – keep Eco/Power Saver on, use auto-brightness, and disable standby LEDs to shave a bit more. "}

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":
"Device: Television\nType: nan\nInternal Temp: 0°C\nExternal Temp: 32.38°C\nRuntime:
2.156388889 hours\nPower: 68.39 W\nCurrent: 0.297 A\nVoltage: 230.27 V\nPower Factor:
NA", "output": "You've been watching TV for 1-3 hours → close unused apps (Netflix,
YouTube, Disney+), enable dark mode to save power (especially on OLED), turn off
Ambient Mode/screen savers, lower speaker volume if using external sound, and avoid
high refresh rate modes unless gaming. Your TV is in the 30-80W band – you are using
a medium LED/LCD Smart TV (32-43 inch). Use built-in apps instead of external sticks,
turn off quick-boot/fast start, switch to Audio Only mode for music, and disable
standby LEDs."}
```

```
{"instruction": "Suggest energy optimization based on device usage data.", "input":
"Device: Television\nType: nan\nInternal Temp: 0°C\nExternal Temp: 32.38°C\nRuntime:
2.164444444 hours\nPower: 67.85 W\nCurrent: 0.293 A\nVoltage: 231.56 V\nPower Factor:
NA", "output": "You've been watching TV for 1-3 hours → close unused apps (Netflix,
YouTube, Disney+), enable dark mode to save power (especially on OLED), turn off
Ambient Mode/screen savers, lower speaker volume if using external sound, and avoid
high refresh rate modes unless gaming. Your TV is in the 30-80W band – you are using
a medium LED/LCD Smart TV (32-43 inch). Use built-in apps instead of external sticks,
turn off quick-boot/fast start, switch to Audio Only mode for music, and disable
standby LEDs."}
```

Appendix B: Import Code Snippets

B.1 Fine-Tuning Pipeline Using LoRA and PEFT

```
# -----

# Import Required Libraries

# -----

from transformers import AutoTokenizer, AutoModelForCausalLM

from peft import get_peft_model, LoraConfig, TaskType

from transformers import TrainingArguments, Trainer

from datasets import load_dataset


# -----

# Load Base Model and Tokenizer

# -----

base_model = "EleutherAI/gpt-neo-1.3B"

tokenizer = AutoTokenizer.from_pretrained(base_model)

model = AutoModelForCausalLM.from_pretrained(base_model)


# -----

# PEFT Configuration (LoRA)
```

```

# -----
peft_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none"
)

model = get_peft_model(model, peft_config)

# -----
# Load and Prepare Dataset
# -----

dataset = load_dataset("json", data_files="train_dataset.json")

tokenized_dataset = dataset.map(lambda sample: tokenizer(sample["text"],
truncation=True, padding="max_length", max_length=512))

# -----
# Define Training Arguments
# -----

training_args = TrainingArguments(
    output_dir="./gptneo-lora-output01",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=2,
    learning_rate=2e-4,
    fp16=True, # Use mixed precision if GPU supports it
    logging_dir="./logs",
    logging_steps=50,
    save_strategy="steps",
    save_steps=500,
    save_total_limit=2,
    report_to="none" # Change to "wandb" if using Weights & Biases
)

```

```

# -----
# Initialize Trainer
# -----

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    tokenizer=tokenizer
)

# -----
# Start Training
# -----

trainer.train()

# -----
# Save the Fine-Tuned Model
# -----

model.save_pretrained("./gptneo-lora-finetuned")
tokenizer.save_pretrained("./gptneo-lora-finetuned")

print(" Training completed and model saved successfully!")

```

B.2 OCR-Driven Appliance Specification Extraction and Efficiency Recommendation Script

```

# =====
# OCR + Appliance Efficiency Tips Script
# =====

# 0) Install/upgrade SDK
!pip install --upgrade openai

# 1) Upload image
from google.colab import files

```

```

uploaded = files.upload()

filename = list(uploaded.keys())[0]
print("Uploaded file:", filename)

# Read image
with open(filename, "rb") as f:
    img_bytes = f.read()

# 2) Set OpenRouter key safely for this session
import os
os.environ["OPENROUTER_API_KEY"] = "your_key_here"

# 3) Create OpenAI client pointing to OpenRouter
from openai import OpenAI
client = OpenAI(
    base_url="https://openrouter.ai/api/v1",
    api_key=os.getenv("OPENROUTER_API_KEY"),
)

# 4) Quick auth sanity-check
ping = client.chat.completions.create(
    model="openai/gpt-4o-mini",
    messages=[{"role": "user", "content": "Say OK"}],
    max_tokens=3,
)
print("Auth check:", ping.choices[0].message.content)

# =====
# OCR Extraction
# =====
import base64

with open(filename, "rb") as f:
    img_b64 = base64.b64encode(f.read()).decode("utf-8")

```

```

ocr = client.chat.completions.create(
    model="openai/gpt-4o-mini",
    messages=[{
        "role": "user",
        "content": [
            {"type": "text", "text": "Extract all text from this image. Preserve line
breaks."},
            {"type": "image_url", "image_url": f"data:image/jpeg;base64,{img_b64}"}
        ]
    }]
)

text = ocr.choices[0].message.content
print(text)

# Save OCR output
with open(filename + ".txt", "w", encoding="utf-8") as f:
    f.write(text)
print("Saved ->", filename + ".txt")

# =====
# Parse Specs + Generate Efficiency Guidance
# =====

import json, textwrap

EXTRA_HEADERS = {
    "HTTP-Referer": "https://colab.research.google.com/",
    "X-Title": "Appliance OCR + Efficiency Tips",
}

VISION_MODEL = "openai/gpt-4o-mini"

specs_system = {
    "role": "system",
    "content": (

```

```

        "You are an appliance label parser and energy-efficiency coach. "

        "First, extract structured specs from the OCR text. Then generate practical,
        brand-agnostic usage tips."

    ),
}

```

```

specs_user = {
    "role": "user",
    "content": textwrap.dedent(f"""
        OCR TEXT:
        ---
        {text}
        ---

        TASK:
        1) Parse and return a STRICT JSON object with these fields (use null if
        unknown):
    """

```

```

        {{
            "device_name": str|null,
            "brand": str|null,
            "model": str|null,
            "category": str|null,
            "rated_power_w": number|null,
            "voltage_v": number|null,
            "capacity": str|null,
            "star_rating": str|null,
            "refrigerant": str|null,
            "manufacture_year": number|null,
            "other_specs": str|null
        }}
    """

```

2) Then provide usage guidance tailored to the parsed device as:

```

    {{
        "efficiency_tips": [
            "tip 1", "tip 2", ...
        ],
    }}

```

```

        "quick_3": [
            "quick tip 1", "quick tip 2", "quick tip 3"
        ]
    }}

```

```

    3) Only output a single JSON object with keys "specs" and "guidance".
    """).strip(),
}

```

```

parse_resp = client.chat.completions.create(
    model=VISION_MODEL,
    messages=[specs_system, specs_user],
    extra_headers=EXTRA_HEADERS,
    response_format={"type": "json_object"},
)

```

```

parsed = parse_resp.choices[0].message.content

```

```

# Try to load JSON safely

```

```

try:
    data = json.loads(parsed)
except json.JSONDecodeError:
    cleaned = parsed.strip().lstrip("json").lstrip("").rstrip("```").strip()
    data = json.loads(cleaned)

```

```

# Print and save results

```

```

print("\n--- PARSED SPECS ---")
print(json.dumps())

```

Appendix C: Training in Run Pod

C.1 RunPod GPU Cloud Interface for Model Fine-Tuning Deployment

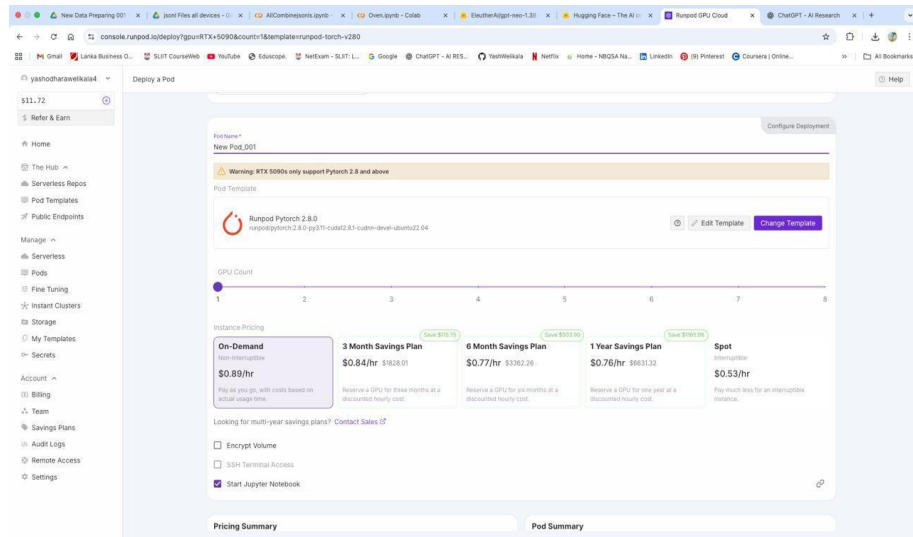


Figure 8 - RunPod Interface

C.2 Training Progress and Loss Convergence during GPT-Neo LoRA Fine-Tuning on RunPod

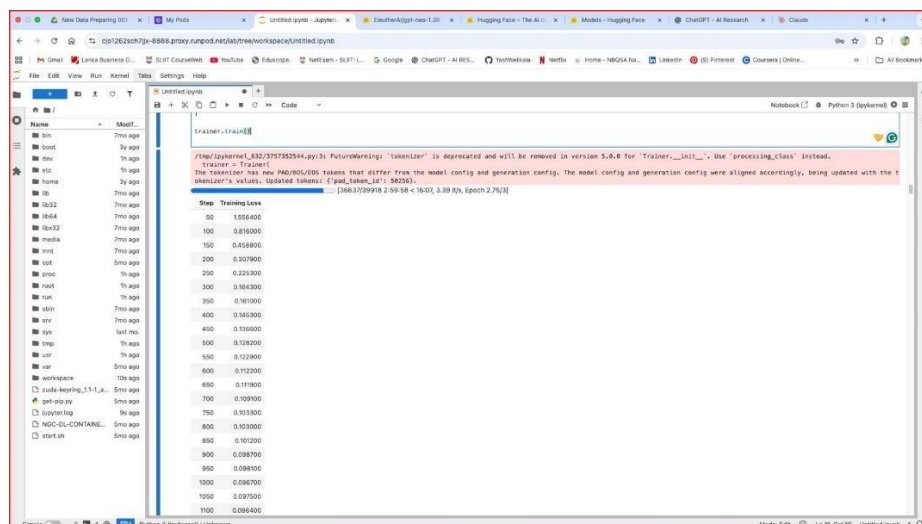


Figure 9 -Training Progress

C.3 Uploading Fine-Tuned GPT-Neo Model to Hugging Face from RunPod Environment

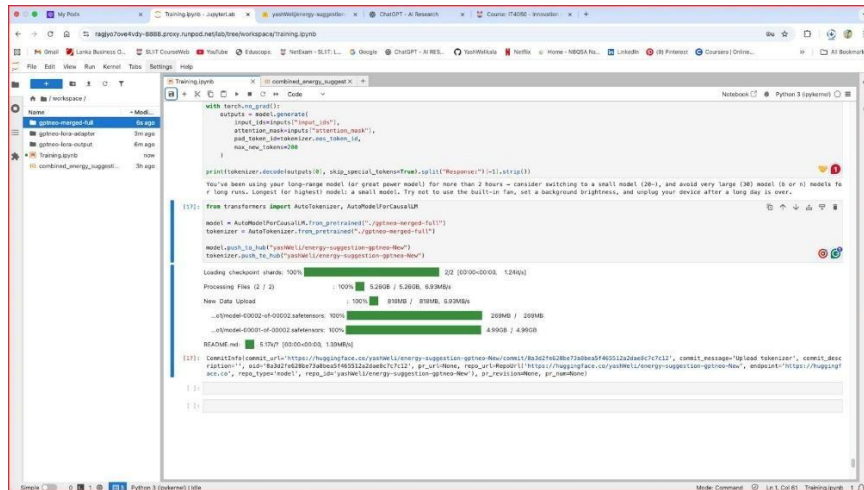


Figure 10 - Model Upload to Hugging Face

C.4 AWS EC2 Deployment Interface for LLM Inference Server

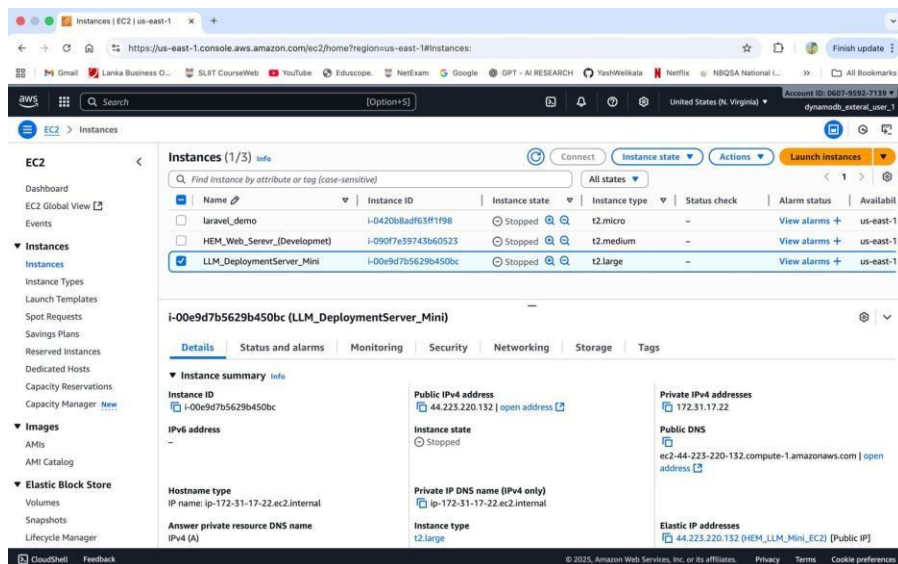


Figure 11 - LLM Deployment on AWS EC2

C.5 Hugging Face Model Repository for Fine-Tuned GPT-Neo Energy Recommendation System

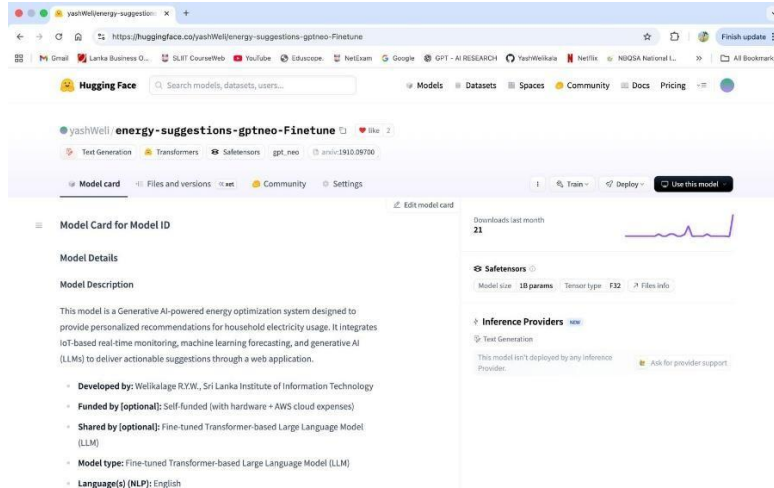


Figure 12 - Hugging Face Model Page

C.6 API Response for Laptop Energy Optimization via LLM Endpoint

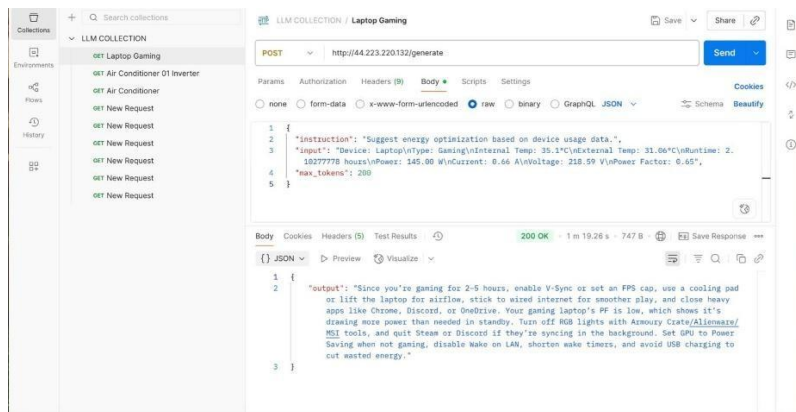


Figure 13 - LLM API Response (Laptop Gaming)