PaperGist: A Cost-Efficient Cloud-Native Research Paper Summarization Platform

Ronit Tushir
rt3068@nyu.edu

Erchi Zhang
ez806@nyu.edu

Joe Braha
jb7044@nyu.edu

Patrick Cheng
pc2720@nyu.edu

***Abstract*—This project presents a robust, cloud-native research paper summarization platform that leverages a suite of AWS services to deliver scalable, efficient academic paper search and summarization. The system features a user-friendly web interface, with search and upload capabilities, connected via Amazon API Gateway to AWS Lambda functions responsible for searching arXiv, checking for cached summaries, and enqueuing new summarization tasks. Manual uploads are handled through Amazon S3, which generates URLs for further processing. Paper metadata and summaries are persistently managed and cached in Amazon DynamoDB, using both arXiv IDs and document hashes to avoid redundant computation. Summarization jobs are orchestrated through an SQS-based GPU tasks queue and processed in batches on EC2 G5 GPU instances running state-of-the-art large language models (LLMs), such as Llama 3.2:latest. Automated scheduling and resource management are achieved using AWS EventBridge and Lambda-based GPU handlers, ensuring that GPU resources are only used on demand, significantly reducing operational costs. This architecture enables asynchronous, scalable, and reliable generation of high-quality research paper summaries, demonstrating the effective synergy of serverless computing, managed databases, and GPU-accelerated AI in modern cloud applications.**

I. Problem Statement

The pace of scientific discovery has accelerated dramatically in recent years, resulting in an unprecedented surge in the publication of academic research papers across a wide array of disciplines. This ever-expanding body of literature presents a significant challenge for researchers, students, and professionals who strive to remain informed about the latest developments in their fields. The traditional approach of manually reading and analyzing each research paper is rapidly becoming impractical, as it demands substantial time and cognitive effort, and increases the risk of overlooking important findings or trends.

Moreover, the diversity and complexity of modern scientific writing—often characterized by specialized terminology, intricate methodologies, and dense technical language—further complicate the task of extracting key insights from individual papers. The sheer volume of new publications, combined with the interdisciplinary nature of much contemporary research, means that even experts can struggle to identify the most relevant and impactful work in their area.

This information overload not only affects individual researchers but also has broader implications for the scientific community and society at large. When critical knowledge is buried within vast digital libraries, the pace of innovation slows, collaboration becomes more difficult, and the potential for cross-disciplinary breakthroughs is diminished. Furthermore, the lack of efficient tools for synthesizing and contextualizing research findings can lead to duplication of effort, missed opportunities, and a general sense of frustration among those seeking to advance their fields.

To address these challenges, there is a pressing need for an intelligent, automated solution that can efficiently search, retrieve, and synthesize the essential information from vast collections of research articles. Leveraging the capabilities of large language models (LLMs) and advanced natural language processing, such a tool must be designed for reliability, scalability, and efficiency in a cloud-based environment. This would empower users to quickly discover relevant research, comprehend critical contributions, and stay abreast of new scientific knowledge with minimal manual effort.

II. Motivation

The motivation for this project stems from the transformative potential that intelligent automation and cloud computing hold for the research community. As the volume and complexity of scientific literature continue to grow, the ability to efficiently access, understand, and apply new knowledge becomes a key driver of progress in both academia and industry. Researchers, students, and professionals are increasingly confronted with the daunting task of filtering through thousands of papers to find those most relevant to their work, often under tight time constraints and with limited resources.

Existing tools for literature search and summarization, while helpful, frequently fall short in several respects. Many rely on basic keyword matching or statistical techniques that do not capture the nuanced context, logical flow, or domain-specific language of scientific texts. As a result, users may receive incomplete or misleading summaries, or miss out on important connections between papers. Furthermore, most current solutions are not designed to scale with the ever-increasing volume of publications, nor do they offer the flexibility and reliability required for integration into modern research workflows.

By harnessing the power of large language models (LLMs) and deploying them within a robust, cloud-based infrastructure, this project aims to bridge these gaps. The goal is to create a system that not only automates the process of summarizing and contextualizing research papers but also integrates seamlessly with existing digital libraries and user interfaces. Such a tool would enable users to rapidly identify key findings, track emerging trends, and make more informed decisions about which papers to read in full. Ultimately, this approach promises to democratize access to scientific knowledge, foster greater collaboration across disciplines, and accelerate the pace of discovery in an increasingly data-driven world.

III. Existing Solutions

A variety of tools and methodologies have been developed to address the challenge of navigating and synthesizing the ever-growing body of academic literature. Early approaches to research paper summarization were predominantly extractive and rule-based. Algorithms such as TextRank and LexRank utilized graph-based and statistical techniques to identify and extract the most salient sentences from a document, while systems like MEAD and SumBasic relied on frequency-based heuristics and manually crafted rules. Although these methods provided a foundation for automated summarization, they often resulted in fragmented and contextually shallov summaries, as they lacked the ability to understand the deepe semantic relationships within scientific texts.

With the advent of machine learning and, more recently deep learning, the field has seen significant advancements Sequence-to-sequence (Seq2Seq) models and attention mech anisms enabled the development of abstractive summarizatio systems capable of generating more coherent and human-lik summaries. Transformer-based architectures, such as BERT SUM, PEGASUS, and various GPT models, have furthe improved the quality of automated summaries by leverag ing large-scale pretraining and contextual embeddings. Thes models are able to capture complex relationships and generat fluent, context-aware summaries that are much closer to thos written by humans.

Despite these advances, several limitations persist. Man state-of-the-art models require substantial computational re sources for both training and inference, which can hinde their deployment at scale, especially in real-time or resource constrained environments. Additionally, most summarization tools are designed as standalone applications or research prototypes, lacking seamless integration with academic search engines or digital libraries. This fragmentation often forces users to manually transfer documents between platforms, reducing workflow efficiency.

Some platforms, such as Semantic Scholar and Scholarcy, have begun to integrate summarization features directly into their search and discovery interfaces. Semantic Scholar, for example, offers a TL;DR feature that provides concise, AI-generated summaries for select papers. However, these solutions are typically limited in scope, may not cover the full breadth of available literature, and often rely on proprietary models that are not easily extensible or customizable by end users.

Furthermore, domain-specific summarization remains a significant challenge. Scientific literature is characterized by specialized terminology, complex argumentation, and diverse formats, which can confound generic summarization models. Tools tailored for specific domains, such as biomedical literature, often require extensive retraining on curated datasets, limiting their generalizability and adaptability to new fields.

Finally, issues of scalability, efficient data management, and semantic search remain largely unresolved in existing solutions. Many systems do not provide robust mechanisms for indexing, storing, and retrieving large volumes of summarized content, nor do they offer advanced search capabilities that leverage semantic understanding rather than simple keyword matching.

In summary, while substantial progress has been made in automated research paper summarization, there remains a clear need for an integrated, cloud-based solution that combines high-quality, context-aware summarization with scalable search, storage, and retrieval capabilities. Such a system would bridge the gap between cutting-edge natural language processing and the practical needs of the research community.

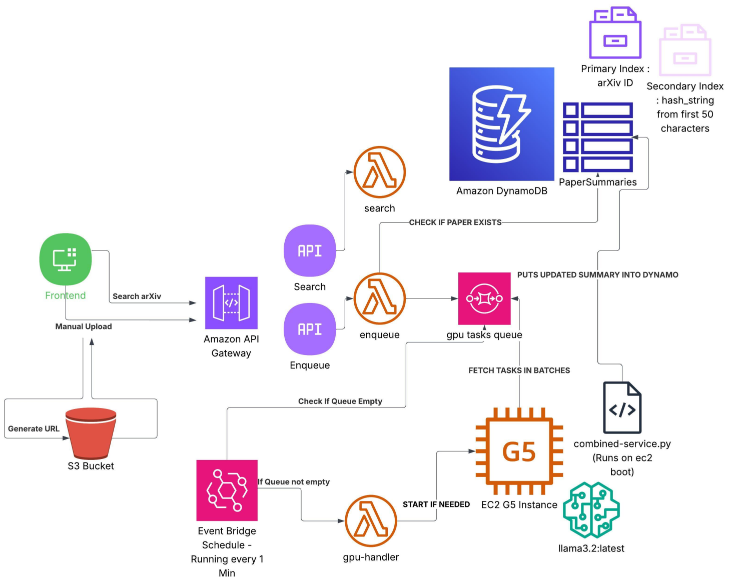


Fig. 1. System Architecture Diagram

IV. Methods

A. System Architecture

As shown in Fig. 1, the application is composed of several cloud-based components designed for scalability and efficiency. The user interacts with a web frontend, which communicates with backend services via AWS API Gateway. For arXiv search, the frontend sends requests through the API Gateway to a Lambda function that queries arXiv and DynamoDB. For manual uploads, the frontend uploads files directly to an S3 bucket, which generates a URL for further processing.

API Gateway routes search and enqueue requests to dedicated AWS Lambda functions. When a summarization task is enqueued, it is placed in the GPU tasks queue (implemented with Amazon SQS). AWS EventBridge triggers the ‘gpu-handler‘ Lambda function every minute to check if there are tasks in the queue. If tasks are present and the GPU instance is not running, the handler starts an EC2 G5 instance. Upon boot, the instance runs the summarization service ('combined-service.py' with 'llama3.2:latest'), which fetches tasks in batches from the queue, processes them, and updates the results in DynamoDB.

Paper metadata and summaries are stored in DynamoDB, which uses the arXiv ID as the primary index and a hash string as a secondary index for efficient querying. Uploaded documents are stored in S3. This architecture ensures high availability, cost-effectiveness, and seamless scalability, making it suitable for both individual researchers and larger organizations.

B. System Component Choices

1) Frontend: Static Web Application on S3: We chose AWS S3 to host the frontend as a static web application because it offers high reliability, global availability, and cost-effectiveness for serving static content. S3's seamless integration with AWS services simplifies deployment, versioning, and scaling, allowing us to deliver a responsive user interface without managing traditional web servers. This approach reduces operational complexity and ensures that users can access the application quickly from anywhere. For manual uploads, the frontend uploads files directly to S3, which stores the documents for further processing and deduplication.

2) API Gateway: AWS API Gateway was selected as the entry point for all client requests due to its ability to provide secure, scalable, and fully managed HTTP endpoints. It supports features such as request validation, throttling, and authorization, which are essential for protecting backend resources and ensuring consistent performance under varying loads. Its native integration with AWS Lambda enables us to build a serverless backend architecture, further enhancing scalability and maintainability.

3) Backend Logic: AWS Lambda: AWS Lambda was chosen for implementing the core backend logic because of its serverless, event-driven execution model. Lambda automatically scales in response to incoming requests, eliminating the need for manual provisioning or management of compute resources. This not only reduces operational overhead but also ensures cost efficiency, as we only pay for actual compute time consumed. Lambda's tight integration with other AWS services enables rapid development and deployment of modular, maintainable backend functions.

4) Task Queue: Amazon SQS: Amazon Simple Queue Service (SQS) was selected to decouple the submission of summarization tasks from their processing. SQS provides a reliable, scalable, and fully managed message queuing service that ensures tasks are processed asynchronously and in the correct order. This design improves system resilience, as it allows the backend to handle spikes in workload without losing or duplicating tasks, and enables efficient batch processing by downstream components.

5) LLM Inference: EC2 G5 GPU Instance: For computationally intensive summarization tasks, we utilize an EC2 G5 GPU instance. This instance type is specifically optimized for machine learning inference workloads, offering high-performance NVIDIA GPUs and fast networking. The LLM inference service (using the Llama 3.2:latest model) runs on this instance and processes queued tasks in batches for efficiency. By running the LLM inference service on a dedicated GPU instance, we can efficiently process large batches of summarization tasks while maintaining flexibility to start or stop the instance as needed, optimizing both performance and cost.

6) Data Storage: DynamoDB and S3: Amazon DynamoDB was chosen as the primary data store for managing paper metadata, processing status, and generated summaries due to its low-latency performance, seamless scalability, and fully managed nature. DynamoDB's NoSQL architecture is well-suited for handling dynamic and unstructured data at scale, and the use of both primary and secondary indexes enables efficient querying. AWS S3 complements this by providing durable, cost-effective storage for large uploaded documents, ensuring that both metadata and raw files are stored securely and can be retrieved efficiently. In our case, for manual uploads, a hash of the document is used as a secondary index in DynamoDB to detect and reuse existing summaries, further reducing redundant computation. Uploaded files are stored in S3 for persistent storage and later processing.

7) Event-Driven Orchestration: EventBridge: AWS EventBridge is used to automate scheduled operations and resource management, such as triggering the GPU handler Lambda function to manage the lifecycle of the GPU instance. This event-driven approach improves system responsiveness, ensures timely processing of queued tasks, and reduces manual intervention.

8) Large Language Model: LLaMA 3: LLaMA 3 was selected as our large language model for summarization due to its state-of-the-art performance in natural language understanding and generation. Its advanced capabilities enable the production of high-quality, contextually accurate summaries of research papers, which is central to the effectiveness of our tool.

9) Hashing Technique for Manual Uploads: To further reduce redundant computation, we implemented a global secondary index (GSI) in DynamoDB based on a lightweight hashing mechanism. When a user uploads a document manually, the system extracts plain text from the file and computes a hash using the first 50 characters of that extracted content. This hash serves as a fingerprint for the document and is checked against the GSI before any new summarization is initiated.

If a match is found, the system directly returns the previously generated summary associated with the hash, avoiding a new upload to S3 or a task queue insertion. This is particularly cost-effective, as manual uploads can vary widely in format and content duplication is common. The use of content-derived hashing ensures that inference on GPU resources is only triggered for genuinely new documents.

C. Cost Optimization

To optimize operational costs, the system employs an event-driven approach to GPU resource management. The EC2 G5 GPU instance is only started when there are summarization tasks in the queue, as detected by a scheduled EventBridge handler. If the queue remains empty for 10 minutes, the GPU instance is automatically shut down, minimizing unnecessary resource usage and cost. This design ensures that expenses are incurred only when computation is required, making the system highly efficient for sporadic or low-traffic usage patterns.

While the current configuration is well-suited for a proof-of-concept and small-scale deployments, these parameters (such as the inactivity timeout) can be further tuned or enhanced for enterprise-scale applications. This flexible, cost-aware approach demonstrates the system's adaptability and potential for future optimization.

V. Results

To demonstrate the functionality and effectiveness of our cloud-based research paper summarization tool, we present several screenshots of the deployed web application and discuss key system outcomes.

A. User Interface and Search Functionality

Figure 2 shows the main search page, where users can enter queries to find relevant research papers. The interface is designed to be intuitive and responsive.

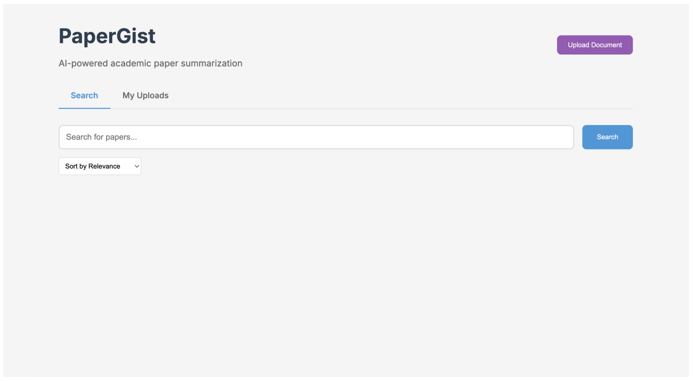


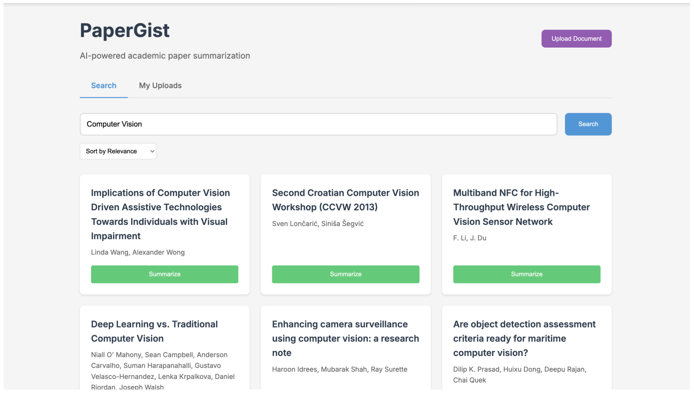
Fig. 2. Main search page of the application.

B. Search Results

Figure 3 shows the search results after user entered some keywords. User can choose to have the results shown sorted by relevance, by date, or by the time last updated.

C. Summarization Page

Figure 4 shows the summary results appearing on the website after the summary has been made.



Implications of Computer Vision Driven Assistive Technologies Towards Individuals with Visual Impairment
Linda Wang, Alexander Wong
Summarize

Second Croatian Computer Vision Workshop (CCVW 2013)
Sven Lončarić, Siniša Šegvić
Summarize

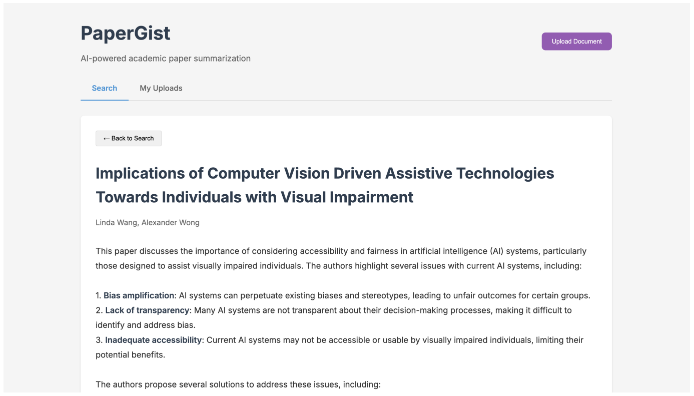
Multiband NFC for High-Throughput Wireless Computer Vision Sensor Network
F. Li, J. Du
Summarize

Deep Learning vs. Traditional Computer Vision
Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco-Hernandez, Lenka Krpalkova, Daniel Riordan, Joseph Walsh

Enhancing camera surveillance using computer vision: a research note
Haroon Idrees, Mubarak Shah, Ray Surette

Are object detection assessment criteria ready for maritime computer vision?
Dilip K. Prasad, Huixu Dong, Deepu Rajan, Chai Quek

Fig. 3. Search results page of the application.



Implications of Computer Vision Driven Assistive Technologies Towards Individuals with Visual Impairment

Linda Wang, Alexander Wong

This paper discusses the importance of considering accessibility and fairness in artificial intelligence (AI) systems, particularly those designed to assist visually impaired individuals. The authors highlight several issues with current AI systems, including:

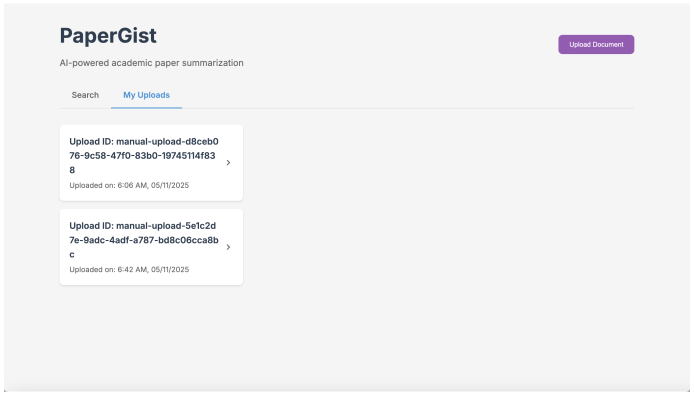
1. **Bias amplification**: AI systems can perpetuate existing biases and stereotypes, leading to unfair outcomes for certain groups.
2. **Lack of transparency**: Many AI systems are not transparent about their decision-making processes, making it difficult to identify and address bias.
3. **Inadequate accessibility**: Current AI systems may not be accessible or usable by visually impaired individuals, limiting their potential benefits.

The authors propose several solutions to address these issues, including:

Fig. 4. Summarization page for application.

D. Upload Functionality

The user can use the "Upload Document" button to manually upload their papers. Then their paper will be queued for summarization, and the paper information as well as the summarization will appear in the "My Uploads" page, as shown in Figure 5.



Upload ID: manual-upload-d8ceb076-9c58-47f0-83b0-19745114f838
Uploaded on: 6:06 AM, 05/11/2025

Upload ID: manual-upload-5e1c2d7e-9adc-4adf-a787-bd8c06cca8bc
Uploaded on: 6:42 AM, 05/11/2025

Fig. 5. Manual uploaded documents page.

E. System Performance

The system is capable of processing and summarizing research papers with high efficiency. When the GPU instance is already running, the average summary generation time is typically 15–20 seconds per paper. However, if the GPU instance has been turned off due to 10 minutes of inactivity (i.e., an empty task queue), there is an additional startup delay. In this case, the EventBridge scheduler, which runs every minute, detects the new task and initiates the GPU instance. The instance startup process generally takes up to 2 minutes, after which the paper is processed as usual. Therefore, in the worst-case scenario—when the GPU is off and a new task arrives just after the last scheduler run—the total time to summary completion can be up to 3 minutes longer than the standard processing time. This event-driven, cost-optimized approach ensures that the system remains both responsive and economical, balancing rapid processing for active periods with resource savings during idle times.

VI. Conclusion

This project tackled the growing challenge of information overload in academic research by developing a scalable, cloud-based research paper summarization tool. Leveraging modern cloud technologies such as serverless computing, managed databases, and GPU-accelerated large language models, our system enables users to efficiently search, retrieve, and understand research papers. By automating the summarization process and providing an intuitive user interface, the tool significantly reduces the time and effort required to stay current with scientific advancements.

The architecture was carefully designed to ensure high availability, cost-effectiveness, and seamless scalability, making it suitable for both individual researchers and larger organizations. Through the integration of AWS Lambda, API Gateway, DynamoDB, S3, and EC2 GPU instances, the system demonstrates how cloud-native solutions can address the computational and storage demands of large-scale natural language processing tasks. The use of advanced LLMs, such as Llama 3, further enhances the quality and contextual relevance of generated summaries, bridging the gap between raw scientific literature and actionable insights.

While the current implementation provides a robust foundation, it also highlights several areas for future improvement, such as expanding repository support, enhancing user personalization, and integrating more advanced evaluation mechanisms. The project underscores the importance of interdisciplinary collaboration, combining expertise in cloud engineering, machine learning, and user experience design to deliver a practical and impactful solution.

Overall, this work demonstrates the transformative potential of integrating state-of-the-art natural language processing with robust cloud infrastructure. By making scientific knowledge more accessible and comprehensible, our system not only empowers researchers to stay informed but also fosters greater innovation and collaboration within the global research community. As the volume of academic literature continues to grow, such intelligent, automated tools will become increasingly vital in supporting the advancement of science and technology in the digital era.

VII. Future Work

While this project successfully demonstrates a scalable and efficient cloud-based research paper summarization tool, there are several promising directions for future enhancement:

- **Stronger and Faster LLM Models:** As large language models continue to advance, integrating more powerful or specialized models could further improve the quality, speed, and domain-adaptability of generated summaries, especially for highly technical or specialized papers.
- **User Authentication and Account Management:** Implementing user authentication and session-based tracking would enable features such as saving search history, bookmarking papers, and managing custom summary requests, thereby enhancing the user experience and enabling persistent user history across devices.
- **Support for Additional Paper Repositories:** Expanding the system to include more research repositories beyond arXiv—such as PubMed, IEEE Xplore, or Springer—would broaden the tool's applicability and provide users with access to a wider range of academic literature.
- **Premium and Instant Summarization APIs:** Providing instant or priority summarization APIs for premium users could offer differentiated service levels and support a sustainable business model.
- **Semantic Search and Discovery:** Integrating semantic search capabilities (e.g., using Typesense or OpenSearch) over the cached summaries would allow users to find relevant papers and insights more effectively than with keyword search alone.
- **Trending and Most-Requested Papers:** Highlighting trending or frequently summarized papers could help users discover important or popular research, while also reducing redundant inference by reusing cached summaries.
- **Enhanced Evaluation and Feedback:** Adding mechanisms for users to rate or provide feedback on generated summaries could help improve summarization quality over time through active learning or model fine-tuning.
- **Mobile and Accessibility Features:** Developing a mobile-friendly interface and adding accessibility features would make the tool more inclusive and convenient for a diverse user base.
- **Local Development and Deployment:** Providing Docker and Terraform scripts for local development and streamlined AWS deployment would facilitate easier onboarding for contributors and reproducibility for researchers.

Due to resource constraints, these enhancements were beyond the scope of the current implementation, but they represent promising directions for future development and potential enterprise-scale deployment.