Cost-Effective On-Premises AI Solution Using CoT Distilled LLMs

Matthew Clark

Computer Science Department
The Masters University
Santa Clarita, CA., United States
mhckqw@gmail.com

March 6, 2025

Abstract

This paper introduces a cost-effective, on-premises Artificial Intelligence (AI) solution that leverages Chain of Thought (CoT) Distilled Large Language Models (LLMs) to address the legal, security, and cost-related challenges associated with vendor-provided AI services. The proposed system includes two practical implementations: an iOS application for mobile access and a Linux-based Python script with Text-to-Speech (TTS) functionality for auditory output. Both applications utilize Disneyland Parks help documentation as a domain-specific knowledge base, demonstrating the system's potential in customer service and information retrieval contexts. The implementation encompasses on-premises infrastructure setup, integration of a Retrieval Augmented Generation (RAG) system, and detailed development of the user interfaces. Evaluation methods are proposed to assess performance across multiple dimensions, and future research directions are outlined to enhance the system's capabilities.

Contents

1 Introduction

The field of Artificial Intelligence (AI) has cultivated remarkable advancements, transforming industries from automation, to decision-making, and personalized services. AI as a Service (AIaaS) platforms, such as those offered by Amazon Web Services (AWS), Google Cloud, and Microsoft Azure, have democratized access to advanced AI capabilities, allowing organizations to integrate machine learning models, natural language processing (NLP), and computer vision into their workflows without building these systems from scratch. Despite their accessibility and scalability, these cloud-based solutions are not a silver bullet solution and present some drawbacks, particularly for small to medium-sized setups and organizations handling sensitive data.

One major concern with AIaaS is data privacy and security. When organizations rely on third-party vendors, they must upload their data to external servers, relinquishing control of how that data is stored, processed, and protected. High-profile data breaches and increasing regulation such as the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA) in the United States have heightened awareness of these risks. For example, a 2022 report by IBM estimated that the average cost of a data breach was around $4.35 million [1], showcasing the financial and reputational risks. For certain applications, such as sensitive HR tools, the business' risk tolerance is too low. Additionally, industries like healthcare, finance, and government, which deal with confidential information, face strict compliance requirements that may prohibit the use of cloud-based AI due to potential legal liabilities.

Another challenge is cost. While AIaaS offers a pay-as-you-go model that appears economical, expenses can rack up quickly as usage scales. For example, processing thousands of queries per day or training custom models on vendor platforms can incur significant costs, making it unsustainable for small setups with limited budgets.

In response to these challenges, on-premises AI solutions are a compelling alternative. By deploying AI models locally or on the edge, organizations retain full ownership of their data and ensure compliance with legal standards. However, traditional on-prem deployments often demand powerful hardware (e.g., high-end GPUs or TPUs) and specialized technical support personnel, posing barriers to adoption.

This paper proposes a cost-effective, on-prem AI solution that overcomes these hurdles by utilizing Chain of Thought (CoT) Distilled Large Language Models (LLMs). CoT Distillation is a technique that transfers the reasoning abilities of large, resource-intensive LLMs to smaller, more efficient models, enabling high performance on modest hardware. My solution integrates this approach with a Retrieval Augmented Generation (RAG) system to enhance response accuracy using external knowledge sources, specifically Disneyland Parks help docs. I demonstrate its practicality through two applications: an iOS app for mobile users and a Python script with TTS. Both applications are aimed at providing information about the Parks.

The objectives of this project are threefold:

1. To develop an on-premises AI system that balances performance with resource efficiency.
2. To address legal and security concerns by keeping data local.
3. To example real-world applications in a customer-facing context.

The paper is organized as follows: Section 2 reviews related work, Section 3 details the implementation, Section 4 outlines the evaluation strategy, and Section 6 concludes with future directions.

2 Related Work

This section surveys existing research and technologies relevant to my solution, categorized into four areas: CoT Distillation, on-premises AI, RAG systems, and mobile/TTS applications.

2.1 Chain of Thought Distillation

Chain of Thought (CoT) Distillation enhances the efficiency of LLMs by training smaller models to mimic the step-by-step reasoning processes of their larger counterparts. According to Wei et al. [2], this method allows a distilled model with fewer parameters to outperform a larger model on reasoning tasks, using significantly less training data. For example, their experiments showed that a 1.5B-parameter model distilled with CoT outperformed a 6B-parameter baseline, making it ideal for resource-constrained environments like on-premises setups.

Similarly, Smith et al. [3] explore applying distilled models to physical agents, such as robots or edge devices. Their findings suggest that CoT-distilled LLMs can generalize reasoning across diverse tasks, a property I leverage for my Disneyland Parks application.

2.2 On-Premises AI Solutions

Deploying AI locally has been studied as a means to mitigate cloud-related risks. Johnson [4] highlights advantages like data sovereignty, reduced latency, and long-term cost savings. However, the paper notes challenges such as high initial hardware costs and the need for in-house expertise—issues my solution addresses by using distilled models that run on standard GPUs.

The technical report on DeepSeek-V3 [5] describes a distilled LLM optimized for on-premises use, achieving near state-of-the-art performance with a 32B-parameter model that fits within 24GB of GPU memory, aligning with my hardware choice.

2.3 Cost Concerns with Amazon Outposts

Amazon Outposts offer a managed on-premises cloud solution, but their subscription-based pricing, with the smallest configuration at about $1,500 per month for a 3-year term [6], can be a significant cost. This fixed cost structure may not suit innovative AI projects, especially those with unpredictable resource usage, as it lacks the flexibility to scale up and down easily.

Impact on Effective Innovation in AI

For effective innovation, especially in AI, internal projects often require flexibility to adapt to indeterminate resource requirements. The fixed monthly costs of Amazon Outposts, ranging from $1,499 to $14,999 [6], represent a significant expense that may not be sustainable for organizations with variable usage, such as startups or imagineering research teams. Over a 3-year term, the total cost for the smallest configuration would be $53,964, which could strain budgets and limit the ability to experiment and iterate, key aspects of innovation.

Moreover, the lack of flexibility in scaling resources is a concern. Amazon Outposts require a commitment to a specific configuration for the term, and scaling up or down may involve additional costs or complexities, as custom configurations require contacting AWS [6].

GPU Instance Availability and Costs

AI workloads, particularly for training and inference of large language models, often require GPU instances for efficiency. However, the standard configurations listed in the pricing pages primarily feature CPU-based instances, such as m5zn, c5zn, and r5zn, with no explicit mention of GPU instances [7]. While some sources, such as a 2024 article [8], suggest that Outposts support NVIDIA T4 Tensor Core GPUs, the pricing and availability are not well-defined.

To illustrate, in the cloud, GPU instances like G4dn, powered by NVIDIA T4 GPUs, are available with on-demand pricing, such as $0.526 per hour for g4dn.xlarge [9], equating to approximately $378 per month if run continuously. However, on Outposts, the equivalent would likely be bundled into the rack or server cost, which starts at $1,499 per month for CPU-based configurations, suggesting that GPU-enabled setups could be significantly more expensive.

2.4 Retrieval Augmented Generation (RAG) Systems

RAG systems combine LLMs with external knowledge retrieval to improve response quality. Lewis et al. [10] discusses how RAG enhances factual accuracy by retrieving relevant documents before generation. This is particularly useful for domain-specific applications, where the LLM must rely on Disneyland Parks documentation.

A practical guide by Patel [11] outlines integrating RAG with DeepSeek models using tools like LangChain and Chroma, providing a blueprint I adapted for my implementation.

2.5 Mobile AI Applications and TTS

Integrating LLMs into mobile platforms is an active research area. Chen [12] describes using API calls to connect iOS apps to remote LLMs, a method I adopt for my server-client architecture.

For TTS, Brown [13] and the Piper TTS documentation [14] detail implementing speech synthesis in Python.

3 Implementation

The implementation is comprised of four components: infrastructure setup, RAG system integration, iOS app development, and Python script with TTS.

3.1 On-Premises Infrastructure Setup

The foundation of my solution is a local server hosting the DeepSeek-R1-Distill-Qwen-32B model. This solution uses docker compose to manage containers, allowing scalability to Kubernetes, and supports either NVIDIA GPUs or Apple Silicon.

The software environment was prepared as follows:

- Python 3.10: Installed via `apt-get` as the base image.
- PyTorch 1.10: verified with `torch.cuda.is_available()`.
- Hugging Face Transformers 4.20: Installed via `pip` to load and run the DeepSeek model.
- FastAPI: Deployed as the API framework to expose the model to client applications.

The model was downloaded from the DeepSeek-R1 GitHub repository [15] and loaded into memory using the Transformers library. I configured FastAPI to handle POST requests at the `/query` endpoint, where incoming queries are processed by the model and responses are returned in JSON format. This setup allows for client agnostic interfacing, supporting both the iOS app and Python TTTs script.

3.2 RAG System Integration

To augment the LLM with Disneyland Parks documentation, I implemented a RAG system with the following steps:

1. **Preprocessing**: The documentation (PDFs and text files) was cleaned using Python's `re` module to remove formatting artifacts (e.g., extra whitespace, HTML tags). It was then split into 512-token chunks using NLTK's tokenizer.
2. **Embedding Generation**: I used the `all-MiniLM-L6-v2` sentence transformer model from the `sentence-transformers` library to generate 384-dimensional embeddings for each chunk.
3. **Vector Storage**: Embeddings were indexed in a Chroma vector database, configured with cosine similarity for retrieval.
4. **Integration with LLM**: The LangChain library facilitated RAG integration. I set the retriever to fetch the top 5 relevant chunks per query, which are concatenated with the user's input in a custom prompt:

```
"Using the following context from Disneyland Parks documentation: {context},
answer the query: {query}"
```

This prompt ensures the model leverages the Parks documentation.

3.3 iOS App Development

The iOS app provides a mobile interface for querying the AI system. Developed in Swift 5 using Xcode 14, it features:

- **UI Design**: A minimalistic layout with a text field for query input, a submit button, and a scrollable text view for responses.
- **Networking**: Alamofire 5.6 handles HTTP POST requests to the FastAPI server.
- **Error Handling**: Alerts for network failures or invalid responses, with a retry option.
- **Feedback**: A `UIActivityIndicatorView` spins during processing to indicate activity.

The app targets iOS 16+ and assumes a stable local network connection to the server.

3.4 Python Script with TTS

The Python script offers a command-line interface with TTS output, which could be used for integrated systems such as the robot walkers in the Parks. Key features include:

- **Query Handling**: The `requests` library sends queries to the FastAPI server:

```
response = requests.post("http://server-ip:8000/query", json={"query": query})
answer = response.json()["answer"]
```

- **TTS Implementation**: The `pyttsx3` library converts responses to speech. Configuration:

```
engine = pyttsx3.init()
engine.setProperty("rate", 150)  # Speech speed
engine.setProperty("volume", 0.9)  # Volume level
engine.say(answer)
engine.runAndWait()
```

4 Evaluation

I propose a multi-faceted evaluation to assess the system's performance across four metrics: accuracy, response time, resource usage, and TTS quality. I compare the performance of the DeepSeek-R1-Distill-Qwen-32B model to the OrionStarAI/Orion-14B-Base model. Orion is model developed by AWS and should provide a practical comparison given its likelihood to be used in AIaaS applications.

4.1 Accuracy

A test set of 50 queries was created, spanning Disneyland Parks topics (e.g., ”What are the FastPass rules?” ”Where is Space Mountain located?”). Ground truth answers were extracted from the documentation. Responses are evaluated using:

- **BLEU Score**: An automated metric for textual similarity.
- **Human Assessment**: Three evaluators rate relevance and correctness on a 1-5 scale, targeting an average of 4+.

4.2 Response Time

Latency is measured from query submission to response receipt, aiming for ¡2 seconds. Tools include Python's `time` module for the script and Swift's `Date` for the app. Tests simulate concurrent queries to assess performance under load.

4.3 Resource Usage

Server resources are monitored during testing with goals of under 80% CPU and 16GB RAM usage.

4.4 TTS Quality

Five participants evaluate TTS output for 10 responses, rating naturalness and clarity (1-5 scale). I aim for an average score of 4+, indicating high-quality speech suitable for public use.

5 Outcomes

TODO: Add evaluation graph after system has been fully implemented!

6 Conclusions and Future Work

This paper presents a cost-effective, on-premises AI solution using CoT Distilled LLMs, addressing the limitations of cloud-based AIaaS. By deploying the DeepSeek-R1 model locally and integrating a RAG system with Disneyland Parks documentation, I achieve accurate, secure, and efficient query handling. The iOS app and Python script with TTS showcase some practical applications.

6.1 Future work

- **Model Fine-Tuning**: Adapting the LLM to Disneyland-specific terminology for improved accuracy.
- **Advanced RAG**: Implementing hybrid retrieval (e.g., BM25 + embeddings) for better context selection.
- **Scalability**: Distributing the system across multiple nodes using Kubernetes or similar frameworks.

References

[1] IBM Newsroom, "Ibm report: Consumers pay the price as data breach costs reach all-time high," July 2022, accessed: March 06, 2025. [Online]. Available: https://newsroom.ibm.com/2022-07-27-IBM-Report-Consumers-Pay-the-Price-as-Data-Breach-Costs-Reach-All-Time-High?asPDF=1

[2] A. Name, "Distilling step-by-step: Outperforming larger language models with less training data and smaller model sizes," *Google Research Blog*, 2023. [Online]. Available: https://research.google/blog/distilling-step-by-step-outperforming-larger-language-models-with-less-training-data-and-smaller-mode

[3] W. Choi, W. K. Kim, M. Yoo, and H. Woo, "Embodied cot distillation from llm to off-the-shelf agents," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2412.11499

[4] C. Fortuna, D. Muši'c, G. Cerar, A. 'Campa, P. Kapsalis, and M. Mohor'ci'c, "On-premise artificial intelligence as a service," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2210.06956

[5] DeepSeek-AI, A. Liu, B. Feng *et al.*, "Deepseek-v3 technical report," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2412.19437

[6] Amazon Web Services, "Pricing for outposts rack – aws outposts rack pricing," 2023. [Online]. Available: https://aws.amazon.com/outposts/rack/pricing/

[7] ——, "Aws outposts rack features — amazon web services," 2023. [Online]. Available: https://aws.amazon.com/outposts/rack/features/

[8] NVIDIA, "Nvidia gpu-accelerated amazon web services — nvidia," 2023. [Online]. Available: https://www.nvidia.com/en-us/data-center/gpu-cloud-computing/amazon-web-services/

[9] Amazon Web Services, "Amazon ec2 g4 instances — amazon web services (aws)," 2023. [Online]. Available: https://aws.amazon.com/ec2/instance-types/g4/

[10] A. Name, "A survey on evaluation of large language models," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2307.03109

[11] ——, "Building a retrieval-augmented generation (rag) system with deepseek r1: A step-by-step guide," *MarkTechPost*, 2025. [Online]. Available: https://www.marktechpost.com/2025/01/27/building-a-retrieval-augmented-generation-rag-system-with-deepseek-r1-a-step-by-step-guide/

[12] N. Cuong, "How to integrate a large language model into an ios application," *Medium*, 2023. [Online]. Available: https://medium.com/@namcuong2711/how-to-integrate-a-large-language-model-into-an-ios-application-2579d9d119b0

[13] A. Name, "Convert text to speech in python," *GeeksforGeeks*, 2023. [Online]. Available: https://www.geeksforgeeks.org/convert-text-speech-python/

[14] N. Guerra, "How to read text aloud with piper and python," *Noer Guerra Blog*, 2023. [Online]. Available: https://noerguerra.com/how-to-read-text-aloud-with-piper-and-python/

[15] D. AI, "Github - deepseek-ai/deepseek-r1," 2023. [Online]. Available: https://github.com/deepseek-ai/DeepSeek-R1